Florian Evers

Prädiktive Middleware-basierte Mobilitätsunterstützung für multikriterielle Handover

Prädiktive Middleware-basierte Mobilitätsunterstützung für multikriterielle Handover

Florian Evers

Universitätsverlag Ilmenau
2011

Impressum

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Angaben sind im Internet über http://dnb.d-nb.de abrufbar.

Diese Arbeit hat der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität Ilmenau als Dissertation vorgelegen.

Tag der Einreichung:	30. September 2010
1. Gutachter:	Prof. Dr. rer. nat. (habil.) Jochen Seitz (Technische Universität Ilmenau)
2. Gutachter:	Prof. Dr.-Ing. habil. Andreas Mitschele-Thiel (Technische Universität Ilmenau)
3. Gutachter:	Prof. Dr.-Ing. habil. Jochen Schiller (Freie Universität Berlin)
Tag der Verteidigung:	8. April 2011

Technische Universität Ilmenau/Universitätsbibliothek
Universitätsverlag Ilmenau
Postfach 10 05 65
98684 Ilmenau
www.tu-ilmenau.de/universitaetsverlag

Herstellung und Auslieferung
Verlagshaus Monsenstein und Vannerdat OHG
Am Hawerkamp 31
48155 Münster
www.mv-verlag.de

ISBN 978-3-939473-99-2 (Druckausgabe)
URN urn:nbn:de:gbv:ilm1-2011000091

Titelfoto: photocase.com | AlexFlint

Sicht des Drachen

Da es Hände gibt, die mich an die Erde fesseln,
kann ich die Himmelstreppe erklimmen.
Jedesmal, wenn ich meine Schulter schüttelnd gegen den Wind wende,
werde ich Stück für Stück tiefer in den Himmelsschoß gesogen
Da es Hände gibt, die mich an die Erde fesseln,
hängt die Erde an meiner Schnur.

Maoto Ooka, japanischer Dichter

Danksagung

Nachdem jetzt auch die letzte Korrektur eingearbeitet ist, habe ich endlich das erreicht, worauf ich mich schon seit beinahe sechs Jahren als wissenschaftlicher Mitarbeiter im Fachgebiet Kommunikationsnetze der TU Ilmenau freue. Meine Doktorarbeit ist fertig.

Dies wäre jedoch nicht möglich gewesen, wenn mich mein Betreuer Herr Prof. Dr. rer. nat. habil. Jochen Seitz nicht so umfasssend über die Jahre unterstützt und immer ein offenes Ohr für meine Fragen und Probleme gehabt hätte. Sehr dankbar bin ich für sein stetiges Feedback sowie seine damalige Erlaubnis, im entscheidenden Moment von meinem vorherigen Forschungsthema „Sensornetze“ wieder abrücken zu dürfen.

Ein besonderer Dank gilt Herrn Prof. Dr.-Ing. habil. Andreas Mitschele-Thiel sowie Herrn Prof. Dr.-Ing. Jochen Schiller, welche das Korreferat übernommen haben und mir im Vorfeld die Möglichkeit gaben, mich kritischen Fragen zu meiner Ausarbeitung stellen zu können.

Ich habe mich über die Jahre sehr gefreut, Teil eines tollen Team sein zu dürfen. Einen besonders großen Dank möchte ich an Dipl.-Ing. Yevgeniy Yeryomin aussprechen, ohne dessen aktives Mitwirken ich mich wahrscheinlich nicht an die Thematik der SIP-basierten Sprachkommunikation herangetraut hätte. Klasse war auch das Arbeitsklima in unserem Büro H3505, wofür ich meinem Kollegen Dr.-Ing. Maik Debes sehr dankbar bin. Ich werde unsere interessanten Diskussionen vermissen.

Sehr gefreut und weitergebracht hat mich das Engagement einer Vielzahl an Studenten, welche mit ihren Ausarbeitungen zum Gelingen dieser Dissertation beigetragen haben. Dies waren insbesondere, in alphabetischer Reihenfolge, die Studenten Ahmad Ahmad, André Hesse, Danilo Hoffmann, Dominik Schulz, François Dourthe, Jens Hörnlein, Matthias Tosse, Sebastian Delling und Stefan Zech. Ich freue mich wirklich sehr, dass Eure Ergebnisse nicht bloß als Papierstapel in meinem Büro endeten, sondern sich auf den folgenden Seiten wiederfinden lassen. Vielen Dank dafür!

Ein sehr großer Dank gilt meinen Freunden Alexander Ehrenberg, Tilo Ziehn, Holger Hildebrandt, Jan Richarz, Sebastian Göbel und Thomas Leidenfrost, welche mich über die Jahre begleitet haben und mir die nötige Zerstreuung boten. Selbiger Dank gilt den Funkamateuren des Ilmenauer Ortsverbands X30 sowie den Lenkdrachenfliegern des Teams Syndikite.

Nicht zuletzt möchte ich meinen Eltern sowie meiner Schwester Claudia samt Familie danken. Ihr wart immer für mich da, und habt mir den nötigen Rückhalt gegeben, ohne den ich wohl jetzt nicht diese Zeilen schreiben würde. Danke!

Kurzzusammenfassung

Die vorliegende Dissertation befasst sich mit Mobilitätsunterstützungen für moderne mobile Endgeräte, welche auf das Internet zugreifen sollen. Um die Vielzahl der bereits verfügbaren Lösungen zu klassifizieren, werden Anforderungen definiert, welche sich an „unbedarften Nutzern“ orientieren, die ihre Geräte wie gewohnt auch im mobilen Umfeld benutzen möchten. Es stellte sich heraus, dass lediglich die Gruppe der Middleware-basierten Mobilitätsunterstützungen diese Anforderungen erfüllen kann. Da jedoch noch keine der gefundenen Lösungen dazu in der Lage war, wurde die „Roaming-Enabled Architecture“ (REACH) entwickelt.

REACH tritt mit der Maßgabe an, alle gestellten Anforderungen zu erfüllen. Dazu werden Proxyserver eingesetzt, welche als „Ankerpunkte“ fungieren und die Server im Internet vor den negativen Auswirkungen von Mobilitätsereignissen abschotten. Werden mehrere Proxyserver gleichzeitig involviert, kann sogar den negativen Auswirkungen durch das „Dreiecks-Routing“ entgegengewirkt werden. Eine große Herausforderung bestand jedoch darin, die Datenströme der nicht modifizierten Anwendungen abzufangen, damit die Sicherungsmechanismen von REACH greifen können. Dazu wird eine Vielzahl an Möglichkeiten diskutiert, wobei ein besonderes Leistungsmerkmal von REACH darin besteht, diese Mechanismen auch in Kombination anbieten zu können. So stellen beispielsweise längerfristige Isolationssituationen keine Probleme mehr für die Anwendungen dar, und es steht erstmalig eine Mobilitätsunterstützung zur Verfügung, welche in solchen Situationen auch die Benutzer mit einbeziehen kann.

Weiterhin wird ein prädiktiv arbeitender Handoverentscheider vorgestellt, welcher durch Beobachtung von Signalstärkewerten drahtlos arbeitender Netzzugangstechnologien drohende Abrisse erkennen kann. REACH unterstützt prinzipiell beliebige Netzzugangstechnologien, solange diese einen Zugang zum Internet ermöglichen können, und erlaubt „weichere vertikale Handover“ sowie Kanalbündelungsszenarien. So kommen im Demonstrator bereits drahtgebundenes Ethernet, drahtloses arbeitendes WLAN sowie GPRS mittels eines per Bluetooth angebundenen Mobiltelefons zum Einsatz. Der Demonstrator stellt eine praxistaugliche Mobilitätsunterstützung dar, was durch Tests belegt wird und bereits während zahlreicher praktischer Vorführungen einem breit gefächerten Publikum vorgestellt werden konnte.

Abstract

This doctoral thesis deals with mobility extensions for modern mobile devices that want to access resources of the Internet. At first, the existing approaches are classified, with requirements that were derived with focus on normal users. These users want to use their mobile devices in scenarios that involve mobility. It is shown that only the group of middleware-based solutions is able to fulfil these requirements. However, none of the existing solutions was suitable, thus the „Roaming-Enabled Architecture“ (REACH) was created.

REACH implements such a middleware-based approach and was designed to fulfil all requirements. It involves proxy servers that act as „anchor points“ isolating the servers in the Internet from the negative effects of mobility. By connecting to multiple proxy servers at the same time, REACH is able to minimize the negative effects of „triangle routing“. Furthermore, it was challenging to intercept the data streams of the unmodified applications, to apply the protection schemes of REACH. Multiple possibilities are discussed, and it is a special feature of REACH that they are all available and can be combined with each other. Long lasting isolation situations are not problematic, and REACH is the first mobility extension that is able to involve the users.

Additionally, a predictive handover management scheme is presented. It is able to analyze signal strength measurements of wireless networks to predict link loss events before they actually happen. REACH is able to involve any network access technology, as long as it offers access to the Internet. „Softer handovers“ are possible as well as channel bundling scenarios. The testbed already supports Ethernet, WIFI and is able to involve GPRS by accessing a cellphone that is connected via bluetooth. REACH offers a practical solution to allow Internet access in mobile environments. Tests were made in order to underline this. Therefore, REACH was already presented during multiple public demonstrations, where a diversified audience was present.

Inhaltsverzeichnis

1 Einleitung

Die vorliegende Dissertation beschäftigt sich mit Mobilitätsunterstützungen für mobile Endgeräte, welche auf Ressourcen im Internet zugreifen sollen. Es wird gezeigt, dass allein Lösungen aus der Gruppe der Middleware-basierten Architekturen praktikabel und zugleich umfassend sein können. Darin bestand die Motivation zur Erstellung der „Roaming-Enabled Architecture“ (REACH), deren Konzepte in den folgenden Kapiteln vorgestellt und begründet werden. Das vorliegende Kapitel beleuchtet die zu Grunde liegende Problematik und hebt die Zielstellung dieser Ausarbeitung hervor. Abschließend werden die Inhalte der Folgekapitel vorgestellt.

1.1 Motivation

Moderne mobile Endgeräte wie Laptops, „Personal Digital Assistants“ (PDAs) und „Smartphones“ sehen Möglichkeiten vor, sich mit dem Internet zu verbinden. Dazu können eine Vielzahl an drahtgebundenen und drahtlos arbeitenden Technologien zum Einsatz kommen, wobei viele Geräte sogar mehrere Netzzugangstechnologien gleichzeitig anbieten oder um weitere ergänzt werden können. Praxisrelevant sind die Technologien „Ethernet“ mit Übertragungsgeschwindigkeiten bis zu 10 GBit/s, drahtlos arbeitendes „Wireless Local Area Network“ (WLAN) aus der vom „Institute of Electrical and Electronics Engineers“ (IEEE) standardisierten „Familie“ IEEE 802.11, zellulärer Mobilfunk mittels „General Packet Radio Service“ (GPRS) über das „Global System for Mobile Communications“ (GSM), „High Speed Downlink Packet Access“ (HSDPA) über das „Universal Mobile Telecommunications System“ (UMTS) sowie „Long Term Evolution“ (LTE). Aber auch Einwahlverbindungen mittels „analoger Modems“ oder dem „Integrated Services Digital Network“ (ISDN) sind noch relevant und treten vereinzelt auf. Ein Internetzugang mittels „Digital Subscriber Line“ (DSL) ist ebenfalls verbreitet, allerdings wird das Endgerät hierbei schlussendlich wieder über Ethernet oder WLAN angebunden.

Wird ein solches Endgerät „mobil“, weil es beispielsweise von seinem Benutzer durch die Gegend getragen wird oder weil es ein fester Bestandteil eines Fahrzeugs ist, wird

es bezüglich einer jeden Netzzugangstechnologie Areale mit unterschiedlicher Netzabdeckung durchqueren. Der naheliegende Wunsch, dass das mobile Endgerät darauf mit einem Wechsel der aktiven Netzzugangstechnologie reagieren soll, ist im heutigen Internet leider nicht praktikabel. Bei jedem Wechsel der Netzzugangstechnologie, welche als „vertikale Handover“ bezeichnet werden, bekommt das mobile Endgerät eine neue Internetadresse zugewiesen. Da dieser Adresswechsel für die Kommunikationspartner im Internet nicht erkennbar ist, kommt es zu einem Erreichbarkeitsproblem, wobei alle statusbehafteten Transportschichtverbindungen abreißen. Selbst wenn nur der Netzzugangspunkt gewechselt wird, und die involvierte Netzzugangstechnologie dieselbe bleibt („horizontaler Handover“), kommt es zu Problemen, sobald abweichende Internetadressen zugewiesen werden. Hierbei tritt deutlich zu Tage, dass die *heutzutage* im Internet eingesetzten Protokolle mit Blick auf statische Netze entwickelt worden sind, und im mobilen Umfeld versagen.

Um mit den Anforderungen im mobilen Umfeld umgehen zu können, sind verschiedene Ansätze denkbar. Man könnte hierzu auf modernere Protokolle umschwenken oder auf Erweiterungen der bestehenden Protokolle zurückgreifen, welche die genannten Schwachstellen nicht länger aufweisen. Derartige Vorschläge sind zahlreich, allerdings wird hierbei die Modifikation einer unüberschaubaren Anzahl an Systemen erforderlich, weshalb aus praktischen Gründen andere Alternativen gesucht werden sollen.

Außerdem werden im mobilen Umfeld zusätzliche Verwaltungsaufgaben deutlich, da Netzzugänge nicht länger als unveränderlich angesehen werden können, sondern eine andauernde Beobachtung und Eignungsprüfung stattfinden muss. Die hierfür verantwortliche Komponenten müssen konzipiert werden und auf den mobilen Endgeräten die Kontrolle über sämtliche Netzzugangsgeräte übernehmen. Dabei sollen die Endanwender weitestgehend entlastet werden, sodass sie sich nur noch um die Festlegung von Rahmenbedingungen zu kümmern brauchen, und der Rest von der Mobilitätsunterstützung erledigt wird.

Es handelt sich hierbei um ein reales Problem aus der Praxis, dessen negative Auswirkungen bereits viele Nutzer anhand eigener Erfahrungen bemerken mussten. Glücklicherweise wurden bereits eine Vielzahl an Mechanismen von anderen Forschungsgruppen vorgeschlagen, welche aber oftmals mit Blick auf einen besonderen Einsatzzweck entstanden sind, und deren Eignung für einen „normalen Internetzugang“ mit „typischem Nutzungsverhalten“ weiterer Untersuchungen bedürfen.

1.2 Zielstellungen und wissenschaftliche Leistungen

Im Rahmen dieser Dissertation werden diverse Lösungen zu mobilitätsspezifischen Problemstellungen vorgestellt, welche neuartig sind und den Stand der Forschung voranbringen. So ist mit REACH eine „umfassende“ Mobilitätsunterstützung entstanden, was bedeutet, dass ein Gesamtsystem mit Komponenten zur Handoverentscheidung, zur Verwaltung der am mobilen Endgerät verfügbaren Netzzugangsgeräte sowie einer Bedienschnittstelle entwickelt worden ist. Letztere musste allerdings optional bleiben, da die Mobilitätsunterstützung auch auf Geräten einsetzbar sein sollte, welche über keinen Bildschirm verfügen.

Mit der „REACH-Box“ wird ein solches „abgesetztes Gerät“ vorgestellt, welches ein lokales Netzwerk um eine Mobilitätsunterstützung ausstattet, ohne dass die angeschlossenen Systeme in irgendeiner Form modifiziert oder umkonfiguriert zu werden brauchen. Mit dieser als „Netzmobilität“ bezeichneten Anbindungsform können erstmals auch nicht modifizierte Komponenten eines Fahrzeugnetzes, wie beispielsweise Navigationssysteme, Autoradios oder Steuergeräte, „handoversicher“ auf das Internet zugreifen.

Da REACH als Middleware-basierte Architektur „oberhalb“ der Transportschicht angesiedelt ist, muss es sich mit nicht modifizierten Transportschichtprotokollen und dem heutzutage eingesetztem Internetprotokoll (IP) in der Version 4 (IPv4) begnügen, was von Anfang an größtmögliche Kompatibilität mit handelsüblichen Netzzugangsgeräten und den bestehenden Netzzugangspunkten versprach. Damit ist REACH eine wirklich praktikable Lösung, welche im heutigen Internet eingesetzt werden kann.

REACH kann mit beliebigen drahtlosen und drahtgebundenen Netzzugangstechnologien umgehen und durch eine modulare Verwaltung auch zukünftige Varianten unterstützen. Bisherige Handoverentscheider bieten keine derartige Flexibilität, und sind oftmals auf einen ausgewählten Satz an Netzzugangstechnologien beschränkt. So bestand für eine technologieunabhängige Verwaltung von Netzzugängen signifikanter Forschungsbedarf. Es zeigte sich, dass eine kombinierte Betrachtung von Netzzugangsgerät und Netzzugangspunkt erforderlich ist, um die Eignung eines jeden Netzzugangs abschätzen zu können. Dabei sollte der Wechsel von Netzzugangspunkten sowohl innerhalb derselben Technologie („horizontaler Handover“) als auch in Verbindung mit einem Technologiewechsel („vertikaler Handover“) möglich sein. REACH ist hierbei eine von nur sehr wenigen Mobilitätsunterstützungen, welche auch „weichere Handover“ zwecks Erhöhung der Ausfallsicherheit unterstützt. Nach Auffassung des Autors ist REACH zudem die einzige Architektur, welche dem Nutzer eine technologieübergreifende Bündelung von Übertragungswegen anbietet (Kanalbündelungsszenario).

Von besonderem Interesse war eine Betrachtung des Verlaufs von Signalstärkewerten

drahtlos arbeitender Netzzugangsgeräte, da vermutet wurde, dass sich auf ihrer Basis zukünftige Abrissereignisse vorhersagen lassen. Aufbauend auf einer umfassenden Betrachtung der Problematik wurde für REACH ein prädiktiv arbeitender Handoverentscheider entwickelt, dessen Alleinstellungsmerkmal darin besteht, einen Ersatzlink zeitnah noch vor dem Abrissereignis eines Hauptlinks bereitstellen zu können. Der Entscheider ist nachgewiesenermaßen dazu in der Lage, für WLAN sowohl innerhalb als auch außerhalb von Gebäuden überzeugende Abrissprognosen errechnen zu können.

Da Middleware-basierte Mobilitätsunterstützungen die gemeinsame Eigenschaft aufweisen, dass feste „Ankerpunkte“ für den Datenverkehr in Richtung der nicht modifizierten Server benötigt werden, deutete sich Untersuchungsbedarf bezüglich der Suche nach „geeigneten“ Proxyservern an. Für Problematiken wie „Dreiecks-Routing“ oder der Realisierung von Lastverteilungskonzepten stand vorab keine triviale Lösung bereit. Durch eine umfangreiche konzeptionelle Arbeit konnte für REACH ein Mechanismus entwickelt werden, dessen herausragendes Leistungsmerkmal darin besteht, dass er eine Vielzahl an Proxyservern gleichzeitig involviert.

Als eine besondere Herausforderung sollte sich die Einschränkung erweisen, dass zwar beliebige Anwendungen eingesetzt werden sollten, welche aber nicht modifiziert werden durften. Hierfür wurden Konzepte zum Abfangen von Anwendungsdatenströmen untersucht, wobei in dieser Ausarbeitung die jeweiligen Vor- und Nachteile in Bezug auf praxisrelevante Nutzungsszenarien herausgearbeitet werden. Das wesentliche Ergebnis ist, dass jeder Mechanismus für sich alleine gesehen nicht mit allen Anwendungen gleichermaßen zusammenarbeiten kann und in manchen Szenarien sogar komplett versagt. REACH wurde daher nicht auf einen einzigen Mechanismus beschränkt, sondern bietet hierzu eine Vielzahl an „Diensten“ an. Besonders die Aussicht, dass wenn unterschiedliche Dienste gleichzeitig aktiviert werden können auch die größte Abdeckung von Anwendungen und Einsatzszenarien möglich werden würde, war Motivation genug, um ein solches „Dienstekonzept“ für REACH zu erforschen und schlussendlich umzusetzen. REACH ist dabei die erste Mobilitätsunterstützung, welche auch auf die Ebene der Benutzer Einfluss nimmt. So wird in bestehende Telefongespräche eine Ansage eingespielt, welche verhindern soll, dass die Nutzer ein bestehendes Telefonat während einer Isolationssituation irritiert beenden.

Von Beginn an wurde ein funktionierendes Gesamtsystem angestrebt. Es entstand ein Demonstrator, mit welchem ein „anschauliches“ Problem aus der aktuellen Forschungslandschaft einem breiten Publikum vorgeführt werden kann. Dies wurde beispielsweise durch eine praktische Vorführung auf der CeBIT 2009 unterstrichen.

1.3 Gliederung der Dissertation

Kapitel 2 fährt mit der Definition von Anforderungen, welche im Rahmen dieser Ausarbeitung von einer Mobilitätsunterstützung erfüllt werden sollen, fort. Sie dienen anschließend der Bewertung der bereits verfügbaren Ansätze, wobei gezeigt wird, dass lediglich die Gruppe der Middleware-basierten Mobilitätsunterstützungen die notwendigen Funktionen erbringen *kann*.

In Kapitel 3 wird die Architektur von REACH vorgestellt, welche einen solchen Middleware-basierten Ansatz verkörpert und mit dem Ziel antritt, sämtliche der gestellten Anforderungen zu erfüllen.

Die zentrale Aufgabe einer solchen Middleware besteht in der Kooperation mit den Anwendungen, welche nicht modifiziert werden sollen. In Kapitel 4 werden hierzu diverse Mechanismen diskutiert, mit welchen eine Middleware die Datenströme der Anwendungen auf unterschiedlichen Schichten abfangen und behandeln kann. Es folgt eine Vorstellung von Einsatzszenarien aus der Praxis, anhand derer die Eignung der vorgestellten Mechanismen bewertet werden. Da keiner der Mechanismen für alle vorgestellten Szenarien geeignet war, erschien für REACH eine Kombination mehrerer Mechanismen als Erfolg versprechend. Es entstand das so genannte „Dienstekonzept“, welches auf dynamisch ladbaren und miteinander frei kombinierbarer „Relay Plugins“ basiert. REACH realisiert damit einen „Multi-Layer“-Ansatz.

Das Dienstekonzept von REACH fußt auf einem transportorientierten Kern, welcher das „Transportkonzept“ von REACH implementiert, welches in Kapitel 5 vorgestellt wird. Hierbei werden *logische Verbindungen* angeboten, welche in einer stromorientierten und in einer paketorientierten Form verfügbar sind. Spezifische Mechanismen zur Datenstromsicherung schützen vor den negativen Auswirkungen von Handoverereignissen, und eine anforderbare Isolationserkennung stellt die Basis für eine Mobilitätunterstützung auf Benutzerebene dar.

Die Handoverentscheidung sowie die Verwaltung von Netzzugangsgeräten bilden das „Zugangskonzept“ von REACH, welches den Schwerpunkt von Kapitel 6 bildet. Es wird mit einer Betrachtung von Eigenschaften, welche sich zur Bewertung von Netzzugängen eignen, begonnen. Anschließend werden mögliche Zielstellungen für die Handoverentscheidung vorgestellt. Nach einer Klassifizierung von „Handover-Varianten“ folgt die Vorstellung des prädiktiv arbeitenden Handoverentscheiders von REACH. Schlussendlich wird die Notwendigkeit zur nutzerseitigen Einflussnahme auf die Handoverentscheidung beleuchtet und hierfür eine grafische Bedienschnittstelle vorgestellt.

Eng verbunden mit der Verwaltung von Netzzugängen ist die Betrachtung der Ortsabhängigkeit, welche Problematiken wie „Dreiecks-Routing“ und „geographische Lastver-

teilung“ umfassen. Kapitel 7 stellt hierzu das „Suchkonzept“ von REACH vor, welches mehrere Proxyserver berücksichtigen und bevorzugt *nahe* Proxyserver involvieren kann. Abschließend wird die Erreichbarkeit von Proxyservern diskutiert, wobei auch die Überquerung von „Network Address Translation“-Grenzen (NAT) eine Rolle spielt.

Kapitel 8 widmet sich der Vorstellung des Demonstrators, welcher im Rahmen dieser Ausarbeitung entstanden ist. Er wurde bereits zu mehreren Anlässen öffentlich präsentiert und unterstreicht die Praxistauglichkeit von REACH. Während eine solche Demonstration einen Test des Gesamtsystems bedeutet, werden auch die Ergebnisse von Komponententests bezüglich des transportorientierten Kerns vorgestellt.

Kapitel 9 liefert einen Ausblick auf weiterführende Fragestellungen, welche in der vorliegenden Dissertation nicht weiter untersucht werden konnten.

In Kapitel 10 werden die gewonnenen Erkenntnisse zusammengefasst sowie eine Gegenüberstellung zwischen REACH und den bisherigen Middleware-basierten Mobilitätsunterstützungen vorgenommen.

Anhang A stellt die Protokolle des im Transportkonzept von REACH angesiedelten transportorientierten Kerns vor. In Anhang B werden Details bezüglich der „Device Backends“ geliefert sowie eine Betrachtung der Ansteuerungsmechanismen bezüglich Ethernet, WLAN sowie per Bluetooth angebundene Mobiltelefone durchgeführt. In Anhang C werden die Aufgaben sowie die Ansteuerung des „Routing Backends“ vorgestellt, sowie auf die für GNU/Linux-basierte Systeme („GNU is not Unix“ (GNU)) erforderlichen Mechanismen eingegangen. Anhang D widmet sich dem prädiktiv arbeitenden Handoverentscheider, wobei anhand mehrerer Messreihen innerhalb und außerhalb von Gebäuden der Einfluss verschiedener Parameter untersucht wird. In Anhang E wird anhand ergänzender Diagramme der durch REACH verursachte zusätzliche Übertragungsaufwand (Overhead) diskutiert. Die Ausarbeitung schließt mit einer Vorstellung der Komponenten des Demonstrators in Kapitel F.

2 Mobilitätsunterstützungen

Dieses Kapitel beginnt mit der Vorstellung von Anforderungen, welche die gewünschte Mobilitätsunterstützung erfüllen soll. Anhand dieser Anforderungen werden die bereits verfügbaren Mobilitätsunterstützungen, welche den aktuellen Stand der Forschung darstellen, vorgestellt und bewertet. Es wird gezeigt, dass jeder einzelne Mechanismus Defizite aufweist, jedoch eine „Middleware"-basierte Mobilitätsunterstützung prinzipiell in der Lage sein müsste, die gestellten Anforderungen zu erfüllen.

2.1 Definition von Anforderungen

Die Mobilitätsproblematik wurde bereits von vielen Forschungsgruppen untersucht, sodass schon eine Vielzahl an Lösungsvorschlägen hervorgebracht worden sind. Deren Entwicklung erfolgte immer mit Blick auf die Erfüllung bestimmter Anforderungen und Szenarien, wobei jeder Mechanismus in manchen Szenarien seine Stärken zeigt, aber in anderen Szenarien nicht sinnvoll angewendet werden kann.

Daher folgt eine Liste konkreter Anforderungen, welche die erwünschte Mobilitätsunterstützung erfüllen soll. Diese Anforderungen werden im Folgenden vorgestellt, sodass anschließend die bereits verfügbaren Lösungen vorgestellt und bewertet werden können.

- **Keine Einschränkungen der Anwendungen**: Die gesuchte Mobilitätsunterstützung soll keine Vorgaben bezüglich der zu nutzenden Anwendungen machen. Es sollen möglichst alle Anwendungs*typen* unterstützt werden. Zudem sollen keine bestimmten Programme oder bestimmte Versionen eines Programms vorgeschrieben oder verboten werden. Die Mobilitätsunterstützung sollte idealerweise auch mit vorab nicht berücksichtigten Programmen zusammenspielen können.

- **Keine Modifikationen**: Ein besonderer Wert wird darauf gelegt, dass die gesuchte Mobilitätsunterstützung mit „heutigen Anwendungen" und „heutiger Technik" in „heutigen Netzen" funktioniert. Es ist nicht das Ziel, eine „spezielle Lösung für angepasste Anwendungen" zu schaffen. Daher sollen bezüglich der folgenden Komponenten keine Modifikationen erforderlich werden:

- *Die Anwendungen*: Die involvierten Anwendungen sollen keine Änderungen erfahren, sondern müssen in „nicht modifizierter“ Form eingesetzt werden können. Die Gründe hierfür sind vielfältig: zum einen ist es nicht praktikabel, die Vielzahl an Anwendungen, die ein Nutzer einsetzen möchte, vorab durch Modifikationen auf Mobilität vorzubereiten. Zudem sind viele Programme proprietär, liegen also nicht im Quelltext vor, und können damit nicht modifiziert werden. Da solche Programme ebenfalls eingesetzt werden, darf eine Modifikation der Anwendungen keine Voraussetzung sein.
- *Die Betriebssysteme*: Selbiges gilt für die Betriebssysteme der mobilen Endgeräte und eventuell benötigter Zwischensysteme (Proxyserver). Da hier ebenfalls mit eine Vielfalt an Systemen zu rechnen ist, darf nur auf die bereits vom Betriebssystem angebotenen Mechanismen aufgesetzt werden.
- *Die Server und Protokolle*: Zudem dürfen keine Änderungen der „Server“ im Internet sowie der Protokolle, welche diese „sprechen“, in Betracht kommen. Die gesuchte Mobilitätsunterstützung muss mit den heute verfügbaren Gegebenheiten zurecht kommen, sodass eine Modifikation einer nicht abschätzbaren Anzahl an Systemen keine Option darstellen soll.
- *Die Netzzugangsgeräte und das Kernnetz*: Für den Zugang zum Internet sollen keine neuen Technologien erforderlich werden, sondern stattdessen das zum Einsatz kommen, was ein mobiles Endgerät bereits „ab Werk“ aufweist oder das, was problemlos nachgerüstet werden kann. Hierbei sollen nur die vorgesehenen Schnittstellen und Treiber verwendet werden. Zudem ist auf eine Kompatibilität bezüglich des „Kernnetzes“ zu achten. Neue Protokolle könnten zu Problemen mit Routern oder Firewalls führen. Solche Probleme gilt es von Beginn an zu vermeiden.
- *„Blackbox“-Szenario*: Es kann passieren, dass ein Endgerät vorliegt, auf welchem keinerlei Änderungen vorgenommen werden können oder sollen (eine „Blackbox“). Ein Beispiel könnte ein streamingfähiges Autoradio oder ein Navigationssystem mit Internetzugang darstellen. Möchte man solche Geräte „fit für das mobile Umfeld“ machen, wird es problematisch, denn eine *direkte* Anpassung solcher Geräte wäre nicht praxistauglich. Es besteht aber die Möglichkeit, diese Geräte durch ein „Vorschaltgerät“, welches entsprechende Mechanismen anbietet, zu unterstützen.

- **Flexibilität**: Die gesuchte Mobilitätsunterstützung soll zudem den Anspruch erfüllen, *flexibel* zu sein. Damit ist gemeint, dass eine möglichst vollständige Un-

terstützung aller erdenklichen Anwendungen angestrebt wird. Wie gezeigt werden wird, ist keine der bislang vorgestellten Mobilitätsunterstützungen für alle Szenarien gleichermaßen geeignet. Um eine möglichst große Anzahl an Szenarien unterstützen zu können, stand die Idee im Raum, die positiven Eigenschaften der bereits verfügbaren Mechanismen gezielt aufzugreifen und miteinander kombinieren zu können, sodass die jeweiligen Nachteile durch die Vorteile eines anderen Mechanismus überdeckt werden. Diese Idee wurde umfassend umgesetzt, und die beschriebenen Eigenschaften können am Demonstrator nachvollzogen werden.

- **Sicherungsmechanismen**: Eine zentrale Aufgabe einer Mobilitätsunterstützung umfasst den Schutz von Datenströmen und Sitzungen.
  - *Datenströme*: Transportschichtverbindungen sind statusbehaftet, was zu Problemen führt, wenn sich die IP-Adresse eines Verbindungsendpunkts ändert. Bezüglich des „Transmission Control Protocols“ (TCP) kommt es zu einem Verbindungsabriss, und ein Sender erhält keine Informationen darüber, welche Daten vor dem Abriss noch bis zum Empfänger durchgedrungen und welche verloren gegangen sind.
  - *Minimierung von Ausfallzeiten*: Nach dem Verlust eines Netzzugangs kommt es zu einer kurzfristigen Ausfallzeit, wenn das mobile Endgerät erst einen „neuen“ Internetzugang etablieren muss. Diese Ausfallzeiten können als störend empfunden werden, was besonders bei Sprachkommunikation zu Tage tritt. Durch den Einsatz redundanter Übertragungsverfahren, welche mehrere Netzzugänge gleichzeitig involvieren, kann diesem Problem entgegen gewirkt werden („weichere Handover“).
  - *Kanalbündelungsszenario*: Da es möglich sein soll, mehrere Netzzugänge in einen redundanten Betriebsmodus zu versetzen, spräche auch nichts dagegen, stattdessen jeweils *neue* Daten über unterschiedliche Wege zu versenden. Man erhält hierbei ein „Kanalbündelungsszenario“, wobei die dabei möglich werdende Datenrate höher sein kann als die eines einzelnen Netzzugangs.
  - *Isolationsunterstützung*: Ein mobiles Endgerät kann zudem für eine lange Zeitspanne vom Internet isoliert werden, wenn der Benutzer beispielsweise mit seinem mobilen Endgerät einen Tunnel durchquert. Während solcher Isolationssituationen treten weiterführende Probleme auf. So reißen TCP-Verbindungen ab, wenn über einen Zeitraum von ca. 9 Minuten keine Bestätigungen empfangen werden konnten („Retransmission Timeout“). Unabhängig davon kommt es auf Anwendungsebene zu einem Sitzungsabriss, wenn sich ein Inter-

net-Telefon nicht mehr zyklisch an seiner Vermittlungsstelle anmelden kann. Auf *Nutzerebene* kommt es ebenfalls zu Beeinträchtigungen, da ein Telefonat wahrscheinlich getrennt werden wird, wenn sich die Teilnehmer nicht mehr gegenseitig hören können. Für derartige Probleme müssen Lösungen angeboten werden.

- **Integrierte Verwaltungsmechanismen**: Auf einem jeden mobilen Endgerät muss es eine *zentrale Instanz* geben, welche sich um mobilitätsspezifische Verwaltungsaufgaben kümmert. Damit ist gemeint, dass diese Instanz idealerweise wie ein Bestandteil des Betriebssystems agieren sollte. Müsste sie stattdessen erst manuell durch einen Benutzer gestartet werden, wäre eine kosistente Verwaltung nicht möglich. Auf einem Mehrbenutzersystem könnte beispielsweise nicht sichergestellt werden, dass immer genau eine Instanz verfügbar ist, welche für die Verwaltung der Netzzugangsgeräte des mobilen Endgeräts verantwortlich wäre.
  - *Netzzugangsgeräte*: Es muss eine zentrale Komponente geben, welche die „Hoheit“ über alle verfügbaren Netzzugangsgeräte übernimmt. Damit ist gemeint, dass jedes Netzzugangsgerät nur von einer einzigen Steuerungsinstanz „bedient“ werden darf, damit es zu keinen Problemen durch konkurrierende Zugriffe kommen kann. Diese Steuerungsinstanz darf nur einmal pro mobilem Endgerät gestartet werden können, sonst käme es zu Konflikten. Zudem sollen jegliche denkbaren Netzzugangstechnologien unterstützt werden können.
  - *Handoverentscheider*: Weiterhin wird ein Handoverentscheider benötigt, welcher auf jedem mobilen Endgerät eine zentrale Komponente darstellen muss. Dieser soll sowohl „horizontale“ als auch „vertikale Handover“ unterstützen und zwischen „harten“, „weichen“ und „weicheren Handovern“ sowie Kanalbündelungsszenarien unterscheiden können. Zudem soll neben einem „reaktiven“ auch ein „prädiktives“ Verhaltensmuster implementiert werden können, was „proaktive“ Handoverentscheidungen ermöglicht. In Abschnitt 6.1.2 werden diese Begriffe umfassend erläutert.
  - *Systemdienst*: Heutige Betriebssysteme stellen „Mehrbenutzersysteme“ dar, sodass ein mobiles Endgerät nicht auf einen einzigen Benutzer beschränkt zu sein braucht. Daraus folgt allerdings, dass es nicht im Verantwortungsbereich der Anwender liegen darf, die für die Mobilitätsunterstützung erforderlichen Komponenten starten zu müssen. Stattdessen müssen diese Komponenten den Anspruch eines „Systemdienstes“ erfüllen, also losgelöst von den Nutzern bereits beim „Hochfahren“ des Systems aktiv werden. Diese „Kernkomponen-

ten“ dürfen zudem keine grafische Benutzerschnittstelle voraussetzen, da sie auch auf Systemen eingesetzt werden sollen, die über keine grafische Oberfläche verfügen.

  - *Bedienschnittstelle*: Obschon der „Kern“ der Mobilitätsunterstützung keine grafische Bedienschnittstelle aufweisen darf, wird eine solche benötigt, um beispielsweise den Zustand des Systems in Erfahrung zu bringen oder um auf den Handoverentscheider einwirken zu können. Eine solche Bedienschnittstelle muss „optional“ sein und soll bei Bedarf durch einen Nutzer gestartet werden können. Es muss allerdings von vornherein berücksichtigt werden, dass mehrere Nutzer gleichzeitig eine Bedienschnittstelle starten können und dabei ein konkurrierender Zugriff erfolgen wird.

- **Kontextabhängiges Verhalten**: Eine Lösung zur Mobilitätsunterstützung sollte zudem „Zusatzwissen“ mit berücksichtigen und somit beispielsweise den aktuellen Aufenthaltsort oder die Charakteristika der gestarteten Anwendungen mit einbeziehen können.

  - *Anwendungsabhängiges Verhalten*: Mit Zusatzwissen über die gerade aktiven Anwendungen, also dass beispielsweise gerade ein Telefonat stattfindet, kann die Mobilitätsunterstützung gegebenenfalls *bessere* Ergebnisse erzielen. Dies kann die Auswahl der Netzzugangstechnologien in Blick auf bestimmte Dienstgüteparameter betreffen, oder eine zeitweise Isolationserkennung aktivieren, welche auf höherfrequent versendeten Paketen basiert (siehe Abschnitt 5.4). Da allerdings keine Anwendungen modifiziert werden sollen, muss die Mobilitätsunterstützung solche Gegebenheiten selbstständig in Erfahrung bringen können.

  - *Minimierung des „Dreiecks-Routings“*: Viele Mobilitätsunterstützungen involvieren, wie noch beschrieben wird, einen „Ankerpunkt“ im Internet, über den sämtliche Daten von und zu den Servern übertragen werden. Durch Mobilität kann es zu einem als „Dreiecks-Routing“ bezeichneten Effekt kommen, wobei sich der Weg der Daten durch das Netz verlängert (siehe Abschnitt 7.1.1) und sich die Übertragungseigenschaften verschlechtern. Es sollte versucht werden, dass diese „Ankerpunkte“ „migriert“ werden, oder falls dies nicht möglich ist, wenigstens mehrere „Ankerpunkte“ gleichzeitig involviert werden.

  - *Ortsabhängiges Verhalten*: Abhängig vom aktuellen Aufenthaltsort soll möglichst der nächstgelegene „Ankerpunkt“ verwendet werden. Dadurch ließe sich eine „geographische Lastverteilung“ realisieren (siehe Abschnitt 7.1.2). Hier-

für wären Suchemechanismen erforderlich, welche „Ankerpunkte“ in der Umgebung ausfindig machen und diese auf ihre Eignung hin bewerten können.

Anhand dieser Anforderungen soll jede der bereits verfügbaren Mobilitätsunterstützungen bewertet werden. Es wird sich zeigen, dass keine der bislang verfügbaren Mechanismen alle Anforderungen erfüllt.

2.2 Aktueller Stand der Forschung

Es sind bereits eine Vielzahl an Mobilitätsunterstützungen vorgeschlagen worden. Viele betrachten lediglich ausgewählte Teilprobleme und lassen sich einer Schicht des Internet-Schichtenmodells zuordnen. Daneben gibt es noch die Gruppe der „Middleware“-basierten Architekturen, welche sich nicht einer bestimmten Schicht zuordnen lassen. Es folgt ein gruppierter Überblick über die verfügbaren Konzepte.

2.2.1 Konzepte auf der Netzzugangsschicht

Es gibt Mobilitätsunterstützungen, welche mit Blick auf mobilitätsspezifische Probleme der Netzzugangsschicht entstanden sind. Diesbezügliche Verbesserungsvorschläge behandeln oftmals die bei einem „horizontalen Handover“ auftretenden Effekte. So wird versucht, Verzögerungszeiten und deren Schwankungen („Jitter“) zu senken sowie die Bitfehlerrate zu vermindern und damit Paketverlusten entgegen zu wirken. Diese Mechanismen greifen allerdings nur innerhalb einer Technologie und sind nur solange für die höheren Schichten *transparent*, wie kein Wechsel eines IP-Subnetzes stattfindet.

Bei Betrachtung der Anforderungen fällt ein solcher Ansatz aus, da mit nicht modifizierten Komponenten gearbeitet werden soll. Weder die Netzzugangsgeräte und deren Treiber noch die zugehörigen Basisstationen dürfen manipuliert werden.

Mit IEEE 802.21 [IEEE08] steht ein Mechanismus bereit, welcher als „Multi-Interface Mobility“ bezeichnet wird und die Verwaltung von Netzzugangsgeräten vereinfachen soll. So werden bereits Technologien der Standards IEEE 802.3 (Ethernet), IEEE 802.11 („Wireless LAN“ (WLAN)), IEEE 802.16 („Worldwide Interoperability for Microwave Access“ (WiMAX)) sowie zelluläre Netze gemäß des „3rd Generation Partnership Projects“ (3GPP) und 3GPP2 unterstützt. Mit Hilfe so genannter „Media Independent Handover Functions“ (MIHF) soll eine technologieunabhängige Bewertung und Verwaltung der Netzzugänge möglich werden, aber es werden keine Vorgaben bezüglich der höheren Schichten gemacht. Es handelt sich ausschließlich um ein Werkzeug, welches die höheren Schichten unterstützen soll. IEEE 802.21 muss von den Betriebssystemen der mobilen Endgeräte unterstützt werden, was heutzutage noch nicht der Fall ist.

2.2.2 Konzepte auf der Vermittlungsschicht

Eine Vielzahl an Mobilitätsunterstützungen setzen auf der Vermittlungsschicht an und basieren auf Erweiterungen des „Internet Protokolls“ (IP). Standardisierte Vertreter sind „Mobile IP“ („Request for Comments“ 3344 (RFC 3344), [Perk02]) und dessen auf IPv6 basierende Variante „Mobile IPv6“ (MIPv6, RFC 3775, [JoPA04]). Sie basieren auf dem Einsatz von „Agenten“ als „Ankerpunkte“ für den Datenverkehr und dem Konzept des „Tunnelns“. MIPv6 bietet zudem die Möglichkeit der „Route Optimization“ durch „Binding Updates“, wodurch „Dreiecks-Routing“ unterbunden werden kann.

Eine Vielzahl weiterer Varianten schlagen Verbesserungen von „Mobile IP“ und MIPv6 vor. Hierarchisch arbeitende Ansätze wie „Hierarchical Mobile IP“ (HMIP [GuJP02]) und „Hierarchical Mobile IPv6“ (HMIPv6, RFC 4380 [Huit06]) versprechen einen schnellen Ablauf von Signalisierungsoperationen, und bei „ProxyMIPv6“ (RFC 5213 [GLDC+08]) kommen Proxyserver zur beschleunigten „Übergabe“ von Datenströmen zum Einsatz. Für eine umfassende Übersicht sei ein Blick in [Diab10] empfohlen.

Ein Sonderfall, den der Autor ebenfalls der Vermittlungsschicht zuordnet, ist das „Host Identity Protocol“ (HIP, RFC 5201 [MNJH08]) welcher sich als „Shim layer“-basierter Ansatz bezeichnet. Diese Zwischenschicht liegt oberhalb von IP, aber noch unterhalb der Transportschicht. Für die Transportschicht arbeitet HIP *transparent*.

Trotz der Vielzahl an Mechanismen lassen sich Gemeinsamkeiten erkennen, welche insbesondere bei Blick auf die Liste an Anforderungen zu Tage treten. So wird keine der genannten Methoden dem Anspruch gerecht, dass keine Modifikationen der Betriebssysteme oder der Netzzugänge durchgeführt werden sollen. „Mobile IP“ benötigt Agenten auf den Gateways, was heutzutage in der Praxis bei Verwendung von GPRS zu Problemen führen wird. Die Gateways, auf denen die Agenten installiert werden müssten, gehören in diesem Fall den Mobilfunkprovidern. Ob oder wann die Provider den Privatkunden besagte Agenten zur Verfügung stellen, ist ungewiss. Zwar benötigen die auf IPv6 basierenden Lösungen keine derartigen Agenten mehr, jedoch stehen diese Mechanismen für einen Praxiseinsatz noch nicht zur Verfügung, da das heutige Internet immer noch auf IPv4 basiert.

Keine der genannten Lösungen kann Netzzugangsgeräte verwalten oder benennt einen Handoverentscheider. Diese Problematiken werden ausgeklammert und müssen anderweitig gelöst werden. Zudem erlaubt „Mobile IP“ beispielsweise keine „weicheren Handover“ oder Kanalbündelungsszenarien, da es nur einen „aktiven“ Tunnel zum „Heimatagenten“ geben kann. Gäbe es mehrere Tunnel, müsste der „Heimatagent“ die in Richtung des mobilen Endgeräts zu sendenden Daten auf mehrere Tunnel aufteilen können. Eine derartige Funktionalität ist jedoch nicht vorgesehen; es fehlen beispielsweise Signalisierungsmecha-

nismen, über welche den Agenten mitteilt werden kann, ob eine Kanalbündelung oder eine Redundanzerhöhung durchgeführt werden soll.

Probleme, welche bei längerfristigen Isolationssituationen auftreten können, werden nicht betrachtet. Hierfür wäre eine zusätzliche Betrachtung der höheren Schichten erforderlich, sodass keiner der hier aufgeführten Ansätze die Anforderungen erfüllt.

2.2.3 Konzepte auf der Transportschicht

Diverse Mobilitätsunterstützungen auf Ebene der Transportschicht trennen die Übertragungsstrecke in mehrere Abschnitte auf, wobei sich die Mobilität nur auf den Abschnitt nahe des mobilen Endgeräts beschränken soll. Die Mechanismen „Indirect TCP“ (I-TCP, [BaBa95]) und „Mobile TCP“ (M-TCP, [BrSi97]) sehen hierfür Zwischensysteme vor, welche die mobile Teilstrecke gegenüber dem „festen“ Internet „abschotten“ können. Die vielfältigen Lösungen lassen sich dahingehend unterscheiden, dass beispielsweise bei M-TCP die Ende-zu-Ende-Semantik erhalten bleibt und dass lang anhaltende Isolationssituationen hier zu keinen Verbindungsabrissen mehr führen.

Andere Vorschläge betreffen TCP-interne Mechanismen, welche mit Blick auf ein verbessertes Übertragungsverhalten im mobilen Umfeld abzielen. Zudem sind auch für das verbindungslos arbeitende „User Datagram Protocol“ (UDP) Mobilitätsunterstützungen verfügbar, wie „Mobile UDP“ (M-UDP [BrSi96]).

Erwähnenswert sei noch das „TCP Migrate“ [SnBK01], welches die redundante Versendung von Daten über mehrere Wege ermöglicht, indem mehrere TCP-Übertragungen an einem gemeinsamen „Ankerpunkt“ zusammengeführt werden. „Multipath TCP“ (MPTCP, [HSHS+06]) ist hierzu ebenfalls in der Lage, zeigt sein Potential allerdings nur, wenn auch die Server mit einer modifizierten Instanz ausgestattet worden sind.

Weiterhin wurde mit dem „Stream Control Transmission Protocol“ (SCTP, RFC 4960 [Stew07]) ein drittes Transportschichtprotokoll neben TCP und UDP vorgeschlagen. SCTP unterstützt „Multi-Homing“ und ist damit in der Lage, ein redundantes Übertragungsschema anzubieten. Für SCTP wurden wiederum Verbesserungen hinsichtlich Mobilität vorgeschlagen, wie das „Mobile SCTP“ (mSCTP [RiTu05]), welches „weichere Handover“ ermöglicht. Weder SCTP noch mSCTP unterstützen Kanalbündelungen.

Alle vorgeschlagenen Mechanismen basieren darauf, dass die heutzutage verwendeten Transportschichtprotokolle TCP und UDP in ihrer jetzigen Form nicht länger eingesetzt werden. Damit drohen Inkompatibilitäten, da heutige „Network Address Translation“-Gateways“ (NAT-Gateways) häufig nur mit TCP und UDP umgehen können. Zudem werden Modifikationen im Betriebssystem der mobilen Endgeräte oder in Zwischensystemen benötigt, was mit den gestellten Anforderungen unvereinbar ist.

2.2.4 Konzepte auf der Anwendungsschicht

Der Anwendungsschicht zuordbare Mobilitätsunterstützungen weisen den Vorteil auf, dass sie keine Modifikationen der darunter liegenden Schichten benötigen. So besteht insbesondere keine Notwendigkeit an speziellen Transportschichtprotokollen oder an einer abgewandelten Form von IP.

Die vorgeschlagenen Ansätze behandeln in erster Linie alle Formen der Sprach- und Videokommunikation, insbesondere die auf dem „Session Initiation Protocol“ (SIP, RFC 3261 [RSCJ+02]) basierenden Varianten. So wird in [ScWe00] ein auf SIP basierender Mechanismus vorgestellt, welcher als „SIP-based Mobility Management“ bezeichnet wird. Es werden mehrere Arten von Mobilität genannt, welche ermöglicht werden: Persönliche Mobilität (eine Adresse, aber viele Endgeräte), Dienstmobilität (erlaubt einen Adressbuchzugriff von unterwegs), Sitzungsmobilität (Übergabe eines Telefonats an ein anderes Telefon), Vor-Sitzungs-Mobilität (Weiterleitung vor Rufannahme) und Während-Sitzungs-Mobilität (Nutzer bewegt sich während er telefoniert).

Handover und damit Datenstromübergaben werden mit Hilfe von SIP-re-`INVITE`-Nachrichten signalisiert, wodurch auch das Problem des „Dreiecks-Routings“ behandelt wird. In [BaAD06] wird ein weiterführender Mechanismus vorgestellt, welcher zusätzliche Proxyserver einsetzt und auch „weichere Handover“ ermöglicht.

Alle vorgestellten Mechanismen basieren auf SIP und eignen sich ausschließlich für Anwendungen aus dem Bereich der „sitzungsorientierten Dienste“, zu denen auch die Sprach- und Videokommunikation gehört. Andere Anwendungen werden nicht unterstützt, und die gestellten Anforderungen damit nicht erfüllt.

2.2.5 Middleware-basierte Konzepte

Durchgängig alle soeben vorgestellten Mobilitätsunterstützungen erweisen sich prinzipiell als ungünstig: da keine Modifikationen der Hardware, der Treiber, der Komponenten des Internet oder der Betriebssysteme in Betracht kommen sollen, erweisen sich alle Mobilitätsunterstützungen bezüglich der unteren Schichten bis einschließlich der Transportschicht als ungeeignet. Eine Modifikation der Anwendungen stellt allerdings ebenfalls keine praktikable Option dar, und eine Einschränkung auf SIP-basierte Anwendungen kommt nicht in Frage.

Eine Lösung hierfür stellen die „Middleware“-basierten Mobilitätsunterstützungen dar, welche als „Zwischenschicht“ unterhalb der Anwendungen aber noch oberhalb der Transportschicht angesiedelt sind. Letzteres impliziert, dass keine Änderungen an den unteren Schichten einschließlich der Transportschicht erforderlich sein *sollten*. Auf Modifikationen der Anwendungen kann ebenfalls verzichtet werden, da die Middleware „unterhalb“

angesiedelt ist. Die Herausforderung besteht jedoch darin, die Datenströme der Anwendungen abzufangen und durch die Middleware zu leiten, was aber auf jeder der darunter liegenden Schicht erfolgen kann. Je nach Schicht unterscheiden sich die dazu geeigneten Mechanismen, womit ein Unterscheidungskriterium zur Klassifizierung der bereits veröffentlichten Middleware-basierten Konzepte zur Verfügung steht.

Mit „Transport Layer Mobility“ (MSOCKS [MaBh98]) wurde ein Mechanismus vorgestellt, welcher auf dem Protokoll SOCKSv5 (RFC 1928 [LGLK+96]) basiert. Hierbei kommen Proxyserver zum Einsatz, welche als feste „Ankerpunkte“ fungieren. Das Abfangen der Anwendungsdaten erfolgt mittels einer modifizierten SOCKSv5-Bibliothek. Obwohl sich der Mechanismus selbst der Transportschicht zuordnet, ist SOCKSv5 ein Protokoll der Anwendungsschicht. Allerdings kommt auf den Proxyservern eine modifizierte TCP-Instanz zum Einsatz, welche ein so genanntes „TCP-Splicing“ betreibt. Bei MSOCKS bleibt im Gegensatz zu SOCKSv5 die Ende-zu-Ende-Semantik erhalten, was allerdings während lang anhaltender Isolationssituationen zu Problemen führt.

Die „Universal Seamless Handoff Architecture“ (USHA [ChSG05, CSYG06]) implementiert einen auf „Virtual Private Networks“ (VPN) basierenden Ansatz. Auf dem mobilen Endgeräten kommt jeweils ein VPN-Client zum Einsatz, während auf einem zentralen Proxyserver ein VPN-Server bereit gestellt wird. So lässt sich auf den mobilen Endgeräten sämtlicher Datenverkehr wahlweise auf Ebene der Netzzugangsschicht oder der Vermittlungsschicht abfangen. Die USHA-eigenen Komponenten betrachten lediglich den „getunnelten“ Datenstrom zwischen VPN-Client und -Server, und leiten diesen gemäß der gefällten Handoverentscheidung über den präferierten Weg. Während längerfristiger Isolationssituationen kommt es jedoch zu einem Abreißen der über den Tunnel geführten Transportschichtverbindungen.

„Interactive Mobile Application Session Handoff“ (iMASH [BBCG+03]) beschränkt sich auf den Sonderfall des „Mobile Continuous Computing“, wobei „Sitzungen einer Anwendung“ zwischen heterogenen Endsystemen ausgetauscht werden sollen. Dazu werden hochgradig auf iMASH zugeschnittene Anwendungen benötigt, da ein so genannter „Savepoint-based handoff“ durchgeführt wird, und die Anwendungen vorher in einen geordneten Zustand überführt werden müssen.

Bei PLASTIC [RFIG07] wird ein „Application Programming Interface“ (API) angeboten, mit dessen Hilfe spezielle Anwendungen erstellt werden können, welche die Middleware benutzen. Da die Vielzahl der bestehenden Anwendungen nicht auf dieser API basieren, können sie nicht mit PLASTIC interagieren.

Die „Drive-thru“-Architektur [OtKu04] ist für Fahrzeuge entwickelt worden. Es wird ein erweiterbares Dienstekonzept vorgestellt, welches Proxyserver involviert, sich allerdings auf „transaktionsorientierte Dienste“ beschränkt. Unterstützt werden die E-Mail-

Protokolle „Simple Mail Transfer Protocol“ (SMTP) und „Post Office Protocol Version 3“ (POP3) sowie das Herunterladen von Webseiten mit Hilfe des „Hypertext Transfer Protocols“ (HTTP). Im Isolationsfall werden Anfragen auf dem mobilen Endgerät in eine Warteschlange eingefügt, um dann später nach einer Neuverbindung abgearbeitet zu werden. Dialogbasierte Kommunikation wie Telefongespräche werden nicht unterstützt.

Mit der „Mobile agent-based Ubiquitous multimedia Middleware“ (MUM [BeCF07]) steht eine Middleware bereit, welche sich auf den Einsatz angepasster Anwendungen beschränkt. MUM ist für die Übertragung von Mediendatenströmen entwickelt worden. Die Anwendungen stellen MUM Informationen über „die Natur“ der zu übertragenden Daten zur Verfügung („Kontextwissen“). MUM ist damit in der Lage, Dienstgüteparameter bei der Handoverentscheidung zu berücksichtigen. Durch die Einschränkung auf Mediendaten sowie der Anforderung, dass die Anwendungen angepasst werden müssen, kann MUM jedoch nicht die Vielzahl an bestehenden Anwendungen unterstützen.

2.2.6 Fazit

Eine Übersicht über die vorgestellten Mobilitätsunterstützungen ist in Tabelle 2.1 aufgeführt. Da lediglich die Middleware-basierten Konzepte in der Lage sind, die geforderten Anforderungen zu erfüllen, werden die restlichen Verfahren nur vereinfacht dargestellt. So wird nicht zwischen „Mobile IP“ und dessen Varianten unterschieden. Wenn dabei ein Kriterium bereits von nur einem der eingeflossenen Mechanismen erfüllt wird, erfolgte eine positive Bewertung des „Sammeleintrags“.

Viele der Middleware-basierten Konzepte erwiesen sich bereits beim „ersten Blick“ als ungeeignet zur Erfüllung der Anforderungen. So basieren beispielsweise einige Ansätze auf der Notwendigkeit von hochgradig angepassten Anwendungen. Aus Sicht des Autors erschien es nach Herausarbeitung eines derartigen Nachteils nicht mehr als lohnenswert, in der Beschreibung detailliert auf die jeweilige Architektur einzugehen.

Die Eignungsbewertung erfolgte anhand der in der jeweiligen Kurzbeschreibung referenzierten Quellen. War zu einem bestimmten Bewertungskriterium keine Information verfügbar, wurde abgeschätzt, ob die vorgestellte Architektur das Kriterium theoretisch erfüllen *könnte*. Falls ja, wurde es mit (s) für „möglich, aber spekulativ“ bewertet, falls nicht, mit (−) für „nein“.

Es lässt sich erkennen, dass keine der bereits verfügbaren Ansätze alle Anforderungen erfüllt. Betrachtet man jedoch die Gruppe der Middleware-basierten Ansätze „als Ganzes“, wird deutlich, dass jedes einzelne Bewertungskriterium bereits von irgendeiner Lösung erfüllt wird, oder wenigstens spekulativ erfüllt werden könnte.

Ansatz: / Bewertungskriterium:	Keine Einschränkungen und Modifikationen: der Anwendungen, Betriebssysteme und Server	der Netzzugangsgeräte, -punkte und des Kernnetzes	„Blackbox“-Szenario („Netzmobilität“)	Flexibles und kombinierbares Dienstangebot	Sicherungsmechanismen: Minimierung von Ausfallzeiten	Kanalbündelungsszenario	Isolationsunterstützung	Betrachtung der Nutzerebene	Integrierte Verwaltungsmechanismen: Netzzugangsgeräte	Handoverentscheider	Bedienschnittstelle verfügbar	Kontextabhängiges Verhalten, Ortsabhängigkeit
Netzzugangsschicht	−	−	−	−	+	+	−	−	+	+	−	−
Vermittlungsschicht:												
Mobile IP etc.	+	+	s	−	+	−	−	−	−	−	−	+
HIP	−	+	−	−	−	−	−	−	−	−	−	−
Transportschicht:												
mod. TCP, UDP	−	−	−	−	+	s	+	−	−	−	−	s
SCTP etc.	+	−	−	−	+	s	+	−	−	−	−	+
Anwendungsschicht:												
SIP etc.	−	+	−	−	+	−	−	−	−	−	+	+
Middleware-basiert:												
MSOCKS	−	+	−	−	−	−	−	−	s	s	s	−
USHA	+	+	s	−	+	−	−	−	+	−	s	−
iMASH	−	+	−	−	s	s	s	−	s	+	s	s
PLASTIC	−	+	−	−	+	−	s	−	+	+	s	+
Drive-thru	+	+	+	+	−	−	+	−	+	+	s	+
MUM	−	+	−	−	+	−	+	−	+	+	s	+

(+) unterstützt, (−) nicht unterstützt
(s) spekulativ: möglich, ggf. sinnvoll, keine Erwähnung bekannt
(mod.) modifiziert, (etc.) Weiterentwicklungen mit betrachtet

Tabelle 2.1: Keine der bereits vorgeschlagenen Mobilitätsunterstützungen erfüllt alle der gestellten Anforderungen.

2.3 Kapitelzusammenfassung

Um die Vielzahl der bereits publizierten Mobilitätsunterstützungen bewerten zu können, wurde in diesem Kapitel mit der Definition von praxisrelevanten Anforderungen begonnen. Ziel war es, eine Mobilitätsunterstützung zu finden, welche alle Anforderungen gleichermaßen erfüllt.

Hierzu wurden die Mobilitätsunterstützungen gemäß der Schicht gruppiert, zu welcher sie sich am ehesten zuordnen lassen. Innerhalb dieser Gruppe erfolgte jeweils eine knappe Vorstellung der zu Grunde liegenden Idee. Dies erfolgte stets mit Blick auf die gestellten Anforderungen, wobei sich zeigte, dass keine der verfügbaren Mobilitätsunterstützungen alle Anforderungen gleichermaßen erfüllen konnte.

Lediglich die Gruppe der Middleware-basierten Konzepte erschien vielversprechend, wobei auch hier noch keine umfassende Lösung verfügbar war. Dies war die Motivation, um eine eigene Middleware-basierte Mobilitätsunterstützung zu entwickeln, welche mit allen vorgestellten Anforderungen zurecht kommt. Sie wird im nächsten Kapitel vorgestellt.

3 Die „Roaming-Enabled Architecture“

Bislang wurde gezeigt, dass lediglich eine Mobilitätsunterstützung aus der Gruppe der Middleware-basierten Ansätze die gestellten Anforderungen erfüllen kann. Jedoch ist keine der verfügbaren Architekturen in der Lage, die gestellten Anforderungen gleichermaßen zu erfüllen. Aus diesem Grund wurde die „Roaming-Enabled Architecture“ (REACH) entwickelt, welche mit dem Ziel antritt, alle in Kapitel 2 vorgestellten Anforderungen zu erfüllen.

3.1 Die Architektur von REACH

Bei REACH kommen neben den mobilen Endgeräten ein oder mehrere Proxyserver zum Einsatz, welche als REACH-Proxyserver bezeichnet werden (siehe Abbildung 3.1). Auf den mobilen Endgeräten läuft der REACH-Client, während auf den Proxyservern der so genannte REACH-Server gestartet wird. Beides sind eigenständige Programme, welche miteinander kommunizieren und die Middleware darstellen. Weder die Anwendungen auf den mobilen Endgeräten noch die Serverdienste im Internet müssen angepasst werden, und auf keinem der beteiligten Geräte ist ein Eingriff in das Betriebssystem notwendig [Ever08].

Die Datenübertragung zwischen REACH-Client und REACH-Server basiert auf nicht modifiziertem TCP und UDP. Es kommt daher zu keinen Inkompatibilitäten mit bestehenden Netzwerkkomponenten, und auch die an NAT-Grenzen auftretenden Probleme werden behandelt.

Die Datenströme der Anwendungen können durch eine Vielzahl an Mechanismen abgefangen werden, welche durch ein modulares „Dienstekonzept“ am REACH-Client und am REACH-Server in Form so genannter „Relay Plugins“ geladen werden können.

Aus Sicht des nicht modifizierten Betriebssystems tritt REACH als ein vom Nutzer gestartetes Programm der Anwendungsschicht in Erscheinung (siehe Abbildung 3.2). Aus Sicht der eigentlichen Anwendungen kann jedoch keine eindeutige Klassifizierung vorgenommen werden. REACH beginnt aus Sicht einer Anwendung auf derjenigen Schicht, auf welcher der Mechanismus zum Abfangen der Datenströme greift, was bei REACH

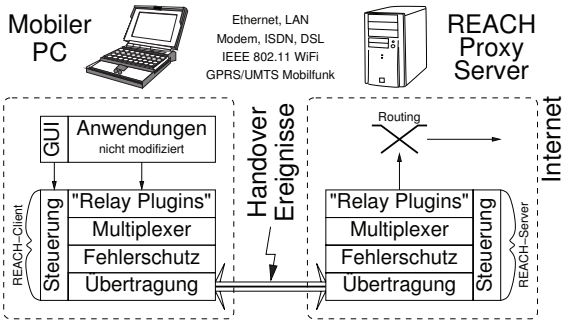

Abbildung 3.1: Die Architektur der Middleware REACH besteht aus den mobilen Endgeräten mit installiertem REACH-Client (links) und den REACH-Proxyservern mit installiertem REACH-Server (rechts).

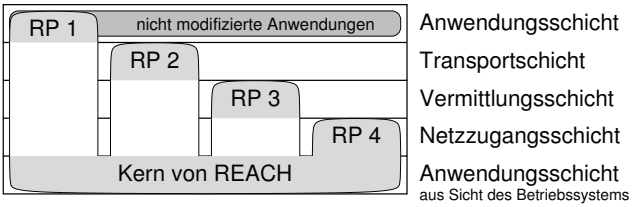

Abbildung 3.2: REACH, welches hier samt vier beispielhafter „Relay Plugins“ (RP) hellgrau dargestellt wird, ist für des Betriebssystems ein Programm der Anwendungsschicht. Aus Sicht einer nicht modifizierten Anwendung (hier dunkelgrau dargestellt) kann ein „Relay Plugin“ von REACH auf jeder Schicht, auch auf der Anwendungsschicht selbst, agieren.

durch „Relay Plugins“ realisiert wird. „Relay Plugins“ können auf jeder Schicht agieren, was im nächsten Kapitel genauer beleuchtet wird.

REACH besitzt einen integrierten Handoverentscheider und übernimmt die vollständige Kontrolle über die Netzzugangsgeräte des mobilen Endgeräts. Neben drahtlos arbeitende Netzzugangstechnologien werden auch drahtgebundene Technologien unterstützt. Für die Erstgenannten kann durch Beobachtung von Signalstärke- und Linkgütewerten sogar eine prädiktive Handoverentscheidung durchgeführt werden. Außerdem besteht die Möglichkeit einer redundanten Datenübertragung (notwendig für „weichere Handover“), und es werden Kanalbündelungsszenarien unterstützt. Zur Einflussnahme auf den Handoverentscheider sowie zur Konfiguration des Dienstangebotes stehen optionale grafische Bedienschnittstellen („Graphical User Interface“ (GUI)) bereit.

3.2 Netzmobilität dank REACH-Box

Ein besonderer Anwendungsfall von REACH ermöglicht so genannte Netzmobilität. Hierbei soll sich nicht bloß ein einzelnes mobiles Endgerät, sondern ein lokales Netzwerk bewegen dürfen, was beispielsweise für zukünftige Fahrzeugnetze von Interesse sein wird.

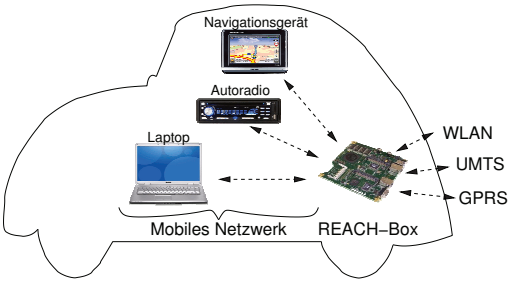

Abbildung 3.3: Der REACH-Client kann auf einem „routerähnlichen“ Gerät installiert werden und seine Mobilitätsunterstützung einem lokalen Netzwerk zur Verfügung stellen. Eine solche REACH-Box soll keinerlei Modifikationen oder Parametrierungen der Endgeräte des lokalen Netzwerks erforderlich machen.

In Abbildung 3.3 ist ein Fahrzeug dargestellt, welches mit einer „On-Board-Unit“ (OBU) ausgestattet ist, welche für alle anderen Geräte des Fahrzeugnetzes als ein Gateway in Richtung Internet angesehen werden soll. Bei dieser „REACH-Box“ handelt es sich im vorliegenden Fall um einen kleinen Computer mit GNU/Linux als Betriebssystem. Dieser Rechner kann über mehrere Netzzugangsgeräte verfügen, um sich beispielsweise mittels WLAN, UMTS oder GPRS/GSM mit dem Internet verbinden zu können. Auf der REACH-Box wird der REACH-Client installiert, dessen Dienstangebot für die Systeme des angeschlossenen „Local Area Networks“ (LAN) eine Mobilitätsunterstützung darstellen, für die sie keinerlei Änderungen benötigen.

3.3 Die vier Standbeine von REACH

Es bietet sich an, die Menge der gestellten Anforderungen in vier Gruppen zu gliedern, woraus sich die „vier Standbeine“ von REACH ergeben (dargestellt in Abbildung 3.4).

Das „Dienstekonzept“ befasst sich mit dem Abfangen von Datenströmen der nicht modifizierten Anwendungen, wobei REACH eine Vielzahl an Varianten anbietet, die miteinander kombiniert werden können. Die abgefangenen Daten müssen mit Blick auf

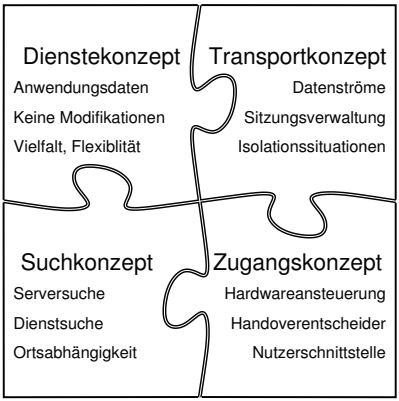

Abbildung 3.4: REACH basiert auf vier Standbeinen, welche im Folgenden als „Konzepte“ bezeichnet werden. Die nächsten Kapitel widmet sich jeweils einem dieser Konzepte, wobei die Reihenfolge durch die Verzahnung der hier dargestellten Puzzleteile unterstrichen wird.

mobilitätsspezifische Probleme gesichert werden, was neben einer robusten Sitzungsverwaltung Bestandteil des „Transportkonzepts“ ist. Damit die gesicherten Datenstöme übertragen werden können, müssen die Netzzugangsgeräte des mobilen Endgeräts verwaltet werden. Die dazu notwendigen Ansteuerungsmechanismen, die Handoverentscheidung sowie eine optionale Bedienschnittstelle für die Benutzer sind Teil des „Zugangskonzepts“. Da ein mobiles Endgerät mehrere REACH-Proxyserver involvieren kann, welche zudem unterschiedliche Dienste anbieten können, sind Suchmechanismen erforderlich. Das „Suchkonzept“ berücksichtigt hierfür die Position des mobilen Endgeräts, wodurch „nahe“ REACH-Proxyserver bevorzugt ausgewählt werden. Die folgenden vier Kapitel sind gemäß dieser vier Konzepte gegliedert, beginnend mit dem „Dienstekonzept“.

4 Das Dienstekonzept

Ein wichtiger Aspekt einer Middleware-basierten Architektur zur Mobilitätsunterstützung ist der verwendete Mechanismus zum Abfangen und Sichern der Anwendungsdatenströme. Im Folgenden werden mehrere Möglichkeiten beleuchtet, welche für nicht modifizierte Anwendungen verwendet werden können. Anschließend werden praxisrelevante Szenarien vorgestellt, anhand welcher die Eignung der einzelnen Mechanismen diskutiert wird. Es wird gezeigt, dass zwar zu jedem Szenario ein idealer Mechanismus existiert, jedoch kein Mechanismus für sich alleine alle Szenarien unterstützen kann. Für REACH wurden die Mechanismen in Form von „Diensten“ implementiert, wobei das sich ergebende „Dienstekonzept“ eine beliebige Kombination der vorgestellten Mechanismen erlaubt. Abschließend wird die Notwendigkeit von Dienstsuchemechanismen erläutert, welche eine Folge des flexiblen Angebotes von Diensten pro REACH-Proxyserver darstellt [EvSe08]. Das Kapitel endet mit einer Betrachtung der durch REACH aufgeworfenen Sicherheitsproblematik.

4.1 Mechanismen zum Abfangen von Daten

Es gibt eine Vielzahl an Mechanismen, welche zum Abfangen von Nutzerdaten mit dem Ziel deren Sicherung herangezogen werden können. Der Übersicht halber bietet es sich an, die Dienste gemäß des Internet-Schichtenmodells zu gruppieren und der Reihe nach vorzustellen. Dabei wird anfangs auf eine detaillierte Vorstellung verzichtet, welche dann ab Abschnitt 4.3.2 für Erfolg versprechende Dienste nachgereicht wird.

4.1.1 Auf der Anwendungsschicht

Eine Anforderung bei der Entwicklung von REACH war es, zum Einsatz keine Modifikationen der vom Nutzer eingesetzten Anwendungen zu erfordern. Im Folgenden werden Mechanismen vorgestellt, welche entweder die Anwendungsdatenströme oberhalb der Transportschicht (und damit im Internet-Schichtenmodell auf der Anwendungsschicht) abfangen, oder die Daten auf niedrigeren Schichten abfangen, um diese dann mit Wissen des zugehörigen Anwendungsschichtprotokolls gezielt zu modifizieren. Letzterer Ansatz

ist mit „Deep Packet Inspection“ (DPI) [Deep] vergleichbar und wird zur Implementierung von Diensten herangezogen, welche ein „Connection Tracking“ benötigen.

4.1.1.1 Die Proxyprotokolle SOCKSv4 und SOCKSv5

Dank der Proxyprotokolle SOCKSv4 und SOCKSv5 (RFC 1928 [LGLK+96]) steht ein Mechanismus bereit, welcher es erlaubt, Anwendungen erst zu deren Laufzeit um eine Mobilitätsunterstützung zu erweitern. Dazu werden praktischerweise deren Quelltexte nicht benötigt. SOCKS setzt dazu in beiden Versionen den Mechanismus der *indirekten Sockets* um, bei welchen alle Nutzdaten nicht länger *direkt* von einem Client an einen Server im Internet gesendet, sondern stattdessen über einen Proxyserver geleitet werden.

SOCKSv4 bietet bereits eine vollwertige Unterstützung für ausgehende TCP-Verbindungen an. Eingehende TCP-Verbindungen werden allerdings nur rudimentär unterstützt, da hiermit nur die Begleitverbindungen bestimmter Anwendungsschichtprotokolle (beispielsweise für das „File Transfer Protocol“ (FTP)) unterstützt werden sollen. Für SOCKSv5 wurde der Funktionsumfang um eine Unterstützung für UDP erweitert.

Dieser Mechanismus kann dazu verwendet werden, die Datenströme von Anwendungen abzufangen. Hierzu wird der SOCKS-Mechanismus so parametriert, dass die Anwendung statt eines „echten“ SOCKS-Proxyservers ein lokal auf dem mobilen Endgerät laufendes Programm kontaktiert. Dieses verhält sich dabei genauso wie ein SOCKS-Proxyserver, ist aber primär dazu da, die Verbindungen der Anwendungen noch vor der mobilen Luftschnittstelle zu terminieren. Seitens des Internets würde dann eine Partnerinstanz die abgefangenen Datenströme an die Server leiten, und somit gleichzeitig einen „Ankerpunkt“ verkörpern.

Vorteile: Mit Hilfe von SOCKS lassen sich ausgehende TCP-Verbindungen und bei SOCKSv5 auch UDP-Assoziationen über einen Proxyserver leiten. Die Unterstützung für eingehende TCP-Verbindungen wird für bestimmte Anwendungsfälle, wie beispielsweise aktives FTP, angeboten. Konzepte zur Namensauflösung sind bereits integraler Bestandteil von SOCKSv5, sodass hierbei auch Funktionen des „Domain Name Service“ (DNS) über den SOCKS-Proxyserver abgewickelt werden können.

Eine auf der Anwendung von SOCKS basierende Mobilitätsunterstützung zeigt keine Probleme bezüglich „horizontaler“ und „vertikaler Handover“. Lang anhaltende Isolationssituationen führen nicht zu einem Abriss abgefangenener TCP-Verbindungen und UDP-Assoziationen, da die mobile Teilstrecke weder für die Anwendungen auf dem mobilen Endgerät noch für die Server im Internet sichtbar ist.

Nachteile: Leider ist es nicht möglich, *alle denkbaren* Anwendungen SOCKS-kompatibel zu machen. Zudem unterstützt SOCKSv5 zwar TCP und UDP, aber für weitere Transportschichtprotokolle wie das „Stream Control and Transmission Protocol“ (SCTP, RFC 4960 [Stew07]), das „Datagram Congestion Control Protocol“ (DCCP, RFC 4340 [KoHF06]) oder gar die Hilfsprotokolle der Vermittlungsschicht „Internet Control Message Protocol“ (ICMP, RFC 792 [Post81]) und „Internet Group Management Protocol“ (IGMP, RFC 1112 [Deer89], RFC 3376 [CDKF+02], RFC 3376 [CDKF+02] und RFC 5519 [ChHa09]) existiert keine Unterstützung. Außerdem können mit Hilfe von SOCKS keine Serverdienste, welche auf einem mobilen Endgeräten laufen sollen, nach außen hin angeboten werden, beispielsweise für einen Fernwartungsdienst. SOCKS kann mit Hilfe seines `BIND`-Kommandos nur eine einzige TCP-Verbindung pro am Proxyserver reserviertem Port „nach innen“ leiten. Für FTP bleibt diese Einschränkung jedoch folgenlos. Schlussendlich bleibt festzustellen, dass durch den Einsatz von SOCKS die Ende-zu-Ende-Semantik von TCP verloren geht.

4.1.1.2 SIP-basierte Internettelefonie

Sprachübertragung über das Internet, im Englischen als „Voice over IP“ (VoIP) bezeichnet, ist eine in der Praxis mittlerweile häufig anzutreffende Technologie. Dabei lassen sich proprietäre Systeme wie beispielsweise Skype [Skyp] von „offenen“ und standardisierten Systemen unterscheiden. Letztere basieren beispielsweise auf dem „Session Initiation Protocol“ (SIP, RFC 3261 [RSCJ+02]) in Verbindung mit dem „Real-Time Protocol“ (RTP, RFC 3550 [SCFJ03]) und dem „Real Time Control Protocol“ (RTCP, ebenfalls RFC 3550 [SCFJ03]), auf welche ein besonderer Fokus in dieser Arbeit gelegt wird.

SIP, RTP und RTCP sind Anwendungsschichtprotokolle, welche mit Hilfe von UDP[1] übertragen werden. SIP-Nachrichten zeichnen sich dadurch aus, dass sie IP-Adressen und Portnummern bezüglich zu etablierender RTP-Assoziationen einbetten, was zu diversen Problemen beim Einsatz in der Praxis führt. Zu nennen wäre hier die so genannte „NAT-Problematik“, welche dann zu Tage tritt, wenn SIP über NAT-Grenzen hinweg eingesetzt werden soll. Die dann von den SIP-Clients in die SIP-Nachrichten eingebetteten IP-Adressen wären in diesem Fall private Adressen, sodass diese nach Überschreitung der NAT-Grenze ihre Funktion nicht länger erfüllen können. Es ist also erkennbar, dass bereits zur Lösung der NAT-Problematik Vorkehrungen getroffen werden müssen, ohne hier auf die Vielzahl an verfügbaren Lösungen einzugehen. Es ist jedoch ersichtlich, dass die SIP-Nachrichten beim Überschreiten der NAT-Grenze angepasst werden müssen, was

[1] SIP-Nachrichten können allerdings auch mit Hilfe von TCP übertragen werden, jedoch lässt sich in der Praxis eher der Versand mittels UDP beobachten.

beispielsweise durch ein „Application Layer Gateway“ (ALG) geschehen kann.

Eine Mobilitätsunterstützung für SIP-basierte Sprachübertragung kann beispielsweise auf dem Angebot eines solchen ALGs basieren. Dadurch können sämtliche auf SIP-basierende Datenströme durch ein „Hilfsprogramm“ geleitet werden, welches mit dem Ziel des Angebotes einer Mobilitätsunterstützung erstellt wurde. Durch Interpretieren (parsen) und Ändern der SIP-Nachrichten können ebenfalls die RTP- und RTCP-basierten Datenströme umgeleitet und abgefangen werden, sodass die Luftschnittstelle für die SIP-Telefone nicht sichtbar ist. Im Fall einer Isolationssituation können sogar künstliche Bestätigungen zur Verhinderung von Fehlerzuständen (Timeouts) in bestehende SIP-Sitzungen eingespielt werden.

Noch „publikumswirksamer“ ist jedoch die Möglichkeit, in bestehende Sprachverbindungen eine künstliche Ansage („Bitte nicht auflegen, die Verbindung wird gehalten!“) einzuspielen, wie auf der CeBIT 2009 anhand der Besucherreaktionen festgestellt werden konnte. Sowohl dem mobilen Telefonierer als auch dem Gesprächspartner hilft eine solche Ansage, um einen temporären Ausfall des Netzzugangs geduldig auszusitzen.

Vorteile: Da speziell die SIP-basierte Sprachübertragung im mobilen Umfeld ohne weitere Vorkehrungen nicht reibungslos funktioniert, bleibt keine andere Wahl, als eine diesbezügliche Mobilitätsunterstützung zu entwickeln. Dabei werden idealerweise die NAT- und die Firewall-Problematik gleich mit gelöst. Darauf aufbauend bietet sich die Möglichkeit an, die Anwender durch künstliche Ansagen oder Wartemelodien während Isolationssituationen zu unterstützen. Anzumerken ist, dass alle auf Mobile IP und dessen Weiterentwicklungen aufsetzenden Schemata eine solche Unterstützungen *nicht* bieten.

Nachteile: Um die Nachteile hinter dieser Idee beleuchten zu können, müssen zwei verschiedene Betriebsmodi unterschieden werden: Die in Form eines ALG bereit gestellte Mobilitätsunterstützung kann entweder wie ein SIP-Server (Als „Private Branch Exchange“ (PBX) oder auch als SIP-Proxyserver) angesprochen werden, oder alternativ *transparent* in den Kommunikationsweg zwischen SIP-Client und SIP-Server eingefügt werden. Erster Fall ist plattformunabhängig implementierbar und erfordert keine Einschränkungen bezüglich der Topologie. Allerdings ist ein erhöhter Konfigurationsaufwand erforderlich. Immerhin muss das SIP-Telefon auf dem mobilen Endgerät so konfiguriert werden, dass er das ALG als SIP-Server benutzt, und dem ALG wiederum muss die Adresse des eigentlichen SIP-Servers mitgeteilt werden.

Im *transparenten* Betriebsmodus braucht dagegen die Konfiguration der SIP-Telefone nicht angepasst zu werden. Das ALG fängt den Datenverkehr unbemerkt von SIP-Telefon und SIP-Server ab und nimmt dabei Einfluss auf die Signalisierungs- und Sprachdaten.

Dieses Schema kann allerdings nur in wenigen Szenarien eingesetzt werden, da es eine bestimmte Topologie voraussetzt. Nur wenn die Mobilitätsunterstützung in Form einer so genannten REACH-Box daherkommt, also auf einem externen Gerät implementiert ist, können UDP-Pakete transparent abgefangen werden. Bezüglich des zum transparenten Abfangen von Daten notwendigen „transparenten Proxyservers“ sei auf die Abschnitte 4.1.2.2 und 4.3.2.2 verwiesen.

In beiden Betriebsmodi darf zudem keine Verschlüsselung der Signalisierungsdaten aktiviert werden. Das ALG wäre dann nicht mehr in der Lage, die SIP-Nachrichten zu lesen geschweige denn zu verändern.

4.1.2 Auf der Transportschicht

Die heutzutage im Internet am häufigsten eingesetzten Transportschichtprotokolle sind TCP und UDP. Es ist möglich, die Datenströme der Anwendungen genau an dieser Stelle im Protokollstapel abzugreifen. Dazu bieten sich zwei Schemata an: Das so genannte „Port Forwarding“ sowie ein Mechanismus bekannt als „transparenter Proxyserver“.

4.1.2.1 Weiterleitung von Ports

Sowohl auf dem mobilen Endgerät als auch auf einem REACH-Proxyserver können TCP- und UDP-Ports zum Zweck der Weiterleitung dort angenommener Verbindungen bzw. Assoziationen belegt werden. So können auf einem mobilen Endgerät Serverdienste angeboten werden, wie beispielsweise ein HTTP-Server oder ein „Secure Shell“-Fernwartungszugang (SSH). Um auf diese Serverdienste dann seitens des Internets zuzugreifen, würde ein *externer* Nutzer einen am REACH-Proxyserver reservierten Port kontaktieren, und dann schlussendlich durch die geschaltete Weiterleitungsregel das mobile Endgerät erreichen. Dieses Schema ist in der Lage, eine Vielzahl an gleichzeitigen Verbindungen und Assoziationen zu verwalten.

Die Weiterleitungsregeln können dabei „nach außen“ (es werden Verbindungen am REACH-Client angenommen und vom REACH-Server ins Internet geleitet) oder „nach innen“ (geöffneter Port am REACH-Server mit Weiterleitung auf ein mobiles Endgerät) gerichtet sein. Unabhängig von der Richtung des Aufbaus ist pro angenommener TCP-Verbindung und UDP-Assoziation ein bidirektionaler Datentransfer möglich. Im Rahmen dieser Dissertation wurde die Weiterleitung der praxisrelevanten Protokolle TCP und UDP untersucht und implementiert. Weitere Transportschichtprotokolle können ebenfalls untersucht und gegebenenfalls integriert werden.

Vorteile: Die lokal oder an einem REACH-Proxyserver reservierten Portnummern für Verbindungen und Assoziationen werden *vom mobilen Nutzer* vorgeschlagen; eine Unterstützung durch einen Administrator ist nicht notwendig. Dabei werden mehrere gleichzeitig aktive Datenströme pro Schaltregel unterstützt. Sowohl weitergeleitete TCP-Verbindungen als auch UDP-Assoziationen können beliebig lange offen gehalten werden und überleben damit auch lang anhaltende Isolationssituationen.

Nachteile: Durch den vorgestellten Mechanismus werden zwar auch *eingehende* Transportschichtverbindungen unterstützt, jedoch weist der Ansatz einen „statischen Charakter“ auf: Bevor Verbindungen abgefangen und weitergeleitet werden können, muss der Nutzer explizit eine Portnummer reservieren und die vollständige Zieladresse für die Weiterleitungsregel festlegen. Von einer „vollwertigen Erreichbarkeit“ (unabhängig davon, ob hierbei jetzt eingehende oder ausgehende Schaltregeln gemeint sind) kann daher nicht die Rede sein. Ein FTP-Server im passiven Betriebsmodus beispielsweise, welcher mit dynamisch reservierten Portnummern für die Datenkanäle arbeitet, kann somit nicht auf dem mobilen Gerät unterstützt werden. Das Weiterleitungsschema wäre dafür „zu statisch“.

Da der Mechanismus auf der Transportschicht aufsetzt, werden IP-Hilfsprotokolle wie ICMP und IGMP nicht unterstützt. In Bezug auf TCP geht zudem durch die Trennung in mehrere Übertragungsabschnitte die Ende-zu-Ende-Semantik verloren.

4.1.2.2 „Transparente Proxyserver“

Der Paketfilter NETFILTER aktueller LINUX-Kernel ist in der Lage, so genannte „transparente Proxyserver“ anzubieten. Bei diesem Mechanismus können alle den lokalen Paketfilter durchlaufenden TCP-Verbindungen abgefangen und auf einen Port des mobilen Endgeräts umgeleitet werden. An diesem Port kann die umgeleitete TCP-Verbindung durch ein Programm (die Proxyanwendung) angenommen werden. Für den Initiator der abgefangenen TCP-Verbindungen bleibt dabei verborgen, dass der Kommunikationspartner ein Zwischensystem ist. Für ihn sieht es so aus, als kommuniziere er mit dem gewünschten Zielsystem. Der Proxyserver ist demnach *transparent*.

Das dazu bei NETFILTER notwendige `REDIRECT`-Kommando greift direkt in den Paketfilter des LINUX-Kernels ein. Dadurch wird es möglich, die ursprüngliche Zieladresse bestehend aus IP-Adresse und Portnummer auszulesen, welche gültig war, bevor die Verbindung zur lokalen Proxyanwendung umgeleitet wurde. Die so ermittelte Zieladresse kann dazu benutzt werden, um eine zweite ausgehende Verbindung zum eigentlichen Zielsystem aufzubauen.

Dieser Mechanismus war ursprünglich für HTTP-Proxyserver gedacht, welche ohne Notwendigkeit der Konfiguration der Browser einen Zwischenspeicher (Cache) realisieren sollten. Die prominenteste Anwendung auf diesem Gebiet ist der Web-Proxyserver SQUID [Squi], zu welchem man viele „Treffer“ erhält, wenn man im Internet nach Informationen zu „transparenten Proxyservern“ sucht. Weiterhin konnte der Autor mit diesem Mechanismus eine Kopplung zwischen Sensornetzen und dem Internet ermöglichen (siehe studentische Arbeiten [Hess06] und [BrHW08]).

Für REACH kann dieser Mechanismus direkt auf dem mobilen Endgerät adaptiert werden. Die auf dem mobilen Endgerät laufende Proxyanwendung wäre in diesem Fall der REACH-Client. Dieser würde sämtliche von den Anwendungen stammenden TCP-Verbindungen abfangen und jeweils die ursprüngliche Zieladresse ermitteln. Diese Zieladresse würde dann am REACH-Server dazu verwendet werden, um über eine zweite TCP-Verbindung das ursprünglich gewünschte Zielsystem zu erreichen. Da sich REACH-Client und REACH-Server mit Hilfe eines handoverresistenten Übertragungsschemas verständigen, sind automatisch auch die abgefangenen TCP-Verbindungen geschützt.

Vorteile: Durch den Einsatz des Mechanismus der „transparenten Proxyserver“ können sowohl die Anwendungen auf dem mobilen Endgerät als auch die Serverdienste im Internet vor der den negativen Auswirkungen durch Handover geschützt werden. Lang anhaltende Isolationssituationen sind unproblematisch. Ideal ist, dass keine weiteren Einstellungen in den Anwendungen vorgenommen zu werden brauchen.

Nachteile: Bei der aktuellen Implementierung des Paketfilters im LINUX-Kernel können lediglich *ausgehende TCP-Verbindungen* in Bezug auf eine Mobilitätsunterstützung abgefangen werden.[2] Zudem treten beim Einsatz bestimmter auf TCP aufsetzender Protokolle, beispielsweise FTP, Probleme auf. Die bei FTP im Steuerdatenstrom eingebetteten IP-Adressen verfehlen durch den REACH-Proxyserver ihren Sinn. Da dieser die abgefangenen Datenströme lediglich durchreicht ohne die eingebetteten IP-Adressen anzupassen, kommt es beim späteren Aufbau des FTP-Datenkanals zu einem Fehlverhalten. Da zudem nur ausgehende Verbindungen abgefangen werden können, muss bei FTP auf den aktiven Modus verzichtet werden. Beim passiven Betriebsmodus treten dagegen keine der genannten negativen Beeinflussungen auf.

Auch hier gilt, dass die Ende-zu-Ende-Semantik von TCP verloren geht. Der Mechanismus der „transparenten Proxyserver“ ist zudem nur für LINUX und manche UNIX-

[2] Es gibt allerdings noch den Sonderfall, dass eine so genannte REACH-Box zum Einsatz kommt. Nur dann kann der „transparente Proxyserver“ auf Wunsch so geschaltet werden, dass er auch ausgehende UDP-Assoziationen mit abfängt.

artigen Betriebssysteme verfügbar. Im Vorfeld muss eine Paketfilterregel hinterlegt werden, wozu Systemverwalterrechte benötigt werden. Unter Microsoft Windows sind nach aktuellem Kenntnisstand des Autors keine „transparenten Proxyserver“ realisierbar.

4.1.3 Auf der Vermittlungsschicht

Auf der Vermittlungsschicht können IP-Pakete mit Hilfe so genannter Tunnelendpunkte abgefangen werden. Diese tragen unter GNU/Linux die Bezeichnung `tunX`, mit `X` beginnend bei `0` aufsteigend nummeriert. Unter Microsoft Windows ist dieser Mechanismus ebenfalls verfügbar. Durch eine solches virtuelles Netzzugangsgerät wird dann der Datenverkehr geleitet, was durch Setzen einer Route geschieht.

Hinter einem solchen „virtuellen Netzzugangsgerät“ verbirgt sich eine Anwendung, wie beispielsweise der VPN-Client aus der OpenVPN Programmsammlung [OVPN]. Dieser nimmt die durch die „virtuelle Schnittstelle“ geleiteten IP-Pakete entgegen, kapselt sie in UDP-Pakete und überträgt diese dann an eine VPN-Partnerinstanz. Da dieses Schema direkt mit IP-Paketen arbeitet, werden automatisch *alle* IP-basierten Protokolle mit erfasst.

Im Detail betrachtet, sind hierbei der auf dem mobilen Endgerät laufende VPN-Client und der im Internet installierte VPN-Server über eine einzige bidirektionale UDP-Assoziation miteinander verbunden. Diese UDP-Assoziation muss durch REACH abgefangen und die UDP-Pakete „handoversicher“ übertragen werden.

Auf Ebene des VPNs kann dabei unterschieden werden, ob für die mobilen Endgeräte öffentliche oder private IP-Adressen (gemäß RFC 1918 [RMKG+96] und RFC 3330 [IANA02]) vergeben werden. Die Vergabe privater IP-Adressen gestaltet sich als praktikabler, weil dazu im Vorfeld kein Block an öffentlichen IP-Adressen reserviert werden braucht. Allerdings erkauft man sich diesen Vorteil durch die Notwendigkeit, am VPN-Server NAT durchführen zu müssen. Dadurch erbt dieser Ansatz alle Probleme, die NAT mit sich bringt, angefangen von der Unterbindung eingehender Verbindungen bis hin zu komplexen Problemen, wie sie sich bei VoIP in Verbindung mit NAT zeigen (siehe RFC 3027 [HoSr01]). Zwar können an der NAT-Grenze des VPN-Servers Regeln zum „Port-Forwarding“ hinterlegt werden, jedoch ist dies mit einem hohen administrativen Aufwand unter Notwendigkeit von Systemverwaltungsrechten verbunden.

Stehen dagegen für die mobilen Endgeräte bezogen auf das VPN öffentliche IP-Adressen zur Verfügung, kann auf NAT verzichtet werden. Es besteht dann eine „vollwertige“ Ankopplung über IP, welche auch jede Form von eingehenden Verbindungen erlaubt.

Vorteile: Das herausragende Merkmal einer VPN-basierten Lösung besteht darin, dass sämtliche auf IP-basierenden Protokolle erfasst werden können. Werden für die VPN-Adapter global gültige IP-Adressen vergeben, stellen sogar eingehende Verbindungen kein Problem mehr dar. Es gibt eine Vielzahl an VPN-Programmpaketen, von denen manche proprietär, manche andere dagegen „offen“ sind. Für die vorgeschlagene Lösung ist dies jedoch irrelevant, da das Abfangen der getunnelten Daten ohne Kenntnis des sich dahinter verbergenden VPN-Systems erfolgen kann.

Die Ende-zu-Ende-Semantik der Transportschichtverbindungen bleibt bei diesem Ansatz erhalten. Kommen jedoch private IP-Adressen zum Einsatz, kann der Einsatz von „Application Layer Gateways“ (ALGs) an der NAT-Grenze erforderlich werden. Für FTP bietet der Linux-Kernel beispielsweise einen eigenen „NAT-Helper“ an, welcher das hierfür notwendige „Connection Tracking“ implementiert.

Ein weiterer Vorteil ist, dass die VPN-Systeme Verschlüsselungsverfahren bereit stellen, sodass ein Mitlesen der getunnelten Daten unterbunden werden kann.

Nachteile: Da es selten möglich sein wird, ein VPN mit global gültigen IPv4-Adressen betreiben zu können, muss in diesem Fällen die Anbindung an das Internet unter Zuhilfenahme von NAT erfolgen. Dabei treten leider alle NAT-spezifischen Probleme bezüglich der Protokolle der höheren Schichten zu Tage. Zudem sind eingehende Verbindungen in diesem Fall nur durch „hart“ vom Systemadministrator geschalteten Regeln zum „Port Forwarding“ möglich.

Ein weiteres Problem tritt während Isolationssituationen auf, da der dargestellte Mechanismus keinerlei Unterstützung der gekapselten Transportschicht- und Anwendungsschichtprotokolle anbietet. Bezüglich der Anwendungsschicht können sich bereits kurzzeitige Isolationssituationen als fatal erweisen, wenn beispielsweise ein aktives Telefonat mangels Sprachdaten verstummt und einer der Gesprächspartner irritiert auflegt.

4.1.4 Auf der Netzzugangsschicht

VPN-Adapter können auch auf der Netzzugangsschicht arbeiten, man spricht dann von so genannten „virtuellen Ethernetkarten“. Diese heißen unter GNU/Linux `tapX`, wobei das `X` wiederum zur Nummerierung verwendet wird.

Da ein solcher VPN-Adapter auf der MAC-Schicht angesprochen wird, kann er mit anderen Ethernetkarten „zusammengebrückt“ werden. Dann werden sogar „Broadcasts“ durch das VPN geleitet. Zudem ist es einer (virtuellen) Ethernetkarte egal, welches Vermittlungsschichtprotokoll über dieses geleitet wird. Somit steht es dem Anwender offen, auch andere Vermittlungsschichtprotokolle als IP einzusetzen.

Vorteile: Die sich hierbei ergebenen positiven Eigenschaften bezüglich des Angebotes einer Mobilitätsunterstützung sind vergleichbar mit denen, die in Bezug auf die IP-basierten VPNs im letzten Abschnitt genannt wurden. Es können jedoch auch Broadcasts unterstützt oder ganz andere Vermittlungsschichtprotokolle als IP getunnelt werden.

Nachteile: Die Nachteile aus dem letzten Abschnitt, in welchem IP-basierte VPNs vorgestellt wurden, sind hier an dieser Stelle ebenfalls gültig. Zudem ist der Aufwand zur Übertragung getunnelter MAC-Rahmen im Vergleich zur Übertragung abgefangener IP-Pakete größer.

4.2 Szenarien

Um die Daseinsberechtigung der unterschiedlichen Mechanismen zum Abfangen von Nutzerdaten zu untermauern, wurden Anwendungsszenarien herausgearbeitet. Diese basieren einerseits auf typischen Endanwenderanforderungen in Bezug auf die Nutzung des Internets im mobilen Umfeld; andererseits muss auf bestimmte Sonderwünsche, wie beispielsweise dem Angebot von Fernwartungszugängen auf mobilen Endgeräten, eingegangen werden. Schlussendlich wird gezeigt, dass zwar für jedes Szenario wenigstens ein geeigneter Mechanismus („Dienst“) verfügbar ist, jedoch keiner der genannten Mechanismen für sich in der Lage ist, wiederum alle Szenarien zu unterstützen.

4.2.1 Unkomplizierter Internetzugang

Einem durchschnittlichen Nutzer eines mobilen Endgeräts wird es ausreichen, wenn es ihm ermöglicht wird, dass er unterwegs weiterhin und ohne negative Beeinflussung seine gewohnten Anwendungen benutzen kann. Der Nutzer ist sich dabei seiner Mobilität bewusst, sie darf aber zu keinen unerwarteten Problemen bezüglich der Kommunikation führen. Kurze Aussetzer bezüglich der Datenrate werden beispielsweise nicht als negativ empfunden, wenn der Anwender gerade einen Tunnel betritt und sein mobiles Gerät von der Netzabschattung betroffen ist.

Wichtig ist hierbei die Tatsache, dass es eine Vielzahl an Anwendungen gibt, welche ein Nutzer unterwegs einsetzen möchte. Dazu gehören diverse Browser, eine Unzahl an Mail- und Chatprogrammen, Online-Spiele und eine Vielzahl weiterer Programme. Diese bieten aber oft kaum Möglichkeiten der Konfiguration an, sodass es am geschicktesten wäre, es bedürfe hier keiner zusätzlichen Änderungen oder Einstellungen.

Dieses Szenario wurde bei der Präsentation von REACH auf der CeBIT 2009 besonders betont, da sich die meisten Interessenten hiermit am ehesten identifizieren konnten. Dabei wurden dem Autor eine Vielzahl an negativen Mobilitätserfahrungen kommuniziert, welche den Messebesuchern bei Einsatz von REACH erspart geblieben wären:

Arbeiten unterwegs in der Bahn: Ein Standbesucher schilderte, dass es ihm unmöglich sei, auf Bahnreisen effektiv mit seinem Laptop zu arbeiten. Er müsse dazu an einem „Content Management System“ (CMS) seiner Firma angemeldet sein, um online arbeiten zu können. Für den Internetzugang benutzte er einen „Universal Serial Bus“ (USB)-Stick zur Einwahl über UMTS. Das Problem des Nutzers manifestierte sich bei der Durchfahrt eines jeden Tunnels, wobei das UMTS-Modem die Verbindung verlor und nach Verlassen des Tunnels eine erneute Einwahl erforderlich wurde. Dabei bekam sein Laptop eine neue IP-Adresse zugewiesen. In der Folge konnte der Server des CMS die Sitzung nicht mehr zuordnen, und beim nächsten Zugriff gab es eine Fehlermeldung. Dabei waren seine bisherigen Eingaben verschwunden, und ein effektives Arbeiten unmöglich.

Durch Installation des REACH-Clients auf dem mobilen Endgerät in Verbindung mit Zugang zu einem REACH-Proxyserver kann dieses Problem effektiv gelöst werden.

Internetzugang eines KFZ: Gleich drei Vertreter ungenannter Automobilhersteller gaben auf der Messe preis, dass bei ihnen an Lösungen zur Internetanbindung von Fahrzeugen gearbeitet wird. Zuerst wurde erfreut festgestellt, dass REACH bereits mehrere Netzzugangstechnologien einbinden und auswählen kann, wie beispielsweise das kostenlos nutzbare WLAN-Netz daheim in der Garage oder das UMTS-Netz, wenn das Fahrzeug unterwegs ist. Nach Aussagen der Vertreter wird der Internetzugang unter anderem für zukünftige Navigationsgeräte benötigt, welche sich errechnete Routen, Kartenmaterial oder gar Firmwareupdates aus dem Internet herunterladen sollen. Ungelöst sind dabei zwei Probleme: Zum einen sind Navigationsgeräte nicht in der Lage, Netzzugangsgeräte zu verwalten. Zum anderen kommen sie nicht mit den mobilitätsspezifischen Problemen zurecht. Wenn während des Downloads eines Kartenausschnittes das Fahrzeug das WLAN-Heimnetz verlässt und auf UMTS umgeschaltet werden muss, oder wenn ein Fahrzeug mit halbem Download für zehn Minuten in einem Tunnel isoliert bleibt, gehen die Systeme in einen Fehlerzustand. Dies betrifft außerdem moderne Radios, welche ihre Musik in Form eines „Streams“ aus dem Internet beziehen oder auf die heimische Musiksammlung des Nutzers zugreifen sollen.

Durch Vorschaltung einer REACH-Box könnte die Verwaltung der Netzzugangsgeräte sowie die Sicherung gegenüber Handoverereignissen ausgelagert werden, sodass Navigationsgeräte und Radios weiterhin auf den bisherigen nicht handoverfähigen Kommunika-

tionskonzepten basieren dürfen. Die REACH-Box wäre dann aus Sicht des Navigationsgeräts ein „Gateway“, über welches sämtlicher „nach außen“ gerichteter Datenverkehr geleitet werden muss. Die REACH-Box könnte zudem eine Basisstation für WLAN darstellen, und so den Insassen während der Fahrt die Möglichkeit zu geben, mit ihren Laptops oder PDAs handoversicher im Internet zu surfen. Diese Geräte benötigen keinen REACH-Client mehr.

Ansätze: Als am geeignetsten erscheint es hierbei, eine VPN-basierte Lösung einzusetzen. Hierbei werden bereits alle Datenströme der Anwendungen transparent abgefangen und über einen REACH-Server geleitet (bzw. schlussendlich über den gewählten VPN-Server). Es werden hierzu keinerlei Einstellungen in den Anwendungen benötigt, und alle relevanten Protokolle wie TCP, UDP und auch ICMP werden unterstützt. Allerdings besteht im Fall einer Isolationssituation kein Schutz der Transportschichtverbindungen.

Alternativ bietet sich der Einsatz von SOCKSv5 an, da dieses ebenfalls Unterstützung für die wichtigen Transportschichtprotokolle TCP und UDP mit sich bringt. Allerdings liefert SOCKSv5 keine Unterstützung für IP-Hilfsprotokolle wie ICMP. Wer allerdings nur im Internet surfen und Standardanwendungen benutzen möchte, dem reicht die Unterstützung von TCP und UDP in der Regel aus.

Als besonders geeignet erweist sich der Mechanismus des „transparenten Proxyservers“, welcher ausgehende TCP-Verbindungen behandeln kann, ohne Erweiterungen oder Einstellungen bezüglich der Anwendungen auf dem mobilen Endgerät zu erfordern. In vorliegenden Anwendungsfall, wo der REACH-Client auf einer eigenständigen REACH-Box installiert ist, können sogar ausgehende UDP-Assoziationen abgefangen werden (vgl. Abschnitt 4.3.2.2). Allerdings wird keine Unterstützung für eingehende TCP-Verbindungen und UDP-Assoziationen, für alternative Transportschichtprotokolle neben TCP und UDP sowie für IP-Hilfsprotokolle angeboten.

4.2.2 Serverdienste auf dem mobilen Endgerät

Es gibt Szenarien, welche Serverdienste auf mobilen Endgeräten involvieren.

Fernwartungszugang: Dies ist zum Beispiel der Fall, wenn zum Zwecke der Fernwartung ein „Secure Shell Daemon“ (`sshd`) gestartet werden soll. Um diesen von außen erreichen zu können, bedarf es einer sorgfältigen Wahl der eingesetzten Mobilitätsunterstützung. Es stehen verschiedene Mechanismen bereit, um solche Serverdienste erreichbar zu machen, allerdings unterscheiden sie sich in Bezug auf ihre Leistungsfähigkeit und den insgesamt notwendigen Konfigurationsaufwand.

Webzugriff auf ein mobiles Gerät: Auf einem mobilen Endgerät könnte ein Webserver installiert sein, welcher auf den Ports `80` (für HTTP) und `443` (für „HTTP Secure“ (HTTPS)) auf TCP-Verbindungen wartet. Dies ist besonders bei eingebetteten Systemen von Interesse, da hierbei häufig mit Hilfe von Webservern Statusinformationen zugänglich gemacht werden. Werden solche eingebetteten Systeme mobil, wird es schwierig, ohne weitere Hilfsmittel zuverlässig mit ihnen zu kommunizieren.

Ansätze: Eine „anwendungsfreundliche“ Lösung erhält man durch Einsatz eines VPNs, allerdings darf hierbei kein NAT zum Einsatz kommen. Die mobilen Endgeräte wären dann über ihre VPN-Adresse seitens des Internets erreichbar, und alle auf IP aufsetzenden Protokolle funktionieren ohne weitere Anpassungen. Nachteilig ist, dass häufig keine globalen IP-Adressen zur Verfügung stehen und private IP-Adressen samt NAT zum Einsatz kommen müssen. Hierbei bedarf es manuell hinterlegter Weiterleitungsregeln an der NAT-Schwelle, wozu aber Systemverwaltungsrechte erforderlich sind. Zudem muss festgestellt werden, dass VPNs nicht für den Einsatz in Szenarien mit längerfristigen Isolationssituationen geeignet sind.

SOCKSv5 wiederum ist nicht in der Lage, die für Serverdienste notwendige Vielzahl an eingehenden TCP-Verbindungen zu unterstützen. Es kann nur eine einzige TCP-Verbindung pro `BIND`-Kommando verarbeitet werden, was beispielsweise für Begleitverbindungen von FTP ausreicht, aber für „echte“ Serverdienste ungenügend ist.

Als am geeignetsten erweisen sich schlussendlich die Port-Weiterleitungsmechanismen von REACH bezüglich TCP und UDP. Diese verarbeiten eine Vielzahl an TCP-Verbindungen und UDP-Assoziationen in Richtung des mobilen Endgeräts. Obschon die weitergeleiteten TCP-Verbindungen handoverresistent und damit langlebig sind, geht die Ende-zu-Ende-Semantik verloren. Es werden keine Konfigurationen bezüglich der zu reservierenden Portnummern am REACH-Server benötigt. Sämtliche Anforderungen und Parameter werden seitens der mobilen Endgeräte durch den Endanwender vorgeschlagen. Man könnte diesen Mechanismus sinngemäß als „weich verdrahtetes, nach innen gerichtetes Port-Forwarding“ bezeichnen.

4.2.3 Wahl des gerade günstigsten Netzanbieters

Unterschiedliche Netzzugänge können sich hinsichtlich vieler Eigenschaften unterscheiden. In diesem Anwendungsfall seien die anfallenden „Geldkosten“ das für den Anwender entscheidende Kriterium. Dabei steht der Wunsch im Raum, immer den gerade günstigsten Netzanbieter zu benutzen.

Tarifierung nach Tageszeit: Dies ist insbesondere bei Einwahlverbindungen über „analoge Modems“ und ISDN der Fall, wobei häufig unterschiedliche Tarife je nach Tageszeit angeboten werden. So ist es gängige Praxis, aus einer Menge gegebener Provider je nach Tageszeit den einen oder anderen auszuwählen, um die anfallenden Telefonkosten zu minimieren. Problematisch ist allerdings, wenn bei bereits bestehender Internetverbindung ein solcher „Umschaltzeitpunkt“ überschritten wird. Dann steht der Nutzer vor einem Dilemma: Entweder er arbeitet weiter wie bisher und nimmt den unnötig hohen Tarif in Kauf, oder er trennt seine Einwahlverbindung und beginnt eine sofortige Wiedereinwahl. Da hierbei alle Transportschichtverbindungen abreißen, ist der Einsatz einer Mobilitätsunterstützung wie REACH sinnvoll.

Bevorzugte Zugangsnetze: Manchmal hat ein mobiler Anwender Zugriff auf örtlich begrenzte, für ihn aber ohne zusätzliche Mehrkosten nutzbare Netzzugänge. Als Beispiel hierfür bietet sich die WLAN-Basisstation in der eigenen Wohnung an, welche für den Anwender als ein „idealer Netzzugangspunkt“ angesehen werden kann. Jedoch wäre die Schlussfolgerung, bevorzugt immer WLAN zu nutzen, falsch: Außerhalb der Wohnung ist das heimische Funknetz unerreichbar, und dem Nutzer stünden in diesem Beispiel nur noch kostenpflichtige „Hot Spots“ zur Verfügung. In diesem Fall kann es geschickter sein, außerhalb der Wohnung auf UMTS umzuschwenken, gerade wenn hierfür eine so genannte „volumenbasierte Flatrate“ zur Verfügung stünde. Um diese allerdings nicht übermäßig zu belasten, würde innerhalb der eigenen Wohnung wieder auf WLAN umgeschaltet werden.

Ansätze: Die sich hierbei ergebenen Anforderungen seitens des Nutzers sind identisch mit denen, die bereits in Abschnitt „Unkomplizierter Internetzugang“ (4.2.1) genannt worden sind. Dort wie hier ist der Nutzer an einer einfach zu bedienenden Lösung interessiert, welche ihm die Nutzung seiner Standardanwendungen erlaubt. Es ist hierbei egal, ob die eigentliche Ursache für einen Zugangswechsel in der Mobilität des Anwenders begründet liegt oder aus seinem Wunsch nach einem günstigeren Provider heraus resultiert.

4.2.4 Langlebige Transportschichtverbindungen

Es gibt Anwendungsfälle, in denen gezielt der Bedarf an langlebigen Transportschichtverbindungen im Vordergrund steht. Der Begriff „langlebig“ bezieht sich hierbei auf eine Resistenz gegenüber dem Wechsel von Netzzugangstechnologien und schließt lange Isolationssituationen mit ein.

Übertragung großer Datenmengen über DSL: Ein Anwendungsfall, welcher den Autor selbst betrifft, umfasst die Durchführung einer sehr umfangreichen Datenübertragung über einen DSL-Anschluss, welcher auf Grund der geringen Übertragungsrate im Uplink mehrere Tage in Anspruch nimmt. Manche Anbieter von „DSL-Flatrates“ führen alle 24 Stunden eine Zwangstrennung durch, wodurch sich die zugewiesene externe IP-Adresse ändert und somit alle TCP-Verbindungen unterbrochen werden. Auch technische Störungen werden, selbst wenn sie nur kurz bestehen, aus selbem Grund alle Transportschichtverbindungen kappen.

Überwachung mobiler Arbeitsmaschinen: Ein Messebesucher der CeBIT 2009 schilderte dem Autor eine interessante Installation: Innerhalb eines großflächigen Hafengebiets werden diverse Gabelstapler eingesetzt. Jeder Gabelstapler ist mit einem eingebetteten System ausgestattet, welches über WLAN mit einer Zentrale kommunizieren kann. Dafür baut jedes System eine einzelne dauerhafte „Telnet“-Verbindung über WLAN zur Zentrale auf. Über diese werden dann Statusinformationen an die Zentrale übermittelt, und in Rückrichtung auch Kommandos an die Gabelstapler versendet.

Problematisch ist hierbei, dass die Gabelstapler mobil sind und sich auch in Arealen ohne Netzabdeckung (innerhalb von Hallen oder hinter Gebäuden) aufhalten können. Kurzzeitige Aussetzer sind dabei unproblematisch, da alle Systeme mit festen IP-Adressen ausgestattet sind und somit verloren gegangene Pakete wiederholt werden können. Ist ein Gabelstapler allerdings länger als 9 Minuten vom Netz getrennt, kann die TCP-Verbindung auf Grund der Funktionsweise des „TCP Retransmission Timers“ absterben. Alle bis dahin in der Sendewarteschlange liegenden Statusmeldungen und Kommandos sind verloren, und das System funktioniert nicht länger wie erwartet.

Durch Einsatz von REACH kann dieser Anwendungsfall vollständig abgedeckt werden.

Ansätze: Für langlebige Transportschichtverbindungen ist eine Auftrennung der Verbindungen und damit der Verlust der Ende-zu-Ende-Semantik erforderlich. So stehen die notwendigen Quittungen zur Verfügung, welche ein Abreißen der Verbindungen im Isolationsfall verhindern.

Diese Eigenschaft wird bereits durch mehrere Ansätze erbracht: Sowohl bei Anwendung von SOCKS als auch den „transparenten Proxyservern“ steht diese Eigenschaft zur Verfügung. Auch der Port-Weiterleitungsmechanismus weist dieses Leistungsmerkmal auf, unabhängig davon, ob er für eingehende oder ausgehende TCP-Verbindungen eingesetzt wird.

4.2.5 SIP-basierte Internettelefonie

Ein weiteres Szenario könnte in dem expliziten Wunsch der Anwender bestehen, SIP-basierte Internettelefonie durchführen zu wollen. Hierbei treten jedoch bereits im „ortsfesten“ Betrieb Probleme auf, besonders, wenn dabei NAT-Grenzen überschritten werden sollen. Soll ein Endgerät zudem mobil werden, werden die Umstände nochmals komplexer. Eine Mobilitätsunterstützung sollte diese Probleme *transparent*, also ohne Zutun der Benutzer, erkennen und lösen können.

Ansätze: Das NAT-Problem bezüglich SIP-basierter Sprachkommunikation lässt sich bereits durch Einsatz eines VPNs lösen, allerdings nur, wenn den mobilen Endgeräten im VPN globale IP-Adressen zugewiesen werden. Das Firewallproblem besteht dann allerdings weiterhin, und mobilitätsspezifischen Herausforderungen wie längerfristigen Isolationssituationen wird nicht begegnet.

Eine umfassende Lösung hierfür, welche auch mit den speziellen Problemen im mobilen Umfeld zurecht kommen kann, muss auf der Anwendung eines ALGs basieren. Diese können sowohl das NAT- als auch das Firewallproblem lösen und während Isolationssituationen ein Fehlverhalten der beteiligten SIP-Telefone verhindern. Die Besonderheit einer solchen Mobilitätsunterstützung besteht darin, dass sogar die Benutzer mit einbezogen werden können. So wird durch das Einspielen einer Wartemelodie oder einer Sprachansage auf die Gesprächspartner eingewirkt, damit diese während einer Isolationssituation möglichst nicht auflegen.

4.2.6 Vergleich aller Mechanismen und Auswahl

In Tabelle 4.1 werden zusammenfassend alle Mechanismen („Dienste“) auf ihre Eignung zur Unterstützung der genannten Szenarien aufgeführt. Es lässt sich erkennen, dass kein Dienst für sich alleine in der Lage ist, alle Szenarien zu unterstützen.

Eines der Leistungsmerkmale von REACH besteht nun darin, diese Dienste auch kombiniert anbieten zu können. Das Ziel dabei ist, dass sich die Stärken der jeweiligen Mechanismen gegenseitig ergänzen, während ihre Schwächen gegenseitig überdeckt werden sollen. Es ist außerdem denkbar, dass die Szenarien in Kombination auftreten. So kann ein möglichst „unkomplizierter Internetzugang“ gewünscht werden, und zudem soll ein SSH-Dienst auf dem mobilen Endgerät für Zugriffe von außen erreichbar gemacht werden. Die SSH-Sitzungen sollen zudem handoverresistent sein, also auch längerfristige Isolationssituationen schadlos überstehen.

Hierfür bietet sich beispielsweise als „Unterbau“ die Verwendung eines VPNs an, sodass bereits ein vollwertiger Internetzugang bereit gestellt wird. Da sehr wahrscheinlich

Szenario: / Dienst:	Relay	SOCKSv4+5	TProxy	VoIP	VPN
Unkomplizierter Internetzugang	−	+	−	++	++
Serverdienste auf mobilem Gerät	++	−−	−−	−−	−
Wahl des Netzanbieters	−	+	−	++	++
Langlebige Verbindungen	++	++	++	−−	−−
SIP-basierte Internettelefonie	−−	−−	−−	++	−

++ sehr gut geeignet, + mit Einschränkungen, − nur Sonderfälle, −− ungeeignet

Tabelle 4.1: Jede der vorgestellten Mobilitätsunterstützungen (Dienste) ist mehr oder weniger gut geeignet, in einem bestimmten Szenario zu funktionieren. Die Bewertung spiegelt die Meinung des Autors in Bezug auf die jeweiligen Vor- und Nachteile wider.

für das VPN nur private IP-Adressen zum Einsatz kommen, werden noch keine eingehenden Verbindungen unterstützt. Hierfür aber am VPN-Server Paketfilterregeln zum nach innen gerichteten „Port Forwarding“ zu schalten wäre zwar möglich, würde aber einen manuellen Eingriff seitens des Administrators erfordern. Die so weiter geleitete Verbindungen würden zudem während längerfristiger Isolationssituationen abreißen.

Praktischer ist es stattdessen, die eingehenden TCP-Verbindungen über den nach innen gerichteten Port-Weiterleitungsmechanismus zu ermöglichen. Für den Fernwartungszugang reicht eine einzelne Weiterleitungsregel aus.

Bei Bedarf können später noch weitere Mechanismen aktiviert werden. Falls seitens des mobilen Nutzers beispielsweise der Wunsch im Raum stehen sollte, ausgehende TCP-Verbindungen zu bestimmten Servern langlebig zu gestalten, bietet sich der Mechanismus der „transparenten Proxyserver“ an. Dieser greift in die Wegewahl ein, und wird immer bevorzugt vor dem VPN aktiv (siehe Abschnitt 4.3.2.2). Das bedeutet, die selektiv durch den „transparenten Proxyserver“ erfassten TCP-Verbindungen werden gesondert behandelt, während alles das, was noch übrig bleibt, durch das VPN geleitet wird.

So lässt sich erkennen, dass es für den Anwender ideal ist, wenn er eine Vielzahl an Mechanismen gleichzeitig aktivieren kann. Da gezeigt wurde, dass jeder Mechanismus für sich gesehen Schwachpunkte hat und in bestimmten Szenarien versagt, ist deren Mischbarkeit ein herausragendes Leistungsmerkmal von REACH, welches keine der bisher vorgestellten Mobilitätsunterstützungen anbietet.

4.3 Das Dienstekonzept von REACH

In Abschnitt 4.1 wurden die verschiedenen Mechanismen vorgestellt, welche zum Abfangen von Datenströmen auf den mobilen Endgeräten zum Einsatz kommen können. Die jeweiligen Eigenschaften sowie Vor- und Nachteile wurden beleuchtet. Darauf aufbau-

end befasste sich Abschnitt 4.2 mit der Entscheidungsfindung, welche Mechanismen in welchen Szenarien die beste Wirkung erzielen.

Die Mechanismen wurden dabei lediglich oberflächlich vorgestellt. Jetzt, wo eine Übersicht vermittelt wurde und die jeweiligen Zielstellungen klar sind, wird für jeden Mechanismus auf die Realisierung als Dienst innerhalb von REACH eingegangen.

4.3.1 Architektur zur Diensterbringung

Bevor allerdings eine Vorstellung aller realisierten Dienste sowie eine detaillierte Beschreibung deren Funktion diskutiert wird, empfiehlt sich eine Vorstellung des transportorientierten Kerns von REACH. Er definiert, wie die Dienste implementiert werden müssen und wie die gesicherten Übertragungskanäle seitens der Dienste angefordert werden können.

Definition 1 (Dienst) *REACH bietet einem Anwender eine Menge an Diensten an, welche je nach gewünschtem Szenario aktiviert werden können. Es können mehrere Dienste gleichzeitig gestartet werden.*

4.3.1.1 Dienstangebot: Dynamisch ladbare „Relay Plugins“

Jede der in Abschnitt 4.1 vorgestellten Methoden zum Abfangen von Datenströmen wurde für REACH in Form dedizierter „Relay Plugins“ implementiert. Das bedeutet, dass es einen „diensteunabhängigen Unterbau“ (den transportorientierten Kern von REACH) sowie eine Menge an instantiierbaren Diensten gibt. In Abbildung 4.1 wird dieses Konzept verdeutlicht. Hier sind diverse „Relay Plugins“ am REACH-Client instantiiert worden. Zudem werden hier die Wechselwirkungen mit den Anwendungen sowie im Fall des VPN-basierten „Relay Plugins“ das Abfangen der UDP-Assoziation des VPN-Clients gezeigt.

Diese „Relay Plugins“ bilden die Grundlage für das modulare Dienstangebot, welches eines der Leistungsmerkmale von REACH darstellt. Es handelt sich dabei um dynamisch ladbare Objekte, so genannte „Dynamically loadable Shared Objects“ (DSOs). Zur Implementierung eines neuen „Relay Plugins“ muss eine C++-Schnittstellenklasse abgeleitet werden, und nur in dieser abgeleiteten Klasse werden sämtliche dienstspezifischen Funktionen implementiert. Alle „Relay Plugins“ verfügen somit über eine uniforme Schnittstelle und können vom Kern gleichermaßen behandelt werden. Die Plugins werden bei REACH mit Hilfe der „libtool dynamic loader library“ (LIBLTDL) geladen. Da diese einen plattformunabhängigen Mechanismus zum Laden von DSOs anbietet, ist der entstandene Quelltext diesbezüglich nicht auf einen Einsatz unter GNU/LINUX beschränkt.

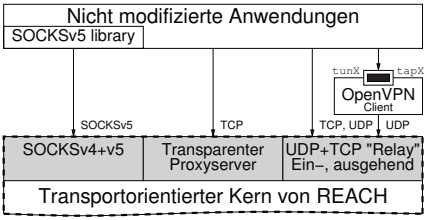

Abbildung 4.1: Sowohl am REACH-Client (hier gestrichelt eingerahmt) als auch am REACH-Server können eine Vielzahl an „Relay Plugins“ (grau eingefärbte Blöcke) instantiiert werden. Ihre Aufgabe ist es, die Datenströme der unmodifizierten Anwendungen gemäß eines bestimmten Schemas abzufangen und durch den transportorientierten Kern zu leiten.

Definition 2 (Relay Plugin) *Als „Relay Plugin“ wird eine dynamisch zu ladende Bibliothek bezeichnet. Durch diese wird das Diensteangebot von REACH in modularer Form bereit gestellt.*

„Relay Plugins“ werden zum Programmstart gemäß der vorgefundenen Konfigurationsdateien geladen und eingebunden. Es handelt sich dabei quasi um „Schablonen“, welche beliebig oft instantiiert werden können. Das bedeutet beispielsweise, dass für drei TCP-Weiterleitungsregeln zwar drei Instanzen des zugehörigen „Relay Plugins“ benötigt werden, aber dessen Bibliothek nur ein einziges mal geladen wird.

Definition 3 (Instanz) *Anhand eines geladenen „Relay Plugins“ können Instanzen erzeugt werden. Diese bieten den durch das „Relay Plugin“ verkörperten Dienst mit jeweils einer ganz bestimmten Konfiguration an. Auf dem REACH-Client kann es dabei mehrere Instanzen pro „Relay Plugin“ geben, auf dem REACH-Server maximal eine.*[3]

Spätere Instantiierungen zur Laufzeit sind ebenfalls möglich, wenn zum Beispiel zusätzliche Weiterleitungsregeln benötigt werden. Instantiierte „Relay Plugins“ werden parametriert und überwacht. Sie können jederzeit mit neuen Parametern versorgt oder beendet werden.

[3]Diese Einschränkung liegt im Dienstsuchemechanismus von REACH begründet. Der REACH-Client kann lediglich erkennen, *welche* „Relay Plugins“ ein bestimmter REACH-Server instantiiert hat. Ist ein bestimmtes „Relay Plugin“ an einem REACH-Server verfügbar, kann dieser ausgewählt werden. Es lassen sich jedoch keine „Variationen“ mehr unterscheiden, sodass am REACH-Server lediglich eine einzige Instanz pro Typ adressierbar ist (siehe Abschnitt 4.4 „Mitteilung des Diensteangebots“ sowie Abschnitt 7.4 „Dienstsuche“).

Ein weiterer Vorteil dieses modularen Ansatzes ist es, dass die „Relay Plugins“ unabhängig vom Kern gepflegt und auch zur Laufzeit des Kerns noch installiert und geladen werden können.

4.3.1.2 Übertragungsdienst: Linkbündel und logische Verbindungen

Eine Herausforderung bestand darin, eine einheitliche Schnittstelle für alle Typen von „Relay Plugins“ zu schaffen. Aus Sicht des Kerns müssen alle geladenen Instanzen dieselbe Schnittstelle aufweisen, was aber bereits durch eine „Vererbungsbeziehung“ sichergestellt wird: Alle „Relay Plugins“ werden von derselben abstrakten Basisklasse abgeleitet.

Weitaus umfangreicher stellt sich die Problematik dar, wenn man aus Sicht eines „Relay Plugins“ auf die Schnittstelle zum transportorientierten Kern hin schaut. Hier müssen im Sinne eines Übertragungsdienstes sämtliche Funktionen angeboten werden, welche benötigt werden, um zwei „Relay Plugin“-Instanzen miteinander assoziieren zu können. Zuallererst ist daher zu klären, welche Anforderungen seitens eines „Relay Plugins“ an den Übertragungsmechanismus des REACH-Kerns bestehen.

Diese Aspekte werden ausführlich im Rahmen des „Transportkonzepts“ in Kapitel 5 beleuchtet. Hier an dieser Stelle soll vorab ein Überblick geschaffen werden, welche die nachfolgende Vorstellung der einzelnen „Relay Plugins“ erleichtert.

- Es werden Transportkanäle zum Austausch der abgefangenen Nutzdaten zwischen jeweils zwei miteinander assoziierten „Relay Plugin“-Instanzen benötigt. Die erste Instanz ist diejenige auf dem mobilem Endgerät, die zweite existiert auf einem REACH-Server. Diese Kanäle werden bei REACH als *logische Verbindungen* bezeichnet.

- Es werden zwei Arten von Daten unterschieden. In Anlehnung an die beiden Transportschichtprotokolle TCP und UDP werden stromorientierte und paketorientierte *logische Verbindungen* unterschieden. Erstere erlauben die gesicherte Übertragung eines byteorientierten Datenstroms, während letztere die ungesicherte Übertragung von Paketen ermöglicht.

- Je nach Dienst kann es notwendig sein, *logische Verbindungen* bereits bei der Annahme genauer klassifizieren zu können, also noch bevor Nutzdaten aus dieser entnommen werden. Bei den für die SIP-basierte Sprachübertragung gedachten „Relay Plugins“ müssen beispielsweise *logische Verbindung* für Sprachdaten von *logischen Verbindungen* für Signalisierungsdaten unterschieden werden. Daher wird noch die Möglichkeit vorgesehen, während des Aufbaus einen „Signalisierungsblock“ mit „out-of-band“-Signalisierungsinformationen zu übergeben.

- *Logische Verbindungen* werden seitens einer beliebigen „Relay Plugin“-Instanz angefordert und dann von einer Partnerinstanz angenommen. Die anfordernde „Relay Plugin“-Instanz kann dabei sowohl auf dem REACH-Client als auch auf dem REACH-Server residieren. Damit sind aus Sicht einer jeden Instanz ausgehende und eingehende *logische Verbindungen* möglich.

Dies wirft allerdings die Frage auf, *wohin* eine an einem REACH-Server angeforderte *logische Verbindung* geleitet werden soll. Sollte bisher kein REACH-Client angemeldet sein, kann kein Ziel benannt werden. Wären dagegen bereits mehrere REACH-Clients angemeldet, wäre eine Zuordnung nicht mehr eindeutig möglich. Daraus folgt, dass eine übergeordnete Sitzung benötigt wird.

Definition 4 (Sitzung) *Jede clientseitige „Relay Plugin“-Instanz kann sich mit einem oder mehreren serverseitigen „Relay Plugin“-Instanzen assoziieren. Da es serverseitig pro „Relay Plugin“ nur eine einzige Instanz geben kann, ist folgende Aussage zutreffend: Jede Instanz eines „Relay Plugins“ kann mit einer Vielzahl an Partnerinstanzen assoziiert sein, also eine Sitzung aufgebaut haben.*

Eine solche Sitzung wird immer seitens einer „Relay Plugin“-Instanz auf einem REACH-Client aufgebaut und schließt eine Partnerinstanz auf einem REACH-Server mit ein. Hierbei wird initial unter Zuhilfenahme eines Dienstsuchemechanismus ein geeigneter REACH-Server gesucht, welcher ebenfalls das entsprechende „Relay Plugin“ instantiiert haben muss. Zwischen beiden Instanzen wird eine Sitzung erzeugt, welche eine Vielzahl an *logischen Verbindungen* umfassen kann. Eine aufgebaute Sitzung ist dauerhaft an einen REACH-Server gebunden und kann niemals auf einen anderen REACH-Server migriert werden. Der Grund hierfür ist, dass sich aus Sicht der Server im Internet die IP-Adresse des Kommunikationspartners (des „Ankerpunkts“) nicht ändern darf.

Innerhalb von REACH werden diese Sitzungen durch so genannte *Linkbündel* repräsentiert. Der Begriff rührt daher, dass dieses Konstrukt als ein „Bündel“ gesehen werden kann, an dessen Endpunkten bidirektional *logische Verbindungen* angefordert und angenommen werden können.

Eine „Relay Plugin“-Instanz besitzt immer maximal *ein Linkbündel* zu einer bestimmten Partnerinstanz, kann aber insgesamt eine Vielzahl an *Linkbündeln* zu diversen Partnerinstanzen verwalten. Dies ist die Voraussetzung für Mechanismen wie Lastverteilung und die ortsabhängige Auswahl von REACH-Proxyservern, welche im Rahmen des „Suchkonzepts“ in Kapitel 7 diskutiert werden.

Definition 5 (Linkbündel) *Ein Linkbündel verkörpert eine Sitzung zwischen einer clientseitigen „Relay Plugin“-Instanz und einer serverseitigen „Relay Plugin“-Instanz. Eine*

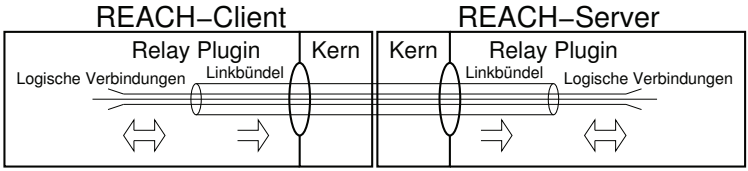

Abbildung 4.2: Zwei miteinander assoziierte „Relay Plugin“-Instanzen sind durch genau ein *Linkbündel* miteinander assoziiert. Diese werden *immer* ausgehend vom mobilen Gerät aus aufgebaut und erlauben schlussendlich die beidseitige Anforderung einer Vielzahl an *logischen Verbindungen*.

Sitzung bedient sich genau eines Linkbündels. Jede Instanz kann jedoch eine Vielzahl an Linkbündeln verwalten.

Definition 6 (Logische Verbindung) *An den Endpunkten eines jeden Linkbündels können beidseitig logische Verbindungen angefordert und angenommen werden. Diese stellen den eigentlichen Übertragungsdienst bereit. REACH unterscheidet stromorientierte und paketorientierte logische Verbindungen. Jedes Linkbündel kann eine Vielzahl beider Typen führen. Welche Arten und wie viele logische Verbindungen in welcher Richtung konkret angefordert werden, hängt lediglich vom jeweiligen „Relay Plugin“ ab.*

4.3.1.3 Konfiguration

Die Konfiguration bezüglich der Dienste, also der zu instantiierenden „Relay Plugins“, ist sowohl am REACH-Client als auch am REACH-Server in der Datei `relayplugins.xml` hinterlegt. Diese Datei wird beim Programmstart eingelesen. Sie dient in erster Linie der persistenten Speicherung der vom Nutzer gewählten Zusammenstellung an „Relay Plugin“-Instanzen.

Die Konfigurationsdatei weist eine definierte Struktur auf. Sie ist sowohl menschen- als auch maschinenlesbar und wurde entworfen, um die folgenden Anforderungen zu erfüllen:

- Sie muss die Dateinamen aller „Relay Plugin“-Bibliotheken erfassen, damit diese dynamisch geladen werden können.
- Clientseitig kann jede „Relay Plugin“-Bibliothek zur Erzeugung einer Vielzahl von Instanzen benutzt werden.
- Serverseitig wird pro „Relay Plugin“-Bibliotheken immer nur genau eine Instanz erzeugt.[4]

[4]Das liegt in der beschränkten Implementierung des Dienstefindungsmechanismus begründet. Dieser

- Instanzen können parametriert werden. Parameter werden in Form von Schlüssel-Werte-Paaren abgelegt.
- Es kann notwendig sein, die Freiheit des Dienstefindungsmechanismus einzuschränken. Diese Möglichkeit wird von mehreren Diensten in Anspruch genommen.

```
<RelayPlugins>
  <RelayPlugin loadlibrary="LIBRARYNAME">
    <Instance>
      <Configuration>
        <Item key="KEY" value="VALUE" />
        <!-- ... weitere Schlüssel-Werte-Paare -->
      </Configuration>
      <Discovery>
        <Item key="KEY" value="VALUE" />
        <!-- ... weitere Schlüssel-Werte-Paare -->
      </Discovery>
    </Instance>
    <!-- Nur am REACH-Client: Weitere Instanzen -->
  </RelayPlugin>
  <!-- ... weitere RelayPlugins -->
</RelayPlugins>
```

Abbildung 4.3: Die Konfigurationsdatei `relayplugins.xml` definiert die zu instantiierenden „Relay Plugins“. Es soll die allgemeine Struktur verdeutlicht werden.

Für die Konfigurationsdateien wird die „Extended Markup Language“ (XML) benutzt. Die resultierende XML-Struktur entspricht dabei dem Schema aus Abbildung 4.3. Sie besteht aus einer äußeren `<RelayPlugins>`-Schachtelung, welche eine Menge an `<RelayPlugin>`-Unterschachtelungen enthält. Jede ist für ein bestimmtes „Relay Plugin“ zuständig und definiert in erster Linie den Dateinamen der zu ladenden Bibliothek (Attribut `loadlibrary`) sowie eine Menge an zu erstellenden Instanzen (Schachtelungen `<Instance>`). Für jede Instanz können Parameter in Form zweier Listen an Schlüssel-Werte-Paaren hinterlegt werden. Diese sind frei definierbar und werden durch das jeweilige „Relay Plugin“ vorgegeben.

Die Angabe des `<Configuration>`-Blocks ist optional, da speziell am Server oftmals keine Parameter für die Instanzen benötigt werden. Zusätzlich zum Konfigurationsblock

bietet momentan beim Aufbau eines *Linkbündels* keine Möglichkeit an, mehrere Instanzen desselben „Relay Plugins“ zu unterscheiden. Eine solche Unterscheidung war bislang bei keinem Dienst sinnvoll oder gar erforderlich.

kann noch ein Parameterblock zur Einflussnahme auf die Dienstsuche (Schachtelung `<Discovery>`) abgelegt werden. Diese werden aber nicht an die betroffenen Instanzen übergeben, sondern bereits durch den transportorientierten Kern ausgewertet. In manchen Fällen ist es notwendig, dass ein bestimmter REACH-Proxyserver für die Dienstserbringung festgelegt wird. So kann beispielsweise ein „Sprungbrett“ in ein „geschützte Netze“ geschaffen werden. Bislang wurden hierfür zwei Parameter definiert:

`key="transition":` Im `value`-Feld kann durch Angabe von `"true"` oder `"false"` festgelegt werden, ob eine Instanz einen einmal gewählten REACH-Server durch einen anderen ersetzen darf. Die Defaulteinstellung lautet `"true"` und erlaubt damit mehrere Sitzungen pro Instanz. Ein „Relay Plugin“ kann diesen Parameter allerdings eigenmächtig auf `"false"` setzen, wobei auch ein manuelles Aktivieren dann nicht mehr möglich ist.

`key="use reachserver":` Bei Angabe dieses Parameters nutzt eine Instanz ausschließlich den im `value`-Feld referenzierten REACH-Server, unter vollständiger Umgehung des Dienstsuchemechanismus. Sollte sich herausstellen, dass der referenzierte Server nicht in der Lage ist den Dienst zu erbringen, versagt die hiesige Instanz ihren Dienst. Anzumerken ist, dass bei Angabe dieses Parameters implizit der Parameter `"transition"` unveränderlich auf `"false"` gesetzt wird.

Die erläuterte Struktur der Konfigurationsdatei gilt sowohl für den REACH-Client als auch für den REACH-Server. Bei Letzterem darf es allerdings pro „Relay Plugin“ nur eine einzige Instanz erzeugt werden. Zudem wird serverseitig die `<Discovery>`-Schachtelung nicht ausgewertet.

4.3.2 Realisierte „Relay Plugins“

Da für jeden zu unterstützenden Dienst sowohl am REACH-Client als auch am REACH-Server bestimmte Funktionen hinterlegt werden müssen, treten „Relay Plugins“ immer paarweise auf. Zu jedem Dienst gibt es ein clientseitiges und ein serverseitiges „Relay Plugin“. Damit ein Dienst erbracht werden kann, muss sich eine clientseitige „Relay Plugin“-Instanz mit wenigstens einer serverseitigen „Relay Plugin“-Instanz desselben Typs assoziiert haben.

Im Folgenden werden die bereits für REACH entwickelten „Relay Plugins“ vorgestellt und die sich dahinter verbergenden Mechanismen im Detail beleuchtet.

4.3.2.1 SOCKSv4 und SOCKSv5

SOCKS wurde ursprünglich dazu entwickelt, den Datenverkehr an Firewalls feingranular steuern zu können: Eine restriktiv eingestellte Firewall würde dafür sämtlichen Datenverkehr sperren, aber die indirekte Passage über einen SOCKS-Proxyserver gestatten (welcher praktisch parallel zur Firewall angeordnet ist und diese somit umgehen kann). Auf diesem SOCKS-Proxyserver können dann umfangreiche Regelsätze pro Nutzer und Anwendung hinterlegt werden. Ein solcher Einsatzfall – allerdings ohne die genannte Firewall – wird in Abbildung 4.4 dargestellt.

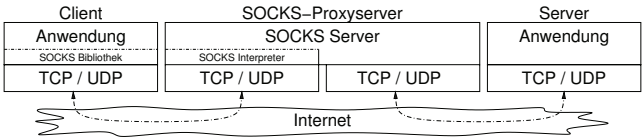

Abbildung 4.4: In diesem Anwendungsfall, für den SOCKS ursprünglich entwickelt worden ist, werden TCP-Verbindungen und UDP-Assoziationen seitens der Anwendungen nur noch *indirekt* über den SOCKS-Proxyserver abgewickelt.

Die relevante Tatsache hierbei ist, dass die Anwendungen dann sämtliche Daten über den Proxyserver leiten, und zwar unter Zuhilfenahme der Protokolle SOCKSv4 oder SOCKSv5. SOCKSv5 ist der Nachfolger von SOCKSv4 und erweitert dieses um die Unterstützung von UDP sowie um eine umfangreiches Angebot an Authentifizierungsschemata.

Abbildung 4.5 illustriert den Ablauf beim Aufbau eines „indirekten Sockets“, wenn SOCKSv4 oder SOCKSv5 zum Einsatz kommen. Die Spalte links im Ablaufdiagramm repräsentiert die Anwendung (den SOCKS-Client), welche mit dem in der mittigen Spalte angeordneten SOCKS-Proxyserver unter Zuhilfenahme des SOCKS-Protokolls kommuniziert. Der SOCKS-Proxyserver kann die Server im Internet (rechte Spalte) erreichen, was den Anwendungen auf dem Client unter Umständen nicht „direkt“ gestattet ist.

SOCKS-Protokollfunktionen: Bei Einsatz von SOCKSv5 baut der Client TCP-Steuerverbindungen zum SOCKS-Proxyserver auf. Diese Verbindungen werden authentifiziert, wobei SOCKSv5 hierfür eine Vielzahl an Möglichkeiten vorsieht. Der Client schickt hierfür eine Liste der von ihm unterstützen Mechanismen zum SOCKS-Proxyserver. Der Proxyserver ignoriert alle eventuell nicht unterstützen Mechanismen und filtert zudem alle nicht erwünschten Varianten (`NO AUTHENTICATION REQUIRED` stellt je nach gewählter Sicherheitsstufe eine solche unerwünschte Variante dar). Aus der Menge der übrig

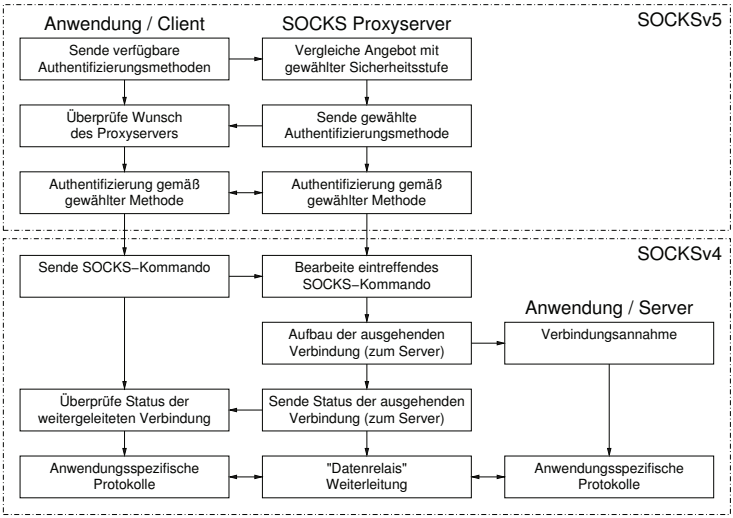

Abbildung 4.5: Der Lebenszyklus einer SOCKSv5-Sitzung ist gegenüber einer SOCKSv4-Sitzung um optionale Authentifizierungsschemata erweitert worden (nach [Sock]).

gebliebenen Varianten wählt der Proxyserver eine aus und kommuniziert seine Wahl an den Client. Konnten sich beide Instanzen auf ein Schema einigen, beginnt der eigentliche Authentifizierungsvorgang. Geht bei einem dieser Schritte etwas schief, beendet der Proxyserver die Steuerverbindung.

Ab dieser Stelle verhalten sich SOCKSv4 und SOCKSv5 (bei erfolgreich durchlaufener Authentifizierung) ähnlich. Der Client hat jetzt die Möglichkeit, ein so genanntes „SOCKS-Kommando“ an den SOCKS-Proxyserver zu versenden. Es werden folgende drei Protokollfunktionen unterschieden:

- `CONNECT`

  Mit Hilfe des `CONNECT`-Kommandos werden *ausgehende* TCP-Verbindungen geschaltet. Der Client übermittelt dazu die gewünschte IP-Adresse und TCP-Portnummer an den SOCKS-Proxyserver, welcher dann seinerseits die Verbindung zum gewünschten Zielsystem aufbaut. Konnte die ausgehende Verbindung etabliert werden, schickt der Proxyserver eine positive Bestätigung zum Client und schaltet danach in einen transparenten Weiterleitungsmodus. Alle Daten vom Client zum Zielsystem und umgekehrt werden so lange „umgeschaufelt“, bis einer der beiden Kommunikationspartner die TCP-Verbindung schließt. Anzumerken ist, dass die

Bestätigung des SOCKS-Proxyservers im positiven Fall noch die IP-Adresse und Portnummer enthält, mit welcher die ausgehende TCP-Verbindung am Proxyserver sichtbar ist.

- `BIND`
  Das `BIND`-Kommando ist für *eingehende* TCP-Verbindungen verantwortlich. Dazu wird am SOCKS-Proxyserver ein TCP-Port „gebunden“, um auf diesem im Anschluss eine TCP-Verbindung annehmen zu können. Der Client erhält auf ein `BIND`-Kommando zwei Antworten: In der ersten Antwort wird ihm mitgeteilt, auf welcher IP-Adresse und TCP-Portnummer am Proxyserver auf die eingehende TCP-Verbindung gewartet wird. Diese Information muss vom Client dahingehend benutzt werden, dass diese Adresse über eine bereits bestehende Kommunikationsbeziehung an den Kommunikationspartner im Internet gesendet werden muss. Diese Partnerinstanz kann daraufhin eine Verbindung zu der spezifizierten Adresse aufbauen, was den SOCKS-Proxyserver dazu veranlasst, eine zweite Bestätigung für den Client zu formulieren: Sie signalisiert die erfolgreiche Verbindungsannahme und benennt die IP-Adresse und Portnummer des Initiators der eingehenden Verbindung. Anschließend wird wiederum in einen transparenten Datenweiterleitungsmodus geschaltet, so wie es beim `CONNECT`-Kommando bereits erläutert worden ist.

  Da die vom Client aufgebaute Steuerverbindung nach erfolgreichem `BIND`-Kommando nur noch zum Datenaustausch mit dem Zielsystem verwendet wird, ist die Annahme weiterer TCP-Verbindungen an der reservierten Annahmestelle nicht mehr möglich. SOCKS unterstützt lediglich die Annahme einzelner „Begleitverbindungen“, was zum Beispiel für FTP benötigt wird. Es ist nicht möglich, Annahmestellen zu schalten, welche mehr als nur eine einzige TCP-Verbindung annehmen und „nach innen“ leiten. Eine solche Funktion wäre aber notwendig, um Serverdienste auf dem Client anbieten zu können.

- `UDP ASSOCIATE`
  Mit Hilfe dieser Protokollfunktion steht Unterstützung für das verbindungslose UDP bereit. Hierfür erstellt der Client eine Annahmestelle für UDP-Pakete, dessen Adresse er mit Hilfe des `UDP ASSOCIATE`-Kommandos dem SOCKS-Proxyserver mitteilt. Dieser erstellt daraufhin einen so genannten „UDP Relay Server“, welcher clientseitig wiederum eine Annahmestelle für UDP-Pakete besitzt. Dessen Adresse wird dem Client in einer Bestätigungsnachricht über die TCP-Steuerverbindung mitgeteilt. Client und Proxyserver können sich nun UDP-Pakete zusenden, wobei jedes UDP-Paket einen SOCKS-Protokollkopf (Header) aufweisen muss. In die-

sem Header steht pro UDP-Paket verzeichnet, wohin das Paket vom Proxyserver gesendet werden soll bzw. von welchem Kommunikationspartner ein am SOCKS-Proxyserver empfangenes UDP-Paket stammt.

Nach Übertragung der Bestätigung wird die TCP-Steuerverbindung nicht länger zur Datenübertragung herangezogen. Sie dient von nun an lediglich dazu, den „UDP Relay Server“ am SOCKS-Proxyserver zu erhalten. Dieser bleibt solange bestehen, bis der Client die zugehörige TCP-Steuerverbindung schließt.

SOCKSv4 unterstützt keine UDP-Assoziationen und kann zudem nur mit IPv4-Adressen in Form 32 Bit langer Datenfelder arbeiten. SOCKSv5 dagegen kann alternativ auch mit IPv6-Adressen und sogar mit DNS-Namen umgehen. In letzterem Fall würde die Namensauflösung erst vom SOCKS-Proxyserver durchgeführt werden, und am Client bräuchte kein DNS-Server bekannt zu sein.

Der Aufbau der Nachrichten von SOCKSv4 und SOCKSv5 ist unterschiedlich. Da allerdings in jeder SOCKS-Protokolldateneinheit das erste Byte für die Versionsnummer vorgesehen ist, kann der SOCKS-Proxyserver erkennen, nach welchem Schema die Nachrichten einer jeden Steuerverbindung aufgebaut sein sollen. Es stellt also kein Problem dar, eine SOCKS-Annahmestelle zu implementieren, welche sowohl SOCKSv4 als auch SOCKSv5 unterstützt.

Einsatz von SOCKS in der Praxis: Es lässt sich erkennen, dass durch den Einsatz von SOCKS ein breites Spektrum an Anwendungen abgedeckt werden kann. Ausgehende TCP-Verbindungen werden unterstützt, für eingehende TCP-Verbindungen müssen allerdings gravierende Einschränkungen in Kauf genommen werden. So ist es nicht möglich, mehr als eine einzige TCP-Verbindung pro am Proxyserver reservierter Portnummer anzunehmen und deren Inhalt weiterzuleiten. Mit SOCKSv5 steht zudem eine umfassende Unterstützung für den indirekten Versand von UDP-Paketen bereit. Was nicht angeboten wird, ist beispielsweise die Unterstützung von ICMP oder anderer Transportschichtprotokolle wie SCTP oder DCCP. Die für die meisten Anwendungen relevanten Datenströme können aber durch SOCKS gehandhabt werden. So wird SOCKS auch für die Anonymisierungsprogramme TOR [Tor0] (Abk. für „The Onion Router“) und I2P [I2P0] (Abk. für „Invisible Internet Project“) eingesetzt, um die Datenströme der Anwendungen durch den jeweiligen Anonymisierer zu leiten. Auch zur Realisierung eines IPv4-zu-IPv6-Gateways konnte SOCKSv5 bereits erfolgreich herangezogen werden (RFC 3089, [Kita01]).

Es muss allerdings geklärt werden, wie man die Anwendungen dazu bringen kann, nur noch indirekt mit Hilfe des SOCKS-Protokolls zu kommunizieren. Glücklicherweise

gibt es Anwendungen, die bereits „von Haus aus“ diese Möglichkeit anbieten. Bekannte Beispiele sind der Webbrowser Mozilla Firefox [Fire] sowie die Desktopumgebung KDE [KDE0], welche beide in ihren Einstellungsdialogen den Einsatz von SOCKS vorsehen.

Für Anwendungen, die eine solche Möglichkeit nicht vorsehen, steht der Mechanismus der „Socksifizierung“[5] zur Verfügung. Hierbei wird durch Beeinflussung des „Linkers“ beim Start der Anwendung die „SOCKS Library“ statt der ursprünglich vorgesehenen „Sockets Library“ geladen. Die in dieser Ersatzbibliothek hinterlegten Funktionen besitzen dasselbe „Application Programming Interface“ (API) wie die ersetzten Socket-Routinen, sodass die Anwendung ab diesem Moment nur noch SOCKS-Funktionen aufruft und somit indirekt über den SOCKS-Proxyserver kommuniziert. Der Austausch der Kommunikationsbibliothek wird durch die Anwendungen nicht bemerkt.

Hierfür bietet sich unter GNU/Linux der Einsatz des Programms `socksify` aus dem Programmpaket Dante [Dant] an, während unter Microsoft Windows die Software SocksCap[6] diese Funktionalität bereitstellt. Diese Vorgehensweise versagt allerdings, wenn ein Programm „statisch gelinkt“ wurde, und damit ein nachträgliches Austauschen der Kommunikationsbibliothek durch Beeinflussung des Linkers nicht mehr möglich ist. Sie versagt außerdem, wenn der Start der betroffenen Anwendung für den Nutzer unerreichbar „in den Tiefen des Betriebssystems“ abgewickelt wird. Dies ist beispielsweise beim Internet Explorer unter Microsoft Windows der Fall, da dessen Komponenten bereits zum Systemstart initialisiert werden, noch lange bevor der Nutzer die Möglichkeit hat, eine Software wie SocksCap zu starten. Dann ist es nicht möglich, die Datenströme der betroffenen Anwendung mit Hilfe von SOCKS über einen Proxyserver zu leiten.

Für das Programmpaket Dante sieht die Konfiguration der clientseitigen SOCKS-Bibliothek so aus wie in Abbildung 4.6 dargestellt. Exakt diese Konfigurationsdatei kommt auch im Demonstrator zum Einsatz. Ohne deren Inhalt tiefgreifend vorstellen zu wollen, lassen sich dennoch wichtige Details erkennen, die eine gesonderte Erwähnung verdienen. Zum einen wird hier die Art der Namensauflösung festgelegt. SOCKSv5 bietet es an, die Namensauflösung wie üblich über UDP seitens des mobilen Endgeräts durchzuführen, oder alternativ statt aufgelöster IP-Adressen den DNS-Namen des gewünschten Zielsystems im Klartext in die SOCKS-Kommandos einzubetten. In diesem Beispiel wird explizit eine Auflösung durch den SOCKS-Client unter Zuhilfenahme von UDP festgelegt. Damit die Namensauflösung nicht ebenfalls durch SOCKS abgefangen wird, enthält die Konfigurationsdatei hierfür eine Ausnahmeregel. Diese Ausnahmeregel wird durch

[5] Socksifizierung: Freie Übersetzung des englischen Kunstworts „to socksify“, was sinngemäß „mit SOCKS-Funktionalität ausstatten“ bedeutet.

[6] Mittlerweile abgekündigt und ohne Homepage.

```
resolveprotocol: udp

route {
  # Anfragen an Nameserver immer direkt zustellen!
  from: 0.0.0.0/0 to: 0.0.0.0/0 port = domain via: direct
}

route {
  # Anfragen an Localhost immer direkt zustellen
  from: 0.0.0.0/0   to: 127.0.0.0/8  via: direct
  command: connect udpassociate
}

route {
  from: 0.0.0.0/0 to: 0.0.0.0/0 via: 127.0.0.1 port = 1080
  proxyprotocol: socks_v4 socks_v5
  command: bind bindreply connect udpassociate udpreply
}

route {
  from: 0.0.0.0/0 to: . via: 127.0.0.1 port = 1080
  proxyprotocol: socks_v4 socks_v5
  command: bind bindreply connect udpassociate udpreply
}
```

Abbildung 4.6: Die Konfigurationsdatei `/etc/socks/socks.conf` parametriert die SOCKS-Clientbibliothek des DANTE-Pakets.

den ersten `route`-Block definiert, welche als Filterkriterium die Zielportnummer `domain` verwendet. Das Alias `domain` wird unter Zuhilfenahme der Datei `/etc/services` in die Portnummer `53` aufgelöst. Der zweite `route`-Block definiert eine weitere Ausnahmeregel, welche sämtlichen lokalen TCP- und UDP-Datenverkehr ausschließt. Nur so wird gewährleistet, dass weiterhin lokale Sockets zur Interprozesskommunikation funktionieren können. Die letzten beiden Blöcke erfassen sämtlichen TCP- und UDP-basierten Datenverkehr, der noch übrig bleibt, und leiten diesen unter Anwendung von SOCKSv4 oder SOCKSv5 über den SOCKS-Proxyserver.

SOCKS mit Blick auf eine Mobilitätsunterstützung: Für alle ursprünglichen Einsatzszenarien von SOCKS gilt, dass die Modifikation der Anwendungen zur Erreichung der „indirekten Konnektivität“ nicht praxistauglich ist. Das macht SOCKS für den Einsatz in REACH interessant, da hier dieselbe Randbedingung gilt: Das Kommunikations-

verhalten der Anwendungen soll ohne deren Modifikation beeinflusst werden, mit Blick auf die Bereitstellung handoverresistenter Kommunikationsmechanismen.

Ein Lösungsvorschlag, welcher diese Idee umsetzt, wurde vom Autor in [Ever04] vorgestellt. Wenn man sich nochmals Abbildung 4.4 auf Seite 49 anschaut, erkennt man, dass durch den Einsatz von SOCKS die Kommunikationsbeziehungen jeweils in zwei eigenständige Teilstrecken aufgebrochen werden. Im Zentrum steht der SOCKS-Proxyserver, welche beide Teilstrecken miteinander verknüpft. Keine der beiden Teilstrecken darf durch Mobilitätsereignisse negativ beeinflusst werden, da sowohl die Anwendungen (erste Teilstrecke) als auch die Server im Internet (zweite Teilstrecke) damit nicht umgehen können.

So wie SOCKS regulär eingesetzt wird, ist bereits die Teilstrecke zwischen SOCKS-Proxyserver und den Servern im Internet nicht durch Mobilitätsereignisse betroffen. Die Teilstrecke am Client involviert jedoch die Luftschnittstelle, was während Handovererereignissen zu Problemen führen wird. Würde man den SOCKS-Proxyserver stattdessen auf dem mobilen Endgerät installieren, wären die Anwendungen effektiv vor der Luftschnittstelle abgeschottet, jedoch wären dann die Verbindungen der zweiten Teilstrecke in Richtung der Server im Internet den negativen Auswirkungen von Mobilität ausgesetzt.

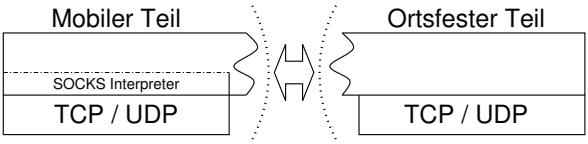

Abbildung 4.7: Eine bereits in [Ever04] geäußerte Idee sieht die Teilung eines SOCKS-Proxyservers in einen „mobilen Teil“ und einen „ortsfesten Teil“ vor, um sowohl die Clients als auch die Server vor der Luftschnittstelle abschotten zu können.

Um beide Teilstrecken vor der Luftschnittstelle abzuschotten, *könnte* ein SOCKS-Proxyserver wie in Abbildung 4.7 gezeigt in zwei Teile getrennt werden. Der „mobile Teil“ wird auf dem mobilen Endgerät installiert und von den dort laufenden Anwendungen als SOCKS-Proxyserver verwendet. Der Rest bildet den „ortsfesten Teil“, welcher auf dem Proxyserver installiert wird. Er schottet seinerseits die Server vor der Luftschnittstelle ab und bietet einen „Ankerpunkt“ für sämtlichen Datenverkehr. Mobiler und ortsfesten Teil müssen durch einen handoverresistenten Übertragungsmechanismus miteinander verbunden werden, sodass beide Teile wieder eine funktionale Einheit bilden, welche einem vollwertigen SOCKS-Proxyserver ähnelt.

Dieser Ansatz wurde vom Autor in [Ever04] verworfen, da es sich als zu kompliziert herausstellte, einen bereits bestehenden SOCKS-Proxyserver wie den aus dem DANTE-Paket [Dant] zu nehmen und zu teilen. Problematisch war, dass es sich hierbei um ein kompliziertes und umfangreiches Programm handelte, welches durch seine gegebene Struktur eine völlige Umstrukturierung erfordert hätte. Ein solches Vorgehen würde neue Probleme mit sich bringen: der dabei entstandene Quelltext hätte mit der Ursprungsversion nicht mehr viel zu tun gehabt, und eventuelle Verbesserungen oder Fehlerbereinigungen in neueren Versionen des DANTE-Pakets hätten nicht länger eingebunden werden können. Gerade durch das vielfältige Angebot an Authentifizierungsmethoden und Konfigurationsmöglichkeiten wäre die Pflege der Software extrem erschwert gewesen. Daher wurde eine Alternative gesucht, welche ohne eine Trennung der Proxysoftware auskommen konnte.

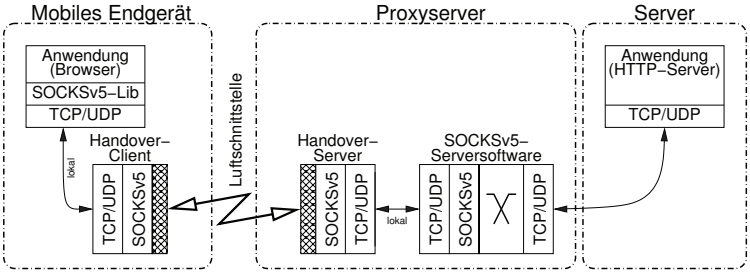

Abbildung 4.8: In [Ever04] wurde eine SOCKSv5-basierte Mobilitätsunterstützung vorgeschlagen, welche ohne die Trennung einer SOCKS-Proxysoftware auskam. Stattdessen kamen ein „Handover-Client“ und ein „Handover-Server“ zum Einsatz, welche durch einen mit Blick auf Mobilitätsereignisse entwickelten Übertragungsmechanismus (die zwei gerasterten Blöcke) miteinander in Kontakt standen. Weiterhin war noch der nicht modifizierte vollwertige SOCKSv5-Proxyserver aus dem DANTE-Programmpaket erforderlich.

Dabei erwies sich eine Struktur wie in Abbildung 4.8 dargestellt als vielversprechend. Sie konnte schlussendlich in Form eines Prototypen implementiert werden. Dabei kam weiterhin die SOCKS-Proxysoftware DANTE zum Einsatz, welche allerdings weder modifiziert noch aufgeteilt werden brauchte. Somit standen sämtliche SOCKS-Protokollfunktionen zur Verfügung, und es kam seitens des Proxyservers zu keinem Wartungsproblem. Für die Isolation der Luftschnittstelle kamen zwei vom Autor einzig für diesen Zweck erstellte Programme zum Einsatz: der so genannte „Handover-Client“ und der „Handover-Server“. Ersterer wird auf dem mobilen Endgerät installiert und bietet eine SOCKSv5-

fähige Annahmestelle, ohne jedoch einen vollwertigen Proxyserver zu implementieren. Er terminiert damit die von den lokalen Anwendungrn kommenden SOCKS-Kommunikationskanäle und ist in der Lage, die an der SOCKSv5-Annahmestelle angenommenen TCP-Verbindungen und UDP-Assoziationen zu verwalten und deren Daten mittels eines handoverresistenten Übertragungsmechabnismus über die Luftschnittstelle an den Handover-Server zu versenden. Der Handover-Server ist zusammen mit der DANTE SOCKS-Proxyserversoftware auf dem Proxyserver installiert. Der Handover-Server tritt seinerseit als SOCKS-Client gegenüber dem SOCKS-Proxyserver auf, wobei auch hier die Luftschnittstelle nicht mehr „sichtbar“ ist.

Diese Lösung konnte umgesetzt werden und funktionierte auch, zeigte jedoch einige Schwachpunkte. Einerseits wurde neben Handover-Client und Handover-Server noch der SOCKS-Proxyserver benötigt, was die Anzahl der zu pflegenden Komponenten und damit die Komplexität erhöhte. Zudem stellte sich heraus, dass gar nicht alle angebotenen Authentifizierungsmechanismen und Verschlüsselungsmechanismen eingesetzt werden dürfen. Immerhin müssen Handover-Client und -Server den SOCKS-Protokolldatenfluss mitlesen können, um zwischen eingehenden und ausgehenden TCP-Verbindungen unterscheiden zu können und um gegebenenfalls eigene „UDP Relay Server“ zu instantiieren. Es musste sowohl auf Authentifizierung als auch auf Verschlüsselung verzichtet werden. Damit verschwand allerdings auch einer der Gründe, welche für den Einsatz einer vollwertigen SOCKS-Proxysoftware sprachen.

Die beiden größten Einschränkung resultierten jedoch aus der gewählten Form der Umsetzung: So wurden abgefangene UDP-Pakete zwischen Handover-Client und Handover-Server aus Gründen des Implementierungsaufwandes intern ebenfalls mit Hilfe von TCP transportiert. UDP ist jedoch ein ungesichert funktionierendes, nicht flussgesteuertes Protokoll. Tunnelt man UDP über TCP, kommt es zu einem unerwünschten Verhalten. Beispielsweise bei Sprachkommunikation würde sich eine erhöhte Latenzzeit und ein stockendes Übertragungverhalten bemerkbar machen. Außerdem waren beide Programme derart monolithisch aufgebaut, dass die zu Grunde liegenden Sicherungsschemata nur für SOCKS eingesetzt werden konnten, und nachträgliche Erweiterungen nicht möglich waren. So bot es sich an, im Zuge der Weiterentwicklung die Struktur komplett zu überdenken und schlussendlich die SOCKS-Funktionen in Form von „Relay Plugins“ vom transportorientierten Kern zu entkoppeln.

Realisierung für REACH: Für die Implementierung der „Relay Plugins“ für REACH wurde wiederum nicht der Weg beschritten, einen bestehenden SOCKS-Proxyserver zu analysieren und in zwei Teile zu trennen. Stattdessen wurden beide Komponenten von Grund auf neu erstellt. Ziel war es, aus logischer Sicht einen SOCKS-Proxyserver zu er-

halten, welcher aus zwei getrennten Teilen besteht, welche aber handoverresistent miteinander verbunden sind. Für den handoversicheren Transport zwischen beiden Komponenten konnte direkt auf den transportorientierten Kern von REACH mit seinen *logischen Verbindungen* aufgesetzt werden, wodurch sich für die „Relay Plugins“ eine klare und problemorientierte Struktur ergab. Zudem bietet REACH auch paketorientierte *logische Verbindungen* an, was für die Implementierung der Protokollfunktion `UDP ASSOCIATE` wichtig war. So werden abgefangene UDP-Pakete zwischen REACH-Client und REACH-Server auch paketorientiert übertragen, und nicht wie beim veralteten Handover-Client/-Server mittels TCP.

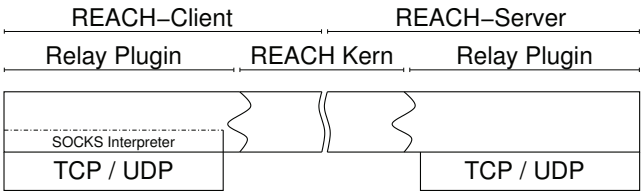

Abbildung 4.9: Für REACH wurde ein zweigeteilter SOCKS-Proxyserver implementiert. Beide Teile sind in Form von „Relay Plugins“ erstellt. Der transportorientierte Kern von REACH implementiert den eigentlichen Datentransport und sorgt schlussendlich dafür, dass zwei „Relay Plugin“-Instanzen zusammen wieder einen SOCKS-Proxyserver ergeben.

Die für REACH umgesetzte Struktur ist in Abbildung 4.9 dargestellt. Sie besteht aus dem REACH-Client und dem REACH-Server, welche miteinander in Kontakt stehen und handoverresistente Kommunikationsbeziehungen ermöglichen. Das für SOCKS relevante clientseitige „Relay Plugin“ bietet eine gemeinsame Annahmestelle für SOCKSv4 und für SOCKSv5 an und implementiert alle drei Protokollfunktionen `CONNECT`, `BIND` und `UDP ASSOCIATE`. Die Menge der angebotenen Authentifizierungsschemata wurde auf die Methoden `NO AUTHENTICATION REQUIRED` und `USERNAME/PASSWORD` (gemäß RFC 1929 [Leec96]) beschränkt. Authentifizierung bezieht sich sowieso nur auf die Kommunikation *am* mobilen Endgerät, da sowohl der SOCKS-Client (die „socksifizierten“ Anwendungen) als auch der SOCKS-Proxyserver (das clientseitige „Relay Plugin“) auf ein und demselben Gerät residieren. Daher reicht es in der Regel aus, auf ein Authentifizierungsschema zu verzichten. Da es sich bei unixartigen Betriebssystemen aber um Mehrbenutzersysteme handelt, wurde wenigstens noch die Möglichkeit vorgesehen, dass sich Anwendungen durch einen Nutzernamen und ein Passwort am „Relay Plugin“ authentifizieren müssen. Auf Verschlüsselung wurde verzichtet, da sie nur auf dem mobilen Endgerät wirksam wäre und somit keine zusätzlich Sicherheit bringen würde.

Eine brauchbare Konfiguration für ein clientseitiges „Relay Plugin“ zeigt Abbildung 4.10. Sie nennt alle Parameter, welche zur Erstellung einer Instanz auf dem mobilen Endgerät benötigt werden.

```
<RelayPlugins>
  <RelayPlugin loadlibrary="librrpc_socks.so">
    <Instance>
      <Configuration>
        <Item key="listener ip" value="127.0.0.1" />
        <Item key="listener tcp port" value="1080" />
        <Item key="listener udp port" value="1080" />
        <Item key="authentication username" value="" />
        <Item key="authentication password" value="" />
      </Configuration>
    </Instance>
  </RelayPlugin>
</RelayPlugins>
```

Abbildung 4.10: Beispielhafte Konfiguration einer „Relay Plugin“-Instanz bezüglich des SOCKS-Dienstes am REACH-Client.

Drei der fünf aufgeführten Parameter definieren die SOCKS-Annahmestelle: Der Parameter `"listener ip"` spezifiziert die IP-Adresse, auf welcher die Steuerverbindungen und bei SOCKSv5 die UDP-Pakete angenommen werden sollen. Im vorliegenden Fall wurde die IP-Adresse des „loopback devices“ (`"127.0.0.1"`) angegeben, wodurch nur Datenströme, welche vom mobilen Gerät selbst stammen, akzeptiert werden. Alternativ kann man auch `"0.0.0.0"` spezifizieren, dann wären beide Annahmestellen auch für externe SOCKS-Clients erreichbar. Die zwei Parameter `"listener tcp port"` und `"listener udp port"` definieren die Portnummern, welche durch die Instanz reserviert werden sollen. Hier werden SOCKS-Steuerverbindungen und UDP-Pakete der SOCKS-Clients angenommen. Schlussendlich können optional ein Nutzername und ein Passwort angegeben werden. Sind beide Felder leer, bietet das „Relay Plugin“ lediglich das Authentifizierungsschema `NO AUTHENTICATION REQUIRED` an. Ist dagegen wenigstens ein Parameter gesetzt, wird nur noch das Schema `USERNAME/PASSWORD` erlaubt.

An der Annahmestelle werden die Protokolle SOCKSv4 und SOCKSv5 unterstützt. Sowohl die indirekten TCP-Verbindungen als auch die UDP-Assoziationen werden handoversicher verwaltet. Es gibt keine Verbindungsabrisse bei langanhaltenden Isolationssituationen.

Am REACH-Server sieht die Konfiguration des „Relay Plugins“ für SOCKS so aus wie in Abbildung 4.11 dargestellt. Hier muss als einziger Parameter die IP-Absenderadresse

```
<RelayPlugins>
  <RelayPlugin loadlibrary="librrps_socks.so">
    <Instance>
      <Configuration>
        <Item key="listener ip" value="141.24.92.184" />
      </Configuration>
    </Instance>
  </RelayPlugin>
</RelayPlugins>
```

Abbildung 4.11: Beispielhafte Konfiguration einer „Relay Plugin“-Instanz bezüglich des SOCKS-Dienstes am REACH-Server.

angegeben werden, mit deren Hilfe die TCP-Verbindungen und UDP-Assoziationen zu den Servern aufgebaut werden sollen. Auf dieser IP-Adresse wird zudem beim `BIND`-Kommando auf eingehende TCP-Verbindungen gewartet. Hierfür dürfen weder `"0.0.0.0"`, `"127.0.0.1"` noch die private IP-Adresse einer LAN-Schnittstelle angegeben werden. Es muss sich um eine global gültige IP-Adresse handeln, welche auch seitens des Internets erreichbar ist. Andernfalls werden eingehende TCP-Verbindungen sowie UDP-Assoziationen nicht funktionieren.

4.3.2.2 „Transparente Proxyserver“

Der Mechanismus der „transparenten Proxyserver“ eröffnet einen bequemen Weg, Daten abzufangen. Er kommt völlig ohne Konfigurationsaufwand seitens der Anwendungen aus, ist aber im Gegenzug lediglich in der Lage, *ausgehende* TCP-Verbindungen zu unterstützen. Eine Unterstützung ausgehender UDP-Assoziationen ist möglich, erfordert allerdings gravierende Einschränkungen, sodass ein transparentes Abfangen von UDP-Paketen nur im Sonderfall der REACH-Box Sinn ergibt.

Funktionsweise: Ein „transparenter Proxyserver“ besteht aus zwei Komponenten: Zuerst werden innerhalb des Paketfilters (bei aktuellen Linux-Kerneln: Netfilter) mit Hilfe einer so genannten „Redirection“ (bei GNU/Linux: Kommando `iptables` und Netfilter-Regel `REDIRECT`) bestimmte Portbereiche ausgewählter Protokolle umgeleitet. Diese Umleitung richtet alle betroffenen Verbindungen auf die zweite Komponente, eine lokal auf demselben Rechner laufende Anwendung, die Proxysoftware. Diese nimmt die umgeleiteten Verbindungen (im Fall von UDP wären es Assoziationen) an und es entsteht ein Verbindungsendpunkt, welcher die Teilstrecke zur initiierenden Anwendung terminiert. Für jeden dieser Verbindungsendpunkt können drei Adressen bestimmt wer-

den: Die IP-Adresse und Portnummer des Initiators, das effektive Ziel des Sockets (die lokale IP-Adresse und Portnummer) sowie die ursprüngliche Zieladresse, welche noch vor der Umleitung galt (ermittelbar mit Hilfe der Socket-Option `SO_ORIGINAL_DST`). Diese „Original Destination“ ist die interessanteste der drei Adressen, da diese verwendet werden kann, um anschließend das ursprünglich gewünschte Zielsystem zu kontaktieren. Die Anwendungen bekommen von alledem nichts mit. Für sie verhält sich der Proxyserver nicht anders als ein Router, welcher lediglich die Datenpakete in Richtung des gewünschten Zielsystems weiterleitet.

Varianten: Der Mechanismus der „transparenten Proxyserver“ ist leider nicht unter Microsoft Windows verfügbar und bleibt damit auf einen Einsatz unter unixartigen Betriebssystemen beschränkt. Seitens des Linux-Kernels gibt es mittlerweile sogar zwei verschiedene Mechanismen, die hierfür eingesetzt werden können. Leider zeigte sich aber, dass der modernere und speziell auf „transparente Proxyserver“ zugeschnittene Mechanismus nicht für eine Mobilitätsunterstützung *auf* dem mobilen Endgerät eingesetzt werden kann, sondern er darf wie im Fall der REACH-Box nur auf einem abgesetzten Gerät aktiviert werden. Daher lohnt sich eine genauere Beleuchtung beider Verfahren.

Es gab bis einschließlich der 2.2er Versionsreihe des Linux-Kernels eine integrierte Unterstützung für „transparente Proxyserver“. Diese wurde in der 2.3er Entwicklungsreihe fallen gelassen, als der Paketfilter umstrukturiert wurde. Mit Beginn der als stabil gekennzeichneten 2.4er Reihe im Januar 2001 stand diese Variante dann nicht länger zur Verfügung. An ihre Stelle trat der bereits angesprochene Mechanismus, welcher auf der Hinterlegung von `REDIRECT`-Netfilter-Regeln in Kombination mit der `SO_ORIGINAL_DST`-Socketoption basiert.

Bevor auf dessen Eigenschaften eingegangen wird, sei angemerkt, dass am 8. Oktober 2008 der so genannte TPROXY-Patch in die Vorabversion des 2.6.28er Linux-Kernels aufgenommen wurde. Dieser stellt nach mehreren Jahren Entwicklung wieder die noch aus Kernel 2.2 bekannten „transparenten Proxyserver“ bereit. Der dazu gehörende „Changelog“-Eintrag (siehe Abbildung 4.12) ist aufschlussreich, da er den für REACH relevanten Unterschied beider Varianten auf den Punkt bringt: Er funktioniert nur für „non-local TCP/UDP traffic“.

In der Dokumentation des Linux-Kernels bezüglich des neueren Mechanismus [Linu] finden sich ebenfalls Hinweise auf die Eigenschaften beider Implementierungen. Der bis dato verwendete auf „Redirection“ basierende Ansatz weist demnach gravierende Einschränkungen auf. Erstens werden hierbei Zieladressen ausgetauscht, es findet also eine Modifikation der betroffenen Pakete statt. Daraus folgt, dass keine Unterstützung für das verbindungslose UDP angeboten werden kann, denn es gibt nach Überschreiben der

```
commit e84392707e10301b93121e1b74e2823db50cdf9e
Author: KOVACS Krisztian <hidden@sch.bme.hu>
Date:   Wed Oct 8 11:35:12 2008 +0200

    netfilter: iptables TPROXY target

    The TPROXY target implements redirection of non-local TCP/UDP traffic to local
    sockets. Additionally, it's possible to manipulate the packet mark if and only
    if a socket has been found. (We need this because we cannot use multiple
    targets in the same iptables rule.)

    Signed-off-by: KOVACS Krisztian <hidden@sch.bme.hu>
    Signed-off-by: Patrick McHardy <kaber@trash.net>
```

Abbildung 4.12: Dieser Eintrag aus dem „Changelog“ des Linux-Kernels belegt die Wiederaufnahme des TPROXY-Patches für „transparente Proxyserver“. Dabei werden auch dessen Einschränkungen genannt.

Zieladresse keine Möglichkeit mehr, auf die ursprüngliche Zieladresse zu schließen, da diese extra pro Paket hätte geführt werden müssen. Für TCP sieht die Sache anders aus, da es verbindungsorientiert arbeitet. Beim Durchlauf durch die „Redirection“ wird eine NAT-Tabelle gepflegt, welche die Zuordnung zur originalen Zieladresse pro Verbindung speichert; schließlich müssen rücklaufende Daten auch wieder unter Anwendung dieser Adresse vom „transparenten Proxyserver“ versendet werden. Die `SO_ORIGINAL_DST`-Funktion greift direkt auf die lokal existierende NAT-Tabelle des Kernels zu, um die gesuchte Zieladresse zu erhalten. Allerdings hat die Anwendung von NAT ihre speziellen Nachteile. So wird in selber Quelle erwähnt, dass bei NAT „Wettlaufprobleme“ („Race Conditions“) auftreten können, da Einträge in NAT-Tabellen einem Alterungsprozess unterliegen.

Die seit Kernel 2.6.28 wieder verfügbare Implementierung kommt dabei ohne NAT-Tabellen aus und ist zudem in der Lage, das verbindungslose UDP zu unterstützen. In Bezug auf die Implementierung der Proxyanwendung ergeben sich nur minimale Änderungen, allerdings sehen die zu hinterlegenden Paketfilterregeln anders aus. Es lässt sich sagen, dass der neue Ansatz dem auf „Redirection“ basierenden Ansatz überlegen ist.

Es zeigte sich aber, dass dieser modernere Mechanismus bislang nicht für eine auf dem mobilen Endgerät ansetzende Mobilitätsunterstützung adaptiert werden kann. Dies wurde durch einen Fehlschlag festgestellt, da das auf „Redirection“ basierende „Relay Plugin“ nach Portierung auf den neuen Mechanismus nicht mehr funktionierte. Der Grund war, dass bislang leider nur „non-local TCP/UDP traffic“ umgeleitet werden konnte, wie es auch der Kommentar in Abbildung 4.12 bestätigte. Da die Datenströme der mobilen Anwendungen allerdings noch innerhalb des mobilen Geräts umgeleitet und terminiert werden müssen, kann dieser Ansatz nicht verwendet werden. Ob eine zukünftige Unterstützung für solche lokal entsprungene Sockets geplant ist, konnte bislang nicht heraus-

gefunden werden. Dann wäre es allerdings lohnenswert, die „Relay Plugins“ zu migrieren und diese um Unterstützung für UDP zu erweitern.

Daher blieb vorerst keine andere Wahl, als für den REACH-Client weiterhin den auf „Redirection“ basierenden Ansatz zu verwenden. Der offensichtliche Vorteil dabei ist allerdings, dass dieser auch GNU/Linux-Systeme mit Kernelversionen vor 2.6.28 unterstützt. Soll der „transparente Proxyserver“ auf einer REACH-Box installiert werden, gilt der abzufangende Datenverkehr jedoch als „non-local“. Dann bietet sich der Einsatz des TPROXY-basierten Mechanismus an, welcher zudem UDP unterstützt.

Fallstricke beim Einsatz des `REDIRECT`-Ansatzes: Laut Dokumentation [Linu] ist das als `REDIRECT` bezeichnete Ziel (Target) nur in der `NAT`-Tabelle gültig. Angewendet werden kann dieses Ziel in den Ketten (Chains) `PREROUTING` und `OUTPUT`. Erstere Kette ist für „von außen“ stammende Pakete zuständig, während die `OUTPUT`-Kette für alle Pakete zuständig ist, die von lokalen Prozessen stammen. Sie ist die für das Vorhaben relevante Kette, da sie für die Daten der lokal laufenden Anwendungen zuständig ist. Ein beispielhaftes `iptables`-Kommando für einen „transparenten Proxyserver“ zeigt Abbildung 4.13.

```
iptables -t nat -A OUTPUT -p tcp -j REDIRECT --to-ports 1082
```

Abbildung 4.13: Dieses `iptables`-Kommando schaltet auf dem betroffenen System die `REDIRECT`-Paketfilterregel für einen „transparenten Proxyserver“. In der `nat`-Tabelle wird in der `OUTPUT`-Kette per `-A` („Append“) eine neue Regel hinzugefügt. Diese betrifft das Protokoll (`-p`) `tcp` und definiert als Aktion `-j` („Jump“) das `REDIRECT`-Ziel. Die dabei umgeleiteten Verbindungen werden auf Port `1082` des lokalen Systems umgeleitet und können dort von einer Proxysoftware angenommen werden.

Das Hauptproblem dabei ist, dass ein lokal entstandenes Paket nur dann durch die `OUTPUT`-Kette geleitet wird, wenn es „geroutet“ werden kann. Es muss also wenigstens einen Eintrag in der Routingtabelle geben, welche die Zieladresse des Pakets betrifft. Da beliebige Zielsysteme adressiert werden sollen, muss hierfür eine sehr allgemeine Regel benutzt werden, und zwar der Eintrag des so genannten „Default Gateways“. Ist dieses nicht gesetzt, und gibt es auch sonst keine Route, die zutrifft, greifen auch die Filterregeln des „transparenten Proxyserver“ nicht! Dies lässt sich experimentell zeigen: Das Programm `nc` (Netcat) [Ncat] liefert in diesem Fall beim Zugriff auf eine beliebige IP-Adresse die Fehlermeldung „Network is unreachable“, obwohl die TCP-Verbindung eigentlich vom lokal gestarteten „transparenten Proxyserver“ hätte abgefangen werden sollen.

Erste Abhilfe schafft eine gesetzte Route auf ein „Default Gateway“. Dann können die Pakete geroutet werden und auch der „transparente Proxyserver“ funktioniert. Dabei zeigen sich aber neue Probleme: Einträge bezüglich eines „Default Gateways“ werden primär beim Aufbau von Internetverbindungen gesetzt, und zwar meistens automatisch. Was passiert aber, wenn der Internetzugang (während eines Handovers) abreißt und zeitweise kein Gateway mehr gesetzt ist? Wie verhalten sich dann bereits abgefangene Verbindungen, wenn anschließend ein abweichender Eintrag (womöglich sogar über eine andere Schnittstelle) gesetzt wird? Sehr problematisch erscheint es zudem, dass bei REACH mehrere Interfaces gleichzeitig aktiviert sein können, aber mehrere gleichzeitig gesetzte „Default Gateway“-Einträge unweigerlich zu Problemen führen. Spätestens dann, wenn ein Gerät sowohl über Ethernet als auch über WLAN verbunden ist, und in beiden Netzen das „Default Gateway“ zufällig dieselbe IP-Adresse besitzt, gibt es ernste Probleme.

Für den Kern von REACH wird dieses Problem dadurch gelöst, dass auf die Notwendigkeit von „Default Gateway“-Einträgen weitestgehend verzichtet wird. Diesbezügliche Details werden später in Abschnitt 5.5.5 beleuchtet. Kurz zusammengefasst, setzt REACH entweder „quelladressbasierte Wegewahl“ ein, wobei der zentrale Gatewayeintrag keine Rolle spielt, oder es wird eine „zieladressbasierte Wegewahl“ durchgeführt, wobei für jeden REACH-Proxyserver eine dedizierte Route gesetzt werden muss. Der Grund für dieses Vorgehen ist, dass sich nur unter Umgehung des „Default Gateway“-Eintrags eine deterministische Nutzung der Netzzugangstechnologien bewerkstelligen lässt.

Dies löst jedoch nicht die genannten Probleme bezüglich des „transparenten Proxyservers“. Da allerdings REACH nicht auf besagten Gatewayeintrag angewiesen ist, darf dieser mit alleinigem Blick auf die „transparenten Proxyserver“ gestaltet werden.

Der direkte Ansatz jedoch, bei welchem der Gatewayeintrag bei der Assoziation des erstbesten Netzzugangsgeräts automatisch gesetzt wird, führt aus folgenden Gründen nicht zum Erfolg:

- Ein Gatewayeintrag wäre nur dann vorhanden, wenn auch mindestens ein Netzzugang verbunden ist. In Fällen ohne aktiven Netzzugang wäre das Routing wiederum nicht möglich, und neue wie bestehende TCP-Verbindungen würden nicht länger durch den „transparenten Proxyserver“ erfasst werden.

- Die gleichzeitige Verwendung mehrerer „Default Gateway“ Einträge ist problematisch. Im Fall „weicherer Handover“, wo zeitweise mindestens zwei Netzzugänge gleichzeitig aktiv sein müssen, wird oftmals nur der erste Gatewayeintrag übernommen. Der zweite Eintrag wird dabei vom „Dynamic Host Configuration Protocol“

(DHCP)-Client ignoriert und bleibt ungesetzt. Fällt dann allerdings der vorherige Netzzugang aus, existiert als Ergebnis überhaupt kein Eintrag mehr.

- Bereits umgeleitete TCP-Verbindungen „frieren ein“, wenn die bei ihrem Aufbau aktiv gewesene Routingregel verändert wird oder verschwindet. Das lässt sich experimentell zeigen und liegt im Aufbau des Paketfilters begründet. Eine genaue Erklärung konnte jedoch nicht gefunden werden, da der hiesige Anwendungsfall unüblich ist.

Daraus ergeben sich folgende Erkenntnisse und Anforderungen:

- Es muss *immer* ein Eintrag bezüglich des „Default Gateways“ gesetzt sein, auch in Fällen, wo kein Netzzugang existiert.
- Der Eintrag muss über die Zeit unverändert bleiben und immer dasselbe Interface mit einbeziehen.
- Es dürfen keine Routingschleifen entstehen, welche die IP-Pakete bis zum Ablauf ihrer „Time to live“ (TTL) auf den mobilen Gerät „kreiseln“ lassen.

Damit steht fest, dass die Schnittstellen der real genutzten Netzzugangsgeräte hierfür ungeeignet sind. Stattdessen muss eine dauerhafte Route hinterlegt werden, welche ein immer vorhandenes und zudem „neutrales“ Netzzugangsgerät mit einbezieht. Ab diesem Zeitpunkt dürfen auch keine weiteren „Default Gateway“-Einträge hinzugefügt werden, um Mehrdeutigkeiten zu verhindern. Hilfsprogramme wie der DHCP-Client-Daemon `dhcpcd` und der „Point-to-Point“ (PPP)-Daemon `pppd` müssen so parametriert werden, dass sie keine wirksamen Einträge bezüglich eines „Default Gateways“ mehr vornehmen.

„Default Gateway“-Eintrag mit persistentem Interface: Es wurden drei Möglichkeiten zur Wahl des Interfaces gefunden, welche alle Erfolg versprechend sind. Ein solches Interface wird beispielsweise unter GNU/Linux durch das so genannte „Dummy Device“ bereit gestellt. Dieses stellt ein rein virtuelles Netzzugangsgerät dar, ähnlich dem einer Ethernetkarte. Allerdings werden alle Pakete, die in ein solches „Dummy Device“ geroutet werden, stillschweigend verworfen.

Statt des „Dummy Device“ kann alternativ ein Tunnelendpunkt (wie beispielsweise `tun0`) eingesetzt werden. Ein solcher entsteht, wenn auf demselben System eine VPN-Software installiert ist, wie es beim Einsatz des VPN-basierten „Relay Plugins“ (siehe späteren Abschnitt 4.3.2.5) der Fall ist. Dieses Interface wäre ebenfalls persistent und eignet sich gleichermaßen für das Vorhaben.

Als letzte Variante kann auch das „Loopback Device“ benutzt werden, welches bereits ohne weiteren Konfigurationsaufwand verfügbar ist. Zwar funktioniert in diesem Fall der „transparente Proxyserver“ wie gewünscht, allerdings werden die IP-Pakete der nicht durch den „transparenten Proxyserver“ erfassten Protokolle wie UDP und ICMP durch das „Loopback Device“ wieder an den lokalen Rechner zugestellt. Dadurch kann eine Routingschleife entstehen, welche unnötigerweise Ressourcen beansprucht.

Da das „Dummy Device“ alle Daten verschwinden lässt, welche durch dieses geroutet werden, werden Seiteneffekte effektiv vermieden. Auf die Nutzung des „Loopback Devices“ sollte jedoch aus selbigem Grunde verzichtet werden, jedoch ist dessen Einsatz theoretisch möglich. Im Folgenden wird beispielhaft das „Dummy Device“ `dummy0` verwendet, bei Kombinationen mit dem VPN-basierten „Relay Plugin“ *muss* allerdings statt `dummy0` das entsprechende Tunneldevice `tunX` verwendet werden.

Konfiguration des „transparenten Proxyservers“ und der „Relay Plugins“: Das bereits genannte „Dummy Device“ muss bereits aktiviert worden sein, gegebenenfalls durch die Startskripte des Betriebssystems. Dabei muss es mit einer IP-Adresse versehen worden sein, damit das Device mit in die Wegewahl einbezogen wird. Diese IP-Adresse sollte so gewählt werden, dass eine Kollisionen beim Zugriff auf Rechner des Internets ausgeschlossen ist. Im vorliegenden Beispiel wurde die private IP-Adresse `10.255.255.1` gewählt, welche im Testaufbau kein zweites Mal vergeben war.

In Abbildung 4.14 wurden alle aktiven Netzzugangsschnittstellen des mobilen Endgeräts mit Hilfe des Kommandos `ifconfig` aufgelistet. Zu diesem Zeitpunkt war der REACH-Client noch nicht gestartet, sodass das `dummy0`-Device sowie das „Loopback Device“ `lo` die einzigen aktiven Schnittstellen waren. Man erkennt die dem `dummy0`-Device zugeordnete IP-Adresse `10.255.255.1`.

Die zentrale Routingtabelle lässt sich auf einem GNU/Linux-System mit Hilfe des Kommandos `route` anzeigen. In Abbildung 4.15 ist dessen Ausgabe für ein korrekt parametriertes System dargestellt. In diesem Fall war der REACH-Client noch nicht gestartet, daher sind auch noch keine weiteren Netzzugangsgeräte sichtbar.

Nach Definition der „Default Route“ können die Paketfilterregeln für den „transparenten Proxyserver“ hinterlegt werden. Wie ein solcher Befehl aussieht, wurde bereits in Abbildung 4.13 gezeigt. Allerdings sind weitaus komplexere Regeln denkbar und auch erforderlich. Die gezeigte Filterregel erfasste nämlich *ausnahmslos alle* lokal erzeugten TCP-Verbindungen, sodass keine Kommunikation zwischen REACH-Client und REACH-Server mehr möglich wäre. Zudem würden ebenfalls alle Verbindungen erfasst werden, welche an „Localhost“ gerichtet sind, also das mobile Gerät nicht verlassen dürfen.

Daher werden Ausschlusskriterien für die „Redirection“ benötigt. Das endgültige Kom-

```
neelix ~ # ifconfig
dummy0    Link encap:Ethernet  HWaddr ae:cb:b0:13:d2:92
          inet addr:10.255.255.1  Bcast:255.255.255.255  Mask:0.0.0.0
          UP BROADCAST RUNNING NOARP  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:1648 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1648 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:110216 (107.6 KiB)  TX bytes:110216 (107.6 KiB)
neelix ~ #
```

Abbildung 4.14: In der Liste der parametrierten Schnittstellen dieses mobilen Endgeräts taucht das „Dummy Device“ auf, welches für den „transparenten Proxyserver“ benötigt wird.

```
neelix ~ # route
Kernel IP routing table
Destination   Gateway   Genmask      Flags Metric Ref   Use Iface
loopback      *         255.0.0.0    U     0      0       0 lo
default       *         0.0.0.0      U     0      0       0 dummy0
neelix ~ #
```

Abbildung 4.15: Der einzige erlaubte Eintrag bezüglich eines „Default Gateways“ involviert das „Dummy Device“ `dummy0`. Dieses lässt alle Daten, welche nicht durch einen „transparenten Proxyserver“ abgefangen werden, verschwinden.

```
iptables -A OUTPUT -t nat -p tcp ! -d 127.0.0.1 ! --dport 10000   \
         -j REDIRECT --to-ports 1082
```

Abbildung 4.16: Dieses `iptables`-Kommando schaltet die Paketfilterregel eines „transparenten Proxyservers“, definiert dabei aber zwei Ausnahmeregeln.

mando einer für REACH durchaus relevanten Paketfilterregel zeigt Abbildung 4.16. Das erste Filterkriterium (`! -d 127.0.0.1`) sorgt dafür, dass alle an „Localhost“ gerichteten TCP-Verbindungen ignoriert werden. Das zweite Kriterium (`! -dport 10000`) ignoriert

alle TCP-Verbindungen, welche als Zielport die Portnummer 10000 adressieren. Diese Portnummer wurde willkürlich für TCP-Verbindungen zwischen REACH-Client und REACH-Server festgelegt. Somit ist sichergestellt, dass diese miteinander kommunizieren können, ohne selbst durch den „transparenten Proxyserver“ beeinflusst zu werden.

Der Paketfilter NETFILTER ist sehr mächtig. Da er durch das Kommando `iptables` parametriert wird, präsentiert sich dieses leider als sehr komplex. Es sind tiefgreifende Kenntnisse der internen Struktur von NETFILTER notwendig, um ein durchdachtes Konzept zu erhalten. Ein solches ist jedoch erforderlich, denn immerhin erfüllt der Paketfilter im vorliegenden Fall wenigstens die Aufgaben einer Firewall und zusätzlich Teilaufgaben eines „transparenten Proxyservers“.

Daher bieten sich „Frontends“ zu `iptables` an, welche die Erstellung der Paketfilterregeln erleichtern. Ein solches „Frontend“ stellt beispielsweise die Software SHOREWALL [East] dar. Hierbei können Filterregeln in aufgabenorientierter Form definiert werden. Die Software kümmert sich dann im Hintergrund darum, aus diesen Regeln gültige `iptables`-Kommandos zu erstellen und auszuführen. Die Nutzung von SHOREWALL ist sehr zu empfehlen und wird vom Autor bei allen seinen GNU/LINUX-basierten Systemen eingesetzt.

```
SECTION NEW

# Regeln fuer REACH-Client: Transparenter Proxyserver
# 1. Zugriff auf Uninetz, internen Zugriff sicherstellen
REDIRECT fw 1081 tcp !10000       - 141.24.0.0/16

# 2. Default-Eintrag: Erfasst alles was noch uebrig ist.
#    Spontaner Wechsel des REACH-Servers moeglich!
REDIRECT fw 1082 tcp !10000       - !127.0.0.1
```

Abbildung 4.17: Für das mobile Endgerät des Demonstrators werden mit Hilfe des „Frontends“ SHOREWALL zwei Paketfilterregeln hinterlegt. Sie definieren zwei „transparente Proxyserver“.

Abbildung 4.17 zeigt auszugsweise die im Demonstrator verwendete Konfigurationsdatei von SHOREWALL, welche die Umleitungsregeln des „transparenten Proxyservers“ definiert. Zuerst fällt auf, dass gleich zwei Regeln definiert wurden. Bevor die Idee dahinter erläutert wird, sei an dieser Stelle erwähnt, dass die zweite Regel genau dem `iptables`-Kommando aus Abbildung 4.16 entspricht. Hier wird ein `REDIRECT` Regel für alle vom System selbst stammenden TCP-Verbindungen definiert (Quelle `fw` „Firewall“, also das lokale System). Ziel der „Redirection“ ist Port 1082 auf dem mobilen Gerät,

geltend für alle `tcp`-Verbindungen mit Ziel ungleich Portnummer `10000` und Zieladresse ungleich `127.0.0.1` („Localhost“).

Bleibt die Frage zu klären, wieso letztgenannte Regel um eine weitere Regel ergänzt wurde. Immerhin war diese für sich gesehen bereits in der Lage, sämtliche relevanten TCP-Verbindungen umzuleiten. Grund dafür ist das in REACH umgesetzte Konzept zur Minimierung des „Dreiecks-Routings“ („triangle routing“). Dieses basiert darauf, mehrere REACH-Server in Abhängigkeit des momentanen Aufenthaltsorts mit einzubeziehen. Für den Mechanismus der „transparenten Proxyserver“ bietet sich dieses Konzept geradezu an, da so die abgefangenen TCP-Verbindungen immer über den topologisch nahesten REACH-Proxyserver geleitet werden können. Es gibt allerdings auch Szenarien, in welchen dieses Verhalten unerwünscht ist. Ein solcher Fall wird hier exemplarisch durch die erste Ausnahmeregel in Abbildung 4.17 beschrieben. Sie leitet alle TCP-Verbindungen, welche als Ziel das Campusnetz der „Technischen Universität Ilmenau“ (TU Ilmenau, `141.24.0.0/16`) aufweisen, auf Port `1081` um. Alle an diesem Port abgefangenen TCP-Verbindungen werden dabei gesondert behandelt und *immer* über ein und denselben REACH-Proxyserver geleitet. Dieser ist definiert (was hier noch nicht erkennbar ist) und befindet sich ebenfalls innerhalb des Campusnetzes der TU Ilmenau. So wird es ermöglicht, dass ein Anwender von überall aus mit seinem mobilen Endgerät auf universitätsinterne Ressourcen zugreifen kann.

Weiterhin ist interessant, wie die beiden TCP-Portnummern `1081` und `1082` auf dem mobilen Endgerät durch REACH verwaltet werden. Da hierfür das „Relay Plugin“ bezüglich des „transparenten Proxyservers“ zuständig ist, wird im Folgenden auf dessen Konfiguration eingegangen.

In Abbildung 4.18 ist die zu den bereits erläuterten SHOREWALL-Regeln passende Konfiguration des clientseitigen „Relay Plugins“ für „transparente Proxyserver“ gezeigt. Es handelt sich dabei um einen exakten Auszug aus der Konfigurationsdatei des Demonstrators.

Verkörpert wird das „Relay Plugin“ durch die Bibliothek `librrpc_tproxy.so`. Es werden zwei Instanzen definiert, wobei erstere den lokalen TCP-Port mit der Portnummer `"1081"` behandelt und letztere die Portnummer `"1082"`. Erstere nimmt mit Hilfe des `Discovery`-Parameters `"use reachserver"` Einfluss auf die Dienstesuche, wobei der REACH-Server mit dem Identifikator `"Equinox"` fest vorgeschrieben wird. Dieser Identifikator referenziert einen REACH-Proxyserver innerhalb des Campusnetzes. Die Aufschlüsselung dieses Identifikators in den konkreten REACH-Proxyserver ist Aufgabe des transportorientierten Kerns. Es ist erkennbar, dass sich die beiden SHOREWALL-Regeln mit den beiden „Relay Plugin“-Instanzen zu zwei „transparenten Proxyservern“ ergänzen. Ersterer leitet seine abgefangenen Verbindungen zudem immer über den REACH-Server

```
<RelayPlugins>
  <RelayPlugin loadlibrary="librrpc_tproxy.so">
    <Instance>
      <Configuration>
        <Item key="listener ip" value="0.0.0.0" />
        <Item key="listener port" value="1081" />
      </Configuration>
      <Discovery>
        <Item key="use reachserver" value="Equinox" />
      </Discovery>
    </Instance>
    <Instance>
      <Configuration>
        <Item key="listener ip" value="0.0.0.0" />
        <Item key="listener port" value="1082" />
      </Configuration>
    </Instance>
  </RelayPlugin>
</RelayPlugins>
```

Abbildung 4.18: Beispielhafte Konfiguration einer „Relay Plugin“-Instanz bezüglich des „transparenten Proxyservers“ am REACH-Client.

`"Equinox"`, was durch das erläuterte Szenario gefordert wird. Letztere Instanz behandelt auf Grund der Reihenfolge der Shorewall-Regeln alle noch übrig gebliebenen TCP-Verbindungen und leitet diese über einen nicht näher definierten REACH-Proxyserver, wobei hierbei topologisch nahe REACH-Proxyserver präferiert werden.

```
<RelayPlugins>
  <RelayPlugin loadlibrary="librrps_tproxy.so">
    <Instance />
  </RelayPlugin>
</RelayPlugins>
```

Abbildung 4.19: Konfiguration einer „Relay Plugin“-Instanz bezüglich des „transparenten Proxyservers“ am REACH-Server.

Bezüglich eines jeden REACH-Servers, welcher diesen Dienst durch Instantiierung des zugehörigen „Relay Plugins“ anbieten möchte, wird eine vergleichsweise knappe Konfiguration benötigt. In Abbildung 4.19 ist diese gezeigt. Es existiert wie für REACH-Server vorgeschrieben nur eine einzige Instanz, welche zudem keine weiteren Parameter benötigt.

4.3.2.3 „TCP-Relay“

Für die Weiterleitung von TCP-Verbindungen wurden zwei „Relay Plugin“-Paare implementiert. Ersteres ist verantwortlich für ausgehende TCP-Verbindungen, während letzteres für eingehende TCP-Verbindungen eingesetzt wird.

Anwendungsfälle: Der vorgestellte Mechanismus ist dem so genannten „Port Forwarding“ ähnlich, welches vornehmlich an NAT-Grenzen geschaltet wird, um dort eingehende Verbindungen zu ermöglichen. Allerdings wird bei dem hier gezeigten Ansatz die TCP-Verbindung aufgetrennt und deren Ende-zu-Ende-Semantik zerstört, was im Umkehrschluss aber auch die Resistenz gegenüber längerfristigen Isolationssituationen ermöglicht.

Die Bereitstellung ausgehend gerichteter Weiterleitungsregeln erscheint dann als sinnvoll, wenn weder das SOCKS-basierte „Relay Plugin“ noch das „Relay Plugin“ bezüglich des „transparenten Proxyservers“ zum Einsatz kommen kann oder soll. Beide Mechanismen wären bereits mächtig genug, die Funktionalität ausgehender TCP-Weiterleitungsregeln abzudecken.

Um ein Beispiel zu nennen, soll ein einzelner TCP-basierter Dienst handoverresistent in Anspruch genommen werden. Eine auf dem mobilen Endgerät geschaltete Weiterleitungsregel kann dazu den lokalen Port `5000` reservieren und diesen beispielsweise mit dem SSH-Daemon eines gewünschten Zielsystems im Internet verknüpfen. Möchte der Anwender jetzt „handoversicher“ mit diesem Zielsystem per SSH kommunizieren, wählt er als Zieladresse für seinen SSH-Client die Adresse `127.0.0.1:5000`. Er kontaktiert damit das „Relay Plugin“, durchläuft das gesicherte Weiterleitungsschema und landet schlussendlich auf dem gewünschten Zielsystem auf Port `22` für SSH. Im Handoverfall bleiben die betroffenen Verbindungen geschützt. Es ist allerdings wiederum erkennbar, dass eine Nutzung von SOCKS oder eines „transparenten Proxyservers“ einfacher zu handhaben wäre. Jedoch ist der Einsatz von SOCKS nicht immer möglich, und nicht auf allen Betriebssystemen werden „transparente Proxyserver“ unterstützt.

Funktionsweise: Für eine solche Schaltregel werden insgesamt drei Parameter benötigt, unabhängig davon, ob es sich um das „Relay Plugin“ für ein- oder ausgehend initiierte Verbindungen handelt. Zuerst ist der Quellport zu benennen, an welchem auf TCP-Verbindungen gewartet werden soll. Für diesen dürfen nur unpriviligierte Portnummern verwendet werden, da „Relay Plugin“-Instanzen in der aktuellen Form ohne Systemverwaltungsrechte laufen. Weiterhin ist das Ziel der Umleitung zu definieren, in Form einer IP-Adresse und einer TCP-Portnummer.

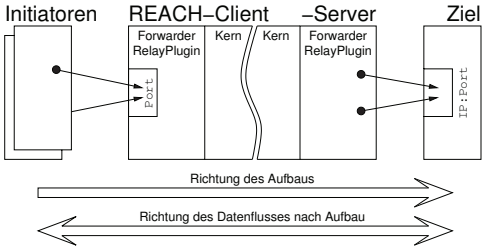

Abbildung 4.20: Pro geschalteter Weiterleitungsregel können eine Vielzahl an TCP-Verbindungen am clientseitigen „Relay Plugin“ entgegen genommen und seitens dessen Partnerinstanz zum Zielsystem geleitet werden.

Dieser Sachverhalt wird in Abbildung 4.20 anhand einer *ausgehenden* Weiterleitung verdeutlicht. Auf der linken Seite werden die Verbindungen am spezifizierten Port durch ein „Relay Plugin“ am REACH-Client angenommen, während sie auf der rechten Seite durch dessen Partnerinstanz am REACH-Server zum Zielsystem „verlängert“ werden. Der Aufbau einer solchen Verbindung erfolgt hier „von innen nach außen“, der anschließende Datenaustausch erfolgt dagegen bidirektional.

Wird als Zieladresse die IP-Adresse `"127.0.0.1"` angegeben, ist damit immer dasselbe System gemeint, auf welchem die weitergeleitete TCP-Verbindung den Weiterleitungsmechanismus verlässt. Das ist entweder das mobile Endgerät oder der REACH-Proxyserver, je nachdem, in welche Richtung die Weiterleitungsregel gerichtet ist. Das wird besonders bei der Schaltung eingehender Weiterleitungsregeln angewendet, da hierbei auf dem mobilen Gerät gestartete Serverdienste angesprochen werden sollen.

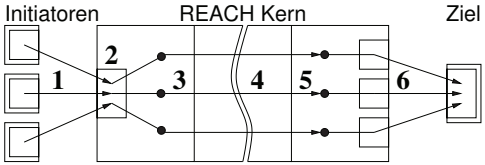

Abbildung 4.21: Pro abgefangener TCP-Verbindung werden eine *logische Verbindung* sowie eine ausgehende TCP-Verbindung an der Partnerinstanz belegt. Dadurch gibt es keine Zuordnungsprobleme.

Abbildung 4.21 hebt hervor, dass jede Weiterleitungsregel eine Vielzahl an aktiven Verbindungen handhaben kann. Für jede an der TCP-Annahmestelle (**2**) angenommene TCP-Verbindung (**1**) wird eine stromorientierte *logische Verbindung* (**3**) angefordert, welche den gesicherten Datentransport (**4**) zwischen beiden „Relay Plugin“-Instanzen si-

cherstellt. An der Partnerinstanz angekommen, wird für jede vorliegende *logische Verbindung* (**5**) eine ausgehende TCP-Verbindung (**6**) zum Zielsystem geschaltet. Im Erfolgsfall schalten beide Instanzen in einen Weiterleitungsmodus, bis jeweils entweder die TCP-Verbindung oder die *logische Verbindung* getrennt wird (letzteres bedeutet einen Abbruch der TCP-Verbindung an der Partnerinstanz). Da für jede TCP-Verbindung zum Zielsystem ein neuer Socket belegt wird, werden Zuordnungsprobleme ausgeschlossen.

Konfiguration: Wie erläutert, müssen sowohl für eingehende als auch für ausgehende Schaltregeln drei Parameter angegeben werden. Dies sind immer die Portnummer der Annahmestelle sowie das Ziel der Weiterleitungsregel in Form einer IP-Adresse und einer Portnummer.

```
<RelayPlugins>
  <RelayPlugin loadlibrary="librrpc_tcprelay_forward.so">
    <Instance>
      <Configuration>
        <Item key="source port" value="21000" />
        <Item key="destination ip" value="127.0.0.1" />
        <Item key="destination port" value="21000" />
      </Configuration>
    </Instance>
  </RelayPlugin>
  <RelayPlugin loadlibrary="librrpc_tcprelay_reverse.so">
    <Instance>
      <Configuration>
        <Item key="source port" value="22000" />
        <Item key="destination ip" value="127.0.0.1" />
        <Item key="destination port" value="22000" />
      </Configuration>
    </Instance>
  </RelayPlugin>
</RelayPlugins>
```

Abbildung 4.22: Beispielhafte Konfiguration zweier „Relay Plugin“-Instanzen am REACH-Client. Erstere ist für ausgehend gerichtete, letztere für eingehend gerichtete TCP-Weiterleitungsregeln verantwortlich.

In Abbildung 4.22 ist eine beispielhafte Konfigurationsdatei eines mobilen Endgeräts (REACH-Client) gezeigt. Dabei wird jeweils eine Instanz einer ausgehenden sowie eine Instanz mit einer eingehenden Weiterleitungsregel beschrieben. Die drei genannten Parameter sind jeweils erkennbar. Da hierbei als Zieladresse immer die IP-Adresse `127.0.0.1` verwendet wird, ist im ersten Fall der REACH-Proxyserver das Ziel (es handelt sich um

eine ausgehende Schaltregel), im zweiten Fall ist es das mobile Endgerät. Da für die ausgehenden Schaltregeln kein `Discovery`-Parameter spezifiziert wurde, können mehrere REACH-Proxyserver mit in die Weiterleitung einbezogen werden.

Eingehend gerichtete Weiterleitungsregeln sind dagegen nicht migrationsfähig. Sie bleiben immer an denselben REACH-Proxyserver gebunden, da die Adresse der Annahmestelle für einen außenstehenden Verbindungsinitiator konstant bleiben muss. Es ist allerdings möglich, durch einen `Discovery`-Parameter einen REACH-Server fest vorzugeben. Fehlt ein solcher Parameter, wird der topologisch naheste relevante REACH-Proxyserver ausgewählt, und dann nicht wieder gewechselt.

Serverseitig kann die Instantiierung der beiden „Relay Plugins“ mit einer Konfiguration erfolgen, wie sie in Abbildung 4.23 gezeigt ist. Für eingehende Schaltregeln muss zur Instantiierung die IP-Adresse der TCP-Annahmestelle angegeben werden. Sie wird später jedem sich assoziierenden „Relay Plugin“ mitgeteilt, damit der Anwender des mobilen Endgeräts erfährt, mit Hilfe welcher IP-Adresse die Annahmestelle erreicht werden kann. Da die Wahl des REACH-Servers ein Ergebnis der Dienstsuche sein kann, könnte der Anwender sonst nicht wissen, wie die genaue Adresse der reservierten Annahmestelle lautet. Hierbei gilt, wie bereits für das SOCKS-basierte „Relay Plugin“ erläutert, dass weder `"0.0.0.0"`, `"127.0.0.1"` noch die private IP-Adresse einer LAN-Schnittstelle angegeben werden dürfen.

```
<RelayPlugins>
  <RelayPlugin loadlibrary="librrps_tcprelay_forward.so">
    <Instance />
  </RelayPlugin>
  <RelayPlugin loadlibrary="librrps_tcprelay_reverse.so">
    <Instance>
      <Configuration>
        <Item key="listener ip" value="141.24.92.184" />
      </Configuration>
    </Instance>
  </RelayPlugin>
</RelayPlugins>
```

Abbildung 4.23: Beispielhafte Konfiguration zweier „Relay Plugin“-Instanzen am REACH-Server. Instanzen für eingehende TCP-Verbindungen benötigen als Parameter die öffentliche IP-Adresse, mit Hilfe derer die TCP-Annahmestelle erreicht werden kann.

4.3.2.4 „UDP-Relay“

Analog zu den im letzten Abschnitt vorgestellten Weiterleitungsregeln für TCP-Verbindungen wurden zwei Paare an „Relay Plugins“ erstellt, welche mit Blick auf die Weiterleitung von UDP-Assoziationen erstellt worden sind.

Anwendungsfälle: Auch hier gilt, dass der vorgestellte Mechanismus dem „Port Forwarding“ an NAT-Grenzen ähnlich ist.

Der Mechanismus zur Weiterleitung von UDP-Paketen ausgehend initiierter UDP-Assoziationen kann beispielsweise zum Austausch von Daten zwischen zwei VPN-Instanzen eingesetzt werden. Hierbei ist ein VPN-Client auf dem mobilen Endgerät installiert, welcher mit Hilfe *einer* UDP-Assoziation einen VPN-Server kontaktiert. Diese UDP-Assoziation kann durch eine „ausgehende Port-Weiterleitung“ behandelt werden, sodass im Endeffekt auch Leistungsmerkmale von REACH wie Kanalbündelungen und „weichere Handover“ für VPNs wirksam sind. Der VPN-basierte Dienst wird im nächsten Abschnitt 4.3.2.5 ausführlicher betrachtet.

Funktionsweise: Bei jeder Schaltregel kann eine Vielzahl an Assoziationen verwaltet werden. Eine Assoziation ist eine Kommunikationsbeziehung zwischen der UDP-Annahmestelle und einem externen Kommunikationspartner, welcher durch eine Kombination aus IP-Adresse und Portnummer eindeutig benannt werden kann. Rücklaufende Pakete werden zu solchen Assoziationen zugeordnet und erreichen somit wieder unterschiedliche Kommunikationspartner. Die UDP-Annahmestelle residiert entweder auf dem mobilen Gerät am REACH-Client (ausgehende Port-Weiterleitung) oder auf dem REACH-Proxyserver am REACH-Server (eingehende Port-Weiterleitung).

An dieser Annahmestelle, deren Portnummer durch den Nutzer vorgeschlagen wird, werden UDP-Assoziationen angenommen. Für jedes Paket, welches von einer bis dato unbekannten IP-Adress-Port-Kombination empfangen wird, entsteht eine neue UDP-Assoziation. Da UDP allerdings verbindungslos arbeitet, verschwinden UDP-Assoziationen an der Annahmestelle wieder, wenn für eine bestimmte Zeitspanne kein Paket über diese Assoziation geleitet wurde. Diese Zeitspanne wurde in REACH auf 180 Sekunden festgelegt. Diese Zeitspanne wurde in Anlehnung an die Standardeinstellung für die Alterung von Einträgen in der NAT-Adressumsetzungstabelle bezüglich UDP verwendet, welche bei einem aktuellen Linux-Kernel (Version 2.6.28) wie in Abbildung 4.24 ermittelt werden kann. Wenn nämlich GNU/Linux-basierte NAT-Gateways mit einem Wert von 180 Sekunden operieren, kann dieser Wert für REACH nicht für unvorhersehbare Probleme sorgen. Er ist zudem durch RFC 4787 [AuJe07] abgedeckt, wo es heißt, dass die

Zeitspanne bis auf eine Ausnahme nicht unter zwei Minuten betragen darf, was hiermit gewährleistet wäre.

```
wesley netfilter # pwd
/proc/sys/net/netfilter
wesley netfilter # cat nf_conntrack_udp_timeout
30
wesley netfilter # cat nf_conntrack_udp_timeout_stream
180
wesley netfilter #
```

Abbildung 4.24: Die in den NAT-Tabellen des LINUX-Kernels geltenden Zeitspannen bezüglich der Erkennung inaktiver UDP-Assoziationen lassen sich durch diese zwei Kommandos ermitteln.

Für jede angenommene Assoziation wird eine paketorientierte *logische Verbindung* etabliert. Alle UDP-Pakete, die von der UDP-Annahmestelle zu dieser UDP-Assoziation zugeordnet wurden, werden anschließend über die zugehörige *logische Verbindung* versendet. Rücklaufende UDP-Pakete, welche einer *logischen Verbindung* entnommen werden, werden über die zugehörige UDP-Assoziation wieder an die IP-Adresse und Portnummer des Kommunikationspartners versendet. Wird dabei die Leerlaufzeit von 180 Sekunden überschritten, „vergisst“ die Annahmestelle die betroffene UDP-Assoziation, was folglich auch zu einem Abbau der zugeordneten *logischen Verbindung* führt.

An der Partnerinstanz der die Assoziationen annehmenden „Relay Plugin“-Instanz müssen schlussendlich die UDP-Pakete in Richtung des Zielsystems (spezifiziert durch die Weiterleitungsregel in Form einer IP-Adresse und UDP-Portnummer) gesendet und auch von diesem wieder empfangen werden. Dazu wird für jede angenommene *logische Verbindung* ein neuer UDP-Socket reserviert. Es wird gewährleistet, dass die vom Zielsystem zurückgesendeten UDP-Pakete immer eindeutig zu *logischen Verbindungen* zugeordnet werden können. Sollte aber in der Zwischenzeit eine Assoziation auf Grund von Inaktivität geschlossen worden sein, ist auch der hier reservierte Socket nicht mehr aktuell und das Paket geht verloren.

Das Schema ist identisch mit dem Port-Weiterleitungsschema bezüglich TCP, nur dass UDP verbindungslos arbeitet und somit eine Überwachung auf Inaktivität durchgeführt werden muss. Eine eindeutige Zuordnung rücklaufender Pakete zum ursprünglichen Initiator einer jeden Ende-zu-Ende-Assoziation wird gewährleistet.

Die beiden „Relay Plugins“ für eingehende UDP-Assoziationen funktionieren nach demselben Schema, nur mit vertauschten Rollen. Die Annahme von Assoziationen findet dabei am REACH-Server statt, und die Weiterleitung zum spezifizierten Zielsystem

am REACH-Client auf dem mobilen Endgerät. Die Ziel-IP-Adresse wird dabei in den meisten Fällen `"127.0.0.1"`, also „Localhost“, lauten.

Konfiguration: Die Konfiguration der beiden „Relay Plugins“ erfolgt analog der „Relay Plugins“ für TCP-Weiterleitungsregeln. Alle dort gemachten Aussagen treffen auch für die hier vorgestellten UDP-basierten „Relay Plugins“ zu.

```
<RelayPlugins>
  <RelayPlugin loadlibrary="librrpc_udprelay_forward.so">
    <Instance>
      <Configuration>
        <Item key="source port" value="21000" />
        <Item key="destination ip" value="127.0.0.1" />
        <Item key="destination port" value="21000" />
      </Configuration>
    </Instance>
  </RelayPlugin>
  <RelayPlugin loadlibrary="librrpc_udprelay_reverse.so">
    <Instance>
      <Configuration>
        <Item key="source port" value="22000" />
        <Item key="destination ip" value="127.0.0.1" />
        <Item key="destination port" value="22000" />
      </Configuration>
    </Instance>
  </RelayPlugin>
</RelayPlugins>
```

Abbildung 4.25: Beispielhafte Konfiguration zweier „Relay Plugin“-Instanzen am REACH-Client. Erstere ist für ausgehend gerichtete, letztere für eingehend gerichtete UDP-Weiterleitungsregeln verantwortlich.

Abbildung 4.25 zeigt eine beispielhafte Konfiguration beider Plugins am REACH-Client, während Abbildung 4.26 das Pendant am REACH-Server zeigt. Auch hier gilt, dass zur Instantiierung des „Relay Plugins“ für eingehende UDP-Assoziationen eine global gültige IP-Adresse spezifiziert werden muss.

Beim Vergleich mit den Konfigurationsdateien der TCP-basierten Weiterleitungsplugins fällt auf, dass sich lediglich die Dateinamen der „Relay Plugins“ geändert haben. Die Parameternamen und ihre Bedeutung sind identisch, und die gewählten Adressen sind lediglich aus Gründen der Vereinfachung gleich.

```
<RelayPlugins>
  <RelayPlugin loadlibrary="librrps_udprelay_forward.so">
    <Instance />
  </RelayPlugin>
  <RelayPlugin loadlibrary="librrps_udprelay_reverse.so">
    <Instance>
      <Configuration>
        <Item key="listener ip" value="141.24.92.184" />
      </Configuration>
    </Instance>
  </RelayPlugin>
</RelayPlugins>
```

Abbildung 4.26: Beispielhafte Konfiguration zweier „Relay Plugin“-Instanzen am REACH-Server. Instanzen für eingehende UDP-Assoziationen benötigen als Parameter die öffentliche IP-Adresse, mit Hilfe derer die UDP-Annahmestelle erreicht werden kann.

4.3.2.5 „Virtual Private Networks“ (VPNs)

Ein sehr interessantes Konzept eröffnet sich durch Einsatz eines „Virtual Private Networks“ (VPN). VPNs dienen in erster Linie dazu, zwischen topologisch getrennten Systemen eine LAN-ähnliche Kommunikationsbeziehung zu ermöglichen. Dies geschieht dadurch, dass man sich die einzelnen Systeme als Bestandteil eines „virtuellen LANs[7]“ („virtuell“, da nicht physikalisch vorhandenen) vorstellt. Physikalisch gesehen brauchen die Systeme dabei nicht „benachbart“ zu sein, sondern können mit Hilfe unterschiedlichster Netzzugangstechnologien vernetzt sein. Wichtig ist nur, dass „IP-Konnektivität“ zwischen den Systemen besteht, damit das „logische VPN“ durch das Vorhandensein „physikalischer Konnektivität“ existieren kann.

VPNs werden rein durch Software realisiert. Dabei übernimmt eines der teilnehmenden Systeme eine herausragende Rolle, da es die Koordinierung des VPNs übernimmt. Dieser so genannte „VPN-Server“ übernimmt beispielsweise Aufgaben zur Authentifizierung und Autorisierung. Zudem würde er in Szenarien mit automatischer Adressvergabe die Hoheit über den Adressvorrat des VPNs besitzen. Zudem ist meist der VPN-Server auch als Gateway konfiguriert und bietet einen Zugang zum Internet an. Diese Aufgabe kann aber auch jedes andere System des VPNs übernehmen; es müssen dazu lediglich die entsprechenden Routen gesetzt werden. Alle anderen beteiligten Systeme benutzen einen „VPN-Client“. Ein solcher VPN-Client verbindet sich mit Hilfe der vorhandenen IP-

[7] Damit sind keine so genannten „Virtual Local Area Networks“ (VLANs) gemeint. Die Wortwahl dient lediglich der Verdeutlichung.

Konnektivität zum VPN-Server. Dort wird er im Idealfall authentifiziert und autorisiert und wird schlussendlich Bestandteil des VPNs.

Es lassen sich zwei Arten von VPNs unterscheiden. Bei so genannten „gebrückten VPNs“ wird von der VPN-Software eine „virtuelle Ethernetkarte“ angeboten, welche eine Weiterleitung von Ethernet-Rahmen erlaubt. Dadurch wird es möglich, „echte“ Ethernet-basierte LANs mit einem VPN-Adapter „zu brücken“ und somit auch Broadcasts sowie alternative Vermittlungsschichtprotokolle durch das VPN zu leiten.

Alternativ kann als VPN-Adapter ein so genannter IP-Tunnelendpunkt angeboten werden. Da dieser in der Vermittlungsschicht ansetzt, sind keine Broadcasts mehr möglich. Man spricht hierbei von „gerouteten VPNs“.

Welche Variante eines VPNs eingesetzt werden soll, hängt von diversen Faktoren ab, wie beispielsweise von der Art des einzusetzenden Vermittlungsschichtprotokolls. Für die hier vorgestellte Architektur ist es aber irrelevant, welche Art des VPNs ausgewählt wird oder durch externe Vorgaben erzwungen wird. Diese Entscheidung kann losgelöst von der Handoverproblematik getroffen werden.

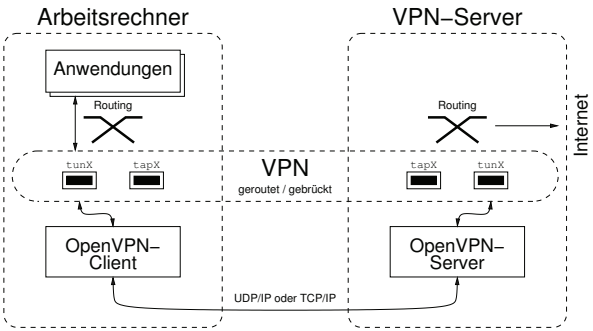

Abbildung 4.27: VPNs werden durch einen zentralen VPN-Server und den dezentralen VPN-Clients gebildet. Die VPN-Clients verbinden sich zum VPN-Server, sodass diese dem VPN beitreten können. Hier werden „gebrückte“ (`tapX`) und „gerouteten“ (`tunX`) VPNs unterschieden.

Der reguläre Einsatzzweck eines VPNs wird durch Abbildung 4.27 verdeutlicht. Hierbei wird allerdings nicht dargestellt, dass an einem VPN eine Vielzahl an VPN-Clients teilnehmen kann, und nicht wie dargestellt bloß einer.

Ferner kann unterschieden werden, welche Art von IP-Adressen (im Fall der Verwendung von IP als Vermittlungsschichtprotokoll) innerhalb des VPNs zum Einsatz kommen. Zum einen können private IP-Adressen gemäß RFC 1918 [RMKG+96] verwendet werden, wobei allerdings am Gateway NAT durchgeführt werden muss. Durch den Einsatz

von NAT entstehen aber neue Probleme, welche dann separat gelöst werden müssen. Alternativ bietet es sich aus genau diesem Grund an, innerhalb des VPNs auf „öffentliche IP-Adressen“ zu setzen. Diese müssen allerdings als reservierter Bereich am VPN-Server verfügbar sein; dann kann auf den Einsatz von NAT verzichtet werden. Eine solche Struktur kommt beispielsweise beim Campusnetz der TU Ilmenau zum Einsatz, wo an VPN-Clients IP-Adressen aus dem öffentlichen Adressbereich der TU Ilmenau zugewiesen werden.

Der nennenswerte Vorteil beim Einsatz eines VPNs als „Dienst“ für die Mobilitätsunterstützung besteht darin, dass es auf Ebene des Internetprotokolls oder darunter (bei gebrücktem VPN) arbeitet. Dadurch werden automatisch auch alle auf IP aufsetzenden Hilfsprotokolle wie ICMP und IGMP sowie alle erdenklichen Transportschichtprotokolle wie TCP, UDP aber auch SCTP und DCCP unterstützt. Dieses Verhalten kann sich aber im Gegenzug als problematisch erweisen, da eben nicht auf die Besonderheiten der jeweiligen Transportschichtprotokolle eingegangen wird. So können während längerfristiger Isolationssituationen TCP-Verbindungen auf Grund von Zeitüberschreitungen abreißen.

Funktionsweise: Der in Abbildung 4.27 dargestellte Verbund aus VPN-Client und VPN-Server muss so miteinander verschaltet werden, dass dieser auch in Szenarien mit Mobilität funktioniert.

Die Kommunikation zwischen VPN-Client und VPN-Server wird in der Regel über das verbindunglos und unzuverlässig arbeitende Transportschichtprotokoll UDP abgewickelt. Der Grund für die Wahl von UDP ist, dass entweder MAC-Rahmen oder IP-Pakete „getunnelt“ werden. Die bei TCP enthaltenen Sicherungsmechanismen sowie dessen Flusssteuerungsalgorithmen wären für den Transport von Paketen kontraproduktiv. Besonders bei der Übertragung von Sprachdaten würden sich die durch TCP verursachten Verzögerungen negativ bemerkbar machen. Daher ist UDP, welches über keine Sicherungs- und Flusssteuerungsmechanismen verfügt, als Basis für den Tunnel zu bevorzugen.

Ziel ist es, die UDP-Assoziation zwischen VPN-Client und VPN-Server abzufangen und mit Blick auf Mobilität gesichert zu übertragen. Dazu eignen sich die bereits im letzten Abschnitt vorgestellten „Relay Plugins“, welche für die Weiterleitung ausgehender UDP-Assoziationen verantwortlich sind (siehe Abschnitt 4.3.2.4). Der auf dem mobilen Endgerät laufende VPN-Client wird so parametriert, dass er seine über UDP getunnelten Daten nicht direkt an den VPN-Server versendet, sondern an die UDP-Annahmestelle des lokal instantiierten „Relay Plugins“ übergibt. Der resultierende Aufbau ist in Abbildung 4.28 dargestellt.

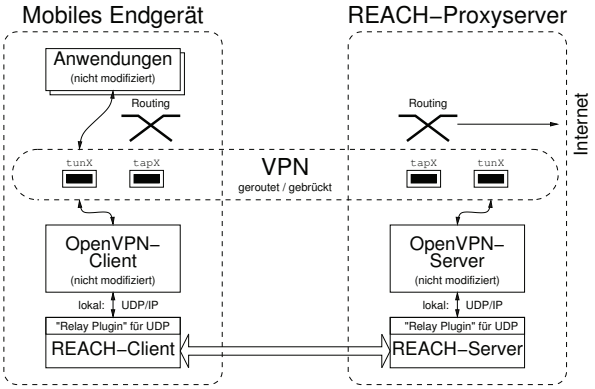

Abbildung 4.28: Wird die vom VPN-Client ausgehende UDP-Assoziation durch den REACH-Client abgefangen und seitens des REACH-Servers zum VPN-Server geleitet, ergibt sich eine handoverfähige Struktur.

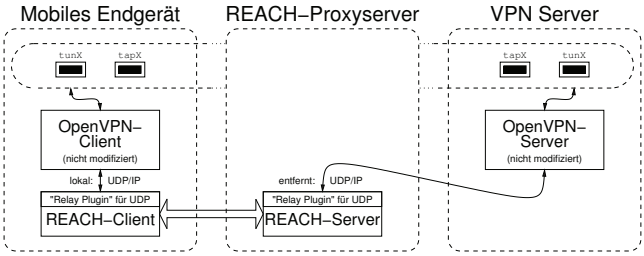

Abbildung 4.29: Der REACH-Server und der VPN-Server können auf unterschiedlichen Systemen installiert sein.

An dieser Stelle muss darauf hingewiesen werden, dass mit dem VPN-basierten Mechanismus zwar ein eigenständiger Dienst für REACH vorgestellt wird, dieser allerdings kein dediziertes „Relay Plugin“ erfordert. Stattdessen wird das bereits vorgestellte und für die Weiterleitung von UDP-Assoziationen gedachte „Relay Plugin“ verwendet. Dennoch soll der VPN-basierte Mechanismus auf „gleicher Ebene“ mit allen anderen Mechanismen vorgestellt werden.

Der VPN-Server braucht dabei nicht direkt auf dem REACH-Proxyserver installiert zu sein. Er kann auch auf einem ganz anderen System installiert sein, wie es in Abbildung 4.29 dargestellt ist. Schließlich muss der Betreiber des REACH-Proxyservers nicht unbedingt derselbe sein wie der Betreiber des VPN-Servers. Der REACH-Proxyserver kann beispielsweise von einer Firma bereit gestellt werden, damit alle mobilen Mitarbeiter

von dessen Vorteilen profitieren können. Zusätzlich kann aber für die Mitarbeiter einer gesonderten internen Abteilung mit gehobenen Sicherheitsanforderungen die Nutzung eines überlagerten VPNs vorgeschrieben sein, wenn diese auf abteilungsinterne Ressourcen zugreifen möchten. Dann wären REACH-Proxyserver und VPN-Server topologisch voneinander getrennt. Anzumerken ist hierbei, dass der Datenaustausch zwischen VPN-Client und VPN-Server mit sehr hoher Wahrscheinlichkeit verschlüsselt stattfinden wird, und somit die ausgetauschten Daten am REACH-Proxyserver nicht entziffert werden können. Somit kann auch ein nicht vertrauenswürdiger REACH-Proxyserver für den VPN-Dienst eingesetzt werden.

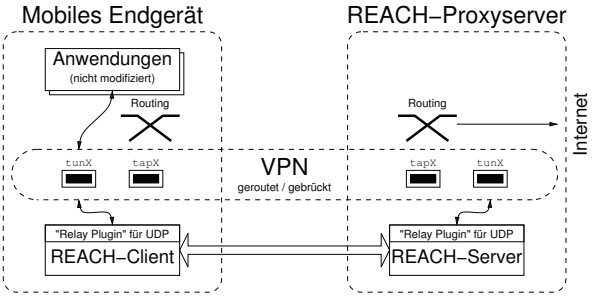

Abbildung 4.30: Im Rahmen einer zukünftigen Weiterentwicklung könnte sogar auf den VPN-Server und die VPN-Clients verzichtet werden. Dazu müssten die „Relay Plugins“ so erweitert werden, dass sie selbstständig mit `tun`- und `tap`-VPN-Adaptern umgehen können.

Die in Abbildung 4.28 gezeigte Architektur kann weiterführend vereinfacht werden, indem die Funktionalität der VPN-Clients und des VPN-Servers in extra „Relay Plugins“ implementiert wird. Es ergäbe sich dann eine Struktur, wie sie in Abbildung 4.30 gezeigt ist. Die Programmierschnittstelle zur Ansteuerung der VPN-Adapter `tunX` und `tapX` ist sehr gut dokumentiert [Kras]. Aus Sicht des „Relay Plugins“ muss eine Gerätedatei geöffnet werden, aus welcher dieser dann „Medium Access Control“-Rahmen (MAC-Rahmen) bzw. IP-Pakete in Form gefüllter Puffer lesen (Daten werden empfangen) und hineinschreiben (Versand) kann. Die Basisfunktionen eines Tunnels könnten somit ohne momentan erkennbare Probleme implementiert werden. Dass bislang dennoch keine Implementierung durchgeführt wurde, hat die folgenden Gründe:

- Innerhalb des VPNs müssen IP-Adressen verwaltet werden. Diese Aufgabe ist bereits zentraler Bestandteil einer VPN-Software, und müsste für REACH implementiert werden.

- Für VPNs bietet sich der Einsatz von Verschlüsselungstechniken an. OpenVPN kommt mit Unterstützung kryptographisch starker Verfahren wie beispielsweise dem „Advanced Encryption Standard“ (AES) daher.
- Der Einsatz der verfügbaren VPN-Systeme gestaltet sich meist als problemlos. Es gibt eine Reihe proprietärer wie auch offener Implementierungen. Die freie Software OpenVPN ist für eine Vielzahl an Betriebssystemen und Plattformen verfügbar.

Theoretisch wäre es möglich, „Relay Plugins“ zu erstellen, welche einen vollwertigen Ersatz für die Funktionen von beispielsweise OpenVPN bieten würden. Gemäß der UNIX-Philosophie [Weis98] ist es allerdings sinnvoller, diese Teilaufgaben weiterhin einer VPN-Software zu überlassen, da diese hierauf spezialisiert ist und aktiv gepflegt wird.

Konfiguration der VPN-Software: Zuallererst muss man sich für eine bestimmte VPN-Software entscheiden. Für eine Auswahl können unterschiedliche Kriterien herangezogen werden. Schreibt beispielsweise der Arbeitgeber eine bestimmte Software vor, erübrigt sich die Auswahl. Dann geht es dem Nutzer nur noch darum, einen zum VPN-Server kompatiblen VPN-Client für sein mobiles Endgerät zu bekommen.

Gibt es dagegen noch keinerlei Vorgaben, kann die Auswahl nach Kriterien wie „Anschaffungskosten“ oder „Verfügbarkeit für unterschiedliche Plattformen“ geschehen. Die endgültige Auswahl hat allerdings keinen Einfluss auf die vorgestellte Architektur; es ist lediglich erforderlich zu wissen, wie die beiden VPN-Instanzen miteinander kommunizieren. Im vorliegenden Fall werden alle auf TCP/IP und UDP/IP basierenden VPN-Systeme unterstützt.

Direkt auf IP aufsetzende VPNs wie „IPsec im Tunnelmodus“ [KeSe05] werden nicht ohne zusätzlichen Aufwand unterstützt, da hierfür noch keine „Relay Plugins“ entwickelt worden sind. Bei einem auf IPsec basierenden VPN-Zugang kommt nämlich keine UDP-Assoziation zum Einsatz, welche durch den hier vorgestellten Mechanismus abgefangen werden könnte. Allerdings wäre es möglich, dass sich ein auf IPsec-basierendes VPN wiederum mit OpenVPN kombinieren lässt, für den Fall, dass ein Anwender gezwungen sein sollte, sich irgendwo per IPsec anzumelden. Als „Unterbau“ dient dann OpenVPN so wie vorgestellt dazu, IP-Pakete über REACH ins Internet zu leiten. Diese IP-Pakete würden dann aber IPsec-Pakete eines überlagerten VPNs sein. Dabei werden also IP-Pakete per IPsec übertragen, welche wiederum durch OpenVPN gekapselt werden. Dieser Ansatz wurde jedoch noch nicht getestet, daher kann hier keine Aussage über eventuell auftretende Probleme gegeben werden. Wenigstens auf die Routingtabelle des mobilen Endgerät muss ein besonderes Augenmerk gelegt werden, damit auch wirklich beide VPNs mit in die Übertragung involviert werden.

Für den Demonstrator wurde die freie VPN-Software OpenVPN ausgewählt. Sie ist plattformübergreifend und kostenlos verfügbar, ist ausgereift und bietet leistungsfähige Verschlüsselungsmechanismen an. Es wurde eine Struktur wie in Abbildung 4.28 gewählt, bei welcher der VPN-Server direkt auf dem REACH-Proxyserver installiert ist. Seine Konfigurationsdatei ist in Abbildung 4.32 abgebildet. Für VPN-Clients werden automatisch IP-Adressen aus dem Subnetz `192.168.10.0/24` vergeben. Die Kommunikation mit den Partnerinstanzen erfolgt über die UDP-Portnummer `1194`. Zudem wird eine auf Zertifikaten beruhende Zugangskontrolle durchgeführt und für den Tunnel auf die AES-Verschlüsselung mit einer Schlüssellänge von 256 Bit gesetzt.

```
dev tun1
client
proto udp
remote 127.0.0.1
port 1194
route 0.0.0.0 0.0.0.0 vpn_gateway

cipher AES-256-CBC
ns-cert-type server
ca openvpn-reach-ca.crt
cert openvpn-reachclient-equinox-neelix.crt
key openvpn-reachclient-equinox-neelix.key

resolv-retry infinite
comp-lzo
keepalive 10 60
ping-timer-rem
persist-tun
persist-key
nobind
```

Abbildung 4.31: Die Konfigurationsdatei des OpenVPN-Clients

Die Konfigurationsdatei eines OpenVPN-Clients ist in Abbildung 4.31 dargestellt. Einige der hier aufgeführten Parameter sind identisch mit denen der Konfiguration des VPN-Servers (siehe Abbildung 4.32), andere sind dagegen eindeutig für die Rolle als VPN-Client relevant. So wird beispielsweise mit Hilfe des `route`-Parameters eine „Default Route“ gesetzt, welche durch den Tunnel hindurch auf den VPN-Server zeigt. Ein sehr wichtiger Parameter ist zudem die Adresse des VPN-Servers. Dieser muss vom VPN-Client kontaktiert werden können, um das VPN zu bilden. In diesem Fall ist der VPN-Server über die IP-Adresse (Parameter `remote`) `127.0.0.1` und Portnummer (Pa-

rameter `port`) `1194` mit Hilfe von UDP (Parameter `proto`) zu erreichen. Dieses ist jedoch nicht die Adresse des VPN-Servers, sondern die der „Relay Plugin“-Instanz am lokal gestarteten REACH-Client.

```
dev tun1
server 192.168.10.0 255.255.255.0
proto udp
port 1194
ifconfig-pool-linear
ifconfig-pool-persist reachserver-ipp.txt
client-to-client
client-config-dir reachserver-ccd

cipher AES-256-CBC
ca openvpn-reach-ca.crt
cert openvpn-reachserver-equinox.crt
key openvpn-reachserver-equinox.key
dh openvpn-reachserver-dh2048.pem
#crl-verify openvpn-reachserver-crl.pem

comp-lzo
keepalive 10 60
ping-timer-rem
persist-tun
persist-key
```

Abbildung 4.32: Die Konfigurationsdatei des OpenVPN-Servers

Es sind noch weitere Dateien notwendig, welche durch beide Konfigurationsdateien referenziert werden. Diese dienen der „Public Key“-Infrastruktur und verweisen auf Zertifikate, private Schlüssel sowie eine Liste mit zwischenzeitlich gesperrten Zertifikaten. Diese Dateien müssen gemäß der OpenVPN-Installationsanleitung [OVHT] im Vorfeld angelegt werden.

Konfiguration der „Relay Plugins“: Da beim Einsatz von OpenVPN lediglich eine einzige UDP-Assoziation abgefangen werden muss, bietet sich die auf die Weiterleitung von UDP-Assoziationen spezialisierten „Relay Plugins“ aus Abschnitt 4.3.2.4 an.

Es wird lediglich eine einzige Instanz benötigt, welche auf dem mobilen Endgerät die ausgehende UDP-Assoziation des OpenVPN-Clients abfängt. Die Annahme von Assoziationen muss dabei auf demselben UDP-Port passieren, welcher vom VPN-Client kontaktiert wird. UDP-Assoziationen werden dabei gemäß Abbildung 4.33 auf Port

```
<RelayPlugins>
  <RelayPlugin loadlibrary="librrpc_udprelay_forward.so">
    <Instance>
      <Configuration>
        <Item key="source port" value="1194" />
        <Item key="destination ip" value="127.0.0.1" />
        <Item key="destination port" value="1194" />
      </Configuration>
      <Discovery>
        <Item key="use reachserver" value="Equinox" />
      </Discovery>
    </Instance>
  </RelayPlugin>
</RelayPlugins>
```

Abbildung 4.33: Für den VPN-Dienst muss eine einzige UDP-Assoziation abgefangen werden, wofür sich die bereits vorgestellten „Relay Plugins“ zwecks Weiterleitung von ausgehend aufgebauten UDP-Assoziationen eignen.

`"1194"` angenommen und seitens des „Relay Plugins“ am REACH-Proxyserver an Adresse `"127.0.0.1"` mit Port `"1194"` weitergeleitet. Dort wartet schlussendlich der VPN-Server auf die UDP-Pakete der VPN-Clients.

Es ist erforderlich, auf die Dienstsuche Einfluss zu nehmen. Da der Weiterleitungsmechanismus prinzipiell eine Vielzahl an REACH-Proxyservern involvieren darf, aber nur auf einem einzigen Server auch der OpenVPN-Server installiert ist, muss die Weiterleitung auf den korrekten REACH-Proxyserver beschränkt werden. Das geschieht mit Hilfe des Parameters `"use reachserver"`, welcher hier den REACH-Proxyserver `"Equinox"` spezifiziert. Dieser Parameter wurde bereits in Abschnitt 4.3.1.3 vorgestellt.

4.3.2.6 SIP-basierte Internettelefonie

Für die Betrachtung von Internettelefonie wurde die SIP-basierte Protokollfamilie ausgewählt. Für sie sind eine Vielzahl an „offenen“ Programmen erhältlich, angefangen mit der „Private Branch Exchange“ (PBX, engl. für Vermittlungsstelle) Asterisk [Aste], diversen „Softphones“ wie X-Lite [XLte], Twinkle [dBoe] oder KPhone [Kphn] und Protokollanalysatoren wie Wireshark [Wshk]. In Verbindung mit der umfassenden Dokumentation erhält man schnell eine lauffähige und erweiterbare Testumgebung.

Funktionsweise von SIP-basierter Telefonie Bei SIP-basierter Telefonie wird eine ganze „Familie“ an Protokollen eingesetzt. SIP wird zu Signalisierungszwecken verwen-

det und dient dem Auf- und Abbau von Sitzungen. Worum es sich genau bei einer solchen Sitzung handelt, also dass es sich um ein Telefonat für einem bestimmten „Coder and Decoder“ (Codec) handelt, kommuniziert das „Session Description Protocol“ (SDP, RFC 4566 [HaJP06]). Auch Adressinformationen bezüglich des Austauschs von Mediendaten werden mit Hilfe von SDP ausgehandelt. SDP-Nachrichten werden als Nutzlast bestimmter SIP-Nachrichten übertragen. Für den eigentlichen Transport der Mediendaten kommt das „Real-Time Transport Protocol“ (RTP, RFC 3550 [SCFJ03]) zum Einsatz, welches vom „RealTime Control Protocol“ (RTCP, ebenfalls RFC 3550) zum Zwecke des Aushandelns und Überwachens von „Quality-of-Service“ (QoS)-Parametern begleitet wird. SIP kann wahlweise über UDP oder TCP übertragen werden, während RTP und RTCP ausschließlich UDP verwenden. Ein TCP-basierter Austausch von SIP-Nachrichten konnte jedoch nicht beobachtet werden, sodass beschlossen wurde, nur die UDP-basierte Variante zu berücksichtigen.

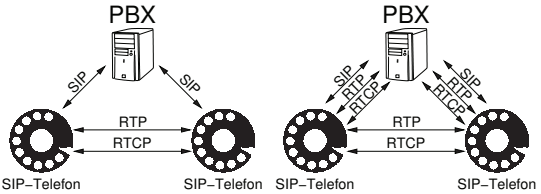

Abbildung 4.34: Bei SIP-basierter Internettelefonie kommen SIP-Telefone und eine so genannte „Private Branch Exchange“ (PBX) zum Einsatz. Je nach Betriebsmodus können *auch* von der PBX Mediendatenströme ausgehen.

Abbildung 4.34 zeigt die bei SIP-basierter Telefonie involvierten Komponenten. Zum einen gibt es eine Menge an SIP-Telefonen („SIP User Agents“), welche entweder als Software in Erscheinung treten („Softphone“) oder sich für den Nutzer wie ein „gewöhnliches“ Telefon darstellen („Hardphone“). Eine zentrale Komponente stellt die Vermittlungsstelle (PBX) dar. Sie vermittelt SIP-Nachrichten zwischen SIP-Telefonen und stellt zudem ein Aufenthaltsregister zur Verfügung („SIP Registrar“).

Ein jedes SIP-Telefon muss Daten mit „seiner“ PBX sowie mit anderen SIP-Telefonen austauschen können. Im linken Diagramm von Abbildung 4.34 ist der „gewöhnliche“ Fall gezeigt, bei welchem die PBX lediglich für Signalisierungszwecke benutzt wird ein Austausch von Mediendatenströmen nur zwischen SIP-Telefonen erfolgt. Die im Demonstrator verwendete PBX Asterisk zeigt jedoch das im rechten Diagramm gezeigte Schema, wobei auch ein Austausch von Mediendaten mit der PBX festgestellt werden kann. Hiermit sollen Meldungen und Töne übertragen werden können, wie man sie beispielsweise von einer Tarifansage her gewohnt ist. Das ist eine wichtige Erkenntnis, denn zu einem

einzigen Telefonat gehören dann Mediendatenströme zu zwei Kommunikationspartnern, sodass am SIP-Telefon insgesamt fünf Datenströme anfallen.

Probleme im mobilen Umfeld In der Praxis können Schwierigkeiten auftreten, wenn SIP-basierte Internettelefonie eingesetzt werden soll. Auf das so genannte „NAT- und Firewall-Problem“ stößt man bereits bei üblichen Heiminstallationen, während man im mobilen Umfeld noch mit zusätzlichen Herausforderungen konfrontiert wird. Diese sollen im Folgenden beleuchtet werden.

- **Das NAT-Problem**: Das so genannte NAT-Problem tritt auf, sobald sich ein SIP-Telefon innerhalb eines *privaten Netzes* befindet und seine Kommunikationspartner oder die PBX nur durch Überschreitung einer „NAT-Grenze“ erreicht werden können. Hierbei treten sowohl hinsichtlich der Übertragung von Signalisierungs- als auch von Mediendaten Probleme auf.

  Bezüglich SIP muss die zugehörige UDP-Assoziation daher *immer* aus dem privaten Netz heraus initiiert werden, sodass die NAT-Grenze anfangs von innen nach außen überschritten wird und ein Eintrag in der NAT-Tabelle hinterlegt wird. Damit Antworten auf die Signalisierungsnachrichten auch wieder in Rückrichtung die NAT-Grenze passieren können, muss dieselbe UDP-Assoziation verwendet werden. SIP sieht hierfür den Betriebsmodus des „SIP Symmetric Response Routing“ vor, welcher in der PBX aktiviert werden muss. Da Einträge bereits nach 180 Sekunden Inaktivität wieder aus den NAT-Tabellen verschwinden, muss in den SIP-Telefon die Versendung so genannter „Keep-Alive“-Pakete aktiviert werden.

  Mediendatenströme, welche beispielsweise Daten eines Telefongesprächs enthalten, werden über SIP-`INVITE`-Nachrichten aufgebaut. Hierfür schicken beide SIP-Telefone ihrer Partnerinstanz Adressinformationen bezüglich lokal geöffneter UDP-Ports, an welchen Mediendaten entgegen genommen werden sollen. Das SIP-Telefon im privaten Netz besitzt jedoch keine öffentliche IP-Adresse, sodass die in der SIP-Nachricht eingebetteten Adressinformationen für die Gegenstelle wertlos sind. In der Praxis fällt dies durch „simplexartige“ Telefonate auf, wobei ein Teilnehmer den Anderen hören kann, jedoch der Rückkanal nicht funktioniert.

  Da NAT häufig vorgefunden wird, wurden hierfür bereits Lösungsmechanismen entwickelt. So stehen beispielsweise das „Simple Traversal of UDP through NAT“ (STUN, RFC 5389 [RMMW08]), das „Traversal Using Relay NAT“ (TURN) und das „Universal Plug and Play“ (UPnP, [Wiki10]) zur Auswahl. STUN benutzt „Hilfsserver“ im Internet, um die an der NAT-Grenze „nach außen hin“ sichtbare IP-Adresse und Absenderportnummer zu ermitteln. Die Idee von STUN basiert

darauf, dass theoretisch auch die UDP-Pakete anderer Kommunikationspartner an das hinter der NAT-Grenze liegende System weitergeleitet werden sollten, wenn die Pakete an genau diese vorab ermittelte IP-Adress-Port-Kombination adressiert werden (erfordert Betriebsmodus „Full Cone NAT“). Da der LINUX-Kernel jedoch kein „Full Cone NAT“, sondern „symmetrisches NAT“ durchführt, und DSL-Router häufig auf LINUX basieren, kann STUN nicht eingesetzt werden. TURN kommt dagegen mit „symmetrischem NAT“ zurecht, und greift bei Überschreitung von NAT-Grenzen ebenfalls auf die Unterstützung durch externe Server zurück. Diese externen Server dienen als Vermittlungsstellen, wobei dann keine Überschreitung von NAT-Grenzen „von außen nach innen“ mehr erforderlich ist. TURN basiert jedoch auf einem proprietären Protokoll, welches von den SIP-Telefonen unterstützt werden muss. UPnP wiederum ist „offen“ und eignet sich dazu, an der NAT-Grenze eine eingehende UDP-Assoziation anzumelden und dabei die öffentlich sichtbare IP-Adresse und Portnummer in Erfahrung zu bringen. Von Nachteil ist jedoch, dass nicht jeder NAT-Router UPnP unterstützt und dass UPnP seitens der SIP-Telefone unterstützt werden muss [NATT].

Ein weiterer Ansatz liegt in der Verwendung so genannter „Intermediaries“ (engl. *Mittelsleute*). Es handelt sich um „Application Layer Gateways“ (ALGs), welche in der Lage sind, Protokolldatenströme der Anwendungsschicht abzufangen und gegebenenfalls zu ändern. Ein ALG kann beispielsweise die mittels SDP übertragenen Adressinformationen manipulieren und damit die Überquerung von NAT-Grenzen ermöglichen. Im Fall von VoIP werden solche Systeme je nach Umfang als „Back-to-back User Agent“ (B2BUA) oder als „Session Border Controller“ (SBC) bezeichnet. Sie stellen ein Gateway sowohl für die Signalisierungs- als auch für die Mediendatenströme dar und müssen direkt an der NAT-Grenze positioniert werden. Als nachteilig erweist sich die nicht mehr vorhandene „Ende-zu-Ende-Semantik“, und auf eine Verschlüsselung der Signalisierungsdaten muss verzichtet werden.

- **Das Firewall-Problem**: Für die Übertragung von Mediendatenströmen öffnen die beteiligten SIP-Telefone bestimmte UDP-Ports, wobei es je nach Software passieren kann, dass hierbei keine Vorgaben gemacht werden können. Es werden dann wahlfrei UDP-Ports reserviert, auf denen das SIP-Telefon auf eingehende UDP-Pakete wartet. Ist auf dem betroffenen System eine Firewall installiert, werden die eingehenden Pakete jedoch gefiltert und eine Sprachkommunikation kommt nicht zu Stande. Das SIP-Telefon muss also auf die Verwendung bestimmter Portnummern oder -bereiche beschränkt werden, sodass diese an der Firewall explizit freigegeben werden können. Kann eine solche Begrenzung jedoch nicht durchge-

führt werden, müsste die Firewall bezüglich UDP komplett „geöffnet“ werden, was wiederum Sicherheitsprobleme aufwirft.

- **Sich ändernde IP-Adressen**: Im Fall eines Handovers kommt es zu einem Adresswechsel seitens eines Kommunikationspartners, was, wie bereits allgemein erläutert wurde, problematisch ist. Bei SIP-basierter Telefonie werden zudem IP-Adressen in den Protokolldatenstrom eingebettet, was die Unterstützung eines Adresswechsels abermals komplizierter macht. Im schlimmsten Fall werden alle bestehenden Telefonate unterbrochen, und das betroffene SIP-Telefon kann erst nach einer erneuten Anmeldung am SIP-Registrar wieder erreicht werden.

- **Sitzungsmanagement**: Im mobilen Umfeld kann es passieren, dass ein Teilnehmer vom Netz getrennt wird und eine Zeit lang isoliert bleibt. In dieser Zeitspanne kann einiges passieren: ein Teilnehmer kann beispielsweise auflegen und sogar seinen Computer herunterfahren. Da keine Verbindung zum Netz besteht, wurde die Sitzung in diesem Fall nur einseitig beendet, während die andere Seite davon nichts mitbekommen hat. Steht später wieder eine Verbindung zur Verfügung, wird es zu einem Sitzungsfehler kommen, da keine Partnerinstanz mehr für einen ordentlichen Sitzungsabbau zur Verfügung steht. Es drohen somit inkonsistente Zustände bezüglich der PBX und der beteiligten SIP-Telefone untereinander.

- **Isolationsverhalten**: Wird ein mobiles Endgerät isoliert, kann es zu Fehlerfällen bezüglich des gestarteten SIP-Telefons kommen, denn jedes SIP-Telefon muss sich zyklisch am SIP-Registrar melden. Ist jedoch keine Kommunikation möglich, erhält das SIP-Telefon keine Quittungen mehr von der PBX, sodass die Anmeldung fehlschlägt. Das Programm X-Lite wechselt hierbei in einen Fehlerzustand, aus dem es von alleine nicht mehr herauszukommen versucht. Der Nutzer muss dies bemerken und dann manuell eine Neuanmeldung anstoßen.

  Ein weiteres Problem betrifft die Übertragung von Mediendaten während eines bestehenden Telefongesprächs. Wird die Verbindung zum Internet getrennt, können keine Mediendaten mehr ausgetauscht weden, und das Gespräch verstummt. Für eine Mobilitätsunterstützung ist ein solches Verhalten nicht praktikabel, da die Nutzer hierdurch irritiert werden und sie das Gespräch beenden, anstatt bloß abzuwarten. Eine Lösung besteht in der Einspielung einer künstlichen Ansage, welche beide Teilnehmer auf die Isolationssituation hinweist und sie um Geduld bittet.

Möchte man eine brauchbare Unterstützung von SIP-basierter Sprachkommunikation im mobilen Umfeld anbieten, müssen die genannten Probleme ausnahmslos gelöst werden. Eine Erweiterung der SIP-Telefone oder der PBX soll vermieden werden.

Eignungsprüfung bisheriger „Relay Plugins“: Es muss festgestellt werden, dass alle bislang vorgestellten „Relay Plugins“ nicht in der Lage sind, mit SIP-basierter Sprachkommunikation umzugehen. SOCKSv5 ist ungeeignet, da dessen `UDP-ASSOCIATE`-Kommando zwar UDP-Assoziationen unterstützt, man jedoch hierbei keine *benachbarten* Portnummern anfordern kann. Einzig das VPN-basierte Schema kann in der Lage sein, *manche* der aufgeführten Probleme zu lösen. Dazu darf am VPN-Server kein NAT durchgeführt, und am mobilen Endgerät muss sichergestellt werden, dass das Softphone mit einer lokalen Firewall zusammenspielt. Jedoch stehen in diesem Szenario keine Unterstützungsmechanismen für Isolationssituationen bereit.

Aufteilung eines „Session Border Controllers“: Es war unumgänglich, für SIP-basierte Telefonie dedizierte „Relay Plugins“ zu erstellen. Die Problematik der sich bei einem Handover ändernden IP-Adressen wird dabei bereits durch den Kern von REACH gelöst. Allerdings ist es nicht ausreichend, bloß UDP-Pakete abzufangen und zwischen REACH-Client und REACH-Server zu tunneln. Stattdessen müssen alle SIP-Nachrichten interpretiert und sorgfältig zu Sitzungen gruppiert werden. Enthaltene Adressinformationen müssen so verändert werden, dass sämtliche Mediendatenströme ebenfalls durch die „Relay Plugins“ geleitet werden und dass dabei weder das NAT- noch das Firewallproblem eine Rolle spielen. Während Isolationssituationen sollen künstliche Signalisierungsnachrichten generiert werden, um Fehlersituationen wegen ausbleibender Bestätigungen zu verhindern. Da zudem die Nutzer nicht verunsichert werden dürfen, soll in bestehende Sprachverbindungen beidseitig eine künstliche Ansage eingespielt werden.

Die dazu notwendige Funktionalität entspricht der eines vollwertigen „Session Border Controllers“ (SBC), welcher auf zwei „Relay Plugins“ verteilt wird und zudem Mechanismen zur Behandlung handoverspezifischer Problemstellungen aufweist. Da diese Aufgaben sehr komplex sind, wurde zunächst ein eigenständiger SBC entwickelt, welcher anschließend in einen client- und einen PBX-seitigen Teil getrennt worden ist. Jeder Teil ergab schlussendlich ein „Relay Plugin“ für REACH [EvYS08].

Der hierfür umgesetzte SBC wird als SIP-RTP-Proxy (SRP) bezeichnet. Er ist für sich gesehen bereits in der Lage, sowohl das NAT- als auch das Firewallproblem zu lösen. Zudem muss er im Gegensatz zu herkömmlichen SBCs nicht direkt auf dem NAT-Gateway installiert werden, sondern kann auch auf einer beliebigen Maschine innerhalb eines lokalen Netzwerks gestartet werden. Am NAT-Gateway müssen hierbei lediglich Regeln für zwei eingehende UDP-Portweiterleitungen hinterlegt werden. Für eventuelle Firewalls kann ein begrenzter Portnummernbereich spezifiziert werden, wobei „nach Außen hin“ die Öffnung von nur drei Ports ausreichend ist (für SIP, RTP und RTCP). Der SRP ist in [YeES08] veröffentlicht worden, wobei dort neben einer ausführlichen Darle-

gung der NAT- und Firewallproblematik tiefgreifend auf den hier nur knapp umrissenen Lösungsweg eingegangen wird.

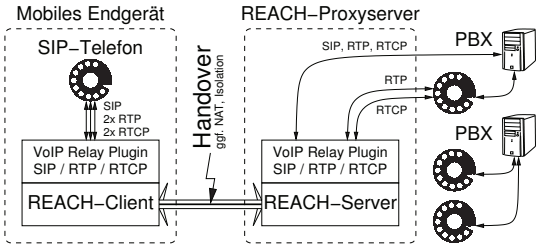

Abbildung 4.35: Die am REACH-Client und am REACH-Server geladenen „Relay Plugins“ ergeben in ihrer Summe einen „Session Border Controller“, der um handoverspezifische Funktionalität erweitert worden ist. Alle Datenströme werden durch REACH abgefangen und entsprechend behandelt. Dabei gelten an einem REACH-Server keine Einschränkungen bezüglich der Menge der externen Vermittlungsstellen (PBX) sowie der Erreichbarkeit der dort registrierten SIP-Telefone.

Nach Aufteilung des SBCs und der Erstellung der beiden „Relay Plugins“ ergab sich die in Abbildung 4.35 dargestellte Architektur. Auf dem mobilen Endgerät müssen sowohl der Signalisierungsdatenstrom (SIP) als auch die Mediendatenströme (RTP und RTCP) abgefangen werden. Das „Relay Plugin“ am REACH-Client kommuniziert lediglich mit SIP-Telefonen, während das Gegenstück am REACH-Server neben anderen SIP-Telefonen auch mehrere PBX als Kommunikationspartner aufweisen kann. Die zu verwendende PBX kann von jedem Nutzer an „seiner“ lokalen „Relay Plugin“-Instanz benannt werden.

Die dargestellte Architektur fängt sämtliche Datenströme in beide Richtungen ab und bietet eine handoverresistente Sitzungsverwaltung. Dabei müssen weder die clientseitigen SIP-Telefone, die der Kommunikationspartner noch die PBX modifiziert werden. Einzig die SIP-Telefone auf dem mobilen Endgerät müssen so parametriert werden, dass sie als PBX das lokal instantiierte „Relay Plugin“ benutzten, welches beispielsweise auf die Adresse UDP `127.0.0.1:5060` gebunden wird. Damit steht dem SIP-Telefon eine *virtuelle PBX* zur Verfügung, welche auch im Fall einer Isolationssituation erreichbar ist. Die Adresse der eigentlichen PBX wird dem lokal laufenden „Relay Plugin“ als Konfigurationsparameter mitgegeben. Diese Adresse wird schließlich an die Partnerinstanz am REACH-Proxyserver kommuniziert.

Sitzungsverwaltung: Sowohl der Signalisierungsdatenverkehr (SIP) als auch die Mediendatenströme (RTP und RTCP) müssen beidseitig durch die „Relay Plugins“ abgefangen werden. Signalisierungsdaten werden dabei jedoch anders behandelt als Mediendaten:

- **Signalisierungsdaten**: Jedes SIP-Telefon tauscht Signalisierungsnachichten mit einer PBX aus, wobei beim Einsatz von REACH diese Nachrichten durch die „Relay Plugins“ abgefangen werden. Zwischen serverseitigem „Relay Plugin“ und einer PBX genügt eine einzige UDP-Assoziation, welche für alle abgefangenen SIP-Telefone aller angemeldeten REACH-Clients verwendet wird. SIP-Nachrichten sind textbasiert und können anhand eindeutiger Identifikatoren zu SIP-Sitzungen gruppiert werden. Die hierfür relevanten Informationen sind die `Call-ID`, das `tag`-Feld sowie die Nutzernamen [RSCJ+02].

  Da manche SIP-Nachrichten wiederum SDP-Nachrichten enthalten, müssen diese sorgfältig auf Adressinformationen hin untersucht werden. Zudem muss REACH dafür sorgen, dass alle Mediendatenströme wiederum an die „Relay Plugins“ gerichtet werden. Der Transport der abgefangenen und zwischenzeitlich modifizierten SIP-Nachrichten zwischen „Relay Plugin“-Instanzen erfolgt mittels einer paketorientierten *logischen Verbindung*.

- **Mediendaten**: RTP- und RTCP-Datenströme werden mittels UDP transportiert. Da die involvierten IP-Adressen und Portnummern mit Hilfe von SDP übertragen werden, sind sie den „Relay Plugins“ bekannt. Die Adressen werden vermerkt und dahingehend geändert, dass der spätere Empfänger der Signalisierungsnachrichten seine Mediendatenströme an das „Relay Plugin“ richtet. Dadurch wird sichergestellt, dass RTP- und RTCP-Datenströme immer durch REACH geleitet werden. Auch hier kommen *paketorientierte logische Verbindungen* zum Einsatz.

Verwaltung von Portnummern: Für RTP und RTCP werden benachbarte UDP-Portnummern benötigt, wobei bezüglich RTP eine *gerade* Portnummer vorgegeben werden muss und RTCP die benachbarte Portnummer verwendet. Für dieses paarweise Belegungsschema reservieren die „Relay Plugins“ im Vorfeld einen „durchgehenden Block“ an UDP-Ports, sodass genügend „Portpaare“ zur Verfügung stehen. Des Weiteren wurden die *Relay Plugins* so gestaltet, dass ein einzelnes Portpaar eine Vielzahl an UDP-Assoziationen bedienen kann, also mehrere SIP-Telefone gleichzeitig Mediendaten über dasselbe Portpaar senden und empfangen können.

Eine Instanz am REACH-Client benötigt genau fünf Portnummern, was in Abbildung 4.36 gezeigt ist und eine Folge des in Abbildung 4.34 dargestellten Umstandes ist, dass

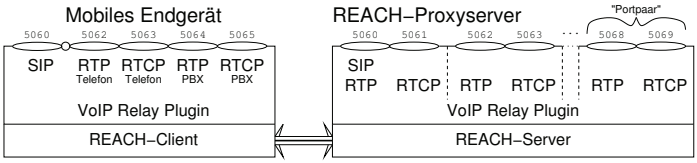

Abbildung 4.36: Das „Relay Plugin“ am REACH-Client weist ein anderes Portbelegungsschema auf als das am REACH-Server. Während am REACH-Client genau fünf UDP-Ports pro Instanz benötigt werden, kann serverseitig eine Vielzahl an „Portpaaren“ involviert werden.

ein SIP-Telefon auch von seiner PBX Mediendatenströme erhalten kann. Da das Softphone X-Lite für beide RTP-Datenströme (in Richtung der PBX und des Partnertelefons) dieselbe Absenderportnummer verwendet, musste die Eindeutigkeit beider UDP-Assoziationen durch unterschiedliche Portpaare am clientseitigen „Relay Plugin“ sichergestellt werden. Nichtsdestotrotz gibt es keine Einschränkungen in Bezug auf die Menge der unterstützten SIP-Telefone und der Anzahl an gleichzeitig führbaren Telefongesprächen, welche von einer einzigen Instanz verwaltet werden können.

Bezüglich einer Instanz am REACH-Server reicht sogar ein einziges Portpaar aus, da PBX und Partnertelefone immer anhand ihrer Adressen unterschieden werden können. Der erste Port des ersten Paares wird zudem für SIP mitbenutzt.

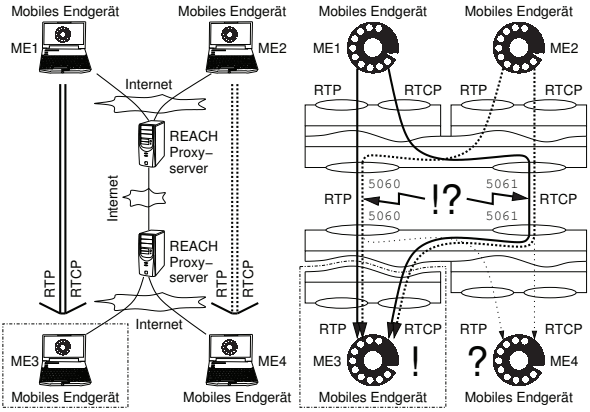

Abbildung 4.37: UDP-Pakete aus unterschiedlichen Telefonaten werden miteinander vermischt, wenn mehrere Gespräche zwischen zwei serverseitigen „Relay Plugin“-Instanzen stattfinden. Die Diagramme zeigen nur eine Übertragungsrichtung, in Rückrichtung besteht das Problem ebenfalls.

Probleme tauchen allerdings auf, wenn mehrere „Relay Plugin“-Instanzen existieren, und sich mobile SIP-Telefone gegenseitig anrufen sollen. So lange beide SIP-Telefone dieselbe „Relay Plugin“-Instanz am selben REACH-Server benutzen, kommt es zu keinem Problem, denn hier wird das Gespräch bereits intern „gebrückt“. Werden allerdings unterschiedliche Instanzen involviert, weil beispielsweise wie in Abbildung 4.37 verschiedene REACH-Server beteiligt sind, kann die Eindeutigkeit von UDP-Assoziationen nicht länger gewährleistet werden. So schickt hier das mobile Endgerät **ME1** Daten an **ME3**, während **ME2** Daten an **ME4** sendet. Kommt wie dargestellt an den serverseitigen Instanzen nur ein einziges Portpaar zum Einsatz, weisen Pakete beider UDP-Assoziationen bezüglich RTP dieselbe Absender- und Zieladresse auf. Die UDP-Pakete können nicht mehr getrennt werden, und es kommt zu einer Vermischung der Datenströme.

Gelöst wurde dieses Problem dadurch, dass pro serverseitiger Instanz mehrere Portpaare benutzt werden können, was in Abbildung 4.36 verdeutlicht wird. Diese zusätzlichen Portpaare sind nur für den vorgestellten Sonderfall relevant. Die Anzahl verfügbarer Portpaaren bestimmt, wie viele gleichzeitige Telefongespräche mit einer anderen serverseitigen „Relay Plugin“-Partnerinstanz abgewickelt werden können.

Betrachtung des NAT- und Firewallproblems: Da bei typischen Heiminstallationen nur eine einzige öffentliche IP-Adresse zur Verfügung steht, muss NAT durchgeführt werden. Die Nutzung von NAT bringt jedoch nicht vorhersagbare Adressumsetzungen mit sich, was für SIP-basierte Telefonie problematisch ist. Für REACH wurde eine Lösung gefunden, welche die Verwendung von UDP-Weiterleitungsregeln vorsieht. Dieses Schema wird praktisch von allen NAT-Gateways unterstützt, erfordert jedoch den einmaligen Eingiff des Administrators zwecks Hinterlegung dieser Regeln.

Ein solches Szenario ist in Abbildung 4.38 dargestellt. Hierbei ist der REACH-Proxyserver eine eigenständige Maschine, welche sich im LAN „hinter“ der NAT-Grenze befindet. Damit auch die von externen Telefonen gesendeten Mediendaten (**II**) die „Relay Plugin“-Instanz am REACH-Proxyserver erreichen, müssen für alle Portpaare Weiterleitungsregeln an der NAT-Grenze definiert werden. Bei fünf Portpaaren währen das zehn Weiterleitungsregeln, welche die aus dem Internet eintreffenden UDP-Pakete an den REACH-Proxyserver weiterleiten. Hierbei gibt es keine Probleme mit „Timeouts“, und die Menge der an Firewalls zu öffnenden Portnummern ist bekannt.

Alle mobilen Endgeräte, welche mit einem REACH-Client ausgestattet sind, kommunizieren ausschließlich mit dem REACH-Proxyserver, wofür REACH-eigene Kommunikationsprotokolle (**I**) zum Einsatz kommen. Die auf dem mobilen Endgerät gestarteten SIP-Telefone bekommen von der NAT-Grenze nichts mit, da sich „ihre“ Kommunikationspartner scheinbar immer auf dem lokalen System befinden.

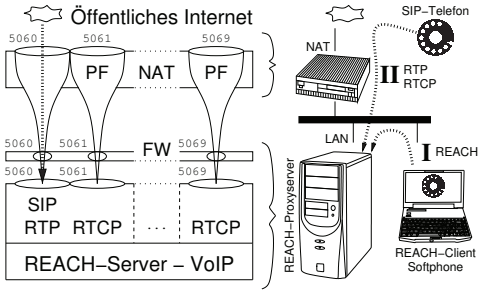

Abbildung 4.38: Mit REACH lassen sich sowahl das NAT- als auch das Firewallproblem lösen, selbst wenn sich der REACH-Proxyserver *hinter* der NAT-Grenze befindet. In diesem „Heimnutzungsszenario“ müssen *nach innen* gerichtete UDP-Weiterleitungsregeln („Port Forwarding“ (PF)) an der NAT-Grenze hinterlegt sowie Ausnahmeregeln für die Firewall (FW) definiert werden.

Sowohl das NAT- als auch das Firewallproblem wurden damit gelöst, wäre da nicht noch ein letzter Haken: es kann passieren, dass die Internetverbindung an der NAT-Grenze getrennt wird, und das NAT-Gateway eine neue öffentliche IP-Adresse erhält. In diesem Fall sind die Mechanismen von REACH unwirksam, und schlimmer noch, die serverseitige „Relay Plugin“-Instanz kennt dann die extern sichtbare IP-Adresse der NAT-Grenze nicht mehr. Diese Adresse ist allerdings notwendig, damit SIP-`INVITE`-Nachrichten verschickt werden können, welche den SIP-Telefonen im Internet mitteilen, zu welcher IP-Adresse RTP- und RTCP-Pakete gesendet werden sollen. Hierfür ist noch keine Lösung verfügbar. Es müsste weiterführend ein Zusammenspiel mit dem NAT-Gateway untersucht werden, um dessen neue IP-Adresse in Erfahrung bringen zu können.

Verhalten im Isolationsfall: Charakteristisch ist für mobile Endgeräte vor allem, dass sie für eine längere Zeit vom Internet getrennt werden können, sie also isoliert sind. Ein besonderes Augenmerk benötigen hier die SIP-Telefone, welche keine Fehlerzustände betreten sollen, sowie die Benutzer, welche nicht den Eindruck abgerissener Telefongespräche gewinnen sollen, welcher sie zum Auflegen verleiten würde.

- **Signalisierungsdaten**: Solange ein mobiles Endgerät über eine Verbindung zum Internet verfügt, können Daten zwischen miteinander assoziierten „Relay Plugin“-Instanzen ausgetauscht werden. Sowohl am REACH-Client als auch am REACH-Server haben die „Relay Plugins“ einen Überblick über alle bestehenden Sitzungen und die zugehörigen Mediendatenströme.

Wird das mobile Endgerät jedoch isoliert, ist kein Datenaustausch mehr mit der PBX möglich, und es können keine Telefongespräche mehr aufgebaut werden. Da SIP-Telefone in diesem Fall keine Quittungen mehr von der PBX erhalten, wechseln sie in einen Fehlerzustand, sobald eine bestimmte Anzahl an Versuchen fehlgeschlagen ist. Daher ist das clientseitige „Relay Plugin“ in der Lage, im Isolationsfall künstliche SIP-Nachrichten zu erstellen, damit die hiesigen SIP-Telefone keinen Unterschied bemerken. Hierbei findet jedoch keine „Auffrischung“ am SIP-Registrar statt, und die SIP-Telefone drohen unerreichbar zu werden, selbst wenn die Internetverbindung später wieder besteht. Um die SIP-Telefone zu einer raschen Neuanmeldung zu zwingen, verweisen die künstlich erzeugten SIP-Nachrichten auf einen „Sitzungs-Timeout“ von nur fünf Sekunden, sodass sich die SIP-Telefone mit gesteigerter Frequenz wieder und wieder anmelden werden. Erst wenn die Internetverbindung wieder besteht und der SIP-Registrar erreicht werden kann, gilt wieder ein „gewöhnlicher Sitzungs-Timeout“ von beispielsweise 3600 Sekunden.

Eine weitere Aufgabe, die beide „Relay Plugin“-Instanzen umfasst, betrifft den geordneten Abbau von Sitzungen. Zwar können während einer Isolationssituation keine neuen Telefonate aufgebaut, jedoch bestehende Telefonate beendet werden: ein Nutzer braucht hierzu lediglich sein SIP-Telefon „aufzulegen“. Das jeweilige „Relay Plugin“ erkennt diesen Fall anhand der zugehörigen SIP-Nachrichten und ist in der Lage, eigenmächtig einen ordentlichen Sitzungsabbau durchzuführen. Sobald die Isolationssituation beendet wurde, wird die Partnerinstanz über die getrennte Sitzung informiert, sodass sie ihrerseits einen geordneten Sitzungsabbau durchführen kann. Beide „Relay Plugin“-Instanzen agieren im Isolationsfall autonom und erzeugen eigene SIP-Nachrichten. Diese Fähigkeit macht REACH zu einem „Session Border Controller“, denn ein „Back-to-Back User Agent“ arbeitet lediglich statuslos und kann selbst keine Signalisierungsnachrichten erzeugen.

- **Mediendaten**: Es ist unvermeidlich, dass während einer Isolationssituation keine Sprachdaten mehr übertragen werden können. Dennoch kann beruhigend auf die Gesprächspartner eingewirkt werden, sodass sie nicht vorschnell auflegen. Für REACH wurde hierfür eine praktikable Lösung gefunden: während einer Insolationssituation spielen die „Relay Plugins“ beidseitig eine Ansage ein, welche die Teilnehmer auf die zeitweilige Unterbrechung hinweist. Am REACH-Server kann zudem eine andere Ansage hinterlegt werden als am REACH-Client, da der mobile Nutzer sich der Trennungssituation eher bewusst ist als ein Teilnehmer im Festnetz.

  Sobald die Isolationssituation beendet wurde und wieder RTP-Pakete ausgetauscht

werden können, werden die Ansagen gestoppt und das Gespräch wird fortgesetzt. Als Hürde erwies es sich hierbei, dass die Sequenznummern und Zeitstempel der RTP-Pakete manipuliert werden müssen. Ursache ist die so genannte „Voice Activity Detection“ (VAD), welche ein SIP-Telefon dazu bringt, in Sprachpausen keine RTP-Pakete zu versenden. Dabei kommt es auch zu keiner Erhöhung der Sequenznummern mehr. Am „Relay Plugin“ wird jedoch eine durchgängige Ansage eingespielt, wobei die Sequenznummer durch die künstlichen RTP-Pakete erhöht wird. Daher ist es nicht ausreichend, nach einer Aufhebung der Isolationssituation bloß die RTP-Pakete weiterzuleiten. Die Sequenznummern müssen so angepasst werden, dass der Empfänger weiterhin eine linear aufsteigende Paketfolge erhält.

Die Ansagen können aus beliebigen Quellen gewonnen, müssen jedoch in ein spezielles Format umgerechnet werden. REACH unterstützt den nicht komprimierenden Codec G.711 mit Quantisierung gemäß A-law und μ-law, deren Verwendung durch Betrachtung des „Payload Type“-Felds abgefangener RTP-Pakete festgestellt wird. Komprimierende Codecs können bislang noch nicht benutzt werden, da unklar wäre, zu welchen Zeitpunkten ein RTP-Paket gesendet werden müsste.

Eine wichtige Aufgabe des transportorientierten Kerns von REACH besteht in der Erkennung von Isolationssituationen, was eine besondere Herausforderung darstellt, da der REACH-Server diese ausschließlich durch ein Ausbleiben von Daten erkennen kann (siehe Abschnitt 5.4). REACH erkennt solche Situationen *beidseitig* in weniger als 1,5 Sekunden, was kurzfristig genug ist, um die Ansagen rechtzeitig starten zu können.

Konfiguration der „Relay Plugins“: Abbildung 4.39 zeigt die Konfiguration bezüglich eines mobilen Endgeräts. Es muss die IP-Adresse und die Portnummer der als „virtuelle PBX“ agierenden UDP-Annahmestelle spezifiziert werden. In diesem Fall wird sie auf die IP-Adresse `"127.0.0.1"` gebunden, sodass lediglich lokal gestartete Anwendungen (Softphones) einen Zugriff erhalten. Soll der REACH-Client jedoch auf einer REACH-Box gestartet werden, muss hier die IP-Adresse einer LAN-Schnittstelle angegeben werden, sodass Teilnehmer aus dem LAN heraus die „virtuelle PBX“ erreichen können.

Die spezifizierte Portnummer der SIP-Annahmestelle `"5060"` impliziert die Nutzung der Ports `5062`, `5063`, `5064` und `5065`, was dem Portbelegungschema aus Abbildung 4.36 entspricht. Die nächsten beiden Parameter benennen die IP-Adresse und die Portnummer der *eigentlichen* PBX, welche seitens des REACH-Servers kontaktiert werden soll. Im REACH-Box-Szenario kann ein Nutzer aus dem LAN heraus keine PBX mehr auswählen, da alle Sitzungen bezüglich einer „Relay Plugin“-Instanz dieselbe PBX involvieren. Es spricht allerdings nichts dagegen, eine weitere Instanz zu erzeugen, und

```
<RelayPlugins>
  <RelayPlugin loadlibrary="librrpc_srp.so">
    <Instance>
      <Configuration>
        <Item key="listener ip" value="127.0.0.1" />
        <Item key="listener port" value="5060" />
        <Item key="pbx ip" value="141.24.92.183" />
        <Item key="pbx port" value="5060" />
        <Item key="announcement alaw" value="ansage.alaw" />
        <Item key="announcement ulaw" value="ansage.ulaw" />
      </Configuration>
    </Instance>
  </RelayPlugin>
</RelayPlugins>
```

Abbildung 4.39: Am REACH-Client müssen pro Instanz die lokalen Annahmestellen, die Adresse einer PBX sowie die Namen von zwei Audiodateien mit unterschiedlich kodierten Ansagen spezifiziert werden.

diese beispielsweise ab Portnummer `5070` zu verankern. Diese Instanz könnte dann mit einer anderen PBX verknüpft werden. Die letzten beiden Parameter benennen die Dateinamen der Audiodatei mit der Ansage, wobei als Codec G.711 mit A-law bzw. μ-law verwendet werden muss.

Im dargestellten Fall gibt es bezüglich einer Firewall nichts zu beachten, da sich die Kommunikation auf das „Loopback“-Device beschränkt. Im REACH-Box-Szenario hingegen, wo Daten aus einem LAN empfangen werden sollen, müssen pro „Relay Plugin“-Instanz die fünf UDP-Portnummern freigegeben werden.

Eine mögliche serverseitige Konfiguration ist in Abbildung 4.40 dargestellt. Hier muss die öffentlich sichtbare IP-Adresse, unter welcher der REACH-Server erreichbar ist, angegeben werden. Zudem wird eine UDP-Portnummer sowie eine Anzahl an „Portpaaren“ genannt, welche in Anlehnung an Abbildung 4.36 die Menge der zu belegenden Portnummern beschreibt. Die Portnummer muss *gerade* sein, und es ist wenigstens *ein* Portpaar erforderlich. Der gesamte sich ergebende Portnummerbereich muss an der lokalen Firewall freigegeben werden. Falls Regeln zur Weiterleitung von Ports an einer NAT-Grenze hinterlegt werden sollen, muss ebenfalls genau dieser Portnummernbereich weitergeleitet werden. Zuletzt benennen zwei Parameter die Namen der Ansage-Dateien.

```
<RelayPlugins>
  <RelayPlugin loadlibrary="librrps_srp.so">
    <Instance>
      <Configuration>
        <Item key="listener ip" value="141.24.92.184" />
        <Item key="listener port" value="5060" />
        <Item key="port pairs" value="5" />
        <Item key="announcement alaw" value="ansage.alaw" />
        <Item key="announcement ulaw" value="ansage.ulaw" />
      </Configuration>
    </Instance>
  </RelayPlugin>
</RelayPlugins>
```

Abbildung 4.40: Anhand der Konfiguration einer serverseitigen Instanz ist ersichtlich, dass mit Portpaaren gearbeitet wird. Auch hier können die Dateinamen von zwei Audiodateien genannt werden.

4.4 Mitteilung des Diensteangebotes

Wenn auf einem mobilen Endgerät ein „Relay Plugin“ instantiiert wird, muss dieses mit einer Partnerinstanz, welche von einem REACH-Server geladen worden ist, assoziiert werden. Dabei muss ein geeigneter REACH-Proxyserver ausfindig gemacht werden.

Der für REACH umgesetzte Dienstsuchemechanismus ist komplex, da eine Vielzahl an Faktoren berücksichtigt werden müssen. So können REACH-Proxyserver automatisch ausgewählt oder durch den Nutzer fest vorgegeben werden (mittels `Discovery`-Parameter). Manche „Relay Plugin“-Instanzen können sich mit mehreren Partnerinstanzen gleichzeitig assoziieren, was den negativen Auswirkungen von „Dreiecks-Routing“ entgegenwirken soll, jedoch ist dies wiederum nicht bei jedem Mechanismus möglich.

Ein REACH-Client muss in der Lage sein, die an jedem relevanten REACH-Proxyserver instantiierten „Relay Plugins“ in Erfahrung zu bringen. Jeder REACH-Server implementiert hierfür einen Auskunftsdienst, welcher seine angebotenen „Dienste“ auf eine „Servicemap“ abbildet. Ein „Relay Plugin“ kann pro REACH-Server maximal einmal instantiiert werden, sodass jeder „Dienst“ durch einen festen Identifikator repräsentiert werden kann. Die „Servicemap“ besteht aus einer Liste an 8bit-Zahlen, welche der REACH-Server an alle assoziierten REACH-Clients überträgt, und ihnen damit die Menge der geladenenen „Relay Plugins“ anzeigt. Wird später ein zusätzliches „Relay Plugin“ instantiiert oder ein anderes entfernt, wird die aktualisierte „Servicemap“ verteilt.

Kapitel 7, welches sich der Auswahl geeigneter REACH-Proxyserver widmet, betrachtet in Abschnitt 7.4 den Dienstsuchemechanismus, welcher die „Servicemaps“ benötigt.

4.5 Sicherheit

Eine berechtigte Frage ist, welche Auswirkungen der Einsatz von REACH auf die Sicherheit hat. Immerhin werden sämtliche Datenströme der Anwendungen durch „Relay Plugins“ abgefangen und über Proxyserver geleitet, sodass an einer zentralen Stelle ein Zugriff auf die Nutzerdaten besteht.

4.5.1 Bedrohungsszenarien

Bei der „Frage nach der Sicherheit“ ist es empfehlenswert, zu klären, was mit Sicherheit eigentlich gemeint ist und wogegen man sich schützen möchte. Hierzu bietet es sich an, Bedrohungsszenarien herzuleiten, und dann die zu prüfende Architektur auf entsprechende Schwachstellen hin zu untersuchen. Betrachtet werden muss hier insbesondere die Kommunikation zwischen den Anwendungen auf dem mobilen Endgerät des Benutzers und den Serverdiensten im Internet. Diese Daten gilt es gegen passive Angriffe (mitlesen) sowie aktive Angriffe (manipulieren) zu schützen. Diese Problematik ist jedoch nicht nur beim Einsatz von REACH zu beachten, sondern betrifft vielmehr auch den alltäglichen Umgang mit dem Internet.

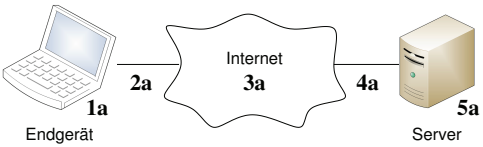

Abbildung 4.41: Bereits beim „gewöhnlichen“ Zugriff auf Server im Internet können Angreifer an verschiedenen Stellen auf die vertraulichen Nutzdatenströme einwirken.

Ohne eine Mobilitätsunterstützung wie REACH ergibt sich eine typische Client-Server-Architektur, wie sie in Abbildung 4.41 gezeigt wird. Die Datenströme der Anwendungen des mobilen Endgeräts **1a** passieren den Netzzugang samt Gateway **2a** und werden durch das Internet **3a** in Richtung des anvisierten Servers **5a** transportiert. Dort passieren die Daten ebenfalls einen Netzzugang **4a**. Geht man davon aus, dass der Nutzer sein persönliches Endgerät „unter Kontrolle“ hat (z.B. als Administrator, aktuelle Virensignaturen und Sicherheitsupdates nach bestem Wissen eingespielt) und er seinem Dienstanbieter **5a** (beispielsweise seine Bank) vertraut, dann kann die für Angriffe relevante Übertragungsstrecke auf **2a**, **3a** und **4a** eingeschränkt werden. Für diesen Fall bleibt jedoch keine Alternative, als die Kommunikation Ende-zu-Ende zu sichern, beispielsweise durch

Anwendung von Verschlüsselungs- und Authentifizierungsmechanismen. Derartige Schemata muss der Nutzer demnach *immer* einsetzen, wenn er sich gegen aktive und passive Angriffe schützen möchte.

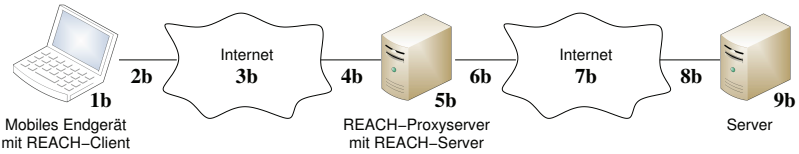

Abbildung 4.42: Beim Einsatz von REACH kommen zusätzliche Komponenten und Übertragungsstrecken hinzu, was die Anzahl der möglichen Angriffspunkte erhöht.

Beim Einsatz von REACH ergibt sich jedoch eine andere Kommunikationsbeziehung (siehe Abbildung 4.42). Aus Sicht des Servers **9b** stammt die Anfrage dann nicht mehr von einer Anwendung auf einem mobilen Endgerät **1b**, sondern scheint beispielsweise von einer proxyserverseitigen „Relay Plugin"-Instanz für „transparente Proxyserver" **5b** auszugehen. Aktive und passive Angriffe an den Netzzugängen **6b** und **8b** sowie im Internet auf der letzten Teilstrecke **7b** können auch hier nur wieder durch den Einsatz von Ende-zu-Ende greifenden Schutzmechanismen behandelt werden. Damit muss allerdings noch die Rolle des REACH-Proxyservers **5b** diskutiert werden. Terminiert dieser die serverseitigen Schutzmechanismen, wären die Datenströme an den Punkten **2b**, **3b**, **4b** und **5b** les- und manipulierbar. Leitet REACH abgefangene Nutzdatenströme jedoch „transparent" durch, können Ende-zu-Ende-Schutzmechanismen auch zwischen den Anwendungen auf dem mobilen Endgerät und den Serverdiensten im Internet etabliert werden. Weder am REACH-Client, am REACH-Server, noch dazwischen könnte dann ein Angreifer sinnvoll auf die Nutzdatenströme einwirken.

4.5.2 Sicherheitsbewertung der „Relay Plugins"

Für den Fall, dass TCP-Verbindungen oder UDP-Assoziationen durch REACH abgefangen werden, entsteht innerhalb von REACH eindeutig ein Sicherheitsproblem, da der Inhalt der Datenströme hier direkt mitgelesen werden kann. Dies betrifft die „Relay Plugins" für „nach innen" und „nach außen" gerichtete Weiterleitungsregeln für TCP-Verbindungen und UDP-Assoziationen, zudem die „Relay Plugins" für „transparente Proxyserver" sowie die „Relay Plugins" für SOCKSv4 und SOCKSv5. Alle diese „Relay Plugins" schränken jedoch die Aktivierung von Ende-zu-Ende-Schutzmechanismen in keinster Weise ein. Für den Nutzer bietet es sich damit an, auf entsprechende Protokolle

wie HTTPS oder Dienste wie SSH zurückzugreifen. Weiterhin wurde bereits gezeigt, dass das UDP-basierte Weiterleitungsschema erfolgreich den verschlüsselten Datenstrom zwischen einem VPN-Client und einem VPN-Server transportieren kann. Damit steht auch bei Anwendung von REACH einem Einsatz von Ende-zu-Ende-Verschlüsselung nichts im Wege.

Beim Einsatz des gerade genannten VPN-basierten Konzepts muss man sich aber bewusst sein, dass das VPN-interne Verschlüsselungsschema am VPN-Server endet. Misstraut man hierbei dem VPN-Server oder der letzten Teilstrecke in Richtung des Servers, muss wieder auf Ende-zu-Ende-Verschlüsselung zurückgegriffen werden.

Einzig die „Relay Plugins“ für SIP-basierte Sprachkommunikation erweisen sich diesbezüglich als problematisch. Wie in Abschnitt 4.3.2.6 dargelegt wurde, darf der Signalisierungsdatenstrom (SIP) nicht verschlüsselt werden, da sonst die eingebetteten Adressinformationen nicht länger angepasst werden könnten. Weiterhin sollten auch die Mediendatenströme nicht verschlüsselt werden, da sonst im Isolationsfall keine künstlichen Sprachansagen mehr eingespielt werden können. Es wäre nur eine geringfügige Erweiterung des serverseitigen „Relay Plugins“ erforderlich, um alle Mediendatenströme am REACH-Server in eine Datei zu schreiben und damit die geführten Telefongespräche zu kompromittieren.

Fazit: *Um Nutzdatenströme vor passiven und aktiven Angriffen schützen zu können, eignen sich Ende-zu-Ende-Sicherungsmechanismen. Diese arbeiten problemlos mit REACH zusammen, mit Ausnahme der Mobilitätsunterstützung für SIP-basierte Telefonie.*

4.6 Kapitelzusammenfassung

In diesem Kapitel wurde das Dienstekonzept von REACH beleuchtet, welches durch die Vielzahl an Möglichkeiten zum Abfangen von Datenströmen der nicht modifizierten Anwendungen motiviert war. Diese Mechanismen wurden gemäß der Schicht, auf der sie jeweils angesiedelt sind, gruppiert und umfassend beleuchtet. Anschließend wurden sie anhand praxisorientierter Nutzungsszenarien bewertet. Diese spiegelten wider, was Nutzer von ihren Endgeräten im mobilen Umfeld erwarten. Das wesentliche Ergebnis lautete, dass zwar zu jedem Nutzungsszenario wenigstens ein geeigneter Mechanismus benannt werden konnte, aber keiner dieser Mechanismen für sich gesehen in der Lage war, wiederum auch alle anderen Szenarien zu unterstützen.

Für REACH wurde ein modulares Dienstekonzept vorgestellt, wobei alle vorgestellten Mechanismen in Form von „Relay Plugins“ umgesetzt werden konnten. Der wichtigster Aspekt liegt hierbei darin, dass diese Plugins beliebig miteinander kombiniert

werden können, wobei sich die Vorteile der einzelnen Mechanismen ergänzen, während ihre Nachteile gegenseitig verdeckt werden. So ist REACH in der Lage, alle vorgestellten Nutzungsszenarien mit gleichbleibender Konfiguration zu unterstützen.

Da REACH architekturbedingt Sicherheitsbedenken aufwirft, wurden speziell die Gefahren passiver und aktiver Angriffe in Bezug auf die vertraulichen Nutzerdatenströme betrachtet. Dabei kam heraus, dass beim Einsatz von REACH weiterhin dieselben Ende-zu-Ende-Schutzmechanismen eingesetzt werden können, wie man sie bereits im „heutigen Internet“ einsetzt.

Zur Laufzeit können eine Vielzahl an „Relay Plugin“-Instanzen dynamisch erzeugt, parametriert und wieder entfernt werden, wobei die Sitzungsverwaltung und die Sicherstellung der Integrität der Nutzdaten bereits durch den transportorientierten Kern erbracht werden. Dieser Thematik widmet sich das nachfolgende Kapitel.

5 Das Transportkonzept

Das vorherige Kapitel widmete sich dem Dienstekonzept von REACH. Wie dort erläutert worden ist, können miteinander assoziierte „Relay Plugin“-Instanzen, welche am REACH-Client und am REACH-Server existieren, durch den „transportierten Kern“ miteinander Daten austauschen. Im Folgenden wird dazu mit einer Übersicht der einzelnen Schichten „des transportorientierten Kerns“ von REACH begonnen, welche anschließend gesondert vorgestellt werden.

5.1 Der transportorientierte Kern von REACH

In Abbildung 5.1 wird die Schichtung eines mobilen Endgeräts, welches über einen REACH-Client verfügt, dargestellt. Die Datenströme der nicht modifizierten Anwendungen werden durch „Relay Plugin“-Instanzen abgefangen und mit der Hilfe von *logischen Verbindungen* an eine Partnerinstanz (eine assoziierte „Relay Plugin“-Instanz selben Typs) transportiert. Eine Vielzahl an strom- und paketorientierten *logischen Verbindungen* werden zu *Linkbündeln* zusammengefasst, wobei ein *Linkbündel* das Ergebnis einer Dienstfindungsprozedur darstellt und eine „Relay Plugin“-Instanz am REACH-Client mit einer „Relay Plugin“-Instanz am REACH-Server assoziiert. In der Abbildung wird allerdings nicht dargestellt, dass jede „Relay Plugin“-Instanz eine Vielzahl an *Linkbündeln* verwalten kann, welche dann an unterschiedliche REACH-Server gerichtet sind (diese Thematik wurde bereits in Abschnitt 4.3.1.2 behandelt). Falls speziell der Zusammenhang zwischen *logischen Verbindungen* und *Linkbündeln* von Interesse sein sollte, empfielt sich ein erneuter Blick auf Abbildung 4.2 auf Seite 46.

Der transportorientierte Kern von REACH bietet zwei verschiedene Arten *logischer Verbindungen* an, da stromorientierte und paketorientierte Daten unterschieden werden. Alle *logischen Verbindungen*, welche denselben REACH-Client bzw. REACH-Server zum Ziel haben, passieren dasselbe Multiplexer (MUX)-Paar. Multiplexer treten paarweise auf, da für die beiden Typen an *logischen Verbindungen* gesonderte Mechanismen benötigt werden. Die Multiplexer haben die Aufgabe, die Daten vieler *logischer Verbindungen* zu einem einzigen Summendatenstrom (pro Typ) zusammen zu fassen. Die Verwaltung von *logischen Verbindungen*, was deren Auf- und Abbau mit einschließt, fällt ebenfalls

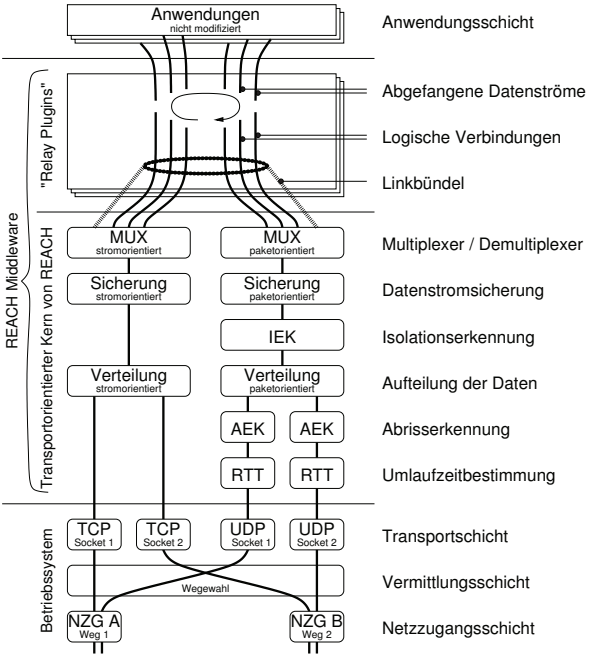

Abbildung 5.1: Die vollständige Schichtung von REACH zeigt, dass die Komponenten des Dienstekonzepts auf einen transportorientierten Kern aufsetzen. Dieser Kern implementiert verschiedene Protokollinstanzen, welche nacheinander durchlaufen werden und verschiedene Aufgaben erfüllen.

in ihren Verantwortungsbereich. Die Multiplexer werden in Abschnitt 5.2 genauer beleuchtet.

Da speziell der Summendatenstrom des stromorientierten Multiplexers besondere Anforderungen an die Übertragungsstrecke stellt, welche durch TCP allein im mobilen Umfeld nicht erbracht werden können, sind unterhalb der Multiplexer Sicherungsinstanzen angesiedelt. Sie haben die Aufgabe, die beiden Summendatenströme vor den negativen Auswirkungen eines Handovers zu schützen, damit die Übertragung über die mobile Teilstrecke keine negativen Auswirkungen auf die Multiplexer haben kann. Die Funktionsweise der beiden Sicherungsinstanzen wird in Folgeabschnitt 5.3 vorgestellt.

Unterhalb der Sicherungsinstanzen liegen zwei gesicherte Summendatenströme vor, welche mittels nicht modifiziertem TCP/IP und UDP/IP auch im mobilen Umfeld übertragen werden können. Vorher durchläuft der paketorientierte Datenstrom noch die „Iso-

lationserkennung“ (IEK), welche bei Bedarf für einen Mindestdatenstrom sorgt, damit empfängerseitig beim Ausbleiben von Daten auf eine Isolationssituation geschlossen werden kann. Dieser Mechanismus bildet den Schwerpunkt von Abschnitt 5.4.

Die an dieser Stelle vorliegenden Datenströme können schließlich versendet werden, wobei das Ergebnis der Handoverentscheidung berücksichtigt werden muss. Es kann eine Vielzahl an aktiven Netzzugängen und damit Übertragungswegen geben, welche für den Datenaustausch zwischen REACH-Client und REACH-Server herangezogen werden können. Dabei können Daten auch redundant verschickt werden oder mit Blick auf ein Kanalbündelungsszenario auf verschiedene Wege „aufgefächert“ werden. Die sich für REACH ergebenen Übertragungsmechanismen werden in Abschnitt 5.5 diskutiert. Unterabschnitt 5.5.1 beleuchtet die Aufteilung der Datenströme auf die verfügbaren Übertragungsstrecken. In Unterabschnitt 5.5.2 wird die Übertragung mittels TCP/IP und UDP/IP diskutiert. Da hierbei NAT-Grenzen überschritten werden und Probleme durch Firewalls auftreten können, wird in Unterabschnitt 5.5.3 auf die Notwendigkeit von „Keep-Alive“-Paketen eingegangen. Diese Pakete eignen sich zudem dazu, „abgerissene“ Transportschichtverbindungen erkennen zu können, womit eine „Abrisserkennung“ (AEK) zur Verfügung steht. In Unterabschnitt 5.5.4 wird anschließend ein Mechanismus zur Bestimmung der aktuellen Paketumlaufzeit pro Übertragungsstrecke vorgestellt, dessen Ergebnis für die Übertragungswiederholung innerhalb der Sicherungsinstanzen wichtig ist. Schlussendlich wird in Unterabschnitt 5.5.5 auf das Problemfeld einer deterministischen Wegewahl eingegangen, da Datenströme am mobilen Endgerät an bestimmte Netzzugänge (NZG) gebunden werden müssen.

Anhand von Abbildung 5.1 lässt sich abermals deutlich erkennen, dass REACH eine Middleware darstellt. Aus Sicht des Betriebssystems tritt der transportorientierte Kern von REACH als eine Anwendung in Erscheinung, während die eigentlichen Anwendungen nicht modifiziert werden brauchen, wobei sich REACH „unterhalb“ von ihnen befindet. Am REACH-Server sieht die Architektur ähnlich aus, es gibt jedoch Unterschiede. Während der dortige transportorientierte Kern bis auf Details identisch ist, spielt dort beispielsweise die Kopplung von Datenströmen an bestimmte Netzzugänge keine Rolle mehr.

5.2 Die Multiplexer

Seitens der „Relay Plugins“ kann eine Vielzahl an *logischen Verbindungen* anfallen. Da REACH zwei Arten *logischer Verbindungen* unterscheidet, gibt es auch zwei verschiedene Multiplexer. Ihr Ziel ist es, die Datenströme der *logischen Verbindungen* auf einen

zentralen Summendatenstrom abzubilden, damit dieser durch eine nachgeschaltete Sicherungsinstanz vor den negativen Auswirkungen von Handoverereignissen geschützt werden kann.

5.2.1 Benennungs- und Adressierungsproblem

Da also Daten einer Vielzahl an *logischen Verbindungen* im senderseitigen Multiplexer in einem Summendatenstrom münden, muss der empfängerseitige Multiplexer die Datenströme wieder voneinander trennen können, sodass die ursprünglichen *logischen Verbindungen* wieder zum Vorschein kommen. Da jede *logische Verbindung* einen Verbindungsaufbau durchläuft, muss zudem die Adressierungsproblematik geklärt werden. Ziel ist es, dass eine eingehende *logische Verbindung* an die korrekte „Relay Plugin"-Instanz gereicht werden kann. Dazu müssen beim Verbindungaufbau Zusatzinformationen mitgegeben werden, welche beispielsweise einen eindeutigen Identifikator für die Verbindung festlegen sowie die „Relay Plugin"-Instanz benennen. Letztgenannte Instanz wird über das zugeordnete *Linkbündel* definiert, welches eine übergeordnete Organisationseinheit darstellt, welche zwischen miteinander assoziierten „Relay Plugin"-Instanzen besteht Besitzt eine „Relay Plugin"-Instanz ein *Linkbündel*, kann es an seinem Endpunkt *logische Verbindungen* anfordern und annehmen (siehe Abbildung 4.2).

Logische Verbindungen werden bei REACH intern mittels numerischer Identifikatoren unterschieden. Diese Identifikatoren werden als Verwaltungsinformation mit den zu übertragenden Nutzdaten verknüpft, sodass der Empfänger des Summendatenstroms die einzelnen Datenströme wieder voneinander trennen kann.

Identifikatoren müssen eindeutig vergeben werden, was im Rahmen des Verbindungsaufbaus geschehen muss. Hierbei ist zu berücksichtigen, dass sowohl am REACH-Client als auch am REACH-Server *logische Verbindungen* angefordert werden dürfen. Da *logische Verbindungen* eindeutig benannt werden müssen, kommt es zu einem Vergabeproblem, welches in Abbildung 5.2 gezeigt wird.

Es lässt sich erkennen, dass weder der REACH-Client noch der REACH-Server in der Lage sind, *global eindeutige* Identifikatoren zu vergeben. Auch eine „nur" systemweit über mehrere Multiplexer hinweg erfolgende eindeutige Vergabe von Identifikatoren führt nicht zum gewünschten Erfolg, da die Kombination aus zeitgleich eintreffenden und ausgehenden *logischen Verbindungen* keine robuste Kollisionsvermeidung zulässt. Daher muss die Eindeutigkeit von Identifikatoren jeweils auf zwei miteinander assoziierte Multiplexer-Partnernstanzen beschränkt werden (siehe Abbildung).

Allerdings löst diese Maßnahme nur einen Teil des Vergabeproblems. Bisher teilen sich beide Multiplexer-Partnerinstanzen einen gemeinsam genutzten Vorrat an Identifikato-

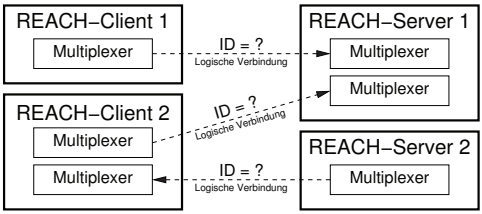

Abbildung 5.2: Die Vergabe von Identifikatoren *logischer Verbindungen* ist problematisch. Weder ein REACH-Client noch ein REACH-Server können robust einen als frei geltenden Identifikator festlegen. Gründe dafür sind ein konkurrierender Zugriff mehrerer REACH-Clients und REACH-Server auf den Vorrat an Identifikatoren sowie die Tatsache, dass *logische Verbindungen* beidseitig angefordert werden können.

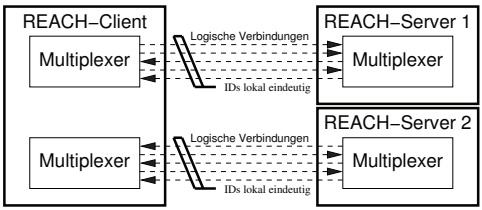

Abbildung 5.3: Identifikatoren werden nur zwischen zwei miteinander assoziierten Multiplexer-Partnerinstanzen eindeutig vergeben.

ren, und es wurde noch keine Aussage darüber getroffen, welche der beiden Multiplexer-Instanzen Identifikatoren vergeben darf. Dieses Problem spielt ebenfalls bei X.25 eine Rolle, wo *logische Verbindungen* (dort als „Virtual Channel“ (VC) bezeichnet) mit einem einzigen „Virtual Channel Identifier“ (VCI) aus einem Vorrat von $2^{12} = 4096$ verschiedenen Identifikatoren benannt werden. X.25 ist ein Netzzugangsprotokoll, welches konkret zwischen *ankommenden* und *abgehenden Verbindungen* unterscheidet. Die Vermittlungsstelle belegt dabei die VCIs aufsteigend vom unteren Ende des Vorrates her, während die Teilnehmerseite die VCIs vom oberen Ende herunter belegt. Hin- und rücklaufende Pakete derselben *logischen Verbindung* weisen einen identischen VCI auf. Bei der Vergabe kann es allerdings zu Kollisionen kommen. Sollten Vermittlungsstelle und Teilnehmer gleichzeitig eine Verbindung aufbauen wollen und dabei denselben VCI belegen, so setzt sich die Vermittlungsstelle gegenüber dem Teilnehmer durch [Sieg99].

Diese Vorgehensweise wäre für REACH aus mehreren Gründen ungeeignet, da bei auftretenden Kollisionen unabhängig von der Art und Weise ihrer Behandlung Verzögerungen beim Aufbau der *logischen Verbindung* auftreten. Wären dagegen Kollisionen

ausgeschlossen, erspare dieses deren Behandlung und es träten keine durch Neuverhandlungen oder Ablehnungen provozierten Verzögerungen auf. Zusätzlich sollen bei REACH immer möglichst kleine Zahlen als Identifikatoren benutzt werden, da im Kopf einer jeden nutzdatenbezogenen „Protocol Data Unit“ (PDU) direkt fünf Bits zur Benennung der *logischen Verbindung* herangezogen werden, wodurch die ersten 32 *logischen Verbindungen* bereits durch das erste Byte dargestellt werden können. Erst Identifikatoren zahlenmäßig größer gleich 32 verlangen nach einem dynamisch wachsenden Protokollkopf und damit einem steigenden Overhead. Daher erweist sich eine Vergabe vom oberen Ende des Sequenznummernvorrates herunter für REACH als ineffizient.

Fazit: *Der Vergabemechanismus von X.25 bietet eine gute Basis, muss aber dahingehend angepasst werden, dass keine Kollisionen mehr auftreten können und dass beidseitig bevorzugt kleine Zahlen als Identifikatoren vergeben werden.*

5.2.2 Allgemeine Funktionsweise beider Multiplexer

Verbindungsaufbau Für den Verbindungsaufbau und die Festlegung des Identifikators (ID) ist eine „Ein-Wege-Verbindungsanzeige“ ausreichend, da der Initiator einer *logischen Verbindung* bei REACH immer davon ausgehen darf, dass am Gerufenen die Verbindung angenommen wird und der Identifikator dort auch verfügbar ist. Beide Instanzen haben innerhalb ihrer Sitzung prinzipiell den vollen Zugriff auf den gesamten Vorrat möglicher Identifikatoren, ohne dass es bei der Vergabe zu Kollisionen kommen kann. Damit dieses Ziel erreicht werden konnte, musste eine Vorkehrung getroffen werden. Da gleiche Identifikatoren beidseitig vergeben werden können, wird ihre Eindeutigkeit daran festgemacht, welche der Partnerinstanzen sie vergeben hat. Mit Hilfe eines als „Initiator-Bit“ bezeichneten Flags wird im Protokollkopf der PDUs vermerkt, ob der enthaltenene Identifikator ein lokal vergebener Identifikator (ausgehend) ist oder ob er durch die Partnerinstanz festgelegt wurde (eingehend).

In Abbildung 5.4 ist ein solcher Fall dargestellt. Beide Partnerinstanzen beginnen zeitnah mit dem Aufbau einer *logischen Verbindung* und vergeben jeweils den ihrerseits freien Identifikator `ID=5`. Bestandteil der *logischen Verbindung* ist allerdings jeweils die Zusatzinformation, dass der Identifikator „ausgehend“ festgelegt worden ist. Auf dem Medium überkreuzen sich beide Anforderungen (die hierfür verantwortlichen `SYN`-PDUs) und lösen am jeweiligen Empfänger eine Verbindungsannahme aus. Hierbei wird die von der Partnerinstanz gewählte `ID=5` in beiden Fällen als frei erkannt, da bislang noch keine Verbindung mit dieser ID (`5`, eingehend) existiert. Es kommt zu keiner Kollision.

Abbildung 5.5 verdeutlicht die genannte Unterscheidung eingehender und ausgehender logischer Verbindung bei deren interner Speicherung. Es gibt hierfür in jeder Multiplexer-

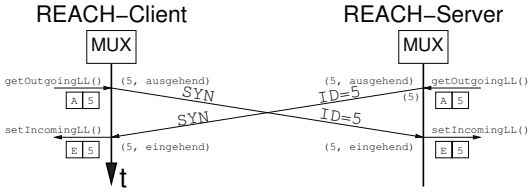

Abbildung 5.4: Jede Multiplexer-Partnerinstanz hat die alleinige Hoheit über die Vergabe der Identifikatoren ausgehender Verbindungen. Eingehende Verbindungen werden mit Zahlen aus demselben Zahlenvorrat benannt. Die Zuordnung innerhlab dieser Sitzung bleibt aber eindeutig, da nicht alleine auf die Zahl geschaut wird, sondern zusätzlich auf deren Herkunft (`A`=ausgehend, `E`=eingehend).

Instanz zwei Listen, eine zur Speicherung lokal initiierter (ausgehender) und eine für eintreffende *logische Verbindungen*.

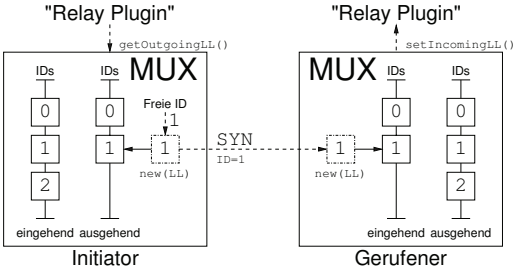

Abbildung 5.5: Beim Aufbau einer *logischen Verbindung* („Logical Link“ (LL)) wird vom Initiator ein Identifikator (ID) vergeben und der Partnerinstanz durch Versendung einer SYN-PDU mitgeteilt.

Bisher wurde noch nicht abschließend geklärt, was mit einer angenommenen *logischen Verbindung* passieren soll. Immerhin können oberhalb einer gerufenen Multiplexer-Instanz eine Vielzahl an „Relay Plugin“-Instanzen verfügbar sein, oder es könnte sich auch um eine Folgeverbindung handeln, welche zu einer bereits bestehenden Dienstabwicklung zugeordnet werden muss. Dazu wurde der Verbindungsaufbaumechanismus um die Möglichkeit der Übertragung eines zusätzlichen Signalisierungsdatenblocks erweitert. Dieser benennt, welchem *Linkbündel* eine eintreffende *logische Verbindung* am Empfänger zugeordnet ist, und referenziert damit eine bestimmte „Relay Plugin“-Instanz.

Fazit: *Es wurde ein kollisionsfrei arbeitender Verbindungsaufbaumechanismus vorgestellt, welcher wie gewünscht bevorzugt kleine Zahlen als Identifikatoren vergeben kann.*

Nutzdatenversendung Daten-PDUs müssen bei Ihrer Versendung mit dem entsprechenden Identifikator der zugehörigen *logischen Verbindung* gekennzeichnet werden. Dabei ist aber die Zusatzinformation nötig, ob es der Absender war der den Identifikator vergeben hat, oder ob der Empfänger der Daten-PDU der Initiator der *logischen Verbindung* war. Abhängig davon wird die zugehörige *logische Verbindung* am Empfänger entweder in der Liste der ausgehenden oder der eingehenden *logischen Verbindungen* gesucht.

Hierfür wurden die fünf Bit des Protokollkopfes, welche wie bereits genannt in der Lage wären 32 verschiedene Identifikatoren darzustellen, neu organisiert. Das fünfte Bit (Wertigkeit 16) wird nun als „Initiator-Bit“ (I-Bit) verwendet. Ist es gesetzt, heißt dies, dass der Absender der Nachricht der Initiator der *logischen Verbindung* war und demnach auch den Identifikator festgelegt hatte. Es verbleiben vier Bits für den eigentlichen Identifikator, was die Zahl der direkt darstellbaren *logischen Verbindungen* pro Richtung auf 16 beschränkt. Somit können beidseitig „von unten her“ aufsteigende (kleine) Zahlen als Identifikatoren vergeben werden. Abbildung 5.6 zeigt beispielhaft, wie Nutzdaten-PDUs unter Berücksichtigung des „Initiator-Bits“ versendet werden.

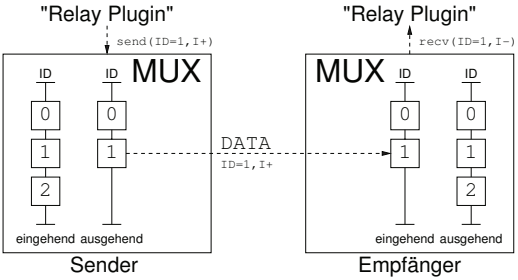

Abbildung 5.6: Nutzdaten werden beim Versand mit dem Identifikator der *logischen Verbindung* gekennzeichnet. Das gesetzte Initiator-Bit an, dass der Absender der Initiator der *logischen Verbindung* war, und damit auch den Identifikator festgelegt hatte.

Grundannahmen Die vorgestellte Betriebsweise der „Ein-Wege-Verbindungsanzeige“ basiert auf der optimistischen Grundannahme, dass erstens *logische Verbindungen* grundsätzlich vom Gerufenen angenommen werden und dass sich zweitens `SYN`-PDU und `DATA`-PDUs nicht gegenseitig überholen oder verloren gehen können. Ersteres ist durch den kollisionsfrei arbeitenden Vergabemechanismus sichergestellt, während die Einhaltung der zweiten Grundannahme in den zwei folgenden Abschnitten speziell für stromorien-

tierte (Abschnitt 5.2.3) und paketorientierte *logische Verbindungen* (Abschnitt 5.2.4) betrachtet werden muss.

Während in [Ever04] auf Grund des dort noch verwendeten „Zwei-Wege-Handshakes“ *logische Verbindungen* im Voraus aufgebaut wurden – um den zeitraubenden „Handshake“ im Vorfeld durchlaufen zu lassen – ist eine solche Logik bei REACH nicht mehr notwendig. *Logische Verbindungen* brauchen erst bei Bedarf aufgebaut und können zur sofortigen Versendung von Nutzdaten herangezogen werden.

Verbindungsabbau Zum Schließen *logischer Verbindungen* wird ein „Zwei-Wege-Handshake“ durchlaufen. Dazu werden zwei `CLOSE`-PDUs ausgetauscht (siehe Abbildung 5.7).

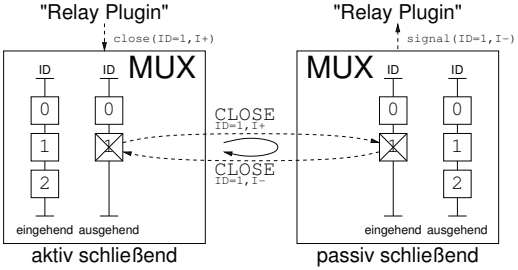

Abbildung 5.7: Der Abbau *logischer Verbindungen* wird mit Hilfe eines beidseitigen Austauschs zweier `CLOSE`-PDUs durchgeführt.

`CLOSE`-PDUs werden wie `DATA`-PDUs mit dem Identifikator der *logischen Verbindung* in Kombination mit dem Initiator-Bit benannt.

Falls ausgeschlossen werden kann, dass sich `SYN`-, `DATA`- und `CLOSE`-PDUs gegenseitig überholen oder verloren gehen können, handelt es sich hierbei um einen robusten Mechanismus zur Handhabung *logischer Verbindungen*. Auch ein simultanes Schließen einer *logischen Verbindung* seitens beider Multiplexer-Partnerinstanzen kann den Verwaltungsmechanismus nicht verwirren. Es ist sogar denkbar, dass eine *logische Verbindung* aufgebaut, zur sofortigen Datenversendung herangezogen und augenblicklich wieder geschlossen wird, wie es in Abbildung 5.8 dargestellt ist. Die Nutzdaten werden empfängerseitig (am Gerufenen) ausgewertet, allerdings müssen eventuell zurückgesendete Daten am Initiator verworfen werden, da der dortige Endpunkt der *logischen Verbindung* bereits geschlossen worden ist. Es kommt dabei zu keinem Zuordnungsproblem, da auf Grund der vorausgesetzten Reihenfolgetreue immer alle PDUs eindeutig zu *logischen Verbindungen* zugeordnet werden können.

Der lokale Zuordnungseintrag wird jeweils in dem Moment wieder freigegeben, wo

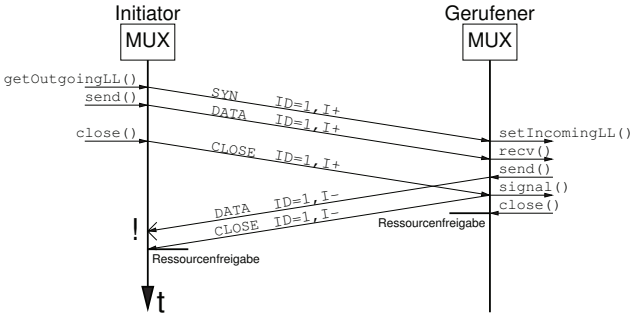

Abbildung 5.8: Eine logische Verbindung kann angefordert, zur sofortigen Nutzdatenversendung herangezogen und augenblicklich wieder geschlossen werden. Eventuell rücklaufende DATA-PDUs müssen zwar verworfen werden, was aber unproblematisch ist.

sowohl ein lokaler `close()`-Aufruf bezüglich der *logischen Verbindung* stattfand als auch eine `CLOSE`-PDU von der Partnerinstanz zu dieser *logischen Verbindung* eintraf. Dabei ist es egal, ob diese `CLOSE`-PDU die Antwort auf die versendete `CLOSE`-PDU ist („Zwei-Wege-Handshake“) oder ob die Partnerinstanz simultan die Verbindung geschlossen hat (sich überkreuzende `CLOSE`-PDUs).

REACH unterscheidet zwischen stromorientierten und paketorientierten *logischen Verbindungen*. Beide werden im Folgenden vorgestellt.

5.2.3 Stromorientierte logische Verbindungen

Eigenschaften Die Übertragung stromorientierter Daten ist mit der Dienstleistung von TCP vergleichbar. Die Daten sollen in Form eines Bytestroms so am Empfänger ankommen, wie sie versendet worden sind, ohne Verluste, Duplizierungen oder Vertauschungen. Dieser Typus *logischer Verbindungen* eignet sich damit dazu, die abgefangenen Daten eines TCP-Sockets zwischen zwei „Relay Plugin“-Instanzen zu transportieren.

Der Verbindungsaufbau einer stromorientierten *logischen Verbindung* läuft so ab, wie es bereits allgemein im letzten Abschnitt erläutert worden ist. Es wird eine „Ein-Wege-Verbindungsanzeige“ durchgeführt, bei welcher der Identifikator der *logischen Verbindung* vom Initiator festgelegt wird, und dann dem Gerufenen mitgeteilt wird. Nutzdaten dürfen sofort übertragen werden, bis zu dem Punkt, wo eine Instanz die Verbindung schließt und die *logische Verbindung* mittels eines „Zwei-Wege-Handshakes“ abgebaut wird. Das entwickelte Protokoll, welches die genannte Funktionalität erbringt, wird als „Stream Control Protocol for REACH“ (SCPR) bezeichnet.

Einhaltung der Grundannahmen SCPR verlangt nach der Einhaltung der geforderten Grundannahmen: SCPR-PDUs dürfen nicht verloren gehen oder sich gegenseitig überholen können. Sollte dies passieren, würden die Statusmaschinen im Multiplexer nicht mehr ordnungsgemäß funktionieren. Daher muss eine Sicherungsinstanz nachgeschaltet werden, welche den Gesamtdatenstrom, welcher aus den SCPR-PDUs aller stromorientierten *logischen Verbindungen* besteht, gesichert überträgt. Diese Sicherungsinstanz für den stromorientierten Summendatenstrom wird später in Abschnitt 5.3.1 beschrieben.

Notwendigkeit einer Flusssteuerung Die Hauptaufgabe jeder Multiplexer-Instanz besteht in der Bündelung vieler *logischer Verbindungen* zu einem Gesamtdatenstrom (multiplexen). In der Partnerinstanz angekommen wird der empfangene Gesamtdatenstrom wieder in seine Komponenten, die SCPR-PDUs der einzelnen *logischen Verbindungen*, zerlegt (demultiplexen). Da Datenströme gesichert übertragen werden müssen, also keine Verluste toleriert werden, wird auf Ebene der Multiplexer eine Flusssteuerung benötigt.

Nur eine Flusssteuerung kann verhindern, dass eine sendende „Relay Plugin"-Instanz Daten mit einer höheren Rate an eine *logische Verbindung* übergibt, als wie die empfangene Instanz sie wieder entnimmt. Ohne eine Flusssteuerung käme es zu „Verklemmungen" beim Demultiplexen, da ein vorliegendes Datenfragment nicht unmittelbar in den Empfangspuffer der zugehörigen *logischen Verbindung* geschrieben werden könnte. Es gäbe nur zwei Möglichkeiten: Entweder wird die wartende SCPR-PDU verworfen – damit der Mechanismus nicht blockiert – oder es wird gewartet, bis der Empfangspuffer der *logischen Verbindung* abgerufen und damit geleert wurde. Ersteres zerstört den Datenstrom, letzteres blockiert den gesamten Demultiplexer (die empfängerseitige Multiplexer-Instanz) und damit alle Datenströme [Ever04].

Die daher notwendige Flusssteuerung arbeitet bei REACH mit Bestätigungen, welche einer sendenden SCPR-Instanz (erster Multiplexer) Auskunft über den Zustand des Empfangspuffers der empfängerseitigen SCPR-Instanz (assoziierter Multiplexer) geben. Dieser Zustand ändert sich immer dann, wenn Daten aus dem Empfangspuffer abgerufen werden und somit wieder freier Platz entsteht. Dabei werden bei REACH jedoch weder Sequenznummern noch der verbleibende Platz im Empfangspuffer kommuniziert (wie es bei TCP geschieht), sondern mit einer kumulativ zu interpretierenden Blockgröße gearbeitet. Wenn also beispielsweise 4096 Bytes aus dem Empfangspuffer abgerufen worden sind, kann dieser frei gewordene Block als „erneut versendbare Datenmenge" quittiert werden.

Der Vorteil gegenüber der Verwendung von Sequenznummern liegt in einem geringeren Übertragungsaufwand. Eine kumulative Quittung kommt bei REACH mit einem einzigen Byte als Bestätigungsfeld aus. Es werden Blöcke zu jeweils 4096 Byte be-

stätigt, was Sammelbestätigungen ermöglicht, welche bis zu 1 MiB abdecken können: 256×4096 Byte $= 1048576$ Byte $= 1$ Megabinarybyte (MiB).

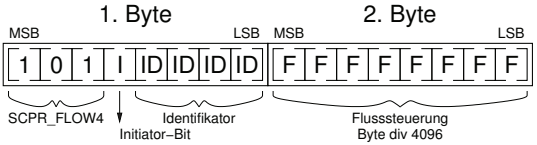

Abbildung 5.9: Quittungen zur Flusssteuerung werden bei SCPR in Form von kumulativ auszuwertenden Blockbestätigungen formuliert. Hier ist beispielhaft eine `SCPR_FLOW4`-PDU gezeigt, welche mit insgesamt zwei Bytes auskommt („Most Significant Bit“ (MSB), „Least Significant Bit“ (LSB)).

Die Flusssteuerungsinformation kann auf zwei Arten übertragen werden. Entweder, falls gerade keine Nutzdaten in Rückrichtung zu versenden sind, über nur für diesen Zweck eingeführte Nachrichten (`SCPR_FLOW`-PDUs, siehe Abbildung 5.9), oder, falls Daten zur Versendung vorliegen, als Bestandteil einer `SCPR_DATA`-PDU.

Zusammenfassung In diesem Abschnitt wurde das „Stream Control Protocol for REACH“ (SCPR) vorgestellt. Es dient der Verwaltung stromorientierter *logischer Verbindungen*. Dabei wurde die Notwendigkeit einer Flusssteuerung dargelegt.

Alle SCPR-PDUs werden in Anhang A.1.1.1 aufgeführt. Die Statusmaschine einer jeden SCPR-Instanz (Multiplexer) ist dort in Abschnitt A.1.1.2 abgebildet.

Es wird allerdings eine gesicherte Übertragung des SCPR-Gesamtdatenstroms vorausgesetzt. Ein solcher gesicherter Transportkanal wurde als gegeben hingenommen. In Abschnitt 5.3.1 wird er in seiner konkreten Realisierung vorgestellt.

5.2.4 Paketorientierte logische Verbindungen

Eigenschaften Auch paketorientierte Daten werden bei REACH über verbindungsorientiert arbeitende *logische Verbindungen* übertragen. Die zu übertragenden Pakete müssen mit einem Identifikator versehen werden, welcher im Rahmen eines Verbindungsaufbaus festgelegt wird. Die Verwaltung paketorientierter *logischer Verbindungen* erweist sich im Vergleich zu den stromorientierten *logischen Verbindungen* als komplizierter. Man kann sich hierbei nicht mehr auf leistungsfähige Sicherungsinstanzen verlassen, sondern muss mit Paketverlusten und Dubletten rechnen.

Pakete sind von ihrer Natur her anders als stromorientierte Daten. Sie werden bei REACH ungesichert übertragen, wodurch jederzeit Verluste auftreten können. Eine generelle Übertragungswiederholung verloren gegangener Pakete wäre unerwünscht, da es

hierbei zu nicht abschätzbar langen Verzögerungen kommen könnte. Dieser Umstand wirft jedoch Fragen bezüglich der Einhaltung der Grundannahme auf, denn auch hier muss ein reibungsloses Funktionieren der Statusmaschinen gewährleistet werden. Das dazu entwickelte „Packet Control Protocol for REACH“ (PCPR) musste dahingehend entworfen werden, dass es „virtuelle Verbindungen“ über ein unsicheres paketorientiert arbeitendes Medium anbieten kann.

Verbindungsaufbau: „Opportunistischer Drei-Wege-Handshake“ Zuerst muss man sich bewusst machen, dass man sich bei der Übertragung von Paketen auf einen ungesichert arbeitenden Dienst verlässt. Jedes Paket, welches verschickt wird, kann auf dem Medium verloren gehen. Zudem kann es durch die Wegewahl zu einer Vertauschung der Reihenfolge oder zu Vervielfachungen kommen. Die Herausforderung bestand darin, einen Verwaltungsmechanismus zu entwickeln, welcher auch unter diesen Randbedingungen robust funktioniert.

Der erste Schritt bestand darin, die bereits vorgestellte „Ein-Wege-Verbindungsanzeige“ des Verbindungsaufbaus durch einen „Drei-Wege-Handshake“ zu ersetzen. Damit wird wie bei TCP eine Robustheit gegenüber Paketverlusten erreicht. Allerdings weist „normales“ TCP die Eigenschaft auf, dass beide Partnerinstanzen mindestens die Dauer eines Paketumlaufs warten müssen, bevor Nutzdaten versendet werden dürfen.

Der „Drei-Wege-Handshake“ wurde für REACH dahingehend modifiziert, dass er eine sofortige Nutzdatenversendung erlaubt, noch bevor er komplett durchlaufen wurde. Dieser hier als „opportunistischer Drei-Wege-Handshake“ bezeichnete Verbindungsaufbau geht dabei – wie bereits bei den stromorientierten *logischen Verbindungen* erläutert – von der optimistischen Annahme aus, dass ein Verbindungsaufbauwunsch immer von der Partnerinstanz angenommen wird. Unter dieser Bedingung wird eine frühzeitige Nutzdatenversendung noch vor Erhalt einer Bestätigung möglich, was in Abbildung 5.10 dargestellt wird. Die hier verwendeten Bezeichnungen `SYN`, `SYNACK` und `SYNACKACK` wurden in Anlehnung an die Namen der PDUs während des TCP-Verbindungsaufbaus gewählt, wobei TCP jedoch keine `SYNACKACK`-PDU kennt.

In Abbildung 5.11 wird verdeutlicht, dass jede der am Verbindungsaufbau beteiligten PCPR-PDUs beliebig oft verloren gehen kann, ohne dass eine nachhaltig negative Beeinflussung stattfindet. Dabei durchlaufen die Statusmaschinen in beiden PCPR-Partnerinstanzen bestimmte Phasen. Die initiatorseitige Instanz durchläuft beim Verbindungsaufbau die SYN-Phase, während die gerufene Instanz die SYNACK-Phase durchläuft. Nach erfolgreichem Durchlauf des Verbindungsaufbaus befinden sich beide Instanzen in der DATA-Phase. Während der SYN- und der SYNACK-Phase findet eine zyklische Versendung von PCPR-PDUs statt, welche sicherstellen soll, dass trotz auf-

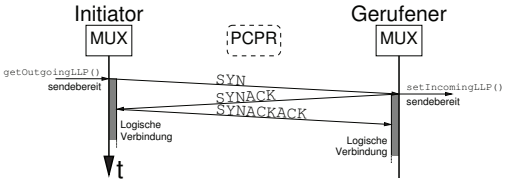

Abbildung 5.10: Ein „opportunistischer Drei-Wege-Handshake“ ermöglicht eine verzögerungsfreie Sendefreigabe.

tretender Paketverluste wenigstens eine `PCPR_SYN`-PDU am Empfänger ankommt. Die `PCPR_SYNACKACK`-PDU dient dazu, die gerufene Instanz von der SYNACK-Phase in die DATA-Phase zu schalten, um die zyklische Versendung von `PCPR_SYNACK`-PDUs zu unterbrechen. `PCPR_SYNACKACK`-PDUs werden seitens des Initiators als spontane Antwort auf erhaltene `PCPR_SYNACK`-PDUs versendet.

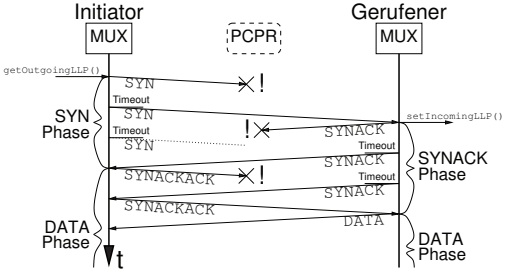

Abbildung 5.11: Die Statusmaschinen beider Instanzen durchlaufen bestimmte Phasen. Durch zyklische Übertragungswiederholung darf jede der am Verbindungsaufbau beteiligten PCPR-PDUs verloren gehen können.

Allerdings müssen noch die Auswirkungen einer Reihenfolgevertauschung beleuchtet werden. Es lässt sich jedoch erkennen, dass der „Drei-Wege-Handshake“ das Auftreten unlogischer Reihenfolgen unterbindet. Es kann lediglich zu einem verzögerten Eintreffen duplizierter PCPR-PDUs kommen, welche dann allerdings einem bestehenden Verbindungsendpunkt zugeordnet werden können und somit für die Statusmaschinen keine Probleme verursachen.

Fazit: *Ein „Drei-Wege-Handshake“ ist gut geeignet, um paketorientierte logische Verbindungen über unzuverlässig arbeitende Medien aufzubauen.*

Nutzdatenversendung Die Übertragung von Nutzdaten erfolgt unbestätigt und ungesichert. Verloren gegangene Nutzdatenpakete werden nicht als solche detektiert und

schon gar nicht wiederholt (siehe Abbildung 5.12). Anwendungen, welche Pakete zur Nutzdatenversendung heranziehen, müssen jederzeit mit Verlusten rechnen und bringen daher bei Bedarf eigene Mechanismen zur Detektierung und Behandlung mit sich.

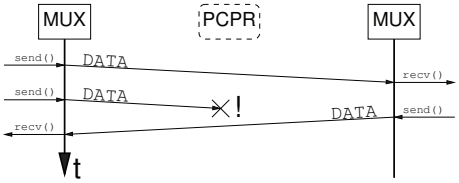

Abbildung 5.12: Der Versand von Nutzdaten (PCPR_DATA-PDUs) erfolgt unbestätigt und ungesichert. Paketverluste werden bewusst in Kauf genommen.

Nutzdatenversendung während des Verbindungsaufbaus Es wurde bereits erläutert, wie der „opportunistische Drei-Wege-Handshake“ funktioniert, aber noch nicht darauf eingegangen, wie bereits während des Verbindungsaufbaus Nutzdaten eindeutig bezeichnet und versendet werden können.

Die Betrachtung beginnt initiatorseitig: Hier wäre es problematisch, gleich im Anschluss an die versendete PCPR_SYN-PDU eine PCPR_DATA-PDU zu versenden. Bei Verlust der PCPR_SYN-PDU würde die gerufene Multiplexer-Instanz lediglich eine PCPR_DATA-PDU empfangen, welche sie jedoch nicht zuordnen könnte. Die hier genannte PCPR_DATA-PDU müsste um die Informationen der PCPR_SYN-PDU erweitert werden, damit beim Gerufenen der Empfang des Verbindungsaufbauwunsches auch im geschilderten Fall sichergestellt ist.

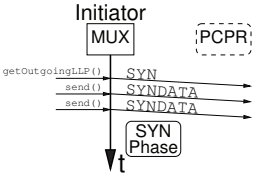

Abbildung 5.13: Der frühe Datenversand am Initiator setzt während der SYN-Phase einen neuen PDU-Typ voraus: Die PCPR_SYNDATA-PDU.

Die Kombination eines Verbindungsaufbauwunsches mit einem Nutzdatenfeld ergibt die so genannte PCPR_SYNDATA-PDU, deren Anwendung in Abbildung 5.13 dargestellt ist. Es ist ersichtlich, dass ein mehrfacher Empfang der SYN-Informationen am Gerufenen zum Normalfall gehört und unproblematisch ist. Die erste PCPR_SYN-PDU wird

augenblicklich nach Anforderung der ausgehenden Verbindung versendet, damit die gerufene Instanz schnellstmöglich über die eintreffende *logische Verbindung* unterrichtet wird. Da weder festgelegt ist, *welche Seite* zuerst Nutzdaten versenden muss, noch, *wann* dies zu erfolgen hat, darf nicht erst auf Nutzdaten gewartet werden, welche eine (dann effizientere) PCPR_SYNDATA-PDU ergeben könnten.

Am Gerufenen löst die erste eintreffende PCPR_SYN- oder PCPR_SYNDATA-PDU den lokalen Verbindungsannahmemechanismus aus. Er beginnt mit der SYNACK-Phase, in welcher zyklisch PCPR_SYNACK-PDUs versendet werden. PCPR_SYNACK-PDUs dienen als Bestätigungen, welche dem Initiator mitteilen, dass dieser die zyklische Versendung von PCPR_SYN-PDUs einstellen kann.

Da auch seitens des Gerufenen eine sofortige Datenversendung möglich sein soll, wird hier ebenfalls eine neue PCPR-PDU benötigt, die so genannte PCPR_SYNACKDATA-PDU. Sie kombiniert eine PCPR_SYNACK- mit einer PCPR_DATA-PDU und wird während der SYNACK-Phase vom Gerufenen zur Nutzdatenversendung herangezogen (siehe Abbildung 5.14).

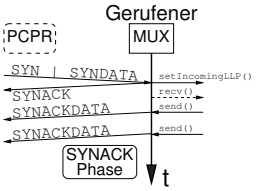

Abbildung 5.14: Der frühe Datenversand am Gerufenen setzt während der SYNACK-Phase eine neue Typ von PDU voraus: Die PCPR_SYNACKDATA-PDU.

Der Initiator verbleibt so lange in der SYN-Phase, bis er eine PCPR_SYNACK- oder eine PCPR_SYNACKDATA-PDU empfängt. Er wechselt dann in die DATA-Phase und beendet damit die zyklische Versendung von PCPR_SYN-PDUs. Der Initiator ist sich zu diesem Zeitpunkt sicher, dass eine PCPR_SYN-PDU beim Gerufenen eingetroffen sein muss.

Jede eintreffende PCPR_SYNACK-PDU wird vom Initiator mit einer PCPR_SYNACKACK-PDU beantwortet. Sie zeigt den erfolgreichen Empfang einer PCPR_SYNACK-PDU an. Allerdings lösen PCPR_SYNACKDATA-PDUs keine Versendung einer PCPR_SYNACKACK-PDU aus, da es sonst zu einer unnötigen Flut an PCPR_SYNACKACK-PDUs kommen könnte. Daraus folgt, dass der Gerufene zyklisch PCPR_SYNACK-PDUs versenden muss, unabhängig davon, ob er zwischenzeitlich PCPR_SYNACKDATA-PDUs versenden konnte oder nicht.

Mit Erreichen der DATA-Phase ist der Verbindungsaufbau für den Initiator abgeschlossen, es werden zur Nutzdatenversendung „normale“ PCPR_DATA-PDUs verwendet und die zyklische Versendung von PCPR_SYN-PDUs wird beendet (siehe Abbildung 5.15).

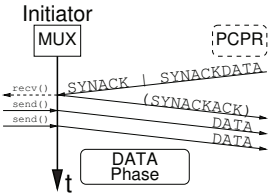

Abbildung 5.15: Datenversendung und Bestätigung von PCPR_SYNACK-PDUs während der DATA-Phase am Initiator.

Am Gerufenen löst der Empfang der ersten PCPR_SYNACKACK-PDU den Zustandswechsel in die DATA-Phase aus. Der Gerufene beendet damit die zyklische Versendung von PCPR_SYNACK-PDUs. Alle drei Wege des „Handshakes“ wurden erfolgreich durchlaufen und die *logische Verbindung* gilt auch am Gerufenen als aufgebaut (siehe Abbildung 5.16).

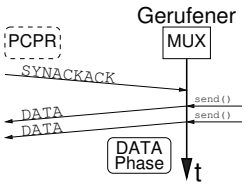

Abbildung 5.16: Start der DATA-Phase am Gerufenen.

Verbindungsabbau Auch der Verbindungsabbau gestaltet sich bei den paketorientierten im Vergleich zu den stromorientierten *logischen Verbindungen* als komplizierter. Da PDUs wiederholt verloren gehen können, musste der Abbaumechanismus so gestaltet werden, dass er mit Übertragungswiederholungen arbeitet. Dabei erwies sich ein „Zwei-Wege-Handshake“ als brauchbar.

Abbildung 5.17 illustriert den PCPR-Verbindungsabbau. Er ordnet den Instanzen unterschiedliche Rollen zu. Diejenige PCPR-Instanz, welche bezüglich der *logischen Verbindung* einen lokalen `close()`-Aufruf erfährt, wird als „aktiv“ bezeichnet. Die Partnerinstanz, an welcher die *logische Verbindung* durch eine eintreffende PCPR_CLOSE-PDU und nicht durch einen lokalen `close()`-Aufruf geschlossen wird, bildet die „passive“ Seite. Die passive Seite bestätigt die PCPR_CLOSE-PDU mit einer PCPR_CLOSEACK-PDU, und alle Ressourcen werden sofort freigegeben. Auf der aktiven Seite wird die eintreffende PCPR_CLOSEACK-PDU als Bestätigung interpretiert und die dortige Instanz ebenfalls ohne weitere Verzögerung freigegeben.

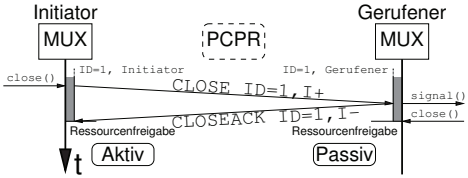

Abbildung 5.17: Beim Verbindungsabbau kommt ein „Zwei-Wege-Handshake“ zum Einsatz.

Kommt es hierbei zu einem Verlust der PCPR_CLOSE-PDU, wird die aktive Seite keine PCPR_CLOSEACK-PDU empfangen. Sie wird zyklisch die PCPR_CLOSE-PDU wiederholen (siehe Abbildung 5.18). Dies geschieht ohne eine obere Zeitschranke, sodass auch Isolationssituationen kein Problem darstellen.

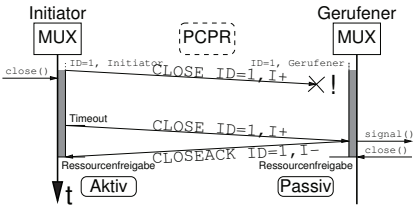

Abbildung 5.18: Geht eine PCPR_CLOSE-PDU verloren, wird dies durch Ausbleiben der Bestätigung bemerkt. Die PCPR_CLOSE-PDU wird dann wiederholt.

Sollte die rücklaufende PCPR_CLOSEACK-PDU verloren gehen, macht dies für die aktive Seite keinen Unterschied. Es kommt zu einer erneuten Versendung der PCPR_CLOSE-PDU (siehe Abbildung 5.19). Allerdings wurden zwischenzeitlich auf der passiven Seite die Ressourcen freigegeben, und die *logische Verbindung* ist hier nicht länger bekannt. Das stellt allerdings kein Problem dar, denn PCPR_CLOSEACK-PDUs können auch in diesem Fall formuliert werden.

Für den Sonderfall, dass es zu einer simultanen Schließung beider Verbindungsendpunkte kommt, übernehmen beide Seiten die Rolle einer „aktiven“ Instanz. Dieses „simultane Schließen“ ist in Abbildung 5.20 gezeigt und wird ebenfalls durch die Statusmaschinen berücksichtigt.

Dublettenproblematik Da bei der Vergabe von Identifikatoren bevorzugt „kleine“ Zahlen verwendet werden sollen, ist die Wahrscheinlichkeit hoch, dass ein soeben frei gewordener „kleiner“ Identifikator sofort wiederverwendet wird. Da nach einer schnellen Neuvergabe aber noch verspätete PCPR_CLOSE-PDUs der vorherigen *logischen Verbindung*

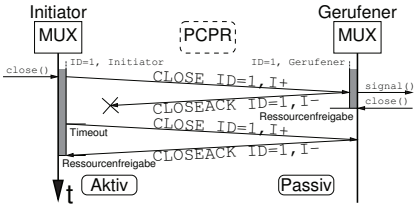

Abbildung 5.19: Geht eine PCPR_CLOSEACK-PDU verloren, mündet dies ebenfalls in einer erneuten Versendung einer PCPR_CLOSE-PDU. Diese kann bestätigt werden, obwohl die *logische Verbindung* am Kommunikationspartner bereits freigegeben worden ist.

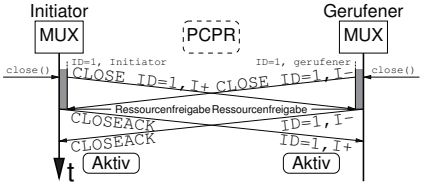

Abbildung 5.20: Auch ein simultanes Schließen von paketorientierten *logischen Verbindungen* wird unterstützt.

eintreffen können, ergibt sich ein Problem: Deren Identifikator ist mittlerweile wieder gültig, sodass die PCPR_CLOSE-PDUs hier fälschlicherweise der frisch erzeugten Instanz zugeordnet werden (siehe Abbildung 5.21).

Statt des in Abbildung 5.21 dargestellten Fehlverhaltens wäre ein anderes Verhalten wünschenswert: Initiatorseitig sollte die „alte“ PCPR_CLOSE-PDU zwar erneut durch eine PCPR_CLOSEACK-PDU bestätigt, allerdings nicht zur Trennung der dortigen „neuen“ *logischen Verbindung* herangezogen werden. Am Gerufenen dagegen sollte die eintreffende PCPR_SYN-PDU die „alte“ *logische Verbindung* entfernen und dabei gleichzeitig eine „neue“ Verbindungsinstanz anlegen.

Diese Problematik wurde dadurch gelöst, dass PCPR-PDUs nicht bloß mit dem Identifikator der *logischen Verbindung* ausgestattet, sondern zusätzlich dazu alle paketorientierten *logischen Verbindungen* linear aufsteigend nummeriert werden. Ein zusätzliches *Referenzfeld* ermöglicht es dann dem Empfänger einer PCPR-PDU, festzustellen, ob der enthaltene Identifikator noch aktuell ist oder ob er zwischenzeitlich erneut vergeben wurde und die PDU ignoriert werden muss. Dazu muss die Verbindungsinstanz herausgesucht werden und – falls vorhanden – ein Vergleich der Referenznummern stattfinden.

Diese Referenznummer dient zur Sicherstellung der Funktion der Statusmaschinen,

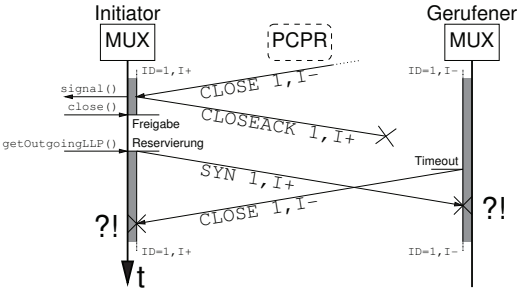

Abbildung 5.21: Verzögert eintreffende oder während des Verbindungsabbaus verloren gegangene PCPR-PDUs führen zu ernsten Problemen, sollten keine weiteren Vorkehrungen getroffen werden. Hier ist beispielhaft gezeigt, dass ein Identifikator nach seiner Freigabe sofort wiederverwendet werden kann. An der Partnerinstanz kann allerdings noch ein veralteter Verbindungsendpunkt existieren, welcher diesen Identifikator blockiert. Die dortige Instanz versendet zudem zyklisch `PCPR_CLOSE`-PDUs, welche am Initiator die Beendigung der nachfolgenden Verbindungsinstanz auslösen.

aber nicht zur Filterung von Nutzlast-Dubletten. Mit Blick auf den zusätzlichen Übertragungsaufwand wurde entschieden, lediglich die Steuerungs-PDUs zu sichern, um den reibungslosen Ablauf der Statusmaschinen zu gewährleisten. Bei den `PCPR_DATA`-PDUs wird auf das Referenzfeld verzichtet. Dabei können allerdings verzögert eintreffende `PCPR_DATA`-PDUs früherer *logischer Verbindungen* irrtümlich zu später aufgebauten *logischen Verbindungen* zugeordnet werden. Die möglicherweise negativen Konsequenzen dieses Kompromisses können leider nicht allgemeingültig bewertet werden, da sie von den involvierten „Relay Plugins“ und letztendlich auch von den Anwendungen abhängen. Die Gefahr einer negativen Beeinflussung wird vom Autor als vernachlässigbar gering eingeschätzt, sie ist aber definitiv vorhanden.

Zusammenfassung Mit dem „Packet Control Protocol for REACH“ (PCPR) können paketorientierte *logische Verbindungen* über unzuverlässig arbeitende paketorientierte Medien verwaltet werden. Die dazu notwendigen Protokollfunktionen wurden vorgestellt und diskutiert. Eine Aufschlüsselung aller PCPR-PDUs befindet sich in Anhang A.1.2.1. Die Statusmaschinen zur Abwicklung dieses Protokolls sind in Anhang A.1.2.2 zu finden. Für den Transport von Nutzdaten werden PDUs mit einem möglichst geringen Anteil an Verwaltungsinformationen – im Idealfall reicht ein einziges zusätzliches Byte pro Paket aus – eingesetzt.

5.3 Datenstromsicherung

Durch die beiden Multiplexer entstehen zwei Summendatenströme, welche zu den entsprechenden Partnerinstanzen (die assoziierten Multiplexer an der Gegenstation) übertragen werden müssen. Während der erste Summendatenstrom stromorientierter Natur ist und am Ziel exakt in derselben Form wieder vorliegen muss, ist der zweite Datenstrom paketorientierter Natur. Stromorientierte Datenströme unterliegen in Bezug auf die Übertragung anderen Anforderungen als paketorientierte Datenströme. Im Folgenden werden die beiden Sicherungsinstanzen vorgestellt, welche die erforderlichen Sicherungsmechanismen implementieren.

5.3.1 Sicherung des stromorientierten Summendatenstroms

Zur Übertragung stromorientierter Daten eignen sich TCP-Verbindungen, wobei TCP an dieser Stelle ungeeignet ist, da es im mobilen Umfeld versagt und die Konsistenz des Summendatenstroms nicht sicherstellen kann. Die für stromorientierte Daten erforderliche Sicherungsinstanz muss eine mit TCP vergleichbare Übertragungsqualität aufweisen, jedoch zusätzlich um mobilitätsspezifische Sicherungsschemata erweitert worden sein. Der so gesicherte Datenstrom kann anschließend bedenkenlos mittels TCP übertragen werden, wobei im Fehlerfall die Mechanismen der Sicherungsinstanz greifen sollen. Das hierfür entwickelte Übertragungsprotokoll wird „Stream Protection Protocol for REACH“ (SPPR) genannt, dessen PDUs in Anhang A.2.1 aufgeführt sind. Die erforderlichen Protokollfunktionen werden im Folgenden herausgearbeitet.

5.3.1.1 Sitzungsaufbau

Jede stromorientierte Sicherungsinstanz ist mit einer Partnerinstanz assoziiert, sodass auf dieser Ebene genau *eine* stromorientierte Kommunikationsbeziehung zwischen einem REACH-Client und einem REACH-Server dargestellt wird. Eine solche „Sitzung“ ist langlebig und zeitlich nicht von den Standzeiten der einzelnen Übertragungskanäle zwischen REACH-Client und REACH-Server abhängig. Es findet ein einmaliger Sitzungsaufbau statt, bei welchem sich die beiden Sicherungsinstanzen miteinander synchronisieren.

Während dieser Synchronisierungsprozedur werden die Größen der beidseitig vorhandenen Empfangspuffer ausgetauscht. Diese Information wird benötigt, damit beide Sicherungsinstanzen wissen, wie viele Daten versendet werden dürfen ohne den Empfangspuffer der Partnerinstanz zu überfluten.

Für diese Synchronisation tauschen beide Sicherungsinstanzen eine `SPPR_SYN`-PDU

aus, wobei Nutzdaten versendet werden dürfen, sobald die SPPR_SYN-PDU der Partnerinstanz empfangen worden ist. Erhaltene SPPR_SYN-PDUs müssen mit einer SPPR_SYNACK-PDU bestätigt werden. Die Partnerinstanz wird daraufhin die Versendung weiterer SPPR_SYN-PDUs einstellen.

5.3.1.2 Datenversendung und Quittierung

Da REACH zur Übertragung TCP-Verbindungen benutzen soll und hierbei jederzeit mit Abrissen und damit mit Datenverlusten rechnen muss, implementieren die stromorientierten Sicherungsinstanzen einen Bestätigungsmechanismus. Nutzdaten werden dabei in Form von Fragmenten versendet, welche mit Sequenznummern versehen sind, sodass ihre Position im Gesamtdatenstrom bekannt ist. Der Empfänger hat die Aufgabe, die eintreffenden Fragmente wieder zu einem lückenlosen Gesamtdatenstrom zusammenzufügen, wobei ordnungsgemäß erhaltene Fragmente quittiert werden.

REACH zieht zwei unterschiedliche Informationen zur Formulierung von Bestätigungen heran. Zum einen wird die momentan höchste Sequenznummer des empfängerseitig lückenlos vorliegenden Datenstroms quittiert, was im Fall einer ungestörten Übertragung bereits ausreichend ist, da der Empfangsdatenstrom hierbei lückenlos anwächst. Auf Grund von Datenverlusten durch Handoverereignisse oder bei einer „Mehrwegeversendung“ über unterschiedlich schnelle Medien kann es jedoch zu Lücken im Empfangsdatenstrom kommen, wobei die besagte obere Sequenznummer nicht weiter ansteigen wird sich damit und zur Bestätigung eintreffener Nachfolgefragmente als ungeeignet erweist. Damit aber auch Fragmente mit Sequenznummern oberhalb von Lücken bestätigt werden können, erlaubt REACH eine zusätzliche Bestätigung selektiver Sequenznummernbereiche. Der Mechanismus ist verwandt mit der selektiven Blockbestätigungen moderner TCP-Implementierungen („TCP Selective ACK“ (TCP-SACK, [MMFR96])).

Datenfragmente des Typs SPPR_DATAACK sowie reine Bestätigungsnachrichten des Typs SPPR_ACK benennen *immer* die höchste lückenlos empfangene Sequenznummer. Falls sich diese zwischenzeitlich jedoch nicht geändert haben sollte, können Datenfragmente auch ohne Quittungsfeld verschickt werden, wozu sich eine SPPR_DATA-PDU anbietet. Alle drei PDU-Typen können bei Bedarf bis zu 63 SACK-Einträge mitführen. Somit können Übertragungswiederholungen erfolgreich empfangener Datenblöcke unterdrückt werden, wenn sich diese außerhalb des lückenlos empfangenen Bereichs befinden.

Die SPPR-Bestätigungsinformationen werden bei REACH pro Übertragungskanal nur einmal verschickt. Es macht keinen Sinn, einen SACK-Block mehrfach anzuzeigen oder eine sich nicht geänderte „lückenlos empfangene Sequenznummer“ wieder und wieder

zu kommunizieren. Als Übertragungskanal kommen schließlich TCP-Verbindungen zum Einsatz, welche „in sich“ keinen Datenverlust erzeugen, sodass pro TCP-Verbindung der Übertragungsaufwand bezüglich der SPPR-Bestätigungen minimiert werden kann.

5.3.1.3 Flusssteuerung

Derartige „Puffer-zu-Puffer“ Betriebsmodi machen „in der Regel“ eine Flusssteuerung notwendig, damit der Empfangspuffer vor Überflutung durch den Sender geschützt wird. Denn sollte der Sender auf Grund einer fehlenden Flusssteuerung *mehr* (im Sinne von „schneller“) Daten senden als der Empfänger aus dem Empfangspuffer entnimmt, käme es zu einem Überlauf, wobei Datenfragmente trotz fehlerfreiem Empfang verworfen werden müssten. Ihr Empfang dürfte nicht bestätigt werden, schließlich müssen die Daten zu einem späteren Zeitpunkt – wenn hoffentlich genügend Platz im Empfangspuffer entstanden ist – erneut versendet werden.

So verworfene Datenfragmente werden allerdings erst nach Ablauf eines Timers vom Sender erneut verschickt. Zwischenzeitlich kommt es zu einem Stocken im Datenfluss, da die im Empfangspuffer entstandene Lücke den Abruf der Daten verhindert. Werden währenddessen noch weitere Fragmente empfangen, welche wiederum nicht in den vollen Empfangspuffer übernommen werden können, verschlimmert sich die Situation weiter. Durch die vielen Fragmentverwerfungen und anschließenden Timeoutbehandlungen kommt es zu einer geringen effektiven Nutzdatenrate.

Diskussion der Notwendigkeit bei REACH Die Frage ist nun, ob die Notwendigkeit einer Flusssteuerung auch im Fall von REACH besteht. Zwar wurde gerade gezeigt, dass ein Sender prinzipiell schneller Daten senden kann als der Empfangspuffer geleert wird, aber bei REACH gilt noch eine andere Randbedingung: Auf Ebene des Multiplexers wird bereits eine Flusssteuerung bezüglich der *logischen Verbindungen* durchgeführt, sodass für dem Multiplexer übergebene Daten *immer* ein genügend großer Empfangspuffer bereit steht. Ist es also notwendig, auf Sicherungsebene nochmals eine Flusssteuerung durchzuführen, wenn durch die Protokollfunktionen von SCPR bereits sichergestellt wird, dass Daten aus dem SPPR-Empfangspuffer (bestehend aus SCPR-PDUs) jederzeit verarbeitet werden können? Reicht es nicht aus, im „Blockierungsfall“, wenn also ein Fragment den Empfangspuffer „am Ende überragt“ und abgeschnitten werden müsste, den Multiplexer „anzustoßen“, damit er den Empfangspuffer abruft? Interessanterweise kann auf eine Flusssteuerung nicht verzichtet werden.

Das Problem wird sichtbar, wenn ein `SPPR`-Datenfragment empfangen, ordnungsgemäß in den Empfangspuffer geschrieben und bestätigt wird. Dabei kann es passieren,

dass der reassemblierte Datenstrom eine Lücke aufweist, und dem Multiplexer noch keine vollständige SCPR-PDU angeboten werden kann, sodass auf weitere SPPR-Datenfragmente gewartet werden muss. Kritisch wird es, wenn genau *die* fehlende SPPR-PDU mit dem lückenfüllenden Fragment verloren geht. Der Sender ist sich nämlich nicht bewusst, dass der *vorderste* Datenblock immer noch im Empfangspuffer liegt, schließlich ist er ordnungsgemäß quittiert worden und hätte schon längst abgerufen werden können. Der Sender geht fälschlicherweise davon aus, dass Platz im Empfangspuffer entstanden ist, und schiebt sein „Sendefenster“ weiter. Damit werden jedoch Datenblöcke versendet, welche definitiv nicht mehr in den Empfangspuffer passen und verworfen werden müssen.

Der ursprüngliche Paketverlust wird also nicht nach Erreichen eines Timeouts mit anschließender Neuversendung behandelt, sondern zieht die Verwerfung weiterer Fragmente mit erneuten timeoutbehafteten Wiederholungen nach sich. Der Datenfluss wird nachhaltig gestört, was eine Flusssteuerung unumgänglich machte.

Möglichkeit zur Flusssteuerung: Quittungsfelder Im Folgenden werden zwei Mechanismen zur Flusssteuerung vorgestellt, welche nacheinander für REACH umgesetzt und untersucht worden sind. Die erste Variante bestand in der Einführung eines Sequenznummernfelds für Flusssteuerungszwecke, welches Bestandteil einer jeden Quittung war. Dieses Feld wurde mit der so genannten „Empfangspuffer-Startsequenznummer“ gefüllt, und gab dem Empfänger dieser Nachricht eine Aussage über den Zustand des Empfangspuffers der Partnerinstanz. Der Erhalt neuer Daten löste die Versendung einer Bestätigung aus, welche den ordnungsgemäßen Empfang anzeigte, jedoch noch nicht das Fenster verschob, da die Daten ja noch im Empfangspuffer lagen. Erst wenn die höhere Schicht (der Multiplexer) die Daten abgerufen hatte, wurde der Empfangspuffer „von vorne her“ geleert und es entstand freier Platz. Die „Empfangspuffer-Startsequenznummer“ erhöhte sich, was mittels einer weiteren Bestätigungsnachricht der Partnerinstanz mitgeteilt wurde.

Diese Form der Flusssteuerung arbeitete zuverlässig, löste die vorgestellte Problematik und wurde eine Zeit lang in REACH eingesetzt. Das Hauptproblem dieser Form der Flusssteuerung bestand jedoch in einem hohen zusätzlichen Übertragungsaufwand. So mussten Fragment- und Bestätigungsnachrichten um ein Bestätigungsfeld für die Startsequenznummer erweitert werden, und es kam zu einem starken Anstieg der Menge an Nachrichten auf dem Medium. Im schlimmsten Fall wurden zwei Bestätigungen pro empfangenem Datenfragment verschickt, wobei erstere den Empfang quittierte, während letztere den Abruf durch den Multiplexer anzeigte (ähnlich dem „Silly Window“-Verhalten von TCP [Clar82]).

Anfangs drohte dieser Mechanismus dauerhaft zu blockieren, wenn der Empfangspuf-

fer komplett gefüllt und alle Daten ordentlich bestätigt waren, aber die nach Abruf zurücklaufende Flusssteuerungs-Bestätigung verloren ging. Dann durfte der Sender keine weiteren Nutzdaten mehr versenden, aber es wurden auch keine neuen Bestätigungen mehr formuliert, da ja alle Fragmente bereits bestätigt worden und der Empfangspuffer geräumt war. Auch dieses Problem wurde gelöst, und zwar durch Einführung einer `SPPR_ACKACK`-PDU, welche benutzt wurde, um Quittungen nach Erhalt nochmals zu bestätigen. Damit wurde sichergestellt, dass Flusssteuerungs-Bestätigung den Empfänger erreichten und der vorgestellte Blockierungsfall nicht länger eintreten konnte. Dieser Mechanismus war komplex in der Implementierung und sorgte für „eine Flut an Bestätigungen“ auf dem Medium.

Die aktuelle Form der Flusssteuerung in REACH Die ursprüngliche Form der Flusssteuerung wurde schließlich durch einen besser geeigneten Mechanismus abgelöst, welcher ohne zusätzliche Paketkopffelder („Startsequenznummer“) und ohne `SPPR_ACKACK`-PDUs auskommt, aber dennoch alle Facetten der vorgestellten Problematik löst. Diese Form der Flusssteuerung funktioniert wie folgt:

1. Jede Seite reserviert einen Empfangspuffer, dessen Größe allerdings der Partnerinstanz nicht mehr komplett mitgeteilt wird. Sie wird um die maximal mögliche Größe einer „Service Data Unit“ (SDU, 8197 Byte als „worst case“ bei einer vollständig gefüllten `SCPR_DATA2O`-PDU) reduziert (genauer: SDU-Größe minus 1 Byte, also um 8196 Byte reduziert). Die so errechnete Empfangspuffergröße wird zu Beginn mit Hilfe der `SPPR_SYN`-PDUs beidseitig kommuniziert.

2. Die geschilderte Situation, dass empfangene Fragmente über das Ende des Empfangspuffers hinausragen und verworfen werden müssen, ist damit ausgeschlossen. Die Daten ragen von nun an maximal noch in den nicht angezeigten Bereich des Empfangspuffers hinein und können daher immer vermerkt werden.

3. Bestätigt wird allerdings nur bis maximal zum Ende des angezeigten Bereichs des Empfangspuffers, das heißt, die „überragenden“ Daten werden zwar inhaltlich vermerkt, aber noch nicht oder noch nicht zur Gänze bestätigt.

4. Kommt es zur Schließung von Lücken durch verspätet eintreffende oder wiederholte Fragmente, verschiebt nach anschließendem Abruf der Nutzdaten der Inhalt des Empfangspuffers, und die bisher unbestätigten Daten rutschen in den angezeigten Bereich. Sie werden dann sofort bestätigt, sodass am Sender idealerweise keine Übertragungswiederholung notwendig wird. Es kommt dabei nicht länger zu einer negativen Beeinflussung der Datenrate.

5.3.1.4 Staukontrolle

Ein weiterer Aspekt bei der Steuerung der Datenrate ist die Staukontrolle. Sie dient nicht dem Schutz des Empfängers, sondern dem Schutz des Netzwerks vor Überlastung durch den Sender. Mechanismen zur Staukontrolle sind komplex, und TCP begegnet diesem Problem mit dem „Slow Start"-Mechanismus, bei welchem die Datenrate „von unten her" an das Optimum angenähert wird und um dieses herum pendelt. Für REACH brauchen derartige Probleme nicht betrachtet zu werden, da der gesicherte stromorientierte Datenstrom schlussendlich über TCP-Verbindungen übertragen, und die Staukontrolle damit bereits erbracht wird.

5.3.1.5 Schutz vor Verfälschung

Die Sicherungsinstanz für stromorientierte Daten benötigt keine eigenen Mechanismen zur Erkennung von Bitfehlern. Diesbezügliche Mechanismen sind bereits Bestandteil von TCP, welches zur Datenübertragung herangezogen wird. REACH benötigt damit zwingend TCP als Transportschichtprotokoll.

5.3.1.6 Mehrfache Versendung und Dublettenerkennung

Idealerweise wird ein Fragment des Summendatenstroms nur einmal versendet und nach Erhalt durch den Empfänger bestätigt. Zum einen können jedoch Datenverluste durch Handoverereignisse auftreten, zum anderen lassen sich unterschiedliche Betriebsmodi bezüglich der Kommunikationswege unterscheiden. Dies wird deutlich, sobald mehrere Netzzugänge gleichzeitig in die Datenübertragung involviert werden.

Beim Kanalbündelungsszenario werden mehrere Übertragungsstrecken in einen „neue Daten"-Modus geschaltet, sodass sich der Gesamtdatenstrom auf verschiedene Wege „auffächern" kann und damit die erreichbare Datenübertragungsrate erhöht wird.

Weiterhin gibt es für „weichere Handover" den „redundante Daten"-Modus. Über Kommunikationswege in diesem Modus sollen nur solche Fragmente versendet werden, welche bereits anderweitig als „neu" verschickt worden sind.

Am Empfänger existiert pro Sitzung (pro Summendatenstrom) ein Empfangspuffer. Für übernommene Fragmente werden Bestätigungen formuliert, wobei je nach Betriebsmodus auch eine redundante Versendung von Bestätigungsnachrichten stattfinden kann. Lückenlos vorliegende Daten werden von der Multiplexer-Instanz entgegen genommen.

Für jedes versendete Fragment wird eine Wartezeit festgelegt. Verstreicht diese, ohne dass eine diesbezügliche Quittung empfangen wurde, wird das Fragment für eine erneute Übertragung herangezogen. Die Wartezeit ist abhängig von der Übertragungsstrecke und basiert auf der aktuellen Umlaufzeit. Deren Ermittlung wird in Abschnitt 5.5.4 erläutert.

Es gibt jedoch keine *obere* Zeitschranke, welche zu einem Sitzungsabriss führen könnte. Zwei miteinander assoziierte stromorientierte Sicherungsinstanzen können beliebig lange voneinander getrennt werden, was die Eignung von REACH für längerfristige Isolationssituationen unterstreicht.

5.3.1.7 Sitzungsabbau

Für den Fall, dass ein REACH-Client oder ein REACH-Server beendet werden, können bestehende Sitzungen geordnet beendet werden. Bei REACH reicht es hierfür aus, wenn die stromorientierte Sicherungsinstanz ihrer Partnerinstanz eine `SPPR_FIN`-PDU schickt. Wird eine solche PDU empfangen, gilt die betroffene Sitzung augenblicklich als ungültig, und alle involvierten *logischen Verbindungen* und *Linkbündel* gelten als „von der Partnerinstanz geschlossen“. Beide Sicherungsinstanzen und beide Multiplexer werden schließlich freigegeben. Das funktioniert allerdings nur, wenn zum Zeitpunkt des „Herunterfahrens“ ein Datenaustausch möglich ist. Im Isolationsfall verbleibt einseitig eine verwaiste Sitzung, welche so lange bestehen bleibt, bis sie durch eine neue Nachfolgesitzung ersetzt wird.

5.3.2 Sicherung des paketorientierten Summendatenstoms

Auch der paketorientierte Summendatenstrom muss vor den negativen Auswirkungen von Handoverereignissen geschützt werden. Die im Folgenden aufgeführten Effekte haben dieselben Ursachen, wie es schon bezüglich der stromorientierten Daten beschrieben wurde, jedoch zeigen sich hier andere Auswirkungen.

5.3.2.1 Auftretende Effekte

Paketorientierte *logische Verbindungen* sollen eine ähnliche Qualität erbringen, wie man sie von IP oder UDP erwartet. „Datenströme“ werden hier als eine lose Folge von Paketen betrachtet, welche über keine Sequenznummer zur Detektierung von Verlusten, Vertauschungen und Duplizierungen verfügen.

Bedingt durch diese Eigenschaften erweist sich der Schutz des paketorientierten Summendatenstroms als weniger umfassend als der bereits präsentierte Schutz für stromorientierte Daten. Dennoch sollen die Auswirkungen von Paketverlusten, von Reihenfolgevertauschungen und von Duplizierungen angesprochen werden.

Paketverluste Die Verwaltung paketorientierter *logischer Verbindungen* wurde bereits mit Blick auf Paketverluste entwickelt, sodass hier keine weiteren Probleme zu erwarten

sind. Die Anwendungen, welche die eigentlichen Verursacher der zu übertragenden Nutzdaten sind, müssen mit Paketverlusten rechnen und bei Bedarf eigene Mechanismen zur Behandlung implementieren. Diese Diskussion wurde bereits in Abschnitt 5.2.4 geführt.

Reihenfolgevertauschungen Ein weiterer Effekt, der durch Handoverereignisse verursacht werden kann, sind Reihenfolgevertauschungen. Dabei erreichen die Pakete den Empfänger in einer anderen Reihenfolge, als der Sender sie losgeschickt hat. Durch Handoverereignisse können nachfolgende Pakete einen „schnelleren“ Weg nehmen als Pakete, welche bereits vor dem Handoverereignis verschickt worden sind.

Reihenfolgevertauschungen können jedoch auch durch die Vermittlungsschicht hervorgerufen werden. Da IP verbindungslos arbeitet, ist nicht sichergestellt, dass aufeinanderfolgende IP-Pakete (und damit auch UDP-Pakete) identische Wege durch das Internet nehmen. Somit kann es auch ohne REACH zu Reihenfolgevertauschungen kommen, mit der Konsequenz, dass die betroffenen Anwendungen ihrerseits bereits je nach Bedarf Schutzmechanismen (zum Beispiel in Form mitgeführter Sequenznummern) vorsehen müssen. Für REACH wird keine zusätzliche Sicherung bezüglich der mobilen Teilstrecke benötigt.

Dubletten Eine Besonderheit von REACH besteht darin, dass Pakete bewusst über mehrere Netzzugänge gleichzeitig versendet werden können, damit die Wahrscheinlichkeit eines „Nutzdatenverlusts“ verringert wird. Dadurch kommt es jedoch zu einem mehrfachen Empfang inhaltlich identischer Pakete an der Partnerinstanz, welche nun die Wahl hat, die Pakete entweder zu filtern (um Dubletten zu verwerfen) oder sie stillschweigend zu akzeptieren und an den Multiplexer „hochzureichen“.

Es ist wichtig zu wissen, ob Dubletten als „schädlich“ betrachtet und durch zusätzliche Mechanismen gefiltert werden müssen. Dafür müsste die Frage beantwortet werden, ob man bei datagrammorientierten Übertragungen wie bei UDP von den Anwendungen erwarten darf, dass sie sich nicht von Dubletten verwirren lassen. Da allerdings nicht generell ausgeschlossen werden kann, dass sich bestimmte Anwendungen auf ein Netzwerk verlassen, welches keine oder nur wenige Dubletten verursacht, erscheint ein entsprechender Schutz für REACH als sinnvoll. Dies gilt besonders, da REACH im Fall von redundanter Nutzdatenversendung bewusst große Mengen an Duplikaten generiert. Wenigstens die aktiv durch REACH erzeugten Duplikate sollen herausgefiltert werden.

5.3.2.2 Markierung redundant versendeter Pakete

Um mehrfach empfangene Pakete als solche erkennen zu können, müssen sie senderseitig mit einer zusätzlichen Kennzeichnung ausgestattet worden sein. Dafür eignen sich Sequenznummern, welche die Pakete in aufsteigender Reihenfolge kennzeichnen und dem Empfänger somit die Möglichkeit geben, Pakete mit einer bereits gesehenen Sequenznummer zu verwerfen.

Als nachteilig erweist sich der erhöhte Verwaltungsaufwand, da einerseits jedes Paket um eine Sequenznummer erweitert werden und zusätzlich am Empfänger eine Prüfung durchgeführt werden muss.

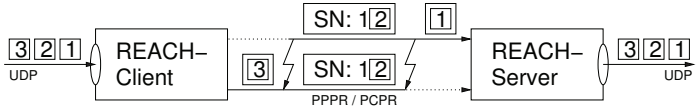

Abbildung 5.22: Das „Packet Control Protocol for REACH“ (PCPR) wird um das „Packet Protection Protocol for REACH“ (PPPR) erweitert, welches bei Bedarf einen Erweiterungsheader mit einer Sequenznummer (SN) in die PCPR-PDU einfügt. So können Dubletten am Empfänger erkannt und gefiltert werden.

Um diesen Mehraufwand bei der Übertragung und der Auswertung zu begrenzen, werden die Sequenznummern nur *bei Bedarf* zu den Paketen hinzugefügt (siehe Abbildung 5.22). Pakete, die ohne weitere Kopien von einer Multiplexer-Instanz verschickt werden, brauchen am Empfänger nicht auf nachfolgende Dubletten hin überprüft zu werden. Die Sequenznummer kann hier eingespart werden und der Empfänger darf das Paket ohne Durchlaufen einer Prüfinstanz sofort auswerten.

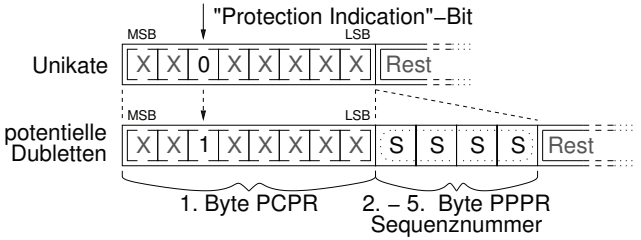

Abbildung 5.23: Das „Packet Protection Protocol for REACH“ (PPPR) erlaubt es, dass bei mehrfach versendeten PCPR-PDUs im ersten Byte des Kopfes das „Protection Indication“-Bit gesetzt und ein 4 Byte großer Erweiterungskopf eingefügt wird. Dieser enthält die Sequenznummer des Pakets.

Im Fall einer redundanten Datenversendung werden Sequenznummern eingefügt. Bei jedem betroffenen Paket wird im ersten Byte des PCPR-Paketkopfes das „Protection Indication“-Bit (PI-Bit) gesetzt, welches das Vorhandensein eines Erweiterungskopfes für die Dublettenerkennung anzeigt (siehe Abbildung 5.23). Dieser Erweiterungskopf enthält die Sequenznummer des Pakets. In den Kopf eines jeden Pakets, welches mehr als einmal verschickt wird, setzt der Sender das PI-Bit und fügt die Sequenznummer ein. So kann der Empfänger durch Prüfung des PI-Bits eine potentielle Dublette erkennen und gegebenenfalls die Sequenznummer prüfen. Im Normalbetrieb ohne redundante Versendung entsteht bis auf das ungesetzte PI-Bit kein zusätzlicher Übertragungs- und Prüfaufwand.

5.3.3 Zusammenfassung

Die Aufgaben der vorgestellten Sicherungsinstanzen von REACH lassen sich wie folgt zusammenfassen: Für die Übertragung des stromorientierten Summendatenstroms wird ein mit TCP vergleichbarer Übertragungskanal angeboten, welcher redundante Übertragungen und Kanalbündelungen, Verbindungsabrisse sowie längerfristige Isolationssituationen erlaubt. Für paketorientierte Daten wird ein Übertragungskanal angeboten, welcher mit UDP vergleichbar ist und lediglich um eine Dublettenerkennung erweitert zu werden brauchte.

5.4 Isolationserkennung

In Abschnitt 4.3.2.6 wurden bereits die Herausforderungen von SIP-basierter Sprachkommunikation im mobilen Umfeld herausgearbeitet. Dabei wurde besonders auf die Problematik von Isolationssituationen aufmerksam gemacht, da diese zu Fehlerzuständen der mobilen SIP-Telefone sowie zu Verwirrungen seitens der Benutzer führen, wenn bestehende Gespräche abrupt verstummen. Die zugehörigen „Relay Plugins“ agieren hierzu während Isolationssituationen autonom, bestätigen eigenständig die Signalisierungsnachrichten der SIP-Telefone und spielen beidseitig eine Ansage in bestehende Gespräche ein. Jedoch blieb bislang offen, wie und vor allem wie schnell der REACH-Client und der REACH-Server solche Isolationssituationen erkennen können, um in den autonomen Betriebsmodus wechseln zu können.

Eine solche Isolationssituation liegt beispielsweise dann vor, wenn das mobile Endgerät über keinen Netzzugang verfügt. Zwar ist sich der Handoverentscheider des mobilen Endgeräts dieser Situation bewusst, jedoch kann er diese Information mangels Netzzugang nicht an den REACH-Server kommunizieren. Da der REACH-Server diese

Information jedoch auch benötigt, muss dieser eine Isolationssituation anderweitig ableiten. So kann er beispielsweise alle eingehenden Datenkanäle mit ihrem momentanen Datenverkehr beobachten und hieraus versuchen, eine Isolationssituation zu erkennen. „Leider“ ist es aber so, dass Transportschichtverbindungen nicht augenblicklich beidseitig abreißen, wenn sich die Kommunikationspartner nicht mehr erreichen können. TCP-Verbindungen reißen im ungünstigsten Fall nach 9 Minuten ab, wenn keine ICMP-Fehlermeldungen zugestellt werden können. Abrisse sind zudem nur erkennbar, wenn Daten versendet werden sollen. Solange sich TCP-Verbindungen „im Leerlauf“ befinden, ist es für die Kommunikationspartner nicht möglich, einen Abriss zu erkennen.

REACH implementiert daher eine eigene Isolationserkennung, welche auf der beidseitigen Auswertung des empfangenen Datenstroms basiert. Bleiben für eine gewisse Zeit Datenpakete aus, wird dies als Isolationssituation gewertet und die lokale Instanz wechselt in den autonomen Betriebsmodus. Das setzt allerdings voraus, dass die Partnerinstanz auch für einen fortwährenden Datenstrom sorgt. Sind keine Nutzdaten zu verschicken, müssen wenigstens „Keep-Alive“-Pakete (bei REACH: `PCPR_ISOPROBE`-PDUs) versendet werden.

„Keep-Alive“-Pakete weisen jedoch das Problem auf, dass sie zusätzlichen Übertragungsaufwand verursachen. Je schneller eine Isolationssituation erkannt werden soll, desto höher muss die Frequenz der zu versendenden Pakete ausfallen. Für REACH wurde daher eine mehrstufige Isolationserkennung implementiert. Sie ermöglicht es, je nach Anforderung auf eine höhere Paketrate umzuschalten, sodass Isolationssituationen wahlweise in kürzerer Zeit erkannt werden können.

Das einzige „Relay Plugin“, welches momentan Gebrauch von der Isolationserkennung macht, ist das „Relay Plugin“ für SIP-basierte Sprachkommunikation. Hierfür war es sinnvoll, mehrere „Auflösungen“ für die Erkennung zu definieren. So wird unterschieden, ob momentan *kein* SIP-Telefon angemeldet ist, ob *ein oder mehrere* SIP-Telefone angemeldet sind oder ob momentan sogar ein Austausch von Mediendaten im Gange ist. An dieser Einteilung orientiert sich die dreistufige Isolationserkennung:

- **Keine Isolationserkennung:** Es werden keine zusätzlichen Pakete verschickt, und der Empfangsdatenstrom wird nicht auf Pausen, welche auf eine Isolationssituation hindeuten könnten, hin untersucht.

- **Isolationserkennung mit niedriger Auflösung:** Senderseitig wird sichergestellt, dass wenigstens alle 2000ms ein Paket verschickt wird. Empfängerseitig wird nach einer Wartezeit von 6000ms eine Isolationssituation angezeigt. Das ist ausreichend kurz, um SIP-Telefone daran zu hindern, in einen Fehlerzustand zu wechseln.

- **Isolationserkennung mit hoher Auflösung:** Besteht ein Mediendatenstrom, wird die Auflösung der Isolationserkennung erhöht. Eine Isolationssituation wird hierbei bereits nach einer Empfangspause von 1500ms angezeigt. Das ist genügend kurz, damit eine Ansage rechtzeitig gestartet werden kann, bevor einer der Nutzer irritiert auflegt. Senderseitig wird wenigstens alle 250ms ein Datenpaket versendet, was besonders bei Verwendung von „Voice Activity Detection“ zur Versendung von zusätzlichen Paketen führen kann.

Wichtig ist, dass die erforderliche Auflösung sowohl am REACH-Client als auch am REACH-Server angefordert werden muss. Dafür sind die „Relay Plugins“ verantwortlich, welche beispielsweise beidseitig einen Gesprächsdatenstrom erkennen können und dann eine hohe Auflösung vom transportorientierten Kern anfordern müssen. Die Isolationserkennung betrifft dann genau eine Sitzung zwischen einem REACH-Client und einem REACH-Server, unabhängig davon, wie viele „Relay Plugins“ diese Sitzung benutzen und wie viele Kommunikationswege (Netzzugänge) in die Sitzung involviert sind. Die Isolationserkennung kann sogar mehrfach angefordert werden (z.B. wenn mehrere Telefongespräche bestehen), was jedoch *nicht* zu einem höheren Datenaufkommen führt, als wenn die Erkennung nur einmal angefordert worden wäre. REACH merkt sich, wie oft die niedrige und die hohe Auflösung angefordert worden sind, und leitet daraus den erforderlichen Betriebsmodus ab.

Die Isolationserkennung ist „oberhalb“ der Datenstromverteilung angesiedelt (vgl. Abbildung 5.1), welche im nächsten Abschnitt vorgestellt wird. Für die Isolationserkennung ist es nicht relevant, ob ein Kommunikationskanal zur Partnerinstanz *existiert*. Wichtiger ist, ob die Pakete das Ziel auch wirklich *erreichen* können. Verfügbare Kommunikationskanäle tragen dazu nämlich nur dann bei, wenn der Handoverentscheider sie auch für die Datenübertragung heranzieht.

5.5 Datenübertragung

Die an dieser Stelle vorliegenden Daten wurden bereits mit Blick auf eine negative Beeinflussung durch Handoverereignisse geschützt und können zudem mit Paketen für die Isolationserkennung durchsetzt sein. Sie sind bereit für die Versendung, wofür nicht modifizierte Transportschichtprotokolle eingesetzt werden sollen.

5.5.1 Verteilung der Datenströme

Bei der Versendung muss das Ergebnis des Handoverentscheiders berücksichtigt werden, wobei für jeden Kommunikationsweg eine bestimmte Nutzungsart vorgesehen wurde. Ein

solcher Kommunikationsweg umfasst eine Kommunikationsbeziehung mittels Transportschichtverbindungen zwischen einem REACH-Client und einem REACH-Server über einen bestimmten Netzzugang des mobilen Endgeräts.

Bezüglich eines jeden Netzzugangs wird vom Handoverentscheider bestimmt, ob Kommunikationsbeziehungen zu REACH-Servern erwünscht sind. Falls dies der Fall ist, werden weitere Betriebsmodi unterschieden, welche die Art und Weise des Datenversands genauer bestimmen. Im späteren Kapitel 6.2.1.3 wird detailliert auf die Handoverentscheidung und deren Ergebnis eingegangen; an dieser Stelle sei jedoch bereits vorweggenommen, dass bezüglich der Versendung gesicherter Nutzdaten pro Netzzugang die folgenden Betriebsmodi unterschieden werden:

- **Sende keine Daten:** Entweder besteht über die zugehörige Netzzugangstechnologie gegenwärtig kein Netzzugang, oder es soll keine Datenübertragung trotz bestehendem Netzzugang durchgeführt werden.
- **Sende *neue* Daten:** Hierbei besteht ein bestimmter Netzzugang, und er soll für den Austausch *neuer* Daten zwischen REACH-Client und den REACH-Servern herangezogen werden. *Neu* bedeutet, dass damit Daten gemeint sind, welche bislang noch nicht versendet worden sind. Werden mehrere Netzzugänge gleichzeitig in diesen Modus geschaltet, erhält man ein Kanalbündelungsszenario. Daten, welche als *neu* verschickt worden sind, werden nicht nochmals über andere Wege versendet, wenn über diese ebenfalls *neue* Daten versendet werden sollen.
- **Sende *redundante* Daten:** In diesem Betriebsmodus wird ebenfalls eine Datenübertragung über einen bestehenden Netzzugang angestrebt, jedoch werden hier ausschließlich Daten versendet, welche vorher als *neu* über einen anderen Netzzugang versendet werden konnten. So lassen sich redundante Übertragungsschemata realisieren, da die Daten einmal als *neu* und zusätzlich noch *redundant* über andere Wege versendet werden können.

Da zu jeder Kommunikationsbeziehung zwischen REACH-Client und REACH-Server der Betriebsmodus bekannt ist, können an dieser Stelle die gesicherten Datenströme auf die verfügbaren Netzzugänge aufgeteilt werden. Die vorgestellten Modi sind lediglich für die Versendung wichtig. Der Empfänger kann nicht mehr erkennen, ob eine eintreffende PDU ursprünglich als *neu* oder als *redundant* versendet worden war.

5.5.2 Verwendete Transportschichtprotokolle

Für die Übertragung gesicherter Nutzdaten wird auf nicht modifiziertes TCP und UDP zurückgegriffen, womit eine größtmögliche Kompatibilität sichergestellt wird. In den stu-

dentischen Arbeiten [Hörn08] und [Dell08] wurden weiterhin die Transportschichtprotokolle „Stream Control Transmission Protocol“ (SCTP, RFC 4960 [Stew07]), „Datagram Congestion Control Protocol“ (DCCP, RFC 4340 [KoHF06]) und „Reliable Datagram Sockets“ (RDS) betrachtet. Es zeigte sich jedoch, dass der Einsatz von REACH im heutigen Internet nur dann problemlos möglich ist, wenn man sich auf den Einsatz von TCP und UDP beschränkt.

Für den Transport des gesicherten stromorientierten Datenstroms wird eine TCP-Verbindung herangezogen, während für den Transport des paketorientierten Summendatenstroms eine UDP-Assoziation benutzt wird. Beide zusammen formen einen „Kommunikationskanal“, zu welchem jedoch noch eine zweite TCP-Verbindung gehört, welche dem Austausch von Steuerungsnachrichten („Servicemaps“, siehe Abschnitt 4.4 und „Linkmaps“, siehe Abschnitt 6.2.1.3) dient. Die zwei TCP-Verbindungen und die UDP-Assoziationen werden seitens des REACH-Clients gemeinsam aufgebaut, passieren am mobilen Endgerät dasselbe Netzzugangsgerät und sind an denselben REACH-Server gerichtet. Beim Aufbau des Kommunikationskanals wird eine Authentifizierungsprozedur sowie eine Sitzungsverwaltung durchlaufen, sodass der REACH-Server den eintreffenden Kommunikationskanal zu einer bestehenden Sitzung – falls vorhanden – zuordnen kann.

Da die Datenströme immer seitens des mobilen Endgeräts aufgebaut werden, sind keine Erreichbarkeitsprobleme beim Überschreiten von NAT-Grenzen zu erwarten. Eine tiefgreifende Betrachtung der durch NAT zu erwartenden Probleme wird in Abschnitt 7.3.2 durchgeführt. Dort wird gezeigt, dass das einzig relevante Problem bezüglich UDP besteht, da diesbezügliche Einträge bei Inaktivität aus den NAT-Tabellen verschwinden.

5.5.3 „Keep-Alive“-Pakete und Abrisserkennung

Um dem Abreißen von UDP-Assoziationen während Phasen der Inaktivität zu verhindern, werden zyklisch „Keep-Alive“-Pakete (bei REACH: `PCPR_KEEPALIVE`-PDUs) versendet. Sie werden nur dann verschickt, wenn für eine Zeitspanne von 10 Sekunden keine anderweitigen UDP-Pakete über eine UDP-Assoziation übertragen worden sind. REACH-Client und REACH-Server verhalten sich diesbezüglich gleich. Die Versendung ist auch in Richtung des mobilen Endgeräts notwendig, da eine Firewall aktiv sein kann, welche eingehende UDP-Pakete nur dann akzeptiert, wenn ihr die UDP-Assoziation „bekannt“ ist. Werden für eine Zeitspanne von 180s weder UDP-Pakete versendet oder empfangen, „vergisst“ der Paketfilter der Firewall die UDP-Assoziation, und später eintreffende UDP-Pakete werden verworfen.

Dieser „kontinuierliche“ Datenstrom erfüllt noch einen weiteren Zweck. Eine Kommunikationsbeziehung zwischen REACH-Client und REACH-Server kann spontan unterbro-

chen werden, wobei dann einer der Kommunikationspartner plötzlich „verschwindet“, sodass keine Möglichkeit eines Datenaustauschs mehr besteht Die Erkennung eines solchen „Abrisses“ kann jedoch lange dauern, was bereits im Rahmen der Isolationserkennung in Abschnitt 5.4 diskutiert worden ist.

Idealerweise sollte der REACH-Client den Abriss eines Kommunikationskanals augenblicklich erkennen, und dann sofort versuchen, einen neuen Kommunikationskanal zu etablieren. Dafür wird die Abrisserkennung nicht ausschließlich an TCP-Socketfehlern festgemacht, sondern zusätzlich der eintreffende Datenstrom einer jeden UDP-Assoziation beobachtet. Da bekannt ist, dass die Partnerinstanz wenigstens alle 10 Sekunden ein UDP-Paket versendet, schließt REACH nach spätestens 25 Sekunden Inaktivität auf einen Abriss und baut den gesamten Kommunikationskanal bestehend aus beiden TCP-Verbindungen und der UDP-Assoziation ab. Die Zeitspanne von 25 Sekunden wurde willkürlich gewählt.

5.5.4 Bestimmung der Umlaufzeit

Der Sicherungsmechanismus für den stromorientierten Summendatenstrom verwendet Bestätigungsnachrichten, welche ordnungsgemäß erhaltene Datenblöcke anzeigen. Bleibt eine solche Bestätigungsnachricht aus, werden Fragmente gemäß des betroffenen Sequenznummernbereichs erneut übertragen. In Abschnitt 4.3.2.6 wurde dieser Mechanismus beschrieben, allerdings blieb die Frage offen, nach welcher Zeitspanne eine Bestätigung als „überfällig“ gilt und die betroffenen Daten für eine erneute Versendung bereit gestellt werden sollen.

Diese Zeitspanne ist abhängig vom gewählten Netzzugang (Netzzugangstechnologie) am mobilen Endgerät sowie vom REACH-Proxyserver, welcher sich „irgendwo“ im Internet befinden kann. Für jede dieser Kommunikationsbeziehungen muss eine Zeitspanne bestimmt werden, welche der Sicherungsinstanz mitteilt, ab wann ein versendetes Fragment als überfällig gelten kann. Diese Zeitspanne wird sowohl am REACH-Client als auch am REACH-Server benötigt, da die Datenstromsicherung in beide Richtungen funktionieren muss.

REACH zieht hierfür die UDP-Assoziation eines jeden Kommunikationskanals heran und bestimmt die Umlaufzeit („Round Trip Time“ (RTT)), um anschließend aus den Messergebnissen eine Zeitspanne für die Übertragungswiederholung zu berechnen.

Die Umlaufzeit wird mit Hilfe zweier Pakettypen bestimmt. Der REACH-Client verschickt hierzu eine `PCPR_RTTREQUEST`-PDU an einen REACH-Server. Diese PDU enthält einen Zeitstempel und eine Sequenznummer, und wird vom REACH-Server schnellstmöglich mit einer `PCPR_RTTREPLY`-PDU beantwortet. Diese enthält denselben Zeitstempel

und dieselbe Sequenznummer, sodass der REACH-Client nach Erhalt der rücklaufenden PCPR_RTTREPLY-PDU die Umlaufzeit ausrechnen kann. Die so bestimmte Umlaufzeit wird mit der nächsten PCPR_RTTREQUEST-PDU, welche hierfür ein weiteres Feld aufweist, an den REACH-Server geschickt. So verfügen beide Endpunkte des Kommunikationskanals über die für UDP-Pakete geltende Umlaufzeit.

Da für einen „frisch“ aufgebauten Kommunikationskanal noch keine Umlaufzeit bekannt ist, wird anfangs sekündlich eine Ermittlung durchgeführt. Die Rate wird schrittweise (nach einer willkürlich festgelegten Tabelle) vermindert, sodass nach 15 erfolgreich durchlaufenen Zyklen eine minimale Rate von einer Messung pro Minute erreicht wird. Anhand der Sequenznummer kann der REACH-Client zudem einen Verlust von Messpaketen erkennen. Er reagiert darauf mit einer Erhöhung der Messrate, wobei die anfängliche Rate von einer Messung pro Sekunde das Maximum darstellt. Damit wird erreicht, dass anfangs eine häufige Bestimmung der Umlaufzeit die Qualität des gemittelten Werts verbessert, aber dieser mit der Zeit immer seltener aktualisiert wird. Bei länger bestehenden Kommunikationsbeziehungen wird somit das Datenaufkommen reduziert. Die Umlaufzeiten werden vom REACH-Client und REACH-Server gemittelt und an die stromorientiert arbeitenden Sicherungsinstanzen übergeben.

5.5.5 Wegewahlmechanismen

Für die Übertragung der Daten werden TCP-Verbindungen und UDP-Assoziationen verwendet, welche vom REACH-Client auf einem mobilen Endgerät initiiert werden und einen REACH-Proxyserver zum Ziel haben. Um „weichere Handover“ und Kanalbündelungsszenarien unterstützen zu können, müssen mehrere Kommunikationsbeziehungen gleichzeitig über unterschiedliche Netzzugänge des mobilen Endgeräts gepflegt werden. Die Herausforderung besteht darin, dass pro Kommunikationskanal am mobilen Endgerät der Weg über einen bestimmten Netzzugang vorgeschrieben werden muss. Ausgehende Daten sollen das gewünschte Interface passieren, und der zugehörige Datenstrom in Rückrichtung soll das mobile Endgerät wiederum über dasselbe Interface erreichen.

Es wurden drei Verfahren gefunden, welche sich für REACH eignen und schlussendlich umgesetzt worden sind:

5.5.5.1 Zieladressbasierte Wegewahl

Zieladressbasierte Wegewahl stellt das „klassische“ Wegewahlschema dar, anhand welchem IP-Pakete im Internet „geroutet“ werden. Unterschiedliche Zielsysteme können anhand eindeutiger IP-Adressen voneinander unterschieden werden, sodass IP-Pakete unterschiedliche Wege durch das Internet nehmen können.

Auch bei REACH kann dieser Mechanismus zum Festlegen des involvierten Interfaces eingesetzt werden, allerdings muss ein REACH-Proxyserver hierbei über mehrere eindeutige IP-Adressen verfügen, wenn ein REACH-Client mehrere Datenströme gleichzeitig über verschiedene Wege an ihn versenden möchte. Jedes Interface am mobilen Endgerät wird mit einer der IP-Adressen des REACH-Proxyservers verknüpft, was durch das „Setzen von Routen“ in der Routingtabelle geschieht. Je nachdem, mit welcher IP-Adresse der REACH-Proxyserver adressiert wird, werden die Daten über einen bestimmten Netzzugang geleitet.

Vorteile: Zieladressbasierte Wegewahl ist für jede Plattform verfügbar, es müssen lediglich Einträge in der zentralen Routingtabelle gepflegt werden.

Nachteile: REACH-Proxyserver sollten mit mehreren IP-Adressen erreichbar sein. Ist ein solcher jedoch nur mittels einer einzigen IP-Adresse erreichbar, sind ihm bezüglich keine „weicheren Handover“ oder Kanalbündelungsszenarien möglich. Die Menge der verfügbaren IP-Adressen eines REACH-Proxyservers bestimmt die Obergrenze der gleichzeitig benutzbaren Netzzugänge am mobilen Endgerät. Ein weiterer Nachteil besteht darin, dass für die Pflege der Routingtabelle Systemverwaltungsrechte erforderlich sind.

5.5.5.2 Zieladressbasierte Wegewahl mit Bindung an ein Interface

Bei der zieladressbasierten Wegewahl besteht die Möglichkeit, am mobilen Endgerät einen Socket explizit an ein bestimmtes Interface „zu binden“. Das geschieht mit Hilfe der Socketoption `SO_BINDTODEVICE`. Die eigentliche Wegewahl erfolgt dann nicht länger anhand der Zieladresse, sondern wurde durch die Festlegung des Interfaces vorweggenommen. Dabei ist es nicht länger erforderlich, dass der REACH-Proxyserver über mehrere eindeutige IP-Adressen verfügen muss. Eine entsprechende Route bezüglich dieses Interfaces muss allerdings trotzdem existieren, nur dürfen jetzt mehrere Routen dieselbe Zieladresse benennen, da eine Eindeutigkeit nicht mehr Vorraussetzung ist.

Vorteile: Durch die explizite Vorgabe des Interfaces besteht keine Notwendigkeit mehr, dass ein REACH-Proxyserver mehrere eindeutige IP-Adressen besitzen muss.

Nachteile: Das Binden eines Sockets an ein bestimmtes Interface erfolgt mittels der `SO_BINDTODEVICE`-Socketoption, welche von der jeweiligen Zielplattform unterstützt werden muss. Nachteilig hierbei ist, dass diese Socketoption Systemverwaltungsrechte erfordert, weshalb der REACH-Client hierfür als Nutzer `root` gestartet werden müsste.

Genauer wird die „CAP_NET_RAW-Capability“ benötigt, welche ein als „normaler Nutzer“ gestartetes Programm nicht besitzt.

5.5.5.3 Quelladressbasierte Wegewahl

Statt der Zieladresse eines IP-Pakets kann auch dessen Absenderadresse für die Wegewahl herangezogen werden. Eine solche quelladressbasierte Wegewahl lässt sich mit Hilfe von „Policy Routing“ realisieren, wobei für die „Policy“ auf dem mobilen Endgerät die Absenderadresse der ausgehenden IP-Pakete betrachtet wird. Die Absenderadresse kann seitens des REACH-Clients pro Socket durch das `bind()`-Kommando spezifiziert werden. Detaillierte Informationen zu „Policy Routing“ und dessen Anwendung in REACH werden in Anhang C.2.2 geliefert.

Vorteile: Für quelladressbasierte Wegewahl ist es ausreichend, wenn jeder REACH-Proxyserver nur über eine einzige eindeutige IP-Adresse erreichbar ist. Zudem benötigt der REACH-Client hierfür keine Systemverwaltungsrechte, da das `bind()`-Kommando, mit welchem der REACH-Client die Wegewahl beeinflusst, keine besonderen Rechte erfordert.

Nachteile: Das vorgestellte „Policy Routing“ wird leider nicht von Microsoft Windows unterstützt. Zudem erfordert das Schema eine aufwendige Konfiguration des mobilen Endgeräts. Für jedes zu nutzende Interface muss im Vorfeld dem Loopback-Device eine lokale IP-Adresse zugewiesen sowie eine NAT-Regel aktiviert werden. Zur Laufzeit müssen Routen und Regeln für das „Policy Routing“ gepflegt werden, wofür Systemverwaltungsrechte erforderlich sind.

5.5.5.4 Bewertung

Alle drei Varianten sind in der Lage, mehrere gleichzeitige Datenströme über unterschiedliche Wege mit demselben REACH-Proxyserver zu pflegen. Sie eignen sich damit allesamt für die Durchführung „weicherer Handover“ und erlauben Kanalbündelungsszenarien. Sehr wichtig ist es für REACH, dass die Wegewahl nicht anhand von „Default Routen“ erfolgen darf, da sonst die Dienste „Transparenter Proxyserver“ (Abschnitt 4.3.2.2) und „Virtual Private Network“ (Abschnitt 4.3.2.5) nicht mehr funktionieren. Der REACH-Client wurde daher so entwickelt, dass zu jedem REACH-Proxyserver explizite Routen gesetzt werden müssen, und somit eventuell vorhandene „Default Routen“ ohne Funktion sind.

Die zieladressbasierte Wegewahl erwies sich als das am wenigsten praxisrelevante Verfahren, da hierbei ein jeder REACH-Proxyserver über mehrere IP-Adressen erreichbar sein muss. Steht hierbei nur eine einzige IP-Adresse zur Verfügung, sind bezüglich dieses REACH-Proxyservers ausschließlich „harte Handover“ möglich. Andererseits ist dieses Wegewahlverfahren auf allen Plattformen verfügbar. REACH benutzt es daher nur dann, wenn keine Alternativverfahren verfügbar sind.

Zieladressbasierte Wegewahl mit Interfacewahl wäre das am besten geeignete Wegewahlverfahren für REACH, jedoch setzt es Systemverwaltungsrechte für den REACH-Client voraus.

REACH favorisiert die quelladressbasierte Wegewahl, und nur wenn diese nicht verfügbar ist, wird eine zieladressbasierte Wegewahl durchgeführt. Wenn die Entscheidung auf die zieladressbasierte Wegewahl gefallen ist, wird untersucht, ob die für eine Interfacewahl notwendigen Rechte zur Verfügung stehen. Falls nicht, muss auf die Wahl verzichtet werden. Im schlimmsten Fall sind dann nur „harte Handover“ möglich.

Für die Manipulation der Routingtabellen und das Pflegen der Regeln des „Policy Routing“ benötigt der REACH-Client selbst keine Systemverwaltungsrechte, da er hierfür auf das so genannte „Routing Backend“ zurückgreifen kann. Es handelt sich hierbei um ein eigenständiges Programm, welches die betriebssystemspezifischen Mechanismen zur Einflussnahme auf die Wegewahl abstrahiert. Das „Routing Backend“ wird zusammen mit dem Handoverentscheider, welcher den Schwerpunkt des folgenden Kapitels darstellt, in Abschnitt 6.4 beschrieben.

5.6 Kapitelzusammenfassung

Der in diesem Kapitel vorgestellte transportorientierte Kern von REACH stellt das Bindeglied zwischen den „Relay Plugins“ und dem Betriebssystem dar. Zur Datenübertragung werden ausschließlich nicht modifizierte TCP-Verbindungen und UDP-Assoziationen herangezogen, was eine größtmögliche Kompatibilität mit dem heutigen Internet verspricht.

Die seitens der „Relay Plugins“ erforderlichen stromorientierten und paketorientierten *logischen Verbindungen* werden durch Multiplexer bereit gestellt, und die Behandlung der negativen Auswirkungen von Mobilität wird durch spezielle Sicherungsinstanzen erbracht. Zusätzliche Anforderungen, wie die Erkennung von Isolationssituationen, wurden diskutiert und sind in die Architektur von REACH mit eingeflossen.

Abschließend wurden drei Möglichkeiten der Wegewahl vorgestellt, welche es erlauben, Datenströme über bestimmte Netzzugangsgeräte des mobilen Endgeräts zu leiten.

Redundante Datenübertragungen oder Kanalbündelungsszenarien sind verfügbar. Damit wird letztendlich das Ergebnis des Handoverentscheiders umgesetzt, welcher den Schwerpunkt des nachfolgenden Kapitels darstellt.

6 Das Zugangskonzept

Eine jede Mobilitätsunterstützung, welche die Anforderungen aus Kapitel 2.1 erfüllen soll, muss die Verwaltung der Netzzugangsgeräte eines mobilen Endgeräts mit übernehmen und dafür einen so genannten Handoverentscheider implementieren. Viele bereits vorgeschlagene Mobilitätsunterstützungen, wie beispielsweise Mobile IP [Perk02] und dessen Weiterentwicklungen, blenden diese Problematik jedoch bewusst aus und kümmern sich lediglich um Teilprobleme. Da REACH jedoch eine „vollständige“ Lösung darstellen soll, müssen diese Aufgaben durch integrale Komponenten übernommen werden. In diesem Kapitel werden diese Aspekte diskutiert und jeweils die für REACH gewählte Form der Umsetzung beschrieben.

6.1 Grundlagen zur Handoverentscheidung

Vorab bietet es sich an, eine Überblick über die Problematik des Wechsels von Netzzugangstechnologien zu schaffen. Darauf aufbauend werden dann Algorithmen beschrieben, welche für die Auswahl und Überwachung von Netzzugängen verantwortlich sind.

Das Ziel ist es, eine vollautomatische Verwaltung der verfügbaren Netzzugangsgeräte zu erhalten, sodass der Nutzer eines mobilen Endgeräts sich nicht mehr mit der Auswahlproblematik zu beschäftigen braucht. Die Handoverentscheidung soll *transparent* im Hintergrund stattfinden, ohne dass zwischenzeitlich beispielsweise aus einer Liste an Alternativnetzen eines manuell ausgewählt oder Passwörter eingegeben werden müssen.

6.1.1 Auswahl eines Netzzugangs

Der Handoverentscheider steht dabei vor der folgenden Problemstellung: Er muss alle am mobilen Endgerät verfügbaren Netzzugangsgeräte verwalten, und dabei entscheiden, zu welchem der verfügbaren Netzzugangspunkte es sich mittels welcher Netzzugangstechnologie assoziieren soll.

Dafür bietet es sich an, in einem ersten Schritt alle verfügbaren Kandidaten gemäß ihrer Eignung zu sortieren, und dann den „am besten geeigneten“ auszuwählen. Hierbei stellt sich allerdings die Frage, anhand welcher Eigenschaften entschieden werden soll,

ob ein gegebener Netzzugang geeignet ist oder nicht. So können zum Beispiel die beiden Wünsche „Maximale Datenrate“ und „Minimierung der Übertragungskosten“ zu einem Widerspruch führen, wenn die schnellere Übertragungstechnologie auch gleichzeitig die höheren Kosten verursacht.

Es lässt sich erkennen, dass vorab eine Zielstellung seitens des Anwenders festgelegt werden muss. Die möglichen Zielstellungen können dabei widersprüchlich zueinander sein. Im Folgenden werden zunächst die Eigenschaften vorgestellt, anhand derer Netzzugänge bewertet werden können. Anschließend wird eine Auflistung möglicher Zielstellungen geboten, um dann im Anschluss einen Algorithmus vorzustellen, welcher eine solche Zielstellung mit den gegebenen Netzzugängen abgleicht, um schlussendlich eine sortierte Liste mit geeigneten Kandidaten zu erhalten.

6.1.1.1 Relevante Eigenschaften von Netzzugängen

Vorab sei erwähnt, dass eine als geeignet erscheinende Netzzugangstechnologie noch keine guten Übertragungseigenschaften bedeuten muss. Zum einen ist überhaupt nicht sichergestellt, dass *hinter* dem involvieren Netzzugangspunkt überhaupt eine Verbindung zum Internet besteht. Und selbst wenn, könnte dies eine vergleichsweise langsame „Digital Subscriber Line“-Anbindung (DSL) sein, welche zudem hohe Verzögerungszeiten mit sich bringt. Dieses Problem lässt sich nicht durch alleinige Betrachtung der Netzzugangstechnologie lösen, sondern erfordert zusätzliche Informationen. Mit Hilfe von Messungen könnten bestimmte Parameter automatisch ermittelt werden, aber dazu müsste ein „interessantes“ Netzzugangsgerät bereits ausgewählt und involviert worden sein. Stellt sich die anfängliche Wahl als falsch heraus, ist wertvolle Zeit verstrichen, in der das mobile Endgerät vom Internet getrennt war. Obwohl eine Betrachtung der Netzzugangstechnologie alleine also nicht ausreicht, liefert sie dennoch entscheidende Informationen für den Handoverentscheider, welcher schlussendlich die Eignung bewerten muss. Als relevante Parameter wurden folgende Eigenschaften herausgearbeitet:

Angebotene Bitrate: Unterschiedliche Netzzugangstechnologien bieten verschiedene Datenübertragungsraten an. So sind mit Ethernetkarten in heutigen Heimrechnern bereits Bruttodatenraten von 1 Gigabit pro Sekunde erreichbar. Ein solcher Netzzugang ist aus Sicht der Übertragungsrate sehr zu empfehlen, *wenn* nicht noch ein DSL-Router als „Flaschenhals“ zu überqueren ist. Das ließe sich allerdings nicht direkt erkennen, sondern müsste dem Handoverentscheider als Zusatzinformation mitgegeben werden.

Drahtlose Netze „hinken“ den drahtgebundenen Medien in Bezug auf die Bitrate hinterher, was den unvorhersehbaren Ausbreitungsbedingungen und den dadurch erforder-

lichen komplexeren Medienzugriffs- und Sicherungsverfahren geschuldet ist. Nichtsdestotrotz lassen sich Netzzugangstechnologien gemäß ihrer erreichbaren Datenrate sortieren, denn die Unterschiede zwischen den Technologien können signifikant sein.

Es ist anzumerken, dass sich Datenraten mit der Zeit ändern können. Einerseits ist die physikalisch verfügbare Datenrate nicht gleichzeitig auch die nutzbare Datenrate, da noch Datenströme anderer Nutzer über denselben Netzzugangspunkt versendet werden können. Zudem spielen bei drahtlosen Zugangsnetzen die sich ändernden Ausbreitungsbedingungen eine Rolle. Bewegt sich ein mobiles Endgerät, ändert sich die Funkstrecke, und die momentan mögliche Datenrate kann sich verbessern oder verschlechtern.

Zudem kann die verfügbare Datenrate auch künstlich begrenzt werden. Es gibt Tarife, in denen dem Anwender ein so genanntes „Freivolumen“ eingeräumt wird, welches er ohne Einschränkung der Datenrate aufbrauchen darf. Ist das Kontingent erschöpft, wird die Datenrate auf ein niedrigeren Wert gedrosselt. Anhand der physikalischen Parameter kann die Drosselung allerdings nicht erkannt werden. Eine andere Variante der Begrenzung stellt das so genannte „Teergruben-Verhalten“ dar, bei welchem die Datenrate schleichend gedrosselt wird, je nachdem, wie viele Daten bereits übertragen worden sind.

Für eine bestmögliche Handoverentscheidung müssten alle diese Eigenschaften bekannt sein, damit ein umfassender Vergleich der Zugangsalternativen untereinander möglich wird.

Übertragungsverzögerung: Ein ebenfalls wichtiger Parameter ist die so genannte Latenzzeit, also die Übertragungsverzögerung. Drahtgebundene Medien weisen in der Regel eine geringe Latenzzeit auf, da hier von geringen Übertragungsfehlerraten ausgegangen und auch Medienzugriffsverfahren mit vergleichsweise geringen Verzögerungszeiten benutzt werden können. Hohe Verzögerungszeiten trifft man bei der DSL-Technologie an, da hier umfassende Sicherungsschemata angewendet werden müssen, welche eine erhöhte Paketlaufzeit verursachen. Noch weitaus höhere Verzögerungszeiten lassen sich im drahtlosen Bereich finden, wie es beispielsweise bei GPRS der Fall ist. Hier ist mit derart hohen Verzögerungszeiten zu rechnen, dass bei Sprachkommunikation kein vernünftiger Dialog mehr durchgeführt werden kann.

Geldkosten: Die Aktivierung und die Nutzung eines bestimmten Netzzugangs kann „Geldkosten“ verursachen. Diese können sehr unterschiedlich ausfallen, daher lohnt sich eine genauere Betrachtung.

Es gibt Tarife, in denen nach der Zeit abgerechnet wird, was besonders häufig bei Analogmodems und ISDN angeboten wird. Das tatsächliche Nutzungsverhalten, also

die Menge der effektiv übertragenen Daten, spielt dabei keine Rolle. Anders verhält es sich bei den so genannte *Volumentarifen*, bei denen zwar die Onlinezeit keine Rolle mehr spielt, dafür aber die Menge der zwischenzeitlich übertragenen Daten als Abrechnungsgrundlage dient. Dieses Modell kann beispielsweise bei GPRS zum Einsatz kommen.

Einen Sonderfall hiervon stellen die fälschlicherweise als „Flatrate“ gekennzeichneten Tarife dar, welche dem Anwender beispielsweise 1 Gigabyte „Freivolumen“ pro Monat einräumen. Wird dieses Kontingent jedoch überzogen, wird auf einen Volumentarif umgeschaltet, wobei dann feingranular abgerechnet wird. Diese Modelle sind jedoch von „echten Flatrates“ zu unterscheiden, welche keine solche obere Volumengrenze aufweisen. In der Regel ist hierbei eine monatliche Pauschale zu entrichten, und es fallen keine weiteren Kosten an. Die Gültigkeitsdauer kann aber auch kürzer sein, wenn beispielsweise tagesweise (Mobilfunk) oder stundenweise (bei „Hot Spots“) abgerechnet wird. Aktiviert man hierbei seinen Internetzugang erstmalig, fällt die Nutzungsgebühr an, jedoch werden bei einer Trennung mit anschließender Neuverbindung keine weiteren Kosten in Rechnung gestellt.

Die Problematik der Kostenbetrachtung ist sehr vielfältig. Es müssen unterschiedlichste Tarifmodelle berücksichtigt werden, wobei auch die Uhrzeit, der Aufenthaltsort (gilt gerade ein Auslandstarif?), das verfügbare sowie verbrauchte Freivolumen sowie eine gegebenenfalls noch zu entrichtende Tagesnutzungsgebühr eine Rolle spielen. Diese Informationen alle durch einen Handoverentscheider auswerten zu lassen wäre theoretisch möglich, erfordert jedoch komplexe Mechanismen beispielsweise zur Erfassung der übertragenen Datenmenge oder zur Verwaltung eines Gebührenkontingents bei so genannten „Prepaid“ Tarifen. Es zeichnet sich ab, dass ein Nutzer im Vorfeld eine Vielzahl an Parametern hinterlegen müsste, damit der Handoverentscheider seine vorliegenden Netzzugangstechnologien bezüglich der Geldkosten bewerten kann. Aus diesem Grund sah der Autor von einer solchen Umsetzung ab. Einem „normalen“ Nutzer kann jedoch noch zugemutet werden, selbst anzugeben, ob ein Netzzugang als „günstig“ oder als „teuer“ einzustufen ist. Diese Angabe ist zwar ungenau, dafür aber vergleichsweise einfach ermittel- und auswertbar.

Dienstgüte: Netzzugangstechnologien können unterschieden werden, indem man die durch sie angebotene Dienstgüte betrachtet. Bei diesem Begriff muss allerdings aufgepasst werden, da der Fokus dieser Arbeit auf „dem Internetzugang“ liegt, und „das Internet“ selbst lediglich „Best Effort“ anbietet. Es können also keine Reservierungen durchgeführt und damit auch keine Garantien ausgesprochen werden.

Nichtsdestotrotz können Unterschiede zwischen den einzelnen Technologien festgestellt werden. Drahtlose Netzzugangstechnologien weisen meist eine erhöhte Anfälligkeit

gegenüber Störeinflüssen auf als drahtgebundene Medien, sodass hier mit höheren Verzögerungszeiten sowie höheren Paketverlustraten zu rechnen ist. Auch die physikalisch verfügbare Bitrate kann, wie es bei WLAN der Fall ist, abhängig von den vorherrschenden Ausbreitungsbedingungen sein.

Bei GPRS ist es möglich, beim Aufbau eines Verbindungskontexts Dienstgüteparameter zu spezifizieren, welche allerdings mit zusätzlichen Gebühren verbunden sein können. Zur Auswahl stehen hier verschiedene „Klassen“, welche die Paketverzögerung und die Verlässlichkeit betreffen. Von einer „echten“ Garantie kann hier jedoch keine Rede sein, da lediglich statistische Werte genannt werden [Walk01].

Für die Fragestellung, ob überhaupt eine bestimmte Dienstgüte erforderlich ist, können sowohl der Nutzer des mobilen Endgeräts befragt als auch die Art der laufenden Anwendungen mit einbezogen werden. Für „Streaming“ oder Sprachkommunikation gelten engere Vorgaben, als beispielsweise beim Abruf von Webseiten. Allerdings ist es nicht möglich, innerhalb einer universellen „Middleware“ wie REACH zu entscheiden, welche Anwendungen ein Anwender starten möchte, um daraus dann Anforderungen an das Medium abzuleiten. Für REACH könnte man allerdings die geladenen „Relay Plugins“ mit einbeziehen. So wäre es möglich, auf Medien mit einer bestimmten Mindestbitrate und Ausfallsicherheit zu bevorzugen, wenn das „Relay Plugin“ für Telefonie (VoIP) (siehe Abschnitt 4.3.2.6) aktiviert worden ist.

Priorität: Für den Fall, dass eine manuelle Sortierung vorgenommen werden soll, kann die Reihenfolge anhand einer Priorität vorgegeben werden. Diese Priorität wird durch eine nutzerseitig wählbare Zahl festgelegt. Zudem können Netzzugänge auch manuell vom Nutzer abgewählt werden, sodass sie garnicht mehr vom Handoverentscheider berücksichtigt werden.

Sicherheit: Netzzugänge können auch gemäß ihrer „Sicherheit“ sortiert werden, wobei dieser Begriff sehr weit gefasst werden kann. Drahtgebundene Netze können eine bessere Abhörsicherheit gewährleisten als drahtlose Netze. Verschlüsselte Netzzugänge sind sicherer als unverschlüsselte Netze. Öffentliche „Hot Spots“ gelten als weniger vertrauenswürdig als der heimische DSL-Router. Wer Wert auf Sicherheit legt, muss zuerst definieren, was „Sicherheit“ für ihn bedeutet. Erst dann kann eine Sortierung gemäß eines genauer definierten Kriteriums erfolgen.

Zuverlässigkeit: Die Zuverlässigkeit eines Netzzugangs kann ebenfalls eine wichtige Rolle spielen. Es können hierbei ähnliche Eigenschaften wie aus der Kategorie „Dienstgüte“ betrachtet werden. Mit Zuverlässigkeit ist hier allerdings nicht die gesicherte Über-

tragung von Daten gemeint, sondern lediglich qualitative Aussagen über die Technologie. So könnte eine Netzzugangstechnologie, bei welcher spontane Linkabrisse auftreten können, als unzuverlässig gelten.

Andere Beschreibungsmöglichkeiten können die Netzabdeckung mit involvieren. So kann spezifiziert werden, ob im Fall von Eigenbewegung eine hohe Handoverwahrscheinlichkeit besteht (LANs, kleine Zellen) oder nicht (große Zellen, „transparenter Handover“).

Energieverbrauch: Mobile Endgeräte werden häufig ohne Stromkabel betrieben, und sind dafür mit Akkumulatoren ausgestattet. Um eine möglichst lange Standzeit zu erreichen, muss der „Energieverbrauch“ des mobilen Endgeräts minimiert werden. Daher wäre es kontraproduktiv, „leistungshungrige“ Netzzugangsgeräte wie Transceiver zu aktivieren, wenn beispielsweise eine drahtgebundene Kommunikationsstrecke zur Verfügung steht, welche einen geringeren „Energieverbrauch“ aufweist. Auch Szenarien wie Kanalbündelungen und „weichere Handover“ sind weniger gut geeignet, wenn der Energiehaushalt ein entscheidende Rolle spielt.

Allerdings ist es schwierig, genaue Daten bezüglich des Energieverbrauchs einer Netzzugangstechnologie zu erhalten. Manche Technologien verbrauchen bereits im Bereitschaftsmodus („eingebucht“) viel Energie, andere dagegen erst bei der Datenübertragung. Hier bietet es sich an, mit wenigen grob abgestuften qualitativen Stufen zu arbeiten.

In diesem Zusammenhang wäre es lohnenswert, wenn dem Handoverentscheider die Informationen vorliegen würde, ob das mobile Endgerät gerade an die Netzstromversorgung angeschlossen ist. Wird dies erkannt, bräuchte der „Energieverbrauch“ zeitweise keine Rolle spielen.

Interferenzen: Werden mehr als eine einzige Netzzugangstechnologie gleichzeitig aktiviert, kann es bei drahtlosen Systemen zu Interferenzen kommen. WLAN nach IEEE 802.11g und Bluetooth funken beispielsweise im selben Frequenzband. Bei WLAN kommt noch die Problematik einer möglicherweise ungünstigen Kanalbelegungen hinzu. Die Möglichkeit drohender Interferenzen ist sehr schwierig in eine Handoverentscheidung mit einzubeziehen, da eine negative Beeinflussung oft nicht direkt erkennbar ist.

Signalstärke- und Linkgütewerte: Drahtlos arbeitende Netzzugangsgeräte bieten häufig die Möglichkeit an, eine Aussage über den Zustand der Luftschnittstelle abzurufen. Bei WLAN erhält man beispielsweise durch eine Suchoperation eine Liste aller sichtbaren Basisstationen in Reichweite, jeweils mit einer Angabe bezüglich der aktuell anliegenden Signalstärke und der Linkgüte. Somit könnte gezielt verhindert werden, dass Basissta-

tionen ausgewählt werden, deren Signal mit einer geringen Signalstärke bzw. Linkgüte empfangen wird.

Providerseitige Vorgaben: Weitere Einschränkungen können durch den Betreiber eines Netzzugangspunkts erfolgen, und zwar völlig unabhängig von der eingesetzten Netzzugangstechnologie. Eine Rolle kann spielen, ob sich das mobile Endgerät am Netzzugangspunkt hinter einer NAT-Grenze wiederfindet, oder ob es stattdessen eine global gültige IP-Adresse erhält. NAT-Grenzen können Probleme verursachen, was bereits in Abschnitt 4.3.2.6 für VoIP beleuchtet wurde und später in Abschnitt 7.3.2 bezüglich REACH diskutiert wird.

In dieselbe Kategorie fallen vertragliche Einschränkungen, wie das Verbot, Sprachkommunikation oder „Instant Messaging“ durchzuführen. So verbieten heutzutage viele Mobilfunkbetreiber ihren Kunden, ihre „Datenflatrate“ für VoIP zu benutzen.

6.1.1.2 Mögliche Zielstellungen

Der Handoverentscheider steht demnach vor einem Bewertung- und Sortierproblem. Ihm stehen eine beliebige Anzahl verfügbare Netzzugänge zur Verfügung, welche sich anhand der soeben vorgestellten Eigenschaften unterscheiden. Das Ziel ist es, eine Liste mit nach ihrer Eignung sortierten Netzzugängen zu erhalten.

Diese Problemstellung kann nicht ohne die Angabe einer Zielstellung gelöst werden, da die wünschenswerten Eigenschaften eines Netzzugangs gegenläufiger Natur sein können. Im Folgenden wird eine Liste an möglichen Zielstellungen vorgestellt, welche jedoch nicht vollständig ist, sondern die vom Demonstrator angebotene Auswahl reflektiert.

Maximale Bitrate: Bei dieser Zielstellung werden die momentan verfügbaren Netzzugänge anhand der angebotenen Bitrate sortiert. Diese Eigenschaft kann sich mit der Zeit ändern, wenn beispielsweise bei einer drahtlos arbeitenden Technologie auf Grund einer sich verschlechternden Linkgüte eine geringere Datenrate ausgehandelt wird.

Minimierung der Kosten: Diese Zielstellung dient dazu, dass anfallende Entgelte minimiert werden. Der Nutzer nimmt dabei in Kauf, dass bei dieser Entscheidung Abstriche bezüglich der Datenrate oder anderer Eigenschaften hingenommen werden müssen.

Minimierung der Übertragungsverzögerung: Mit dieser Zielstellung wird ein besonderes Augenmerk auf eine geringe Übertragungsverzögerung gelegt. Netzzugänge mit hohen Latenzzeiten werden dabei wenn möglich gemieden, also nur dann benutzt, wenn keine besseren Alternativen verfügbar sind.

Minimierung des Energieverbrauchs: Mobile Endgeräte müssen unterwegs mit Hilfe eines Energiespeichers versorgt werden. Hierbei sind Netzzugangstechnologien, welche einen geringen Energiebedarf aufweisen und damit die Laufzeit des mobilen Endgeräts verlängeren können, bevorzugt auszuwählen.

Optimierung auf einen hohen Grad an Mobilität: Ist im Vorfeld bekannt, dass sich das mobile Endgerät mit hoher Geschwindigkeit bewegen wird, können mit Hilfe dieser Zielstellung Netzzugangstechnologien bevorzugt werden, welche vergleichsweise wenig von Verbindungsabrissen durch Handoverereignisse beeinflusst werden.

Hohe Sicherheitsanforderungen: Netzzugänge können in Bezug auf ihre angebotene Sicherheit hin sortiert werden, wobei mit „Sicherheit“ hier beispielsweise die Vertrauenswürdigkeit der Umgebung gemeint sein kann. So genießt beispielsweise das Wohnungsnetzwerk des Benutzers einen „guten Ruf“, gilt also als „sicher“. Ebenso könnte das Firmennetz des Arbeitgebers „mit einer hohen Sicherheit“ bewertet werden. Eine „offene“ Basisstation in einem Internetcafe beispielsweise könnte aber als „unsicher“ eingestuft werden.

An dieser Stelle muss aber festgestellt werden, dass ein als „unsicher“ geltender Netzzugangspunkt durch Anwendung von Verschlüsselung „sicherer“ gemacht werden kann. Kommen bei REACH beispielsweise das VPN-basierte „Relay Plugin“ (siehe Abschnitt 4.3.2.5) zum Einsatz, wäre die „Vertrauenswürdigkeit“ als Auswahlkriterium nicht länger aussagekräftig. Allerdings ist REACH nicht in der Lage, solche Zusammenhänge zu erfassen.

Nutzerseitige Priorität: Mit Hilfe einer nutzerseitig pro Netzzugang definierbaren Priorität steht dem Nutzer ein Werkzeug zur Verfügung, um die Reihenfolge der sortierten Netzzugänge selbst festzulegen. Dieser Mechanismus eignet sich auch für Testzwecke, wenn der Nutzer den Handoverentscheider dazu zwingen möchte, einen bestimmten Netzzugang auszuwählen.

6.1.1.3 Sortierung von Netzzugängen

Für die Sortierung der verfügbaren Netzzugänge hat eine nutzerseitig festgelegte Zielstellung vorzuliegen. Es ist allerdings nicht empfehlenswert, die Bewertung eines Netzzugangs anhand nur einer einzigen Zielstellung vorzunehmen. Wird beispielsweise eine maximale Bitrate gewünscht, sollte nicht bloß die Bitrate eine Rolle spielen, sondern zu einem kleinen Teil auch die anderen Eigenschaften. Sonst könnte es passieren, dass ein

extrem teurer Netzzugang ausgewählt wird (sofern ein solcher zur Verfügung steht), obwohl der zweitschnellste Netzzugang nur unwesentlich langsamer, dafür aber wesentlich kostengünstiger wäre.

Es ergibt durchaus Sinn, möglichst *alle* Eigenschaften in die Beurteilung eines jeden Netzzugangs mit einzubeziehen. Dabei würde je nach Szenario nicht jede Eigenschaft gleichermaßen relevant sein, sondern es müsste abgewogen werden, mit welchem Gewicht ein jeder Parameter in die Entscheidungsfindung einspielen soll. In [NaHH06] wird hierfür die so genannte „Vertical Handoff Decision Function“ (VHDF) eingeführt, hier dargestellt durch Formel 6.1.

$$\begin{aligned} Q_i = \omega_\mathrm{c} \frac{{C_i}^{-1}}{\max\left\{{C_1}^{-1}, \ldots, {C_\mathrm{n}}^{-1}\right\}} + \omega_\mathrm{s} \frac{S_i}{\max\{S_1, \ldots, S_\mathrm{n}\}} + \\ \omega_\mathrm{p} \frac{{P_i}^{-1}}{\max\left\{{P_1}^{-1}, \ldots, {P_\mathrm{n}}^{-1}\right\}} + \omega_\mathrm{d} \frac{D_i}{\max\{D_1, \ldots, D_\mathrm{n}\}} + \\ \omega_\mathrm{f} \frac{F_i}{\max\{F_1, \ldots, F_\mathrm{n}\}} \end{aligned} \tag{6.1}$$

Dabei werden die folgenden Eigenschaften der Netzzugänge zusammen mit ihren jeweiligen Gewichten mit in die Entscheidungsfindung einbezogen:

- Geldkosten („costs“) C_i, Gewicht ω_c
- Sicherheit („security“) S_i, Gewicht ω_s
- Energieverbrauch („power“) P_i, Gewicht ω_p
- Zustand des Netzes D_i, Gewicht ω_d
- Linkgüte („performance“) F_i, Gewicht ω_f

Die Gewichte müssen so gewählt werden, dass ihre Summe 1 ist (siehe Formel 6.2).

$$\omega_\mathrm{c} + \omega_\mathrm{s} + \omega_\mathrm{p} + \omega_\mathrm{d} + \omega_\mathrm{f} = 1 \tag{6.2}$$

Es ist erkennbar, dass manche Eigenschaften *direkt* verrechnet, während andere Eigenschaften *über ihren Kehrwert* berücksichtigt werden. Bei den Geldkosten C_i und dem Energieverbrauch P_i ist man an möglichst kleinen Werten interessiert, während für die Sicherheit S_i, den Zustand des Netzes D_i sowie die Linkgüte F_i jeweils große Werte als vorteilhaft gelten. Der Netzzugang i mit der höchsten Eignung Q_i soll als am besten geeignet gelten. Dazu müssen C_i und P_i reziprok verrechnet werden. Jeder Term wird für

sich gesehen auf den größten verfügbaren Wert normiert und schlussendlich mit seinem Gewicht multipliziert. Für das Gesamtergebnis sind Werte zwischen 0 und 1 möglich, wobei größere Werte für eine bessere Eignung stehen.

Im nächsten Schritt gilt es, vernünftige Werte für die Gewichte zu finden. Für REACH musste die VHDF angepasst werden: zum einen können noch weitere Parameter berücksichtigt werden, beispielsweise in Bezug auf die Latenzzeit und die nutzerseitig festlegbare Priorität. Zum anderen wurde die Bestimmung der Gewichte bislang ausgeklammert, sodass immer nur genau *ein* Gewicht auf 1 gesetzt wird, während alle anderen Gewichte bei 0 bleiben.

In der Praxis gestaltet es sich als schwierig, zu einem jeden Netzzugang Aussagen bezüglich der zu erwarteten Verzögerungszeit, der Sicherheit oder des Energieverbrauchs zu erhalten. Die VHDF wurde zwar implementiert, jedoch findet innerhalb von REACH noch keine Verwaltung dieser Parameter statt. Als besonders problematisch erwies es sich, dass bezüglich mancher der genannten Parameter nur eine grobe Einstufung praktikabel erscheint (beispielsweise beim Energieverbrauch). In der studentischen Arbeit [Viet08] wurde diese Problemstellung aufgegriffen, und es spricht nichts dagegen, die dort vorgestellten Abstufungen der untersuchten Parameter im Rahmen einer zukünftigen Weiterentwicklung auch in die VHDF von REACH zu integrieren.

Die einzige Eigenschaft eines Netzzugangs, welche momentan durchgängig berücksichtigt wird, ist die nutzerseitig festlegbare Priorität. Für die verfügbaren Netzzugangstechnologien Ethernet, WLAN und GPRS ist somit die Festlegung einer praxisrelevanten Reihenfolge möglich, da es hierbei wenigstens bezüglich der Kriterien Geschwindigkeit, Geldkosten und Latenzzeit zu keinen Widersprüchen kommt: Ethernet bietet im Demonstrationsaufbau den schnellsten Netzzugang mit der geringsten Latenzzeit ohne anfallende Verbindunsgentgelte an. Bei REACH beziehen sich diese Parameter jedoch immer auf eine Kombination aus Netzzugangsgerät und Netzzugangspunkt (ergibt „den Netzzugang“), sodass statt „Ethernet ist günstig“ bei REACH Aussagen wie „Ethernet ist im Büro günstig, aber nicht im Hotel“ gelten.

Es erscheint lohnenswert, zukünftig für alle von REACH unterstützen Netzzugangstechnologien diese Parameter entweder automatisch ermitteln zu lassen, oder diese vorab zu bestimmen und bei Bedarf aus einer Wissensbasis zu entnehmen. Stehen damit brauchbare Werte zur Verfügung, sollte die im Handoverentscheider hinterlegte VHDF um den jeweiligen Parameter erweitert werden. Die Struktur der VHDF entspräche weiterhin Formel 6.1, jedoch mit einer beliebig wachsenden Anzahl an Summanden.

6.1.2 Klassifizierung von Handovern

Der Wechsel eines Netzzugangspunkts (Handover) kann nach verschiedenen Gesichtspunkten klassifiziert werden, welche in diesem Abschnitt vorgestellt werden sollen.

6.1.2.1 „Horizontale“ und „vertikale Handover“

Es lässt sich unterscheiden, ob ein Zugangswechsel innerhalb derselben Technologie geschieht, oder ob dabei einen Technologiewechsel stattfindet. Bewegt man sich innerhalb derselben Technologie, spricht man von einem „horizontalen Handover“. Wird dabei jedoch die Technologie gewechselt, bezeichnet man dies als „vertikalen Handover“.

Beispielhafte Technologien, welche mit „horizontalen Handoveren“ arbeiten, sind die Mobilkommunikationsnetze GSM und UMTS sowie die drahtlose Netzzugangstechnologie gemäß IEEE 802.11 (WLAN) und dessen Weiterentwicklungen. Bei GSM müssen „horizontale Handover“ durchgeführt werden, da die Anwender mit ihren Telefonen „mobil“ sind dürfen und sich die Telefone mit einer Basisstation eines zellulären Netzes verbinden müssen. Der Benutzer bekommt von einem „Zellwechsel“ allerdings nichts mit; er findet *transparent* im Hintergrund statt.

Auch bei WLAN können „horizontale Handover“ durchgeführt werden. Ein mobiles Endgerät kann beispielsweise mit einer privaten WLAN-Basisstation assoziiert sein, aber diese auf Grund von Mobilität nicht länger erreichen. Alternativ kann sich das mobile Endgerät dann in ein anderes gerade „sichtbares“ WLAN-Netz einbuchen, beispielsweise einen kostenpflichtigen „Hot Spot“ oder in das WLAN-Netz des Arbeitgebers des Nutzers. Solche Wechsel zählen ebenfalls zu den „horizontalen Handovern“, allerdings sind diese nicht „transparent“. Das mobile Endgerät erfährt hierbei einen Verbindungsabriss, und es muss sich, falls verfügbar, in ein Ersatznetz einbuchen. Beim Zugriff auf das Internet kommt es hierbei zu einem Abriss aller statusbehafteten Transportschichtverbindungen, wenn hierfür keine weiteren Vorkehrungen getroffen werden. Ein Lösungsvorschlag, welche bezüglich WLAN noch auf der Netzzugangsschicht angewendet werden kann, besteht im Angebot eines „Roaming“-fähigen Verbundes aus mehreren Basisstationen. Hierbei können „transparente horizontale Handover“ durchgeführt werden, wobei das mobile Endgerät nicht merkbar vom Netzwerk getrennt wird. Es würde seine zugewiesene IP-Adresse behalten können, und es käme zu keinen unmittelbaren Auswirkungen auf die Transportschichtprotokolle. Wird jedoch der „roaming“-fähige Verbund verlassen, käme es wieder zu dem geschilderten Fehlverhalten.

Von einem „vertikalen Handover“ spricht man dagegen, wenn ein mobiles Endgerät beim Wechsel des Netzzugangs auf eine andere Technologie zurückgreift. Dadurch, dass moderne Mobiltelefone heutzutage bereits mit Unterstützung für UMTS, GSM und

WLAN ausgestattet sein können, kann je nach Aufenthaltsort und verfügbarem Tarif jeweils die eine oder die andere Technologie ausgewählt werden. Seitens des Anwenders *kann* der Handover wiederum „transparent“ erfolgen, wenn wie im Fall eines aufgebauten Sprachkanals ein Wechsel von UMTS auf GSM stattfindet. Zur Anbindung an das Internet können jedoch auch „vertikale Handover“ von WLAN auf UMTS durchgeführt werden, welche dann allerdings auf Grund der sich ändernden IP-Adressen nicht ohne negative Auswirkungen bleiben. Auch hierfür gibt es bereits umfangreiche Lösungsvorschläge, welche sich dieser Problematik angenommen haben. An dieser Stelle geht es jedoch lediglich um die Klassifizierung.

6.1.2.2 „Freiwillige“ und „erzwungene Handover“

Weiterhin kann unterscheiden werden, ob ein Handover „freiwillig“ ist oder „erzwungen“ erfolgt. Beim „freiwilligen Handover“ besitzt das mobile Endgerät eine Verbindung zum Netz, ohne dass diese Verbindung von einem unmittelbaren Abriss bedroht ist. Es gibt in dieser Situation jetzt aber einen weiteren Netzzugangspunkt, welcher über „bessere Eigenschaften“ verfügt, sodass ein Wechsel vorteilhaft wäre. Bei einem „vertikalen Handover“ kann das mobile Endgerät eine parallele Assoziationen zum neuen Netzzungangspunkt aufbauen, ohne sich für die Dauer des Anmeldevorgangs vom alten Netzzugangspunkt „lösen“ zu müssen. Erst, wenn der Ersatzzugang hergestellt werden konnte, wird sich das mobile Endgerät vom alten Netzzugangspunkt trennen. Der Wechsel wäre dann „unterbrechungsfrei“.

Alternativ kann ein Handover auch „erzwungen“ werden, wenn es beispielsweise zu einem spontanen Abriss des gerade aktiven Netzzugangs kommt. Das mobile Endgerät ist dann so lange vom Netzwerk isoliert, wie es Zeit für die Assoziation mit einem neuen Netzzugangspunkt benötigt. „Erzwungene Handover“ sind damit meist mit einer Degradierung der Qualität verbunden, da das mobile Endgerät einerseits kurzfristig isoliert ist und andererseits auf einen weniger gut geeigneten Ersatzzugang umgeschwenkt werden muss.

„Freiwillige Handover“ sind jedoch nur dann *unterbrechungsfrei*, wenn die Möglichkeit besteht, die Ersatzverbindung aufzubauen ohne die bestehende Hauptverbindung trennen zu müssen. Dies ist bei „vertikalen Handovern“ eigentlich immer gegeben, da unterschiedliche Netzzugangstechnologien involviert werden. Will das mobile Endgerät jedoch innerhalb derselben Technologie auf einen anderen Anbieter wechseln, so muss für einen solchen freiwilligen Wechsel erst die bestehende Assoziation abgebaut werden bevor es sich zu anderen Netzzugangspunkt „verbinden“ kann. Diese Einschränkung ist jedoch im hohen Maße technologieabhängig. Ein einleuchtendes Beispiel zur Verdeutli-

chung der Problematik bietet die Einwahl über ein Analogmodem, wobei beispielsweise die Rufnummern zweier Provider gewählt werden können. Soll die Auswahl nach der Uhrzeit geschehen, weil der zweite Provider ab 22 Uhr Abends günstiger ist als der erste Provider, muss für einen „freiwilligen Handover“ erst die Modemverbindung getrennt und eine zeitintensive Wiedereinwahlprozedur durchlaufen werden. Für die Dauer der Einwahl wäre das Endgerät von Netz isoliert, obwohl ein „freiwilliger Handover“ durchgeführt wird.

6.1.2.3 „Reaktive“ und „prädiktive Handover“

Beim einem „reaktiv“ ablaufenden Handover bemerkt der Handoverentscheider für ihn unvorhersehbar, dass ein aktuell benutzter Netzzugang nicht länger geeignet ist, und somit ein Ersatz benötigt wird. Im Unterschied zum „erzwungenen Handover“ ist diese Situation jedoch nicht mit einem „harten“ Abriss gleichzusetzen, sondern kann auch auf der Auswertung der Linkgüte beruhen: ist die aktuelle Linkgüte bereits bedrohlich niedrig, kann ein Wechsel durchführt werden, obwohl der bestehende Netzzugang eigentlich noch nicht abgerissen ist. Es kann sich dabei also auch um eine Form des „freiwilligen Handovers“ handeln.

Dabei kann es auch bei einem „vertikalen Handover“ zu einer Isolationssituation kommen. Der Aufbau einer Datenverbindung mit Hilfe von GPRS kann bei bestimmten GSM-Modems bis zu 22 Sekunden benötigen [Ever04]. In dieser langen Zeitspanne ist die Wahrscheinlichkeit hoch, dass sich die Signalstärke des aktiven Hauptlinks noch weiter verschlechtert, sodass es schlussendlich zu einem Abriss kommt noch bevor der Ersatzlink aufgebaut werden konnte. Es handelt sich dabei um ein *reaktives* Verhalten, wobei die Entscheidung zum Wechsel erst dann getroffen wird, wenn ein Abriss unmittelbar droht.

Dabei muss nicht bis zum drohenden Abrissereignis gewartet werden, da beispielsweise bei drahtlosen Netzzugangstechnologien im Vorfeld häufig eine Verschlechterung der Signalstärke beobachtet werden kann. Das liegt daran, dass die Geräte durch die Gegend getragen werden und sich die Entfernung zur Basisstation soweit vergrößern kann, bis es schließlich zu einem Abrissereignis kommt. Für einen „prädiktiv“ durchgeführten Handover würde man sich den Verlauf der Signalstärke zu Nutze machen. Hierbei könnte ein zukünftiger Abriss durch Betrachtung der jüngsten Vergangenheit prognostiziert werden. Steht ein solcher Vorhersagemechanismus zur Verfügung, kann die Einwahldauer eines potentiellen Ersatzzugangs mit der errechneten Restzeit bis zum prognostizierten Abrisszeitpunkt verglichen werden, und eine zeitaufwendige Einwahl bereits dann gestartet werden, wenn der Hauptlink noch sehr gute Übertragungseigenschaften aufweist. Konnte

die Einwahlprozedur erfolgreich durchlaufen werden, und behält der Vorhersagemechanismus Recht, steht zum Abrisszeitpunkt bereits der prädiktiv (proaktiv) aufgebaute Ersatzlink zur Verfügung.

Eine solche Prognose ist immer mit einer Unsicherheit verbunden, und man darf nicht erwarten, dass perfekte Vorhersagen geliefert werden. Die Signalstärke kann sich plötzlich stark verschlechtern, sodass ein Abriss unerwartet früh erfolgt. Oder die Signalstärkewerte verbessern sich plötzlich wieder, sodass kein Bedarf mehr an einem Ersatzlink besteht. Aber es lässt sich erkennen, dass in beiden Fällen ein besseres Ergebnis erzielt wird, als es im Fall eines reaktiv arbeitenden Handoverentscheiders möglich wäre. Für diesen würde man nämlich entweder *immer* eine Verzögerung im Umfang der Einwahldauer des Ersatzlinks durchlaufen müssen, oder man müsste mit Blick auf Unterbrechnungsfreiheit den Ersatzlink bereits vom ersten Moment an assoziieren („gieriges“ Verhalten).

Der Nutzer muss sich also entscheiden, ob er Wert auf eine möglichst unterbrechungsfreie Anbindung legt; dann hilft nur die dauerhafte Bereitstellung eines Ersatzlinks mit den Nachteilen steigender Kosten oder beispielsweise eines höheren Energieverbrauchs. Alternativ könnte dem Nutzer die Unterbrechnungsfreiheit auch gleichgültig sein, dann wäre ein reaktiv arbeitendes Schema sinnvoller. Ein „prädiktiver Handover“ wäre „in der Mitte“ angesiedelt, da hierbei versucht wird, beide Anforderungen auf einmal zu erfüllen.

6.1.2.4 „Harte“, „weiche“ und „weichere Handover“

Zudem kann bei einem Zugangswechsel unterschieden werden, ob ein Ersatzlink erst bei unmittelberem Bedarf, oder bereits im Vorfeld aufgebaut wird. Bei erstem Fall spricht man von einem „harten Handover“. „Harte Handover“ sind mit Unterbrechnungen des Datenstroms verbunden, da nach Abriss des aktiven Hauptlinks anfangs noch kein Ersatzlink zur Verfügung steht. Bei GSM, welches nur „harte Handover“ unterstützt, dauert eine Übergabe etwa 1/20s [lapt]. Anzumerken ist, dass die Eigenschaft „hart“ und „reaktiv“ (letzter Abschnitt) nicht dasselbe bezeichnen, denn auch ein „prädiktiver Handover“ kann „hart“ verlaufen, falls hierbei kein Ersatzlink zur Verfügung steht, welcher die Zeitspanne zwischen freiwilliger Trennung und abgeschlossener Assoziationsprozedur zu überbrücken hilft.

Beim Wechsel eines Netzzugangspunkts kann der erwünschte Ersatzlink bereits aufgebaut werden, noch während der aktive Hauptlink verwendet wird. Zum Umschaltzeitpunkt stünde dann bereits ein aufgebauter Netzzugang bereit, und es müsste lediglich der Datenstrom von dem einen aktiven Netzzugang auf den andere aktiven Netzzugang umgeschwenkt werden. Ein solcher „weicher Handover“ verursacht geringere Ausfallzeiten als ein „harter Handover“. Auf Grund der dann aber noch auszutauschenden Signalisie-

rungsinformationen bezüglich des Kommunikationskontexts lassen sich die Ausfallzeiten jedoch nicht komplett vermeiden.

Beim „weicheren Handover“ geht man noch einen Schritt weiter, und bezieht den aufgebauten Ersatzlink schon frühzeitig mit in die Übertragung ein. Die über den Hauptlink übertragenden Daten werden dabei zusätzlich noch über Ersatzlinks versendet, es findet also eine *redundante* Datenversendung statt. Reißt dann der vorherige Hauptlink plötzlich ab, kommt es zu keinen Lücken im Datenstrom, da die verloren gegangenen Daten ebenfalls über die Ersatzlinks versendet worden sind. Einer der Ersatzlink wird anschließend zum neuen Hauptlink ernannt, womit der Handover komplett durchlaufen wäre.

An dieser Stelle sei noch erwähnt, dass man den kurzzeitigen Zustand während eines „weicheren Handovers“ auch als einen eigenständigen Betriebsmodus auswählen kann. Durch das Angebot eines permanent verfügbaren Ersatzlinks, über welchen immer dieselben Daten versendet werden sollen wie über den Hauptlink, kann eine Resistenz gegenüber spontanen Abrissereignissen ermöglicht werden. Im hiesigen Fall muss der Handoverentscheider dazu mindestens zwei Internetverbindungen aufbauen, eine davon zur Übertragung „neuer“ Daten heranziehen und alle weiteren Ersatzverbindungen zur Übertragung „redundanter“ Daten benutzen.

Statt einer redundaten Versendung können alternativ auch „neue“ Daten über Ersatzlinks versendet werden. Der „frische“ Strom an zu versendenden Daten würde sich dabei auf mehrere aktive Netzzugänge „auffächern“, sodass eine höhere Gesamtübertragungsrate erreicht werden kann. Ein solches Szenario wird als „Kanalbündelung“ bezeichnet. Es ist aber anzumerken, dass hierbei kein Schutz mehr gegen spontane Verbindungsabrisse besteht. Wenn einer der Netzzugänge abreißt, gehen Daten verloren, und es kommt zu Verzögerungen durch Datenverluste.

6.2 Die Handoverentscheidung

Im diesem Abschnitt wird beleuchtet, welche Komponenten einer Mobilitätsunterstützung für die Handoverentscheidung verantwortlich sein können, wo das Entscheidungsergebnis überall benötigt wird und welche Formen von Handoverentscheidern unterschieden werden können.

6.2.1 Verantwortliche Komponenten

Ein heutiges mobiles Endgerät mit einem marktüblichen Betriebssystem ist nicht handoverfähig, was bereits in Kapitel 2 diskutiert worden ist. Abgesehen davon, dass dies in den heutzutage im Internet eingesetzten Protokollen begründet liegt, fehlt selbst bei modernen Betriebssystem häufig eine zentrale Steuerungsinstanz, welche die „Hoheit“ über die verfügbaren Netzzugangsgeräten übernimmt.

6.2.1.1 Bereits verfügbare „Network Manager“

Unter GNU/Linux war die Verwaltung von Netzzugangsgeräten lange Zeit umständlich. Man war gezwungen, ein Netzzugangsgerät selbst auszuwählen und dieses dann mit Hilfe von Konfigurationsdateien und Befehlen auf der Kommandozeile manuell zu aktivieren. Eine grafische Oberfläche hierfür, beispielsweise eine Integration in die Desktop-Umgebungen KDE oder GNOME, gab es auf Grund der Vielfalt an Betriebssystemen auf denen KDE und GNOME laufen sollten nicht. Die zu Grunde liegenden Konzepte von GNU/Linux, FreeBSD, Unix und neuerdings auch Microsoft Windows waren zu unterschiedlich, als dass beispielsweise ein für KDE implementierter „Network Manager“ alle diese unterschiedlichen Mechanismen beherrschen könnte. So gab es maximal Nischenlösungen wie das grafische Programm KPPP [Wiki09e] („KDE Point-to-Point Protocol“), welches jedoch ausschließlich Einwahlverbindungen kontrollieren konnte.

Erst KDE ab der Version 4, welche seit Januar 2008 verfügbar ist, baut auf den Systemkomponenten „Hardware Abstraction Layer“ (HAL, [Hal0]) und „Desktop BUS“ (DBUS, [DBus]) auf. Der Vorteil besteht darin, dass von nun an KDE-Anwendungen mittels DBUS auf HAL zugreifen können, wobei HAL die Anpassung an das lokal vorliegende Betriebssystem übernimmt. Weitergehend ist HAL in der Lage, angeschlossene Geräte zu erkennen und per DBUS-Meldungen an die Anwendungen zu schicken. Dieses war der notwendige Unterbau, um plattformunabhängige Programme zur Netzwerkkonfiguration auch unter den für GNU/Linux-relevanten Desktop-Umgebungen KDE und GNOME anbieten zu können. Für alle relevanten Zielplattformen ist lediglich eine angepasste Version von HAL notwendig.

Unter GNU/Linux verfügbare Verwaltungsprogramme sind der NetworkManager [NMan] und der Wireless Interface Connection Daemon (Wicd, [Wicd]). Diese Programme sind mittlerweile in der Lage, Ethernet, WLAN, Analogmodems, ISDN und auch Mobiltelefone zu verwalten. Allerdings sind sie mit alleinigem Blick auf eine Schnittstelle zur nutzerseitigen Auswahl entworfen worden, ohne die „Intelligenz“ eines automatisch arbeitenden Handoverentscheiders mitzubringen. Das ist insofern verständlich, da in der heutigen Zeit Mobilitätsunterstützungen wie Mobile IP praktisch keine Relevanz für den Nutzer eines mobilen Endgeräts haben. Folglich sind auch die für die „breite Masse“ entwickelten Verwaltungsprogramme nicht in der Lage, mit so etwas umzugehen. Eine Unterstützung für „weichere Handover“ lässt sich beispielsweise bei keinem der verfügbaren Verwaltungsprogramme erkennen; dafür wäre weitaus mehr Infrastruktur notwendig als ein Handoverentscheider alleine.

Ein Mehrwert, der aus einer näheren Betrachtung dieser Verwaltungsprogramme für REACH gewonnen werden könnte, bestünde in der Nutzung von HAL und DBUS zur Ansteuerung der Netzzugangsgeräte. Im Rahmen dieser Ausarbeitung wurde eine diesbezügliche Betrachtung jedoch nicht durchgeführt, da es sich um sehr junge Projekte handelte und die Anbindung über HAL und DBUS noch nicht ausgereift war. Langfristig erscheint dies allerdings lohnenswert, da die Ansteuerung von Netzzugangsgeräten damit plattformunabhängig realisiert werden könnte. Offen bleibt jedoch die Frage, ob auf diesem plattformübergreifenden Weg noch ausreichend Kontrolle auf die Netzzugangsgeräte ausgeübt werden kann. REACH setzt momentan auf hochgradig plattformspezifische Mechanismen auf, und es erscheint dem Autor als fraglich, dass diesbezügliche Mechanismen zukünftig in einer betriebssystemunabhängigen Abstraktionsschicht wie HAL auftauchen werden.

6.2.1.2 Zentraler Handovermanager

An dieser Stelle stellt sich abermals heraus, dass eine praktikable Mobilitätsunterstützung sich nicht darauf beschränken kann, einzelne Anwendungen durch Modifikation „handoverfähig“ zu machen. Selbst wenn es gelänge, eine ausgewählte Menge an Browsern und Mailprogrammen mit entsprechenden Funktionen auszustatten, bliebe immer noch die Frage offen, welche dieser Komponenten für die Handoverentscheidung und damit die Verwaltung der Netzzugangsgeräte verantwortlich wäre.

Es ist nachvollziehbar, dass diese Aufgabe nicht durch „eine Anwendung“, wie beispielsweise einen „handoverfähig“ gemachten Browser, erfolgen darf. Die Konsequenz wäre, dass dieser Browser dauerhaft laufen müsste. Systemweit dürfte der Browser zudem nur ein einziges Mal gestartet werden, was auf Mehrbenutzersystemen zu Problemen

führen würde. Andere Programme wie Telefonieanwendungen, welche den Verbindungsstatus für eine Wartemelodie in Erfahrung bringen möchten, müssten sich dann mit dem Handoverentscheider des Browsers verbinden. Eine solche Struktur wäre nicht wünschenswert.

Die Handoverentscheidung muss daher von einer externen Komponente übernommen werden, welche losgelöst von den Anwendungen gestartet werden kann. Die in Abschnitt 6.2.1.1 als „Network Manager“ vorgestellten Verwaltungsprogramme kommen diesem Ziel bereits nahe, allerdings handelt es sich hierbei um grafische Oberflächen, welche von einem Nutzer gestartet werden müssen. Sind auf einem Mehrbenutzersystem jedoch mehrere Nutzer gleichzeitig tätig, wäre nicht klar, welcher Nutzer für den Start des Verwaltungsprogramms verantwortlich ist. Interessant wird die Fragestellung vor allem, wenn das mobile Endgerät über keinerlei grafische Oberfläche verfügt. Dann können die vorgestellten Verwaltungsprogramme jedenfalls nicht eingesetzt werden.

Betrachtet man die Anbindung von Kraftfahrzeugen an das Internet, wird dieselbe Problematik ebenfalls deutlich. Soll das Navigationssystem sein Kartenmaterial aus dem Internet herunterladen, benötigt es eine Datenverbindung. Ein streamingfähiges Autoradio würde ebenfalls mit einem eingebauten Modem ausgestattet daherkommen, und wenn dann noch der Beifahrer mit seinem Laptop im Internet surfen möchte, käme ein dritter Netzzugang hinzu. Es liegt auf der Hand, dass ein zentraler Netzzugang, welcher von allen Komponenten gleichermaßen genutzt werden könnte, sinnvoller wäre. Die Frage ist allerdings, welche Komponente diesen „Sternpunkt“ kontrollieren soll. Funktioniert das Navigationssystem nur, wenn ein internetfähiges Autoradio verfügbar ist, oder müsste es umgekehrt sein? Der Ansatz wäre nicht praktikabel.

Erfolg versprechender wäre es, wenn das Fahrzeug selbst als zentrale Komponente den Internetzugang bereit stellt, und sich Navigationssystem, Autoradio und Laptop jeweils nur mit dem Fahrzeugnetz zu verbinden bräuchten. Diese Idee ist in dieser Ausarbeitung unter dem Schlagwort der REACH-Box verfolgt worden (siehe Abschnitt 4.2.1). Allerdings sei an dieser Stelle nur erwähnt, dass sich für eine von allen Anwendungen losgelöste zentrale Verwaltungskomponente entschieden werden musste. Diese kann sich zentral den im mobilen Umfeld zu erwartenden Problemen widmen, und idealerweise auch nicht modifizierten Endgeräten eine transparent arbeitende Mobilitätsunterstützung anbieten.

6.2.1.3 Mitteilung des Entscheidungsergebnisses

Bei REACH wird die Handoverentscheidung alleinig durch das mobiles Endgerät gefällt, denn nur das mobile Endgerät hat eine genaue Übersicht über alle seine verfügbaren Netzzugangsgeräte und die jeweils „sichtbaren“ Netzzugangspunkte. Zudem muss es in der Lage sein, anfangs eine Entscheidung auch ohne Hilfe „aus dem Festnetz“ fällen zu können.

Bei REACH gibt es jedoch noch die ortsfest installierten REACH-Proxyserver. Diese haben zwar keinen Einfluss auf die Handoverentscheidungen der assoziierten mobilen Endgeräte, allerdings müssen sie über die jeweils gefällten Handoverentscheidungen unterrichtet werden. Der Grund ist, dass der Handoverentscheider nicht bloß über den Auf- und Abbau von Kommunikationspfaden entscheidet, sondern auch über deren Nutzungsschema befindet. So unterscheidet REACH zwischen mehreren Betriebsmodi. Damit diese Modi sowohl im Hin- wie auch in Rückrichtung zum Internet angewendet werden können, muss der REACH-Proxyserver wissen, wie die vorliegenden Datenkanäle benutzt werden dürfen.

Diese Informationen werden in einer Nutzungstabelle erfasst, welche in REACH als „Linkmap“ bezeichnet wird und welche das für die REACH-Proxyserver relevante Ergebnis der Handoverentscheidung enthalten. „Linkmaps“ werden, wie in [Ever04] ausführlich dargelegt, in Richtung der REACH-Proxyserver über einen TCP-basierten Steuerkanal übertragen. Eine jede Netzzugangstechnologie kann einen der folgenden Betriebsmodi annehmen:

- **Nicht verbunden:** Seitens des mobilen Endgeräts gilt die entsprechende Netzzugangstechnologie als nicht verbunden.
- **Verbunden, Leerlauf:** Die Netzzugangstechnologie ist verbunden, und es werden Datenverbindungen zu den REACH-Proxyservern angestrebt. Diese werden allerdings nur zum Empfang von „Servicemaps“, welche das Dienstangebot des REACH-Servers beschreiben (siehe Abschnitt 4.4), und zur Versendung von „Linkmaps“ benutzt. Nutzdaten werden nicht übertragen.
- **Verbunden, *neue* Nutzdaten:** Es liegt ein Datenkanal vor, der zusätzlich zur Nutzdatenversendung herangezogen werden soll. Dabei sollen *neue*, also bislang noch nicht über andere Strecken versendete Daten, übertragen werden. Existieren mehrere Pfade mit diesem Betriebsmodus, erhält man ein Kanalbündelungsschema.
- **Verbunden, *redundante* Nutzdaten:** Auch in diesem Betriebsmodus werden

Nutzdaten übertragen, allerdings nur solche, welche bereits über einen anderen Pfad als „neu“ versendet worden sind. Es lassen sich somit redundante Übertragungen erreichen, was für „weichere Handover“ wichtig ist.

„Linkmaps“ werden ausschließlich vom REACH-Client erstellt und vom REACH-Server ausgewertet. „Linkmaps“ erhalten immer Informationen bezüglich aller verfügbaren Netzzugänge. Um eine eindeutige Reihenfolge zu erhalten, sind „Linkmaps“ mit einem Zeitstempel ausgestattet.

6.2.2 Reaktive Entscheider

Ein *reaktiv* arbeitender Entscheider beurteilt die ihm zur Verfügung stehenden Netzzugänge anhand ihres aktuellen Status. Informationen bezüglich der Vergangenheit werden nicht berücksichtigt, sodass weder Signalstärkeverläufe ausgewertet noch wiederkehrende Nutzungsmuster erkannt werden können. Ein Abrissereignis würde damit genau dann sichtbar werden, wenn es stattgefunden hat.

Ein reaktiver Entscheider wird einen momentan verfügbaren Link so lange wie möglich benutzen, da dieser erst im Fall eines Abrissereignisses als nicht mehr benutzbar erkannt wird. Problematisch dabei ist aber, dass erst nach dem Abrissereignis die Notwendigkeit eines Ersatzlinks erkannt wird. Daraus folgt, dass Ersatzlinks „spät“ aufgebaut werden.

Um Netzzugänge untereinander vergleichen zu können, bietet sich die bereits in Abschnitt 6.1.1.3 vorgestellte „Vertical Handoff Decision Function“ (VHDF) an. Mit ihrer Hilfe werden anhand der momentanen Eigenschaften eines jeden Netzzugangs unter Berücksichtigung einer nutzerseitigen Zielstellung „abstrakte Kosten“ errechnet. Diese Kosten machen eine Vergleichbarkeit der Netzzugänge untereinander möglich, was innerhalb von REACH dadurch erreicht wird, dass die Liste der Netzzugänge anhand ihrer „abstrakten Kosten“ sortiert wird. Ändern sich die Eigenschaften der Netzzugänge derart, dass die Elemente der Liste in eine andere Reihenfolge geraten, kann ein bis dato genutzter Netzzugang durch einen anderen Netzzugang ersetzt werden. Je nach Betriebsmodus kann es dabei zu einer Unterbrechung des Datenstroms kommen, wie beispielsweise im Fall des „harten Handovers“.

Möchte man mit einem reaktiv arbeitenden Entscheider die Ausfallzeiten minimieren, bleibt einem keine andere Chance, als mehrere Netzzugänge gleichzeitig zu aktivieren und diese in einem redundanten Modus zu betreiben. Bei einem solchen *proaktiv* arbeitenden Handoverentscheider ist dann allerdings auch die Ressourcenbelegung maximal. Er geht mangels Prognosefähigkeiten „vom Schlimmsten“ aus, und hält den oder die Ersatzlinks dauerhaft vor. Ein solcher Entscheider wird im Folgenden als „gieriger Entscheider“ bezeichnet.

Betrachtung von Signalstärke- und Linkgütewerten

Die im REACH-Client implementierte VHDF berücksichtigt *nicht* die aktuell anliegende Signalstärke oder Linkgüte drahtlos arbeitender Netzzugangstechnologien. Die Betrachtung dieser Werte erfolgt separat, da REACH hierfür einen *prädiktiven* Betriebsmodus anbietet (siehe Abschnitt 6.2.4). Die Betrachtung von Signalstärke- oder Linkgütewerten ist jedoch auch für einen reaktiv arbeitenden Entscheider wichtig, da mit ihrer Hilfe erkannt werden kann, ob ein Netzzugang benutzbar ist.

Eine jede drahtlose Netzzugangstechnologie weist eine maximale Kommunikationsreichweite auf. Wird diese überschritten, fällt die Signalstärke unter einen kritischen Wert und das empfangene Signal kann nicht länger dekodiert werden. Zudem verschlechten hohe Rauschpegel das Signal-Rausch-Verhältnis, sodass die nutzbare Reichweite vermindert wird. Es ist erkennbar, dass der Signalstärkewert für sich alleine betrachtet nicht viel über die Nutzbarkeit einer Funkstrecke aussagt, sondern zudem das Signal-Rausch-Verhältnis mit einbezogen werden müsste. In Bezug auf WLAN steht hierfür ein „Quality"-Wert (Signalqualität) zur Verfügung, welcher das Ergebnis einer Verrechnung zwischen der Signalstärke „Signal level" und weiteren Parametern darstellt [Reyn03, Schm08].

Ein drahtlos arbeitender Netzzugang droht unterhalb eines bestimmten Werts der Signalstärke abzureißen, allerdings ist er bereits vor dem Abrissereignis bei noch höheren Werten nicht mehr sinnvoll nutzbar. Damit macht es keinen Sinn, auf die „harten" Abrissereignisse zu warten, da der Link bereits vorher „unbrauchbar" geworden ist. Es ist daher sinnvoller, eine *Abrissschwelle* zu definieren, deren Unterscheitung auf einen unbrauchbar gewordenen Netzzugang hinweist.

Die Signalstärkewerte fallen bei REACH als zeitstempelbehaftete Messwerte an. Die Einheit dieses Messwerts *kann* unbekannt sein, wenn er beispielsweise als einheitsloser Wert durch den Treiber bereit gestellt wird. Beim ZD1211-WLAN-Treiber des Linux-Kernels ist genau das der Fall, denn hier werden sowohl die Signalstärke als auch die Signalqualität als einheitslose Ganzzahlen im Bereich von 0 bis 255 geliefert. Oftmals wird der Wertebereich auch auf einen prozentualen Wert abgebildet, was jedoch abhängig vom Treiber ist und durch Ausprobieren festgestellt werden sollte [Wild02]. Die WLAN-Treiber für Atheros-Chipsätze, welche in der REACH-Box zum Einsatz kommen, liefern die Signalstärke in „Dezibel bezogen auf 1 Milliwatt" (dBm).

In Abbildung 6.1 ist ein real aufgenommener Verlauf von Linkgütewerten einer WLAN-Schnittstelle dargestellt, welcher in Anhang D.1 tiefgreifend diskutiert wird. Es wurden drei Entscheidungsschwellen definiert, wobei die unterste Schwelle die so genannte *Abrissschwelle* darstellt. Fällt der Linkgütewert unterhalb dieser Schwelle, gilt der Link als

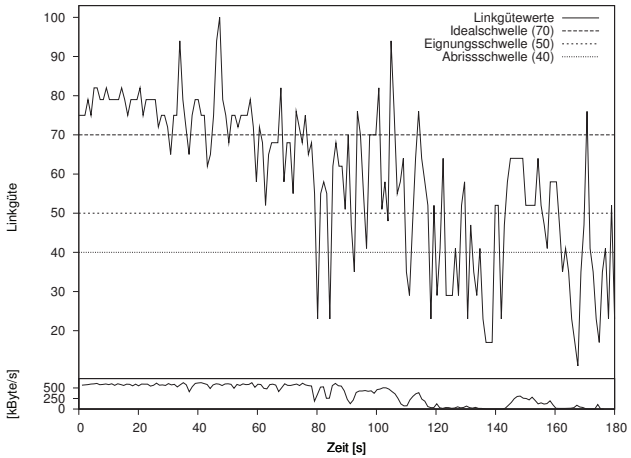

Abbildung 6.1: Zur Beurteilung von Signalstärke- und Linkgütewerten kommen drei Schwellen zum Einsatz. Fällt der Wert unter die *Abrissschwelle*, gilt der Netzzugang als nicht benutzbar. Er muss dann erst wieder die *Eignungsschwelle* überschreiten, um benutzt werden zu können (Hysterese). Überschreitet der Wert die *Idealschwelle*, werden zudem die Ersatzlinks als unnötig erachtet und abgebaut.

nicht länger benutzbar. Um „Flattereffekte“ zu vermeiden, darf der Link erst wieder nach Überschreitung der *Eignungsschwelle* benutzt werden. Nur nach Überschreitung der dritten Schwelle, der *Idealschwelle*, dürfen bestehende Ersatzlinks wieder abgebaut werden. Die drei Schwellen wurden hier mit 40 für die Abrisschwelle, 50 für die Eignungsschwelle und 70 für die Idealschwelle festgelegt, was bezüglich WLAN die im Demonstrator empirisch durch Experimente ermittelten Werte für Linkgütewerte darstellt. Für Betrachtung von Signalstärkewerten liegen die ermittelten Entscheidungsschwellen bei 18, 22 und 30.

Zur Ermittlung der drei Entscheidungsschwellen wurde ein TCP-basierter Datenstrom über den drahtlosen Netzzugang übertragen und dann sekündlich die aktuelle Datenrate zusammen mit dem gelieferten Signalstärke- und Linkgütewert in einer Datei vermerkt. In Abbildung 6.1 ist erkennbar, wie die Datenrate zusammenbrach, als der Linkgütewert die (willkürlich festgelegte) Abrissschwelle unterschritt. Es bietet sich an, eine Vielzahl solcher Messreihen aufzunehmen, um dann eine möglichst gute Schätzung für die drei Schwellen zu erhalten. Zu beachten ist jedoch, dass diese Art der Ermittlung der Entscheidungsschwellen ungenau ist. Es wird beispielsweise nicht berücksichtigt, dass auch an der Basisstation Effekte wie Rauschen und Fading eine Rolle spielen. Da einem mo-

bilen Endgerät diese Werte jedoch nicht zur Verfügung stehen, beschränkt es sich auf die Auswertung der lokal ermittelbaren Parameter.

Reaktive Entscheider sind einfach strukturiert, da zu jedem Zeitpunkt lediglich der vorliegende Status ausgewertet zu werden braucht. Damit sind sowohl Rechen- als auch Speicheraufwand gering. Von Nachteil ist jedoch, dass diverse technische Parameter zu jedem Netzzugang bekannt sein müssen. Die in Abschnitt 6.1.1.3 vorgestellte VHDF funktioniert nur dann ordnungsgemäß, wenn zu jedem Netzzugangspunkt Informationen beispielsweise in Bezug auf die erreichbare Bitrate oder den Energieverbrauch bekannt sind. Jedoch ist unklar, woher diese Werte genommen werden sollen. Ein technisch wenig versierter Nutzer würde durch diese Parameter überfordert werden. Bei drahtlosen Technologien müssen zudem die drei Entscheidungsschwellen ermittelt werden, was vorab anhand von Messungen geschehen, aber nicht jedem Anwender zugemutet werden kann.

Ein weiteres Problem wird deutlich, wenn man sich in Abbildung 6.1 die Wechselwirkungen zwischen den Entscheidungsschwellen und einzelnen verrauschten Messwerten anschaut. Einzelne Messwerte können sowohl nach oben oder nach unten ausbrechen und werden als „Ausreißer“ bezeichnet. Ausreißer sind für einen reaktiv arbeitenden Handoverentscheider problematisch, da wie zum Zeitpunkt 170s selbst bei einem unbrauchbar gewordenen Hauptlink noch die Idealschwelle überschritten werden kann, was zu einem unbrauchbaren Verhalten des Handoverentscheiders führt. Verbessern lässt sich der reaktive Entscheider dadurch, dass beispielsweise vorab eine Mittelung über fünf benachbarte Messwerte durchgeführt wird. Die Messwerte sollten dann aber auch mit einer fünfmal höheren Rate aufgenommen werden.

6.2.3 Lernende Entscheider

Während reaktive Entscheider lediglich anhand *momentan* vorliegendender Statusinformationenen eine Bewertung der Netzzugänge durchführen, können *lernende* Entscheider „erlerntes Wissen“ aus der Vergangenheit mit in die Entscheidungsfindung einfließen lassen.

Ein Anwendungsfall wäre beispielsweise, dass der Handoverentscheider die in Abschnitt 6.1 vorgestellten Entscheidungsschwellen selbst ermitteln und im Laufe der Zeit immer weiter an den idealen Wert annähern könnte. Ausserdem könnte ein lernender Entscheider seine Netzzugangstechnologien in Bezug auf die benötigte Assoziationszeit („Einwahldauer“) oder die effektiv erreichbare Datenrate hin beobachten und somit eine bessere Kenntnis über deren Fähigkeiten erlangen. Dem Nutzer könnte es somit erspart werden, anfangs eine Vielzahl an technischen Parametern zur Beschreibung der verfüg-

baren Netzzugänge hinterlegen zu müssen. Stattdessen würden die relevanten Informationen nach und nach automatisch ermittelt werden, und könnten sich zudem bei sich ändernden Rahmenbedingungen anpassen.

Eine andere Form des lernenden Entscheiders basiert auf der Idee, sich wiederholende Muster durch Beobachtung zu erkennen und dann anhand dieses Wissens zukünftige Ereignisse vorauszusagen. Fährt ein Nutzer beispielsweise jeden Tag dieselbe Strecke von zuhause hin zu seiner Arbeitsstätte, könnte beim Durchfahren eines Tunnels jeden Tag an derselben Stelle der Internetzugang abreißen. Ein lernender Entscheider könnte dieses regelmäßig wiederkehrendes Ereignis bemerken. Er wäre dann in der Lage, das täglich drohende Abrissereignis vorhersagen. In [SNLW08] wird hierzu ein Handoverentscheider vorgestellt, welcher auf der Modellierung eines Markowprozesses basiert. Ein solcher stellt ein gedächtnisloses System dar, in welchem Zustandswechsel anhand einer mittleren Verweildauer pro Zustand sowie durch Übergangswahrscheinlichkeiten zwischen diesen Zuständen beschrieben werden. Anhand eines solchen Systems aus miteinander verketten Zuständen wird versucht, zukünftige Ereignisse vorherzusagen. Das durch bisherige Beobachtungen gewonnene Wissen spiegelt sich in den mittleren Verweildauern sowie in den Übergangswahrscheinlichkeiten wider.

Vorteilhaft ist, dass durch den Einsatz eines lernenden Entscheiders die Menge der vorab zu spezifizierenden Parameter verringert werden kann. Die Parameter könnten sich über die Zeit immer besser an die vorherrschenden Gegebenheiten anpassen. Ein auf Markowprozessen basierender Entscheider könnte zudem wiederkehrende Verhaltensmuster ausnutzen, und somit zu einer effektiveren Handoverentscheidung beitragen.

Als nachteilig erweisen sich ein erhöhter Implementierungsaufwand und ein erhöhter Rechen- sowie Speicherbedarf. REACH implementiert bislang keine Algorithmen, welche eine auf Beobachtung basierende Anpassung von Parametern durchführen. In Bezug auf beispielsweise die benötigte Assoziationszeit eines Netzzugangsgeräts wäre ein solches Schema jedoch sinnvoll. Schließlich ist es einem Nutzer nicht zumutbar, im Vorfeld möglichst genau anzugeben, wie viel Zeit das GPRS-Modem seines Mobiltelefones zum Aufbau einer Verbindung ins Internet benötigen wird.

Die Modellierung vergangener Bebachtungen mit Hilfe eines Markowprozesses wird durch die Menge des erlernbaren und damit speicherbaren Wissens erschwert. Kleine mobile Endgeräte können die mit einem solchen Modell verbundenen Datenmengen und den Rechenaufwand eventuell nicht bewältigen. Zudem kommt es zu Fehlentscheidungen, wenn im vorgestellten Beispiel der Fahrer eines Tages eine andere Route wählt, oder wenn er gar auf Dienstreise ist. Der Entscheider kann nicht erkennen, welche der neuen Eingaben relevant sind und gelernt werden müssen, und ob bereits gelerntes Wissen zum jetzigen Zeitpunkt angewendet werden darf. Auf einer Dienstreise müsste der

Lernprozess idealerweise pausiert werden können, um die Datenbasis nicht mit unnötigen Informationen zu füllen. Nach einem Umzug oder einem Wechsel des Arbeitgebers würde es sich sogar anbieten, die Wissensbasis zu löschen und neu aufbauen zu lassen. Für REACH wurde ein solcher Handoverentscheider bisher nicht umgesetzt.

6.2.4 Prädiktive Entscheider

Reaktive Entscheider werden nur den aktuell vorliegenden Zustand aus, und lernende Entscheider versuchen, bereits erlebte Szenarien auf die aktuelle Situation zu beziehen. Beide sind jedoch nicht in der Lage, zukünftige Ereignisse durch Beobachtung der unmittelbaren Vergangenheit zu prognostizieren. Dies ist die Domäne der *prädiktiven* Entscheider, welche beispielsweise versuchen, durch Extrapolation eines fallenden Linkgüteverlaufs den Zeitpunkt des drohenden Linkabrisses zu errechnen. Hierfür wird im Folgenden ein prädiktiver Entscheider vorgestellt, welcher auf der Anwendung von Regressionsrechnung basiert.

6.2.4.1 Motivation zur Regressionsrechnung

Die Signalstärke und auch die Linkgüte werden hauptsächlich durch die Streckendämpfung bestimmt, welche wiederum abhängig von der Distanz zwischen mobilem Endgerät und Basisstation ist. Die Eigenbewegung des mobilen Endgeräts führt damit zu einer sich mit der Zeit ändernden Signalstärke. Da der Abstand zur Basisstation allerdings unbekannt ist, steht statt einer Distanzangabe nur die voranschreitende Zeit zur Verfügung. Es wird also davon ausgegangen, dass ein Zusammenhang zwischen Ort und Zeit besteht. Eine solche Verknüpfung ergibt jedoch nur dann einen Sinn, wenn sich das mobile Endgerät geradlinig und mit gleichförmiger Geschwindigkeit auf eine Basisstation zu bewegt oder sich von ihr entfernt. Zudem darf die Wellenausbreitung nicht durch Hindernisse beeinflusst werden. In der Praxis kann von einer solchen Einschränkung jedoch nicht ausgegangen werden. Nur während kurzer Zeitabschnitte könnte näherungsweise von einer gleichförmigen Bewegung ausgegangen werden, jedoch selten bei einer längerfristigen Betrachtung.

Entfernt sich das mobile Endgerät von der Basisstation, wird sich die Signalstärke mit der Zeit verschlechtern, bis es zur Unterschreitung der Abrissschwelle kommt. Die Aufgabe des prädiktiven Entscheiders ist es, diesen Abrisszeitpunkt vorauszusagen, was durch Betrachtung vergangener Signalstärkewerte oder Linkgütewerte erfolgen muss. Die „voraussichtlich verbleibende Nutzungsdauer“ stellt den „Mehrwert“ eines prädiktiven Entscheiders gegenüber einem reaktiven Entscheider dar, denn mit Hilfe dieser Information kann ein Ersatzlink zeitnah noch vor Abriss des Hauptlinks aufgebaut werden.

Im Idealfall ist die Überdeckung nahtlos, sodass es weder zu einer Unterbrechung des Nutzerdatenstroms kommt noch unnötig Ressourcen belegt werden.

6.2.4.2 Die lineare Regressionsrechnung

Die einfachste Form der Regressionsrechnung basiert auf der Annahme eines linearen Zusammenhangs; man spricht in diesem Fall von einer *linearen Regressionsrechnung*.

Man nimmt hierfür eine *Punktewolke* bestehend aus den zeitstempelbehafteten Messwerten der jüngsten Vergangenheit, und legt eine *Ausgleichsgerade* durch diese hindurch. Die Ausgleichsgerade muss den Verlauf der Punkte „so gut wie möglich“ nachbilden, was in Abschnitt 6.2.4.4 vorgestellt wird. Anhand der Steigung der Ausgleichsgeraden kann dann bereits erkannt werden, ob ein steigender oder ein fallender Verlauf vorliegt.

Zudem verwendet REACH wie beim reaktiven Entscheider drei Entscheidungsschwellen. Die unterste Schwelle ist wiederum die *Abrissschwelle*, deren Unterschreitung auf einen abreißenden Link hindeutet. Sie stellt die für eine Prognose relevante Schwelle dar. Wird für einen momentan benutzbaren Link eine fallende Regressionsgerade errechnet, kann der Schnittpunkt mit der Abrisschwelle ermittelt werden. Interessant ist dabei der Zeitpunkt des Durchstoßens, welcher eine Aussage über die verbleibende Zeit bis zum drohenden Linkabriss ermöglicht.

Die vorraussichtliche Restzeit wird mit der Aufbauzeit potentieller Ersatzlinks verglichen, sodass schon vor Abriss des Hauptlinks der Aufbau eines Ersatzlinks angestoßen werden kann. Idealerweise wird der Aufbau des Ersatzlinks bereits vor Abriss des Hauptlinks abgeschlossen. Aber selbst wenn der Abrisszeitpunkt nicht exakt berechnet wurde, so ist dennoch erkennbar, dass die Ausfallzeit geringer sein wird als bei einem reaktiven Entscheider. Dieser würde den Ersatzlink nämlich erst dann aufbauen, wenn der Hauptlink bereits abgerissen ist, und damit eine „konstante“ (maximale) Ausfallzeit aufweisen.

Ein Nachteil des prädiktiven Entscheiders besteht jedoch in der Gefahr, dass Ersatzlinks auf Grund von falschen Prognosen aufgebaut werden. Hierbei werden unnötigerweise Ressourcen verbraucht, was beim reaktiven Entscheider nicht der Fall ist.

An dieser Stelle ist ganz klar hervorzuheben, dass es nicht Absicht des Autors ist, eine perfekte Vorhersage anzubieten. Stattdessen wird die Chance ausgenutzt, dass mit einem prädiktiven Entscheider *bessere* Ergebnisse erzielt werden können, als es mit einem reaktiven Entscheider möglich wäre.

6.2.4.3 Funktionsklassen

Die lineare Regressionsrechnung hat den Vorteil, dass sie gut berechenbar ist. Als problematisch erweist es sich jedoch, dass sich Signalstärke und Zeit nicht in einem linearen

Verhältnis zueinander befinden. Die Distanz wird zwar vereinfacht als linear zur Zeit angesehen, da von einer gleichförmigen Bewegung des mobilen Endgerät ausgegangen wird. In die Signalstärke spielen jedoch physikalische Zusammenhänge mit ein, welche sich nichtlinear zur Distanz und damit zur Zeit verhalten.

Dieser Zusammenhang wurde in den studentischen Arbeiten [Ahma07] und [Schu08] untersucht. Dabei wurden *Funktionsklassen* ermittelt, welche sich zur Beschreibung des Zusammenhanges zwischen Zeit- und Gütewerten eignen, und welche sich wiederum auf eine lineare Regressionsrechnung zurückführen lassen. Die folgenden Funktionsklassen wurden dabei beleuchtet:

- **Exponentieller Zusammenhang**:
  Aktuelle WLAN-Treiber des LINUX-Kernels liefern sowohl für die Signalstärke als auch für die Linkqualität einen ganzzahligen Wert, welcher bei der vorliegenden Hardware zudem auf den Wertebereich 0 bis 100 normiert wurde.

  Mit zunehmender Entfernung zur Sendeantenne sinken beide Werte zunehmend ab, allerdings nicht linear, sondern näherungsweise asymptotisch. Eine lineare Ausgleichsgerade würde daher nur eine grobe Schätzung liefern, da die Krümmung des realen Verlaufs nicht dargestellt wird. Daher wurde als Verbesserungsvorschlag statt eines linearen Zusammenhanges ein exponentieller Zusammenhang untersucht. Dieser ist zwar physikalisch nicht zu rechtfertigen,[1] erlaubt jedoch im Vergleich zu einem linearen Zusammenhang eine Darstellung gekrümmter Verläufe [Ahma07].

- **Inversquadratischer Zusammenhang**:
  Der inversquadratische Zusammenhang begründet sich auf dem Prinzip der *ungestörten Freiraumausbreitung*. Es wird angenommen, dass die empfangene Leistung quadratisch mit dem Abstand zur Sendeantenne absinkt.

  Diese Darstellung ist besonders dann von Interesse, wenn „rohe“ Signalstärke- oder Feldstärkewerte vorliegen. Solche Werte weisen einen großen Dynamikbereich auf.

- **Linearer Zusammenhang**:
  Ein linearer Zusammenhang zwischen Signalstärke und Zeit lässt sich nicht mit physikalischen Zusammenhängen begründen. Allerdings ist der lineare Zusammenhang sehr einfach zu berechnen. Interessanterweise lassen sich sowohl der inversquadratische als auch der exponentielle Zusammenhang in einen linearen Zusammenhang überführen. Die dazu notwendigen Transformationen sind in [Papu99] nachlesbar

[1] Ein exponentiell mit steigender Distanz abfallender Signalstärkewert würde die Existenz einer *Halbwertsdistanz* bedeuten, was physikalisch gesehen Unsinn wäre.

und wurden in der studentischen Arbeit [Schu08] für REACH umgesetzt. Es steht dem Anwender frei, auf jegliche Transformationen zu verzichten und von einem linearen Zusammenhang zwischen Zeit und Signalstärke auszugehen.

Abhängig vom jeweils erkennbaren Zusammenhang muss die korrekte Funktionsklasse gefunden werden. Da jedoch häufig keine Informationen bezüglich der Herkunft der (meistens einheitslosen) Werte verfügbar sind, bietet es sich an, die Funktionsklasse durch Experimente empirisch auszuwählen.

6.2.4.4 Durchführung der linearen Regressionsrechnung

Für REACH wurde als Berechnungsverfahren nur die lineare Regressionsrechnung implementiert. Es besteht aber die Möglichkeit, mit Hilfe von auswählbaren Abbildungsfunktionen auch exponentielle und inversquadratische Verläufe zu berücksichtigen. Dazu findet eine Transformation der Messwerte in die „lineare Darstellung“ statt, wo dann die lineare Regressionsrechnung zur Anwendung kommt. In dieser Ausarbeitung wird das kombinierte Verfahren als „exponentielle“ bzw. als „inversquadratische Regressionsrechnung“ bezeichnet.

Die lineare Regressionsrechnung basiert auf der Suche nach einer Ausgleichsgeraden, welche die gegebene Punktewolke *möglichst gut* darstellt.

$$y = f(x) = \mathrm{a}x + \mathrm{b} \tag{6.3}$$

Eine solche Ausgleichsgerade wird durch eine Geradengleichung (siehe Formel 6.3) definiert. Es gilt, die Koeffizienten a und b zu bestimmen, wobei a als Steigung und b als Achsenabschnitt bezeichnet werden.

Die Berechnung der Koeffizienten erfolgt anhand der „Methode der kleinsten Quadrade“, wobei die Lage der Ausgleichsgeraden so erfolgt, dass die mittlere quadratische Abweichung zwischen Ausgleichsgerade und den einzelnen Messwerten minimiert wird.

$$a = \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\sum_{i=1}^{n} x_i^2 - n\bar{x}^2} = \frac{\overline{xy} - \bar{x}\bar{y}}{\overline{x^2} - \bar{x}^2} \tag{6.4}$$

$$b = \bar{y} - a\bar{x} \tag{6.5}$$

Die Berechnungvorschrift für die Koeffizienten a und b wurde aus einem mathematischen Lehrbuch entnommen [Papu99]. Sie lassen sich anhand der Formeln 6.4 und 6.5 berechnen. In die Berechnung fließen diverse gemittelte Werte ein, welche vorab aus

den vorliegenden Messwerten errechnet werden müssen. $\bar{x}$ und $\bar{y}$ stellen die empirischen Mittelwerte der zeitlichen bzw. der gütebezogenen Komponente der Messwerte dar. Weiterhin werden noch der empirische Mittelwert der Produkte aus Zeit- und Gütewerten $\overline{xy}$ sowie der empirische Mittelwert aus den Quadraten aller zeitlichen Komponenten $\overline{x^2}$ benötigt.

Formel 6.4 enthält dazu noch den Parameter n, welcher die Anzahl der an der Mittelwertbildung beteiligten Messwerte darstellt. Es werden nicht *alle* bislang erfassten Messwerte berücksichtigt, sondern immer nur letzten n Werte aus der jüngsten Vergangenheit. Die Mittelwerte werden in REACH als *gleitende Mittelwerte* implementiert, wobei der Nutzer eine bestimmte *Fenstergröße* vorgeben kann.

6.2.4.5 Behandlung von „Ausreißern“

In der Praxis werden sich die mobilen Endgeräte weder im freien Raum befinden noch werden sie sich geradlinig oder gleichförmig bewegen. Vielmehr werden die Verläufe durch Richtungswechsel geprägt sein und durch Fadingerscheinungen beeinflusst werden. Da die aktuelle Ausgleichsgerade jeweils aus Werten der Vergangenheit berechnet wird, kann sie unmittelbar nach einem Richtungswechsel oder bei Verdeckung durch ein Hindernis nicht länger brauchbare Vorhersagen liefern. Zudem können einzelne Messwerte stark durch „Deep Fading“ verfälscht worden sein. Würde man solche „Ausreißer“ mit in die Mittelwertbildung einbeziehen, würden sie die Ausgleichsgerade in hohem Maße negativ beeinflussen. Daher wurde beschlossen, eine *Ausreißererkennung* durchzuführen.

Die Ausreißererkennung läuft wie folgt ab: Es gibt eine bestimmte Menge an Messwerten („Fenster“), welche wie beschrieben zur Bestimmung der Parameter der Ausgleichsgeraden herangezogen werden. Nach Hinzufügen eines neuen Messwerts wird dafür anfangs noch das komplette Wertefenster betrachtet, und damit auch alle Werte zur Bestimmung der Geradenparameter herangezogen.

$$\sigma_f = \sqrt{E\left(\left(f(X) - Y\right)^2\right)} \tag{6.6}$$

Anschließend wird jeder der berücksichtigten Messwerte als potentieller Ausreißer betrachtet. Ein Ausreißer zeichnet sich dadurch aus, dass er einen *zu großen* Abstand zur Ausgleichsgeraden aufweist. Es muss also für jeden Messwert geklärt werden, ob er noch *nahe genug* an der Ausgleichsgeraden liegt. Dazu wird in einem ersten Schritt Formel 6.6 herangezogen. Diese ähnelt der Formel zur Berechnung der empirischen Standardabweichung, nur dass hier der Erwartungswert zu jedem Messwert durch die Ausgleichsgerade vorgegeben wird. σ_f wird in diesem Fall als „die empirische Standardabweichung bezüglich der Ausgleichsgeraden f“ definiert.

In [Schu08] wurde die empirische Standardabweichung hierfür noch mit einem festen *Konfidenzfaktor* gewichtet. Der dabei errechnete „gewichtete Abstand“ wurde zur Bildung eines Vertrauensintervalls um die Ausgleichsgerade herum verwendet. Werte, die sich innerhalb dieses „Konfidenzschlauchs“ befanden, wurden weiterhin berücksichtigt; Werte außerhalb des Schlauchs galten als Ausreißer und wurden ignoriert. Nach diesem Filterungsschritt erfolgte eine erneute Bestimmung der Geradenparameter, allerdings mit dem reduzierten Wertefenster.

Diese Art der Ausreißererkennung ist gut geeignet, um Fadingerscheinungen wie „Deep Fading“ herauszurechnen. Wählt man den Konfidenzfaktor durch empirisches Ausprobieren so, dass durch Fading beeinflusste Werte außerhalb des Konfidenzschlauchs liegen, können sie die Ausgleichsgerade nicht länger negativ beeinflussen. In praktischen Tests hatte sich hierbei ein Konfidenzfaktor von `2,7` bewährt. Nachteilig wirkte sich allerdings aus, dass diese Form der Ausreißererkennung ein träges Verhalten zeigt. Gemeint ist, dass sich der Verlauf der Signalstärkewerte in kurzer Zeit stark ändern kann, wenn beispielsweise eine Häuserecke passiert und dabei eine Basisstation abgeschattet wird. Die aktuellsten Messwerte liegen dann außerhalb des Konfidenzschlauchs, und werden fälschlicherweise als Ausreißer klassifiziert. Es dauert dann erst ein paar Sekunden, bis sich das Fenster weit genug in die Zukunft verschoben hat, bis nicht mehr die aktuellsten, sondern die ältesten Werte als Ausreißer gelten. In dieser „verlorenen Zeit“ hatte der Entscheider keine Möglichkeit, den eventuell drohenden Abriss zu erkennen und den Ersatzlink anzufordern.

Der vorgestellte Ansatz wurde in dieser Ausarbeitung mit dem Ziel weiterentwickelt, den Trägheitseffekten entgegen zu wirken. Erwünscht war eine Form der Ausreißererkennung, welche auch sich rasch ändernden Verläufen folgen kann. Interessant sind dabei vor allem die aktuellsten Messwerte, da ohne Kenntnis der Zukunft nicht entschieden werden kann, ob ein stark abweichender Wert als Ausreißer oder als erstes Anzeichen eines sich ändernden Verlaufs zu werten ist. Da diese Frage noch nicht beantwortet werden kann, dürfen solche Messwerte nicht leichtfertig entfernt werden. Stehen aber später die nachfolgenden Messwerte zur Verfügung, ist die Beantwortung unproblematisch, und der Messwert kann nachträglich als Ausreißer klassifiziert werden. Dann hätte man ihn zwar anfangs berücksichtigt, allerdings wird dieser Wert nicht länger verwendet, nachdem er als Ausreißer erkennbar geworden ist. Es wird also ein Kompromiss zwischen „Trägheit“ und „Glättungseigenschaft“ durchgeführt, wobei das Augenmerk verstärkt auf die Beseitigung von Trägheitseffekten gelegt wird.

Erreicht wird das beschriebene Wunschergebnis durch einen modifizierten Verlauf des Konfidenzschlauchs. Der erste Aspekt besteht darin, dass bei aktuellen Messwerten größere Abweichungen zugelassen werden, wobei in Richtung älterer Messwerte die erlaub-

te Abweichung immer geringer wird. Der Konfidenzbereich erhält dabei die Form eines Trichters, was anhand der folgenden Diagramme verdeutlicht wird. Zudem basiert das verbesserte Verfahren nicht länger auf zwei Berechnungs- und einem Klassifizierungsschritt, sondern arbeitet rekursiv.

Der rekursive Ablauf stellt sich so dar, dass nach Erhalt eines neuen Messwerts anfangs wieder das komplette Fenster berücksichtigt wird, also die Geradenparameter anhand *aller* Werte berechnet werden. Anschließend wird die empirische Standardabweichung bezüglich der Ausgleichsgeraden berechnet. Jetzt kommt der modifizierte trichterförmige Verlauf des Konfidenzfaktors zum Einsatz, welcher mit der empirischen Standardabweichung multipliziert wird und damit den „Konfidenztrichter“ ergibt. Werte außerhalb dieses Trichters sind potentielle Ausreißer, allerdings wird in jedem Schritt immer nur ein einziger Wert als Ausreißer klassifiziert und aus der Menge der aktuell betrachteten Werte des Fensters entfernt. Die Berechnung beginnt dann von Neuem, allerdings mit dem reduzierten Wertefenster, sodass für die hierbei berechneten Geradenparameter bereits der „schlimmste“ Ausreißer nicht länger berücksichtigt wurde.

Hierbei sinkt mit jedem Schritt die empirische Standardabweichung, sodass sich auch der Konfidenztrichter weiter „verjüngt“. Diese rekursive Schleife läuft so lange, bis sich entweder alle verbliebenen Werte innerhalb des Konfidenztrichters befinden, oder bis der minimale Füllstand des Wertefensters erreicht wird. Die zuletzt errechneten Geradenparameter bilden das Ergebnis. Es besteht allerdings die Gefahr, dass der Algorithmus auf Grund des sich mit jedem Schritt verjüngenden Konfidenztrichters immer bis zum minimalen Füllstand des Fensters fortgesetzt wird.

Der „schlechteste“ Wert ist immer derjenige Wert, der *im Verhältnis* zum Konfidenztrichter den größten Abstand zur Regressionsgeraden aufweist. Werte, die sich innerhalb des Trichters befinden, weisen damit ein Verhältnis kleiner 1 auf. Ein Verhältnis von 2 bedeutet beispielsweise, dass der Abstand doppelt so groß ist wie der Radius des Trichters an dieser Stelle. Einem „neuen“ Messwert wird damit eine größere Abweichung zugebilligt als älteren Messwerten, da nicht bloß der Konfidenztrichter breiter ist, sondern immer das Verhältnis betrachtet wird. Der Messwert mit dem größten Verhältnis wird als Ausreißer klassifiziert und entfernt.

In die Berechnung des trichterförmigen Verlaufs des Konfidenzfaktors spielen zwei Komponenten mit ein. Zum einen wird der trichterförmige Verlauf definiert, zum anderen gibt es noch einen füllstandsabhängigen Korrekturfaktor. Der letztgenannte Korrekturfaktor hat die Aufgabe, den sich mit jedem Berechnungsschritt verjüngenden Konfidenztrichter wieder „aufzuweiten“, damit nicht länger bevorzugt die größtmögliche Anzahl an Messwerten entfernt wird. Der Verlauf des Füllstandfaktors ist in Abbildung 6.2 im oberen Diagramm dargestellt. Ist das Wertefenster noch komplett gefüllt, verhält er sich

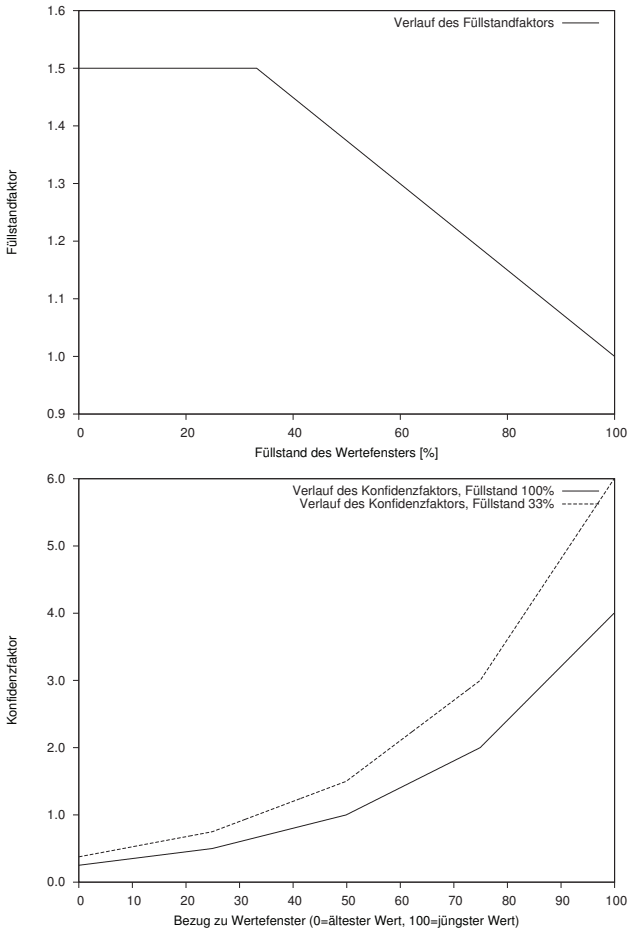

Abbildung 6.2: Im oberen Diagramm ist der Verlauf des Füllstandfaktors dargestellt. Das untere Diagramm zeigt den Verlauf des Konfidenzfaktors für ein minimal und ein maximal gefülltes Wertefenster. Die minimale Fenstergröße betrug `33%`, der Leerungsfaktor `1,5` und der maximale Konfidenzfaktor `4,0`.

neutral (Korrekturfaktor `1,0`). Mit sinkendem Füllstand ändert er sich linear, bis er bei Erreichen des „minimalen Füllstandes“ den so genannten „Leerungsfaktor“ erreicht. Dieser „Leerungsfaktor“ muss spezifiziert werden, wobei durch empirisches Ausprobieren

ein Wert von `1,5` gefunden und anschließend durchgängig verwendet wurde.

Der trichterförmige Verlauf kann durch Angabe eines einzigen Parameters beeinflusst werden, welcher den Radius an seiner weitesten Stelle definiert. Dieser „maximale Konfidenzfaktor“ ist empirisch auf `4,0` festgelegt worden. Der trichterförmige Verlauf wird durch vier linear verlaufende Abschnitte beschrieben, in welchen jeweils eine Änderung um den Faktor 0,5 überstrichen wird. Am linken Rand, entsprechend der Position des ältesten Messwerts im Wertefenster, ist der Konfidenzfaktor auf 1/16 seines maximalen Werts abgefallen. Anschließend werden der aktuelle Leerungsfaktor und der Verlaufswert miteinander multipliziert. Im unteren Diagramm von Abbildung 6.2 ist der sich ergebende Verlauf dargestellt, einmal für ein komplett gefülltes und einmal für ein minimal gefülltes Wertefenster.

Anzumerken ist, dass der beschriebene Verlauf willkürlich ausgewählt worden ist. Ziel war es gewesen, bei älteren Werten geringere Abweichungen zuzulassen, und dafür ist der vorgeschlagene Verlauf geeignet.

In Abbildung 6.3 werden die Auswirkungen der beschriebenen Ausreißererkennung verdeutlicht. Zu erkennen ist der bereits vorgestellte Verlauf von Linkgütewerten, wobei der aktuellste Messwert zum Zeitpunkt 59s hinzugefügt worden ist. Das Fenster überstreckt `30` Sekunden, also `30` Messwerte, bei einem minimalen Füllstand von `10` Sekunden (10 Messwerte). Man erkennt oben den Verlauf der Ausgleichsgeraden, in welche noch alle 30 markierten Werte mit eingeflossen sind. Der eingezeichnete Konfidenztrichter verdeutlicht die zeitliche Ausdehnung des Fensters, und man kann erkennen, dass er in diesem Berechnungsschritt noch weit geöffnet ist. Im unteren Diagramm sind Ausgleichsgerade und Konfidenztrichter nach Durchlauf der rekursiven Ausreißererkennung dargestellt. Von den ursprünglich 30 Messwerten wurden 11 als Ausreißer klassifiziert und entfernt. Damit sank die empirische Standardabweichung, was trotz angestiegenem Leerungsfaktor zu einem „engeren“ Konfidenztrichter geführt hat. Die sich schlussendlich ergebende Regressionsgerade verläuft nicht länger waagerecht, sondern weist einen fallenden Verlauf auf.

Nachdem die Regressionsrechnung samt Ausreißererkennung durchgeführt wurde und aktualisierte Geradenparameter bereit stehen, kann man den Funktionswert der Regressionsgeraden zum Zeitpunkt des soeben hinzugefügten Messwerts errechnen. Dieser Wert ist im unteren Diagramm von Abbildung 6.3 mit einem Kreuz markiert und wird als „gefilterter Linkgütewert“ bezeichnet. Kam dabei die lineare Regressionsrechnung zum Einsatz, wird im Folgenden vom „linear gefilterten Linkgütewert“ gesprochen.

Für Abbildung 6.4 wurde zu jedem Zeitpunkt der linear gefilterte Linkgütewert errechnet und zusammen mit den ungefilterten Messwerten aufgetragen. Es lässt sich erkennen, dass die erkennbaren Ausreißer einen Einfluss auf den Verlauf des gefilterten

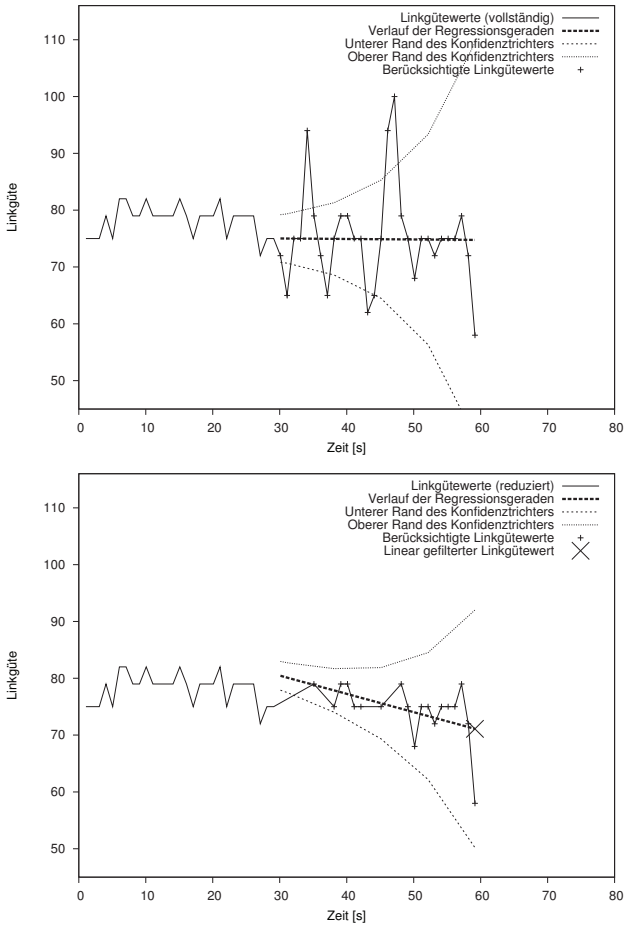

Abbildung 6.3: Oben ist der Verlauf der Ausgleichsgeraden vor, und unten nach Durchführung der Ausreißerbehandlung gezeigt. Es kam lineare Regressionsrechnung mit dem eingezeichneten, trichterförmig verlaufendem Konfidenzbereich zum Einsatz. Im unteren Diagramm ist zudem der linear gefilterte Gütewert eingezeichnet.

Werts haben. Andererseits ist keine nennenswerte Trägheit erkennbar, außer im Bereich von 135s bis 145s, wo der linear gefilterte Verlauf ein wenig nachzueilen scheint. Die linear gefilterten Linkgütewerte werden später für die Handoverentscheidung benötigt.

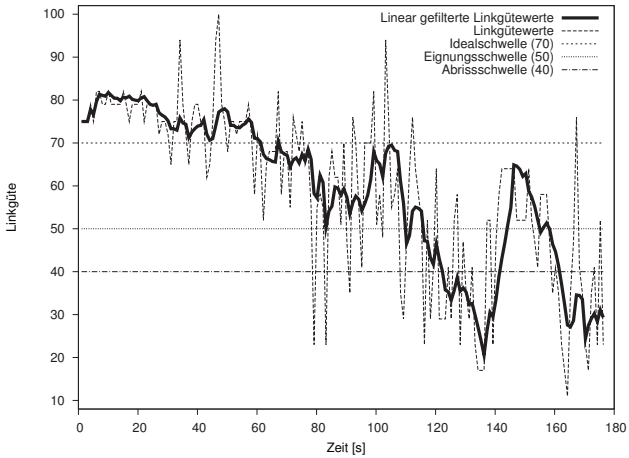

Abbildung 6.4: Die durch Anwendung von linearer Regressionsrechnung gefilterten Linkgütewerte ergeben einen ruhigeren Verlauf als die ungefilterten Messwerte.

6.2.4.6 Verbleibende Zeit bis zum Linkabriss

Nachdem die Parameter der Regressionsgerade berechnet worden sind, steht mit ihr ein Werkzeug zur Prognose eines zukünftigen Abrisses zur Verfügung. Man geht dabei davon aus, dass sich der momentan sichtbare Verlauf (welcher in die Regressionsgerade einfloss) in der nahen Zukunft gleichförmig weiterentwickeln wird. Dazu wird der Schnittpunkt zwischen Regressionsgerade und der Abrissschwelle errechnet. Die Prognose beruht auf der Annahme, dass zum Zeitpunkt des errechneten Schnittpunkts der Hauptlink unbrauchbar wird.

Diese Annahme ergibt allerdings nur dann Sinn, wenn die gewählte Funktionsklasse geeignet ist, den realen Verlauf der Linkgütewerte abzubilden. Abbildung 6.5 zeigt die Extrapolationsergebnisse für die als inversquadratische, exponentielle und lineare Regressionsrechnung bezeichneten Berechnungsverfahren. Die Abrissprognose liegt bei der linearen Regressionsrechnung am nahesten in der Zukunft, während sowohl die inversquadratische als auch die exponentielle Regressionsrechnung einen konkaven (nach oben geöffneten) Verlauf aufweisen und damit der Zeitpunkt des prognostizierten Abrisses in die Zukunft rückt.

Die relevante Information, welche aus Abbildung 6.5 entnommen werden kann, ist die zeitliche Differenz zwischen aktuellem Zeitpunkt (59s) und dem errechneten Schnitt-

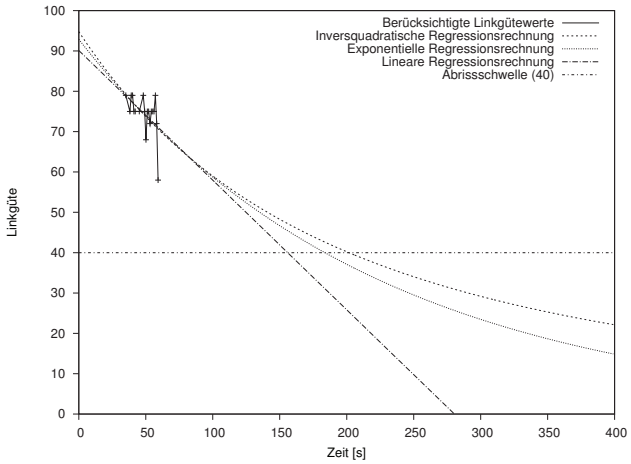

Abbildung 6.5: Die exponentielle, inversquadratische und die lineare Regressionsrechnung unterscheiden sich hinsichtlich des extrapolierten Linkgüteverlaufs. Weiterhin sind die 19 berücksichtigten Linkgütewerte erkennbar.

punkt mit der Abrissschwelle. Trägt man die errechnete zeitliche Differenz über der Zeit auf, erhält man das in Abbildung 6.6 dargestellte „Zeitdifferenzdiagramm“.

Aus dem Zeitdifferenzdiagramm lässt sich jedoch nicht ablesen, ob der Hauptlink zu einem bestimmten Zeitpunkt benutzbar war. Ein „Schnittpunkt 20 Sekunden in der Zukunft“ kann einerseits bedeuten, dass sich die Linkgütewerte gerade kontinuierlich verschlechtern und in 20 Sekunden ein Abriss erwartet wird. Andererseits kann es aber auch auf einen bereits abgerissenen Hauptlink hindeuten, dessen Linkgütewerte jedoch eine steigende Tendenz aufweisen. Dann wäre der errechnete Schnittpunkt nicht als Abrissereignis, sondern als „Erholungsereignis“ zu deuten.

Als hilfreiche Zusatzinformation könnte der „Verlaufstrend“ herangezogen werden, also ob sich die Linkgütewerte aufsteigend oder abfallend entwickeln. Eine ähnliche Aussagekraft weist allerdings der bereits vorgestellte gefilterte Linkgütewert auf, welcher wie in Abbildung 6.3 erkennbar immer auf der aktuellen Regressionsgeraden liegt. Der Entscheider braucht damit nur nachzuschauen, ob sich der aktuelle *gefilterte Gütewert* oberhalb oder unterhalb der Abrissschwelle befindet. Befindet sich der Wert beispielsweise oberhalb der Abrissschwelle, und wird ein Abrisszeitpunkt in der Zukunft errechnet, liegt ein fallender Verlauf vor und es kann zu einem Abriss kommen.

Durch Verknüpfung des Zeitdifferenzdiagramms (Abbildung 6.6) mit dem Verlauf des

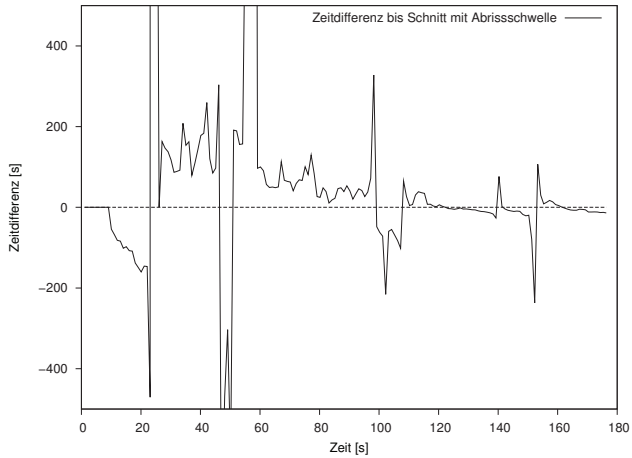

Abbildung 6.6: Im Zeitdifferenzdiagramm ist zu jeden Zeitpunkt die zeitliche Differenz zum errechneten Schnittpunkt zwischen Ausgleichsgerade und Abrissschwelle aufgetragen. In den ersten 10 Sekunden war der minimale Füllstand noch nicht erreicht, und damit keine sinnvolle Prognose möglich.

gefilterten Linkgütewerts (Abbildung 6.4) lässt sich ableiten, ob und wann ein Abriss des Netzzugangs droht. Abbildung 6.7 zeigt das Ergebnis dieser Verknüpfung und wird als „Restzeitdiagramm“ bezeichnet. In diesem wurden zudem alle zeitlichen Prognosen auf einen maximalen Wert von 60 Sekunden begrenzt. Ein Wert von Null steht dabei für einen momentan nicht nutzbaren Link. Ist ein Link benutzbar, und ist kein drohender Abriss erkennbar, lautet das Ergebnis ebenfalls 60 Sekunden.

6.2.4.7 Ableitung einer prädiktiven Handoverentscheidung

Das Restzeitdiagramm liefert eine Aussage, ob ein aktiver Netzzugang von einem Abriss bedroht ist, und falls ja, wieviel Zeit noch verbleibt. Hat man sich bereits für einen bestimmten Ersatzlink entschieden, kann dessen Assoziationszeit mit der verbleibenden Restzeit verglichen werden. Spätestens wenn die verbleibende Restzeit geringer ist als die Assoziationsdauer des Ersatzlinks, sollte der Ersatzlink angefordert werden. Im Restzeitdiagramm 6.7 ist hierfür exemplarisch bei einer Restzeit von 22 Sekunden eine Schwelle eingezeichnet. Diese stellt die vom Ersatzlink benötigte Assoziationszeit, welche für GPRS in [Ever04] mit 22 Sekunden ermittelt wurde, dar. Hier wurde beispielhaft mit diesem Wert gerechnet, jedoch sollten in der Praxis zu diesem Wert noch weitere

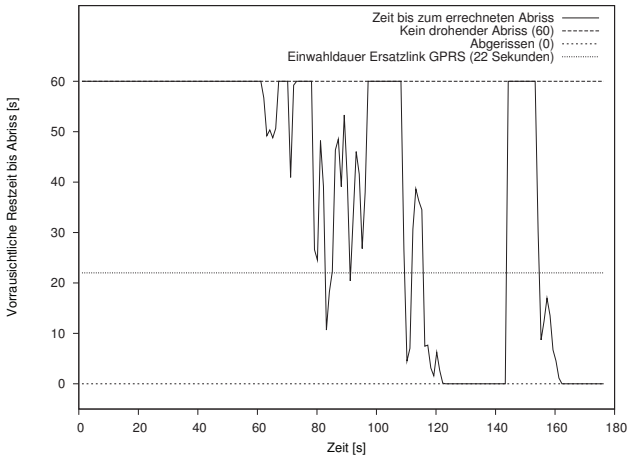

Abbildung 6.7: Das Restzeitdiagramm liefert eine Aussage, ob und wann mit einem Abriss des Hauptlinks gerechnet werden muss. Es kam lineare Regressionsrechnung zum Einsatz. Die horizontale Schwelle bei 22 Sekunden stellt die Assoziationszeit eines exemplarischen Ersatzlinks dar.

Aufschläge hinzuaddiert werden, um beispielsweise beim „weicheren Handover“ noch genügend Zeit zu haben, eine redundante Datenübertragung über den „frisch“ aufgebauten Ersatzlink durchführen zu können.

Damit ist geklärt, wann der prädiktive Handoverentscheider von REACH das Kommando zur Assoziation eines Ersatzlinks fällt. Dieses Schema beantwortet allerdings noch nicht die Frage, wann ein ehemaliger Hauptlink wieder als benutzbar gilt und wann der Ersatzlink wieder abgebaut werden kann.

Hierfür bietet es sich an, wie im Fall des reaktiven Entscheiders (siehe Abschnitt 6.2.2) zu verfahren, allerdings dabei den *gefilterten* Signalstärke- bzw. Linkgütewert heranzuziehen. Überschreitet der gefilterte Wert die Eignungsschwelle, darf der Hauptlink wieder benutzt werden (Hystereseverhalten). Überschreitet er die Idealschwelle, kann zudem der Ersatzlink wieder abgebaut werden. Vor Abbau des Ersatzlinks sollte wenn möglich sichergestellt werden, dass kein erneuter Abriss durch einen fallenden Verlauf droht.

Unter Anwendung dieser Regeln ergibt sich das in Abbildung 6.8 dargestellte Ersatzlinknutzungsdiagramm. Es zeigt das Ergebnis des prädiktiven Handoverentscheiders in Bezug auf einen bestimmten Ersatzlink, in diesem Fall ein simulierter Ersatzlink mit einer angenommenen Assoziationszeit von 22 Sekunden. Zustand **0** repräsentiert einen

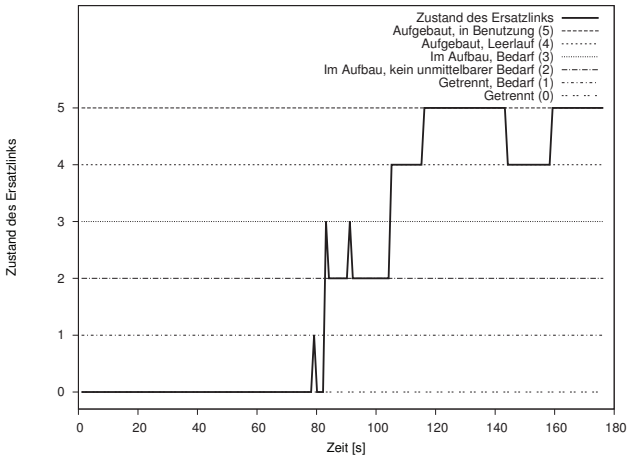

Abbildung 6.8: Das Ersatzlinknutzungsdiagramm verdeutlicht das Ergebnis des prädiktiven Handoverentscheiders von REACH.

nicht assoziierten Ersatzlink, in Zustand **2** befindet er sich im Aufbau und in Zustand **5** gilt er als verfügbar und er wird zur Datenübertragung herangezogen. Diese Zustände gelten als erstrebenswert, und deuten auf eine korrekte Handoverentscheidung hin. Die restlichen Zustände sind dagegen zu vermeiden. Zustand **1** bedeutet, dass der Ersatzlink im Moment benötigt wird, allerdings noch nicht angefordert worden ist. In Zustand **3** wird der Ersatzlink ebenfalls benötigt, er befindet sich allerdings noch im Assoziationsprozess. Es ist anzunehmen, dass das Assoziationskommando zu spät erfolgte. In Zustand **4** steht ein vollständig assoziierter Ersatzlink bereit, allerdings befindet er sich im Leerlauf. Dieser Zustand kann jedoch als postiv bewertet werden, falls der Handoverentscheider wie zum Zeitpunkt 150s erkannt hat, dass der Ersatzlink in Zukunft wieder benötigt werden könnte. Dann wäre eine zwischenzeitliche Trennung kontraproduktiv gewesen.

Im Anhang D ab Seite 337 wurde der hier vorgestelle prädiktive Handoverentscheider anhand von zehn Messreihen untersucht. Es werden der Unterschied zwischen exponentieller, inversquadratischer und linearer Regressionsrechnung verdeutlicht und auf die Auswirkungen verschiedener Erfassungsraten und Fenstergrößen eingegangen. Jede Messreihe wird diskutiert, und ein abschließendes Fazit fasst die gewonnenen Erkenntnisse zusammen. Es kam heraus, dass für die gewählte Kombination aus Hard- und Software bereits die lineare Regressionsrechnung brauchbare Ergebnisse liefert. Der Autor emp-

fielt eine Fenstergröße von 20 Sekunden bei einem minimalen Füllstand von 10 Sekunden und einer Messwerterfassungsrate von 10 Messwerten pro Sekunde. Dabei wurden sowohl innerhalb als auch außerhalb von Gebäuden überzeugende Prognosen errechnet. Zudem wurden durchgängig die in Abbildung 6.2 vorgestellten Konfidenzparameter verwendet.

Um das im Anhang dargelegte Fazit vorwegzunehmen, wertet der Autor die vorgestellte und umgesetzte Idee zur Anwendung von Regressionsrechnung für eine Prognose zukünftiger Abrisse als vollen Erfolg. Die Assoziation des Ersatzlinks wurde in allen Fällen wenigstens ein paar Sekunden vor Abriss des Hauptlinks angestoßen, was einen Zeitvorteil gegenüber einem reaktiv arbeitenden Entscheider darstellt. In machen Fällen wurde der Zeitpunkt sogar genau getroffen. „Weiche“ und „weichere Handover“ sind damit auch ohne einen „gierig“ arbeitenden Entscheider realisierbar. Das gilt allerdings nur, wenn der aktive Link auswertbare Signalstärke- oder Linkgütewerte liefert. Steht als Hauptlink beispielsweise drahtgebundenes Ethernet zur Verfügung, *muss* der Ersatzlink entweder reaktiv aufgebaut werden (nur „harte Handover“ möglich) oder er müsste vorsorglich die gesamte Zeit über vorgehalten werden („gieriges“ Verhalten, dafür wären „weiche“ und „weichere Handover“ möglich).

6.3 Netzzugangsgeräte und Netzzugangspunkte

Der folgende Abschnitt befasst sich mit der Verwaltung von Netzzugangsgeräten und Netzzugangspunkten durch REACH. Eine besondere Herausforderung besteht darin, dass sowohl die Eigenschaften der Netzzugangsgeräte als auch die Eigenschaften der möglichen Netzzugangspunkte mit in die Auswahlentscheidung einbezogen werden müssen. Eine Betrachtung der Geräteeigenschaften alleine reicht nicht aus, da hierbei keine Tarifinformationen vorliegen würden. Stattdessen lediglich die Netzzugangspunkte („Provider“) zu bewerten hieße dagegen, gerätespezifische Eigenschaften wie Bitraten und Latenzzeiten nicht berücksichtigen zu können. Es musste also ein Auswahlmechanismus gefunden werden, welcher die Eigenschaften von Netzzugangsgeräten und Netzzugangspunkten gemeinsam betrachtet. Gesucht wurde eine abstrakte Darstellung, welche einen Vergleich unterschiedlichster Technologien und Provider untereinander ermöglicht.

Ein zusätzliches Problem besteht darin, dass sich die Ansteuerung von Netzzugangsgeräten von Betriebssystem zu Betriebssystem unterscheidet. Daher wird bei REACH die Ansteuerung von Netzzugangsgeräten losgelöst vom technologieunabhängigen Kern durchgeführt. So kann ein und derselbe Handoverentscheider für unterschiedliche Betriebssysteme kompiliert und benutzt werden. Lediglich so genannte „Device Backends“ müssen passend für die jeweils vorliegende Zielplattform erstellt werden.

6.3.1 Repräsentation von Netzzugangsgeräten

Bezüglich eines jeden mobilen Endgeräts muss eine Vielzahl an Netzzugangsgeräten verwaltet werden. REACH setzt dazu „Device Backends“ ein, welche eigenständige Programme darstellen, die jeweils mit Blick auf bestimmte Netzzugangstechnologie entwickelt worden sind. Diese „Device Backends“ werden vom technologieunabhängig implementierten REACH-Client gestartet und verwaltet. Der REACH-Client selbst weist keinerlei Funktionalität auf, um mit einem bestimmten Netzzugangstechnologie umzugehen. Er ist lediglich in der Lage, uniforme „Device Backends“ zu verwalten.

Für jedes gestartete „Device Backend“ wird intern im REACH-Client ein so genanntes „Device Entry“-Objekt angelegt, welches die spezifischen Eigenschaften des verwalteten Netzzugangsgeräts für den Handoverentscheider vorhält und zusätzlich für die Kommunikation mit dem „Device Backend“ verantwortlich ist (siehe Abbildung 6.9). Details bezüglich der Kommunikation zwischen „Device Backend“ und „Device Entry“-Objekt sowie auf die Interna der implementierten „Device Backends“ sind in Anhang B verfügbar. Die Aufgaben eines „Device Entry“-Objekts umfassen:

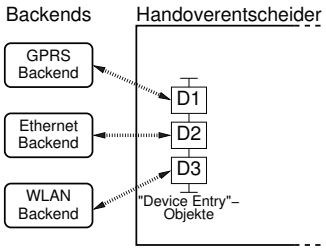

Abbildung 6.9: Jedes gestartete „Device Backend“ wird im REACH-Client durch ein „Device Entry“-Objekt repräsentiert. Da „Device Backends“ eigenständige Programme darstellen, gibt es einen Kommunikationskanal zum zugeordneten „Device Entry“-Objekt.

- **Uniforme Repräsentation eines Netzzugangsgeräts:**
  REACH muss unterschiedlichste Netzzugangsgeräte verwalten können, wobei die eigentliche Auswahlentscheidung durch einen technologieunabhängigen Handoverentscheider erfolgen soll. Dieser arbeitet mit einer Menge an „Device Entry“-Objekten, welche jeweils die Eigenschaften eines durch ein „Device Backend“ verwalteten Netzzugangsgeräts vorhalten. Zu diesen Eigenschaften gehören auch Statusinformationen, sodass unter anderem die folgenden Informationen bereit gestellt werden:

  - aktueller Assoziationsstatus,
  - Signalstärke- oder Linkgütewerte (falls verfügbar) sowie gegebenenfalls auch das Ergebnis der Regressionsrechnung,
  - technische Parameter, wie die momentan ausgehandelte Bitrate, die zu erwartende Latenzzeit oder der Energieverbrauch
  - aber auch die aktuelle IP-Adresse und weitere Konfigurationsdaten, die dem Netzzugangsgerät im Rahmen einer erfolgreichen Assoziationsprozedur zugewiesen worden sind.

- **Kommunikation mit einem „Device Backend“:**
  „Device Backends“ stellen eigenständige Programme dar und können damit nicht direkt vom REACH-Client manipuliert werden. Stattdessen sind zum Austausch von Steuerungs- und Statusinformationen Methoden der *Interprozesskommunikation* erforderlich. REACH benutzt hierfür ein aus XML-Nachrichten bestehendes Protokoll, welches über die Standardein- und Standardausgabe eines jeden „Device Backends“ „gesprochen“ wird. Ein solches XML-basiertes Protokoll erlaubt die Erstellung übersichtlicher Instanzen und ist sehr gut geeignet für Fehlersuche und Komponententests. Details zu diesem Protokoll, welches für alle „Device Backends“ identisch ist, werden in Anhang B.1 dargelegt. Der Protokollablauf bleibt für den Handoverentscheider, welcher lediglich als Nutzer des „Device Entry“-Objekts in Erscheinung tritt, unsichtbar.

- **Statusmaschine:**
  Sollte der Handovermanager mehrmals hintereinander einen identischen Befehl an ein „Device Entry“-Objekt übergeben, wird das Kommando gefiltert und somit nicht bis zum „Device Backend“ durchgereicht. Ein Beispiel hierfür wäre, dass ein doppelter Einwahlbefehl „geblockt“ wird, wenn das „Device Backend“ bereits dabei ist eine Einwahl durchzuführen.

Erstellung von „Device Entry“-Objekten

„Device Backends“ können auf zwei Arten geladen werden. Wenn der REACH-Client gestartet wird, liest er hierfür die Konfigurationsdatei `devices.xml` ein. „Device Backends“ können aber auch später, also bereits zur Laufzeit des REACH-Clients, vom Nutzer angefordert werden. Dazu muss der Nutzer den Dateinamen des „Device Backends“ benennen, sodass dieses vom REACH-Client gestartet werden kann. Konnte es gestartet werden, muss es noch parametriert werden. So benötigt jedes „Device Backend“ einen Gerätenamen, damit es weiß, für welches Netzzugangsgerät es verantwortlich ist.

```
<DeviceEntries>
  <DeviceEntry label="ZD1211 USB">
    <CoreParameters>
      <Item key="description" value="USB-WLAN-Stick" />
      <Item key="activation" value="true" />
      <Item key="backendfilename"
            value="/usr/local/bin/reachplugin-device-wlan" />
      <Item key="timespan_association_ms" value="1500" />
      <Item key="timespan_dissociation_ms" value="1000" />
      <Item key="break_threshold" value="18" />
      <Item key="usage_threshold" value="22" />
      <Item key="ideal_threshold" value="30" />
      <Item key="regression_type" value="linear" />
      <Item key="regression_minsize" value="100" />
      <Item key="regression_maxsize" value="200" />
      <Item key="regression_emptyfactor" value="1.5" />
      <Item key="regression_maxconfidence" value="4.0" />
    </CoreParameters>
    <BackendParameters>
      <Item key="device" value="wlan0" />
      <Item key="driver" value="wext" />
      <Item key="measurement_type" value="fieldstrength" />
      <Item key="measurement_ms" value="100" />
    </BackendParameters>
  </DeviceEntry>
</DeviceEntries>
```

Abbildung 6.10: Die hier dargstellte Konfigurationsdatei `devices.xml` spezifiziert die Parameter zum starten des „Device Backends“ für WLAN.

Abbildung 6.10 zeigt den Aufbau der Konfigurationsdatei `devices.xml`, hier mit einem beispielhaften Parametersatz bezüglich des „Device Backends“ für WLAN. Jede `<DeviceEntry>`-Schachtelung steht für ein zu startendes „Device Backend“, welches durch ein „Device Entry“-Objekt zugänglich gemacht werden soll. Erwähnenswert ist hierbei, dass ein eindeutiges `label` definiert wird. Dieses wird benötigt, um wiederum aus anderen Konfigurationsdateien heraus hiesige „Device Backends“ referenzieren zu können.

Weiterhin enthält jede `<DeviceEntry>`-Schachtelung eine `<CoreParameters>`- sowie eine `<BackendParameters>`-Schachtelung. Beide enthalten Schlüssel-Werte-Paare, wobei die erste Schachtelung durch den REACH-Client ausgewertet wird, während letztere ausschließlich Parameter enthält, welche für das „Device Backend“ interessant sind.

Die `<CoreParameters>`-Parameter haben für alle „Device Backends“ denselben Um-

fang. Mit "description" ist eine textuelle Beschreibung verfügbar, "activation" dient als bool-Schalter zur Anzeige, ob das „Device Entry“-Objekt berücksichtigt werden soll, und der Parameter "backendfilename" spezifiziert den Dateinamen des zu startenden „Device Backends“. Es folgen die Parameter "timespan_association_ms" und "timespan_dissociation_ms", welche dem Handoverentscheider die zu erwartenden Einwahl- und Trennungsdauern mitteilen. Mit Hilfe der Parameter "break_threshold", "usage_threshold" und "ideal_threshold" werden die Abrissschwelle, die Eignungsschwelle sowie die Idealschwelle für das vorliegende Netzzugangsgerät definiert (siehe Abschnitt 6.2.2). Es folgen fünf Parameter bezüglich des prädiktiv arbeitenden Handoverentscheiders, wobei mit Hilfe des Parameters "regression_type" festgelegt werden kann, ob und welche Regressionsrechnung durchgeführt werden soll ("invsqr", "exp", "linear" oder "none"). Zudem können der Umfang des Regressionsfensters und dessen minimaler Füllstand sowie der Leerungsfaktor in Verbindung mit dem maximalen Konfidenzfaktor für die Ausreißerbehandlung spezifiziert werden. Die Bedeutung dieser Regressionsparameter wurde bereits umfänglich in Abschnitt 6.2.4 betrachtet. Wird keine Regressionsrechnung benötigt, können die zugehörigen Parameter weggelassen werden.

Die in der <BackendParameters>-Schachtelung möglichen Schlüssel sind dagegen abhängig vom „Device Backend“ und werden von diesem spezifiziert. Hier lassen sich technologiespezifische Parameter unterbringen, welche jeweils nur für ein bestimmtes „Device Backend“ von Relevanz sind. Wenn ein „Device Backend“ gestartet wird, übergibt es die erforderlichen Schlüssel in Form einer „Schablone“ an den REACH-Client. Sie können dann entweder noch während des Auslesens der Konfigurationsdatei gesetzt, oder durch den Nutzer über eine grafische Oberfläche spezifiziert und geändert werden.

6.3.2 Das „Providerkonzept“ von REACH

Im letzten Abschnitt wurde beschrieben, dass ein Netzzugangsgerät („Device“) durch genau ein „Device Backend“ verwaltet wird, und dabei jedes „Device Backend“ innerhalb des REACH-Clients durch ein „Device Entry“-Objekt repräsentiert wird.

Was allerdings noch fehlt, ist eine Betrachtung der Netzzugangspunkte. Damit ist gemeint, dass ein Netzzugangsgerät nur dann „aktiviert“ werden kann, wenn auch ein Netzzugang möglich ist, sich also ein Netzzugangspunkt in Reichweite befindet. Netzzugangspunkte müssen für drahtlose wie drahtgebundene Technologien gleichermaßen definiert werden, denn erst durch Betrachtung des Netzzugangspunkts können auch Parameter wie Tarifinformationen mit in die Handoverentscheidung einbezogen werden. Die folgenden Überlegungen führten schlussendlich zu dem so genannten „Providerkonzept“, welches Netzzugangsgeräte und Netzzugangspunkte miteinander verknüpft und

eine Liste an Netzzugängen liefert. Es basiert auf den folgenden Aussagen:

- **Mehrere Netzzugangspunkte pro Netzzugangsgerät:**
  Ein bestimmtes Netzzugangsgerät eines mobilen Endgeräts kann sich zu einer Vielzahl an Netzzugangspunkten verbinden bzw. assoziieren. Ein Beispiel wäre eine WLAN-Schnittstelle, welche sich mit unterschiedlichen Basisstationen assoziieren kann. Pro Basisstation können unterschiedliche Authentifizierungsinformationen erforderlich sein, aber auch voneinander abweichende Tarife eine Rolle spielen. Alle „Möglichkeiten" sehen jedoch die Verwendung desselben „Devices" vor, involvieren also im Sinne von REACH dasselbe „Device Entry"-Objekt und damit ein „Device Backend".

- **Statusinformationen pro Netzzugangspunkt:**
  Besonders bei drahtlos arbeitenden Netzzugangstechnologien kann es passieren, dass mehrere alternative Netzzugangspunkte erkannt werden und für einen Netzzugang herangezogen werden können. Bei WLAN geschieht dies durch periodisches „scannen", also durch eine Suche nach Basisstationen in der unmittelbaren Umgebung des mobilen Endgeräts. Nicht alle gefundenen Netzzugänge sind jedoch relevant, da beispielsweise bezüglich unbekannter Netzzugangspunkte keine Authentifizierungsinformationen vorliegen. Zudem lassen sich bei manchen Technologien Signalstärke- oder Linkgütewerte ermitteln, sodass bei einer Vielzahl an möglichen Netzzugängen deren Eignung abgeschätzt werden kann.

- **Maximal *ein* aktiver Netzzugang pro Netzzugangsgerät:**
  Obwohl ein Netzzugangsgerät mit mehreren Netzzugangspunkten assoziiert werden kann, wurde für REACH die Einschränkung getroffen, dass zu jedem Zeitpunkt ein Netzzugangsgerät nur mit maximal einem Netzzugangspunkt assoziiert sein kann. Ein Ethernetanschluss kann zeitgleich nicht an *mehrere* LANs angeschlossen werden, eine WLAN-Schnittstelle nur mit maximal mit einer Basisstation assoziert sein und ein Modem nur zu einer Rufnummer gleichzeitig eine Einwahlverbindung halten. Für ISDN bedeutet dies jedoch eine Einschränkung, da man die zwei „B-Kanäle" („Bearer Channels") theoretisch zu unterschiedlichen Rufnummern verbinden könnte. REACH erlaubt dies zwar nicht, allerdings sieht der Autor darin keine praxisrelevanten Nachteile, zumal ISDN-Kanalbündelungen nicht betroffen sind.

  Steht einem mobilen Endgerät jedoch dieselbe Netzzugangstechnologie mehrfach zur Verfügung (zwei WLAN-Transceiver), muss dies durch mehrfachen Start desselben „Device Backends" gelöst werden. Jedes „Device Backend" wäre dann wieder nur für ein einziges Netzzugangsgerät verantwortlich.

- **Pro Netzzugangspunkt nur ein Netzzugangsgerät:**
  Ein Netzzugangspunkt wird durch eine Menge an Daten repräsentiert, welche beispielsweise Tarifinformationen mit einbeziehen. Zudem gehören zu jedem Netzzugangspunkt Konfigurationsparameter bezüglich seiner Erkennung, und falls erforderlich, zur Authentifizierung. Im Beispiel von WLAN können bekannte Basisstationen anhand ihres „Service Set Identifiers“ (SSID) erkannt werden. Entschließt sich der Handoverentscheider für eine Assoziation mit dieser Basisstation, werden schließlich die Authentifizierungsparameter benötigt. Diese können einen geheimen Schlüssel umfassen oder beispielsweise aus einem Namen, einem Passwort und einem Zertifikat bestehen. Die Möglichkeiten sind vielfältig, und werden vom jeweiligen „Device Backend“ vorgegeben.

  Eine „Netzzugangsbeschreibung“ bezüglich eines Netzzugangspunkts basiert also *immer* auf einer bestimmten Technologie, repräsentiert durch einen Parametersatz („Schablone“), welcher durch ein „Device Backend“ vorgegeben wird. Informationen bezüglich der Einwahlmöglichkeiten eines Analogmodems können nicht gleichzeitig auch mit einer WLAN-Karte verknüpft werden, da die jeweiligen Parameter zu unterschiedlich wären.

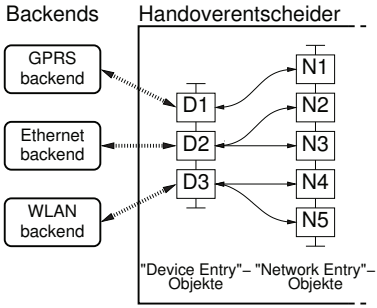

Abbildung 6.11: Ein Netzzugang wird durch ein „Network Entry“-Objekt, welches auf ein „Device Entry“-Objekt (Netzzugangsgerät) verweist, dargestellt.

Repräsentation von Netzzugangspunkten

Der dargelegte Sachverhalt spiegelt sich im Aufbau des Handoverentscheiders wider. Hierzu wurde der Liste der „Device Entry“-Objekte (für die Netzzugangsgeräte) eine zweite Liste zur Seite gestellt (siehe Abbildung 6.11). Sie enthält so genannte „Network Entry“-Objekte, wobei jedes dieser Objekte einen bekannten Netzzugangspunkt beschreibt und

einem bestimmten „Device Entry“-Objekt zugeordnet ist. Ein jedes „Network Entry“-Objekt beschreibt damit genau einen Netzzugang, mit dessen Hilfe sich ein mobiles Endgerät mit dem Internet verbinden kann. Es ist möglich, dass mehrere „Network Entry“-Objekte ein und dasselbe „Device Entry“-Objekt referenzieren, was sinngemäß bedeutet, dass zu einem bestimmten Netzzugangsgerät mehrere alternative Netzzugangspunkte bekannt wären

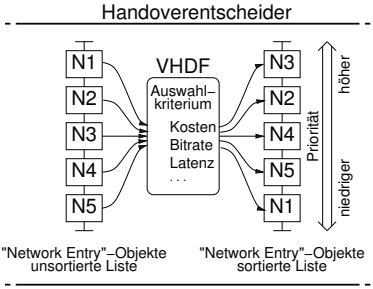

Abbildung 6.12: Die Liste der „Network Entry“-Objekte (Netzzugänge) wird anhand einer nutzerseitig zu wählenden Zielstellung sortiert.

Ablauf der Handoverentscheidung

Es ergibt sich somit eine Kombination aus Informationen bezüglich eines Netzzugangsgeräts und eines Netzzugangspunkts, welche vom Handoverentscheider berücksichtigt wird. Die im Abschnitt 6.1.1.3 vorgestellte „Vertical Handoff Decision Function“ (VHDF) arbeitet mit der Liste der „Network Entry“-Objekte, und erstellt anhand des nutzerseitig gewählten Sortierkriteriums eine sortierte Liste (siehe Abbildung 6.12).

Anhand der gemäß der Zielstellung sortierte Liste an „Network Entry“-Objekten wird die eigentliche Handoverentscheidung durchgeführt. Der Handoverentscheider beginnt wie in Abbildung 6.13 mit dem „Network Entry“-Objekt mit der besten Eignung (höchste Priorität). Er ermittelt den Status des Netzzugangspunkts und des Netzzugangsgeräts und kann schließlich einen Assoziations- oder Dissoziationsbefehl bezüglich des vorliegenden Netzzugangs äußern. Je nach gewünschtem Nutzungsschema kann versucht werden, einen oder mehrere Netzzugänge aufzubauen. Der Handoverentscheider „hangelt“ sich dabei in jedem Entscheidungszyklus vom „geeignetsten“ in Richtung der „weniger geeigneten“ Netzzugänge, und kann dabei auch nicht länger benötigte Netzzugänge wieder freigeben.

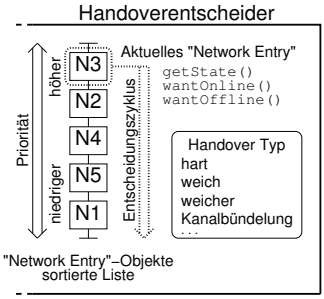

Abbildung 6.13: Die Handoverentscheidung wird anhand der sortierten Liste der „Network Entry“-Objekte (Netzzugänge) durchgeführt.

Erstellung von „Network Entry“-Objekten

Die Instantiierung von „Network Entry“-Objekten erfolgt vergleichbar mit der bereits beschriebenen Entstehung von „Device Entry“-Objekten. Sie können entweder während der Startphase des REACH-Clients anhand einer Konfigurationsdatei erstellt, oder während der Laufzeit durch den Nutzer über eine grafische Oberfläche angefordert und parametriert werden.

```
<NetworkEntries>
  <NetworkEntry label="WLAN REACH AP 1">
    <CoreParameters>
      <Item key="description" value="WLAN ueber linken REACH-AP" />
      <Item key="deviceentrylabel" value="ZD1211 USB" />
      <Item key="activation" value="true" />
      <Item key="priority" value="3" />
    </CoreParameters>
    <BackendParameters>
      <Item key="SSID" value="REACH Access Point 1" />
      <Item key="supplicant_configfile"
            value="/home/reach/reachclient/supplicant_reach1.conf" />
    </BackendParameters>
  </NetworkEntry>
</NetworkEntries>
```

Abbildung 6.14: Die hier dargstellte Konfigurationsdatei `networks.xml` spezifiziert die verfügbaren Netzzugänge, hier beispielhaft für ein „Network Entry“-Objekt, welches eine WLAN-Schnittstelle involviert.

Die Netzzugangsmöglichkeiten werden in der Konfigurationsdatei `networks.xml` (siehe Abbildung 6.14) gespeichert, welche eine vergleichbare Struktur aufweist wie die bereits vorgestellte Konfigurationsdatei `devices.xml`. So findet man hier eine Menge an `<NetworkEntry>`-Schachtelungen, wobei jede Schachtelung für ein „Network Entry“-Objekt steht. Jedes dieser Objekte weist ein eindeutiges `label` auf, damit „Network Entry“-Objekte auch von anderen Komponenten referenziert werden können (Erkennung des Aufenthaltsorts anhand aktiver Netzzugänge, siehe „Zonenkonzept“ in Kapitel 7.2.2). Auch hier finden sich wieder die Schachtelungen `<CoreParameters>` und `<BackendParameters>`, wobei erstere wiederum ausschließlich intern vom REACH-Client ausgewertet wird und letztere über das zugewiesene „Device Entry“-Objekt an das „Device Backend“ „durchgereicht“ werden.

Der Umfang der `<CoreParameters>` ist für *alle* aufgeführten „Network Entry“-Objekte identisch. Sie umfassen eine textuelle Beschreibung `"description"`, den Identifikator `"deviceentrylabel"` eines zugehörigen „Device Entry“-Objekts sowie einen `bool`-Schalter `"activation"` bezüglich der Berücksichtigung durch den Handoverentscheider. Mittels des `"priority"`-Parameters kann nutzerseitig die Reihenfolge der sortierten Liste beeinflusst werden.

Die `<BackendParameters>` sind wiederum vom zugeordneten „Network Entry“-Objekt und damit von einem „Device Backend“ abhängig. Sie umfassen zur Detektion notwenige Parameter und gegebenenfalls noch Authentifizierungsinformationen. Der Umfang hängt alleinig von der zu Grunde liegenden Technologie und deren Betriebsmodus ab. Daher übergibt ein „Device Backend“ nach seinem Start eine Schablone an netzzugangsspezifischen Detektionsparametern an das „Device Entry“-Objekt, welches diese Schablone dann den sich zuordnenden „Network Entry“-Objekten zur Verfügung stellt. Somit kann ein Nutzer auch zur Laufzeit über eine Bedienschnittstelle weitere Netzzugänge hinzufügen. Auf konkrete Parametersammlungen wird in Anhang B.2 eingegangen, wo die für den Demonstrator umgesetzten „Device Backends“ vorgestellt werden.

An dieser Stelle wurde erklärt, wie der REACH-Client sowohl unterschiedliche Netzzugangsgeräte als auch eine Vielzahl an Netzzugangspunkten mit ihren zugehörigen Eigenschaften und Zugangsinformationen in einer universellen Form verwaltet. Diese Struktur wurde als das „Providerkonzept“ eingeführt, da hierbei das bislang offene Problem gelöst wurde, dass sich ein Netzzugangsgerät über verschiedene „Provider“ – also Netzzugangspunkte – mit dem Internet verbinden kann. Die bislang verfügbaren Network- und Handovermanager bieten ein solches „Providerkonzept“ nicht an. Sie sind nicht in der Lage, derart unterschiedliche Netzzugänge miteinander vergleichbar zu machen.

6.3.3 Aufgaben der „Device Backends“

„Device Backends“ stellen wie bereits erläutert vollwertige Programme dar, welche vom REACH-Client mit Systemverwaltungsrechten gestartet und verwaltet werden. Alle „Device Backends“ werden auf dieselbe Art und Weise behandelt, unabhängig davon, für welche Netzzugangstechnologie sie zuständig sind. Sie weisen allesamt dieselbe Schnittstelle auf, und können vom REACH-Client als eine Menge „abstrakter Netzzugangsgeräte“ betrachtet werden.

Im momentanen Ausbauzustand existieren drei lauffähige „Device Backends“ für die GNU/Linux-Plattform. Sie umfassen die Technologien „Ethernet“, „WLAN“ nach IEEE 802.11 (a, b und g) [Toss09] sowie die Nutzung von GPRS-fähigen Mobiltelefonen, welche über Bluetooth mit dem mobilen Endgerät verbunden sind [Hoff08, Hoff10]. Die Aufgaben eines „Device Backends“ umfassen:

- die Detektion des überwachten Netzzugangsgeräts,
- die Erkennung potentieller Netzzugangspunkte,
- sowie den Auf- und Abbau von Assoziationen zu „sichtbaren“ Netzzugangspunkten.

Die Kommunikation zwischen einem „Device Backend“ und dem Handoverentscheider läuft so ab, dass textuelle Nachrichten über die Standardein- und ausgabe des „Device Backends“ ausgetauscht werden. Das bedeutet, man könnte ein jedes „Device Backend“ testweise manuell über die Kommandozeile starten, und direkt die Meldungen des „Device Backends“ auf dem Bildschirm lesen sowie über die Tastatur Nachrichten übergeben. Das begünstigt die Durchführung von Komponententests, da man die möglichen Nachrichten in einem Texteditor sammeln und per Ausschneiden und Einfügen an ein „Device Backend“ senden kann. Anhand dessen Ausgaben kann dann geprüft werden, ob das beobachtete Verhalten mit der erwarteten Reaktion übereinstimmt.

Es wurde jedoch noch nicht beschrieben, wie die auszutauschenden Informationen strukturiert sind. Hierfür wurde ein Protokoll entworfen, welches mit auf XML-basierenden Nachrichten arbeitet. Solche Nachrichten sind sowohl menschen- aus auch maschinenlesbar und erlauben damit sowohl die manuelle Sichtung des Protokollflusses als auch eine übersichtliche Programmierung der Protokollinstanzen. Die Nachrichten wurden gemäß der zu übertragenden Befehle und Statusinformationen entworfen. Ein beispielhafter Protokollablauf wird in Anhang B.1 dargeboten. Dort werden die einzelnen Nachrichten und ihre Parameter anhand abgefangener Daten erläutert. Dem interessierten Leser sei zudem Anhang B.2 empfohlen, welches die Arbeitsweise der drei umgesetzten „Device Backends“ umfassend erläutert.

6.4 Pflege der Routingtabellen

Die Ansteuerung von Netzzugangsgeräten mittels der vorgestellten „Device Backends“ ist allerdings nur „die halbe Miete“, was den Zugriff auf betriebssystemnahe Mechanismen angeht. Spätestens sobald dem REACH-Client mehrere assoziierte Netzzugänge zur Verfügung stehen, wird das bereits in Abschnitt 5.5.5 diskutierte Wegewahlproblem deutlich. Es ist nicht möglich, allein mit Hilfe von „Default Routen“ eine deterministische Wegewahl durchzuführen. Zudem wurde in Kapitel 4.3.2.2 bereits dargelegt, dass es in Bezug auf das „Relay Plugin“ für „Transparente Proxyserver“ zu Problemen kommen kann, wenn Datenstöme über automatisch zugewiesene „Default Routen“ verschickt werden.

REACH musste daher so entwickelt werden, dass keine Einträge bezüglich „Default Routen“ benötigt werden. Das hatte aber zur Konsequenz, dass zu jedem REACH-Proxyserver dedizierte Routen in den Routingtabellen hinterlegt werden müssen. Diese betriebssystemnahe Aufgabe wird vom „Routing Backend“ übernommen.

6.4.1 Das „Routing Backend“

Der REACH-Client soll sich für unterschiedlichste Zielsysteme eignen, und besteht daher aus einem plattformunabhängigen „Kern“. Alle plattformspezifischen Mechanismen werden in so genannte „Backends“ ausgelagert, was für den Fall der Ansteuerung von Netzzugangsgeräten bereits dargelegt worden ist. In Bezug auf die Einflussnahme auf die Wegewahl zeigt sich ein ähnliches Problem, denn die Mechanismen zur Pflege von Routingtabellen sind ebenfalls abhängig vom Betriebssystem. Beispielsweise ist quelladressbasierte Wegewahl nicht auf jedem GNU/Linux-System verfügbar, da es einer besonderen Konfiguration des Kernels bedarf. Zudem müssen hierfür spezielle Systemkommandos verfügbar sein, welche mit „Policy Routing“ umgehen können. Ist die entsprechende Unterstützung nicht im Kernel vorhanden, oder fehlen die notwendigen Systemkommandos, oder läuft der REACH-Client gar auf einem ganz anderen Betriebssystem, wäre keine quelladressbasierte Wegewahl verfügbar.

Gewünscht wurde daher ein „Routing Backend“, welches für alle Belange der Wegewahl zuständig ist. Es sollte über eine plattformunabhängige Schnittstelle verfügen und sämtliche betriebssystemspezifische Aspekte verbergen. Für jede Zielplattform braucht dann lediglich ein solches „Routing Backend“ erstellt zu werden, und da dessen Schnittstelle festgeschrieben ist, kann der REACH-Client es problemlos benutzen.

Eine Herausforderung bestand im Entwurf des Kommunikationsprotokolls zwischen REACH-Client und „Routing-Backend“, da dieses sämtliche Routingschemata unterstüt-

zen muss. Es müssen Ereignisse über neu hinzugekommene und nicht länger verfügbare Netzzugangsgeräte sowie die den Schnittstellen zugewiesenen IP-Adressen kommuniziert werden. Des Weiteren müssen speziell für die zieladressbasierte wegewahl so genannte „Adressbindungen“ gepflegt werden, bei welchen die IP-Adresse eines REACH-Proxyservers mit einem Netzzugangsgerät des mobilen Endgeräts („einem Weg ins Internet“) verknüpft werden.

An dieser Stelle sei auf Anhang C.1 verwiesen. Dort werden die Aufgaben des „Routing Backends“ anhand eines beispielhaften Nachrichtenaustauschs vorgestellt. Im Anschluss wird in Anhang C.2 auf die spezifischen Ansteuerungsmechanismen eingegangen, welche für das im Rahmen dieser Ausarbeitung erstellte „Routing Backend“ unter GNU/Linux angewendet werden mussten.

6.4.2 Erreichbarkeit von DNS-Servern

Als der REACH-Client in Verbindung mit dem Campusnetz der TU Ilmenau getestet wurde, zeigte sich das Problem, dass die Namensauflösung plötzlich nicht mehr funktionierte. In vorherigen Testszenarien, bei welchen eigene WLAN-Basisstationen verwendet worden sind, trat dieses Problem nicht auf.

Es zeigte sich, dass die Server des „Domain Name Systems“ (DNS) nicht erreicht werden konnten, und daher die Namensauflösung fehlschlug. Die Ursache lag in der Wegewahl begründet, da bei REACH die „Default Routen“ der zentralen Routingtabelle ohne Funktion sind. Im Fall des Campusnetzes sind die DNS-Server jedoch nicht *direkt* erreichbar, sodass hierbei eine dedizierte Route gesetzt werden muss.

```
neelix ~ # /sbin/resolvconf -l wlan0
# resolv.conf from wlan0
# Generated by dhcpcd from wlan0
search rz.tu-ilmenau.de
nameserver 141.24.12.2
nameserver 141.24.4.1
```

Abbildung 6.15: Wird das Campusnetz der TU Ilmenau über WLAN betreten, bekommt das mobile Endgerät zwei DNS-Server zugewiesen. Deren IP-Adressen lassen sich mit Hilfe des Kommandos `resolvconf` in Erfahrung bringen.

Dieser Umstand muss von „Routing Backend“ automatisch erkannt werden, damit es in diesem Fall entsprechende Routen zu den DNS-Servern setzen kann. Zur Ermittlung wird unter GNU/Linux das Kommando `resolvconf` benutzt (siehe Abbildung 6.15). Es dient dazu, die IP-Adressen der DNS-Server, welche einem bestimmten Netzzugangs-

gerät zugeordnet worden sind, auszugeben. Die dargestellten DNS-Server stammen aus dem Campusnetz der TU Ilmenau.

Da zu jedem Interface die IP-Adresse und die Netzmaske bekannt sind (diese Informationen werden durch die „Device Backends“ bereit gestellt), kann das „Routing Backend“ jeden hinterlegten DNS-Server prüfen, ob er *direkt* erreichbar ist oder ob eine Route gesetzt werden muss.

```
neelix ~ # route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags  Metric  Iface
141.24.12.2     141.24.127.254  255.255.255.255 UGH    0       wlan0
141.24.4.1      141.24.127.254  255.255.255.255 UGH    0       wlan0
141.24.124.0    0.0.0.0         255.255.252.0   U      10000   wlan0
127.0.0.0       127.0.0.1       255.0.0.0       UG     0       lo
0.0.0.0         0.0.0.0         0.0.0.0         U      0       dummy0
0.0.0.0         141.24.127.254  0.0.0.0         UG     10000   wlan0
```

Abbildung 6.16: Die zentrale Routingtabelle enthält hier bereits zwei dedizierte Routen zu den beiden DNS-Servern des Campusnetzes. Sie wurden durch das „Routing Backend“ hinzugefügt.

In Abbildung 6.16 wird verdeutlicht, dass Routen zu zwei DNS-Servern gesetzt werden mussten. Die untersten zwei Zeilen spezifizieren zwei „Default Routen“, wobei auf Grund des geringeren Metrik-Werts lediglich der Eintrag bezüglich des `dummy0`-Interfaces eine Rolle spielt (Erklärung hierfür in Abschnitt 4.3.2.2 „Transparente Proxyserver“). Es ist jedoch erkennbar, dass die DNS-Server `141.24.12.2` und `141.24.4.1` (ermittelt wie in Abbildung 6.15) nicht *direkt* erreicht werden können, da sich das `wlan`-Interface im Subnetz `141.24.124.0/22` befindet. Es mussten somit zwei Routen hinzugefügt werden, welche explizit einen Weg über das Gateway `141.24.127.254` beschreiben.

Nachdem das „Routing Backend“ um die vorgestellte Funktionalität erweitert worden war, funktionierte REACH auch im Campusnetz der TU Ilmenau. Dem Handoverentscheider standen damit alle Werkzeuge zur Verfügung, um auf das Betriebssystem des mobilen Endgeräts einwirken zu können. Es fehlte lediglich noch eine Bedienschnittstelle, über welche der Nutzer neue Netzzugangsgeräte oder -punkte anmelden oder die Zielstellung des Handoverentscheiders ändern konnte.

6.5 Grafisches „Frontend“ als Bedienschnittstelle

Sowohl der REACH-Client als auch der REACH-Server sind als reine Kommandozeilenprogramme entwickelt worden, und bieten selbst keine grafische Bedienschnittstelle an.

Sie können daher „als Systemdienst“ im Hintergrund laufen („als Dämon“). Somit ist REACH auch auf Systemen einsetzbar, welche wie die REACH-Box ohne einen Bildschirm daherkommen. Zudem stellt sich auf Mehrbenutzersystemen das Problem nicht, *wer* den REACH-Client starten muss: er kann bereits automatisch als Systemdienst gestartet werden, und „gehört“ damit auch keinem der sich anmeldenden Benutzer. Dennoch ist es wünschenswert, dass den Nutzern eine Bedienschnittstelle zur Verfügung steht, über welche sie auf den REACH-Client Einfluss nehmen können.

6.5.1 Bedarf an einer grafischen Bedienschnittstelle

In [Ever04] wurde bereits auf den Bedarf an einer grafischen Bedienschnittstelle hingewiesen. Die als „Frontend“ bezeichnete Software wurde als ein „optionales Zusatzprogramm“ klassifiziert. Das bedeutet, dass für den „reinen Betrieb“ keine Bedienschnittstelle benötigt wird, denn sowohl der REACH-Client als auch der REACH-Server lassen sich bereits vollständig über Konfigurationsdateien parametrieren.

Sollen Konfigurationsparameter allerdings geändert werden, müsste ohne eine grafische Oberfläche der REACH-Client[2] zuerst beendet, und nach Abschluss der Anpassungen erneut gestartet werden. Mit Hilfe einer grafischen Oberfläche könnte allerdings auch „online“ auf den REACH-Client eingewirkt werden, ohne dass dieser zwischenzeitlich gestoppt werden müsste.

Zudem können der aktuelle Status des Handoverentscheiders, die gerade gewählte Zielstellung oder die Menge der momentan geladenen „Relay Plugins“ in Erfahrung gebracht werden. Anwender können auf die Handoverentscheidung Einfluss zu nehmen, ohne dass man ihnen das Recht einräumen muss, den REACH-Client starten und stoppen oder dessen Konfigurationsdateien editieren zu dürfen. Eine grafische Bedienschnittstelle für den vorgestellten Zweck wurde in der Studienarbeit [Zech06] untersucht und bereits prototypisch realisiert, während anschließend in [Dour07] untersucht wurde, wie sich Signalstärkeverläufe und Handoverereignisse grafisch in einem solchen „Frontend“ visualisieren lassen.

Als Herausforderung entpuppte sich das zu entwickelnde Datenaustauschschema zwischen „Frontend“ und REACH-Client, welches in der Lage sein musste, alle relevanten Kommandos und Statusinformationen zu übertragen.

[2]Diese Aussage bezieht sich auch auf den REACH-Server, jedoch wird im Folgenden bevorzugt der REACH-Client genannt. Nur auf diesen sollten Endanwender zugreifen können, während auf einen REACH-Server nur der jeweilige Administrator zugreifen wird.

6.5.2 Zugriff auf REACH-Client und -Server

Bei den vorgestellten Bedienschnittstellen handelt es sich um eigenständige Programme. Damit jedoch ein vom REACH-Client und -Server losgelöstes Programm Kommandos absetzen und den Status des Handoverentscheiders in Erfahrung bringen kann, muss eine Schnittstelle zum Datenaustausch vorgesehen werden. Sowohl der REACH-Client als auch der -Server bieten bereits mehrere Schnittstellen an, welche hierfür verwendet werden können:

- **XML-basierter Zugriff**: Es kann eine Portnummer spezifiziert werden, auf welcher REACH-Client und -Server auf eingehende TCP-Verbindungen warten. Über diese Verbindungen kommt ein selbst entworfenes Protokoll zum Einsatz, welches dem Austausch von Steuer- und Statusnachrichten dient. Die Nachrichten kommen als XML-Telegramme daher. Diese Schnittstelle ist für die Anbindung von „Frontends“ gedacht.

- **Kommandokonsole**: Es stand die Idee im Raum, einen TCP-basierten Wartungszugang anzubieten, welcher in Anlehung an einen „Telnet-basierten Zugang“ eine Kommandokonsole anbietet. Diese Idee wurde allerdings mangels Notwendigkeit bislang nicht zielführend verfolgt, sodass das rudimentär umgesetzte Zugriffsmodul nur wenige beispielhafte Kommandos unterstützt (`exit` und `shutdown`).

- **Debugausgaben**: Für Debugausgaben steht eine eigene Annahmestelle für TCP-Verbindungen bereit. Wenn man sich zu dieser Portnummer verbindet, erhält man einen Datenstrom mit Debugausgaben. Diese sind allerdings nur für den Programmierer interessant, können aber so auch „aus der Ferne“ abgerufen werden.

- **Webserver**: Die Idee einer browserbasierten Bedienschnittstelle kam dem Autor leider erst sehr spät, sodass in der aktuellen Ausbaustufe kein Zugriff mit Hilfe von HTTP angeboten wird. Der Vorteil hiervon wäre, dass man dann auch mit Hilfe von Endgeräten ohne ein installiertes „Frontend“ auf die Handoverentscheidung einwirken könnte. Man würde dazu lediglich einen Browser benötigen. In Szenarien, welche eine REACH-Box involvieren, würde dann zur Bedienung ein Smartphone mit installiertem Browser genügen.

Die Zugriffsschnittstellen des REACH-Clients und des REACH-Servers werden in der Konfigurationsdatei `controls.xml` spezifiziert (siehe Abbildung 6.17). Sie erlaubt bislang drei verschiedene Zugriffsmodule: mit „`debugoutput`“ werden Debugausgaben bereitgestellt, mit „`shell`“ soll eine Kommandokonsole angeboten werden und mit „`xml`“

```
<Controls>
  <Control name="debugoutput">
    <Listener port="6000" />
  </Control>
  <Control name="shell">
    <Listener port="6001" />
  </Control>
  <Control name="xml">
    <Listener port="6002" />
  </Control>
</Controls>
```

Abbildung 6.17: REACH-Client und REACH-Server bieten mehrere Schnittstellen zum Abruf von Statusinformationen und zur Steuerung an. Die Nachrichten werden über TCP-Verbindungen übertragen. Die Portnummern der zugehörigen „Annahmestellen“ müssen in der hier dargestellten Konfigurationsdatei `controls.xml` spezifiziert werden.

steht der für die „Frontends“ relevante XML-basierte Übertragungskanal zur Verfügung. Zu jedem Modul muss die Portnummer eines TCP-basierten `Listener`s angegeben werden, wobei wahlfrei eine freie unprivilegierte Portnummer ausgewählt werden muss.

6.5.3 Funktionsweise

Sowohl für den REACH-Client als auch für den REACH-Server wurde jeweils eine grafische Benutzeroberfläche („Frontend“) implementiert. Die „Frontends“ fragen nach ihrem Start nach der IP-Adresse und der Portnummer, unter welcher der REACH-Client bzw. der REACH-Server erreichbar sind. Im vorliegenden Beispiel (Konfigurationsdatei 6.17) kommt die willkürlich ausgewählte TCP-Portnummer `6002` zum Einsatz.

In Abbildung 6.18 wird ein Bildschirmfoto des „Frontends“ unter KDE 4.3.5 gezeigt, nachdem es sich mit dem REACH-Client verbinden konnte. Es gibt noch eine Version speziell für den REACH-Server, welche zwar ähnlich aussieht, aber keine handoverspezifischen Schaltflächen aufweist.

Die „Frontends“ wurden als reine Visualisierungsprogramme entworfen. Sie enthalten keine Steuerungslogik oder internene Zustände, sondern bekommen sämtliche Informationen vom REACH-Client bzw. vom REACH-Server geliefert. Wird ein Parameter durch den Nutzer geändert, vermerkt das Frontend diesen Zustandswechsel nicht, sondern schickt lediglich eine Änderungsmitteilung an den REACH-Client. Erst wenn dieser die Nachricht interpretieren konnte und es tatsächlich zu einem Zustandswechsel kam, wird das „Frontend“ eine diesbezügliche Änderungsmitteilung erhalten und dem Nutzer die

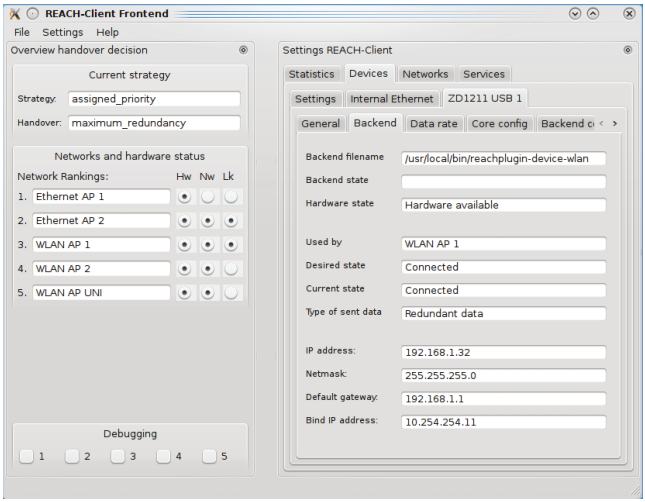

Abbildung 6.18: Das „Frontend“ für den REACH-Client visualisiert dem Nutzer die aktuelle Handoverentscheidung. Zudem kann die Konfiguration des REACH-Clients verändert werden.

Auswirkungen seiner Aktion anzeigen.

Dieses Konzept weist diverse Vorteile gegenüber statusbehafteten „Frontends“ auf. So kann hier der Zustand zwischen „Frontend“ und REACH-Client niemals inkonsistent sein, da das „Frontend“ nur diejenigen Zustände anzeigt, welche vom REACH-Client akzeptiert und übernommen worden sind. Weiterhin ist es kein Problem, mehrere „Frontends“ gleichzeitig zu starten und sie mit demselben REACH-Client zu verbinden. Wird über eines der „Frontends“ ein Parameter geändert, schickt der REACH-Client anschließend Änderungsmeldungen an alle „Frontends“, sodass überall derselbe Status angezeigt wird. Greifen zwei Benutzer konkurrierend auf denselben Parameter zu, setzt sich die letzte Änderung durch, was aber beide Nutzer bemerken werden.

6.5.4 Angebotene Dienste

Prinzipiell soll dem Nutzer eine Bedienschnittstelle zur Verfügung gestellt werden, welche für ihn vergleichbar mit der eines „Network Managers“ ist (siehe Abschnitt 6.2.1.1). Im Unterschied zu besagten „Network Managern“ soll das hiesige „Frontend“ jedoch nur eine optionale Bedienschnittstelle ohne Steuerungslogik sein, dafür aber auch diverse für Mobilität relevante Aspekte mit abdecken.

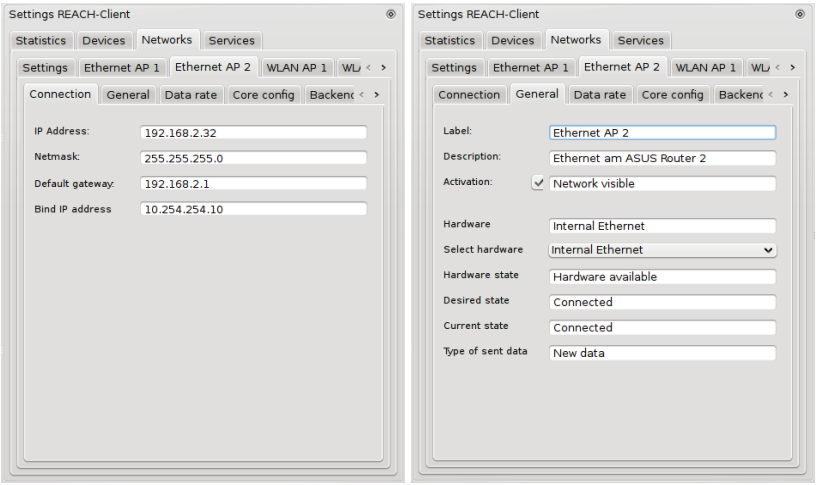

Abbildung 6.19: Das „Frontend“ dient dazu, den aktuellen Zustand des REACH-Clients bzw. REACH-Servers anzuzeigen. Das linke Bild zeigt die IP-Adress-Informationen einer Ethernetschnittstelle, während das rechte Bild Informationen über das zugehörige „Device Backend“ und einen Netzzugangspunkt liefert.

So ist in Abbildung 6.19 gezeigt, wie das „Frontend“ dem Nutzer den Zustand eines Netzzugangs anzeigt. Es können beispielsweise die aktuell gültigen IP-Adressen, die Verknüpfung zwischen Netzzugangsgerät und Netzzugangspunkt sowie der Dateiname eines „Device Backends“ erkannt werden. Das so genannte „Providerkonzept“, welches eines der wesentlichen Elemente von REACH darstellt, ist hier direkt erkennbar.

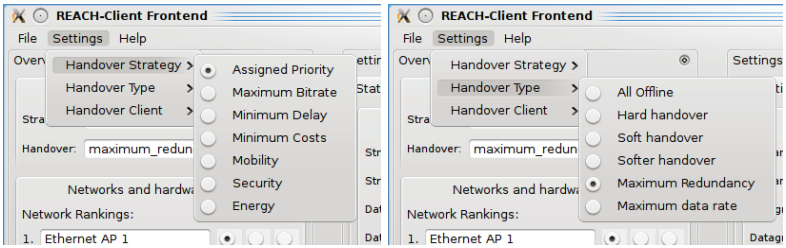

Abbildung 6.20: Der Nutzer kann auf den Handoverentscheider einwirken, indem er beispielsweise die Zielstellung zur Bewertung von Netzzugängen (linkes Bild) oder die Art der Handoverentscheidung (rechtes Bild) auswählt.

Weiterhin kann der Nutzer den Handoverentscheider beeinflussen. In Abbildung 6.20 wird gezeigt, welche Möglichkeiten zur Auswahl einer Zielstellung implementiert worden sind. REACH bewertet die Netzzugänge anhand einer VHDF, bietet allerdings über die Benutzeroberfläche nur eine vorab definierte Menge abstrakter Zielstellungen an. Zudem kann die Art der Handoverentscheidung ausgewählt werden, also ob beispielsweise „harte“, „weiche“ oder „weichere Handover“ oder ein Kanalbündelungsschema gewünscht werden.

Das hier verfolgte Ziel bei der Erstellung der beiden Benutzeroberflächen war es, *dem Programmierer* eine leistungsfähige Oberfläche zur Verfügung zu stellen, welche eine Vielzahl an „interessanten“ Daten visualisieren kann. So gibt es beispielsweise Schaltflächen, welche für Tests und zur Fehlersuche benötigt werden, wie die erkennbaren „Debugging-Schalter“ in Abbildung 6.18. Solche Schaltflächen sind jedoch für den Normalbetrieb ungeeignet und können unerfahrene Benutzer überfordern.

Für einen „normalen“ Benutzer und für den alltäglichen Gebrauch würde sich daher die Erstellung einer angepassten Oberfläche anbieten, welche auf die notwendigsten Funktionen reduziert wurde. Eine solche Oberfläche könnte dann gezielt mit Blick auf Benutzerfreundlichkeit gestaltet werden, und beispielsweise als „Applet“ in die „Taskleiste“ integriert werden. In Bezug auf den REACH-Client und den REACH-Server müssten keinerlei Anpassungen vorgenommen werden. Der einfachste Ansatz wäre es, den vorhandenen Quelltext der jetzigen Bedienschnittstellen zu nehmen und alle unerwünschten Schaltflächen zu entfernen. Dabei würde dann lediglich noch eine Teilmenge der vom REACH-Client und REACH-Server angebotenen Informationen visualisiert werden.

Da der REACH-Client und der REACH-Server problemlos mit einer Vielzahl an „angedockten“ Bedienschnittstellen zurecht kommen, und diese nur „dumme Programme ohne Intelligenz“ darstellen, spricht nichts gegen eine Vielzahl an angepassten Programmen für unterschiedliche Zwecke. Sie können sogar von mehreren Nutzern gleichzeitig auf einem Mehrbenutzersystem gestartet werden. Im Rahmen dieser Ausarbeitung wurde jedoch zu Demonstrationszwecken nur die „komplexen“ Varianten umgesetzt.

6.6 Kapitelzusammenfassung

Das vorliegende Kapitel stellte das Zugangskonzept von REACH vor, welches die Verwaltung von Netzzugangsgeräten am mobilen Endgerät, die Handoverentscheidung, die Pflege von Routingtabellen sowie die Bedienschnittstellen für die Nutzer zwecks Einflussnahme auf die Handoverentscheidung umfasst.

Das für REACH entwickelte „Providerkonzept“ führt eine kombinierte Betrachtung

von Netzzugangsgeräten und Netzzugangspunkten durch, und ist in der Lage, unterschiedlichste Netzzugangstechnologien einzubinden und miteinander vergleichbar zu machen. Aufbauend auf einer uniformen Repräsentation von Netzzugängen arbeitet der Handoverentscheider, welcher „horizontale“ und „vertikale Handover“ in verschiedenen Ausführungen unterstützt. Einen Schwerpunkt bildete die Vorstellung des prädiktiven Betriebsmodus, welcher mittels Regressionsrechnung zukünftige Abrissereignisse vorherzusagen versucht und damit „bessere“ Handoverentscheidungen fällen kann.

7 Das Suchkonzept

Dieses Kapitel befasst sich mit der Auswahlproblematik der mobilen Endgeräte in Bezug auf geeignet erscheinende REACH-Server. Der REACH-Client auf einem mobilen Endgerät bezieht für diese Entscheidung mehrere Aspekte mit ein, wie beispielsweise die Menge seiner geladenen „Relay Plugins“ im Vergleich mit dem Diensteangebot eines jeden REACH-Servers. Ein weiteres Entscheidungskriterium für die Auswahl stellt die jeweilige topologische Entfernung dar, wobei versucht wird, die Kommunikationswege „kurz“ zu halten. Weiterhin werden Anforderungen bezüglich der REACH-Proxyserver diskutiert, damit die mobilen Endgeräte diese immer ordnungsgemäß adressieren und erreichen können.

7.1 Einbeziehung mehrerer REACH-Proxyserver

REACH erlaubt es, dass sich mobile Endgeräte zu mehreren REACH-Proxyservern gleichzeitig verbinden können. Das wird bereits durch das in Kapitel 4 eingeführte *Dienstekonzept* von REACH erforderlich, da nicht jeder REACH-Server alle denkbaren „Relay Plugins“ geladen haben braucht. Als Konsequenz ist es notwendig, dass sich ein REACH-Client zu mehreren REACH-Servern verbinden muss, damit jedes seiner instantiierten „Relay Plugins“ sich mit mindestens einem serverseitigen Gegenstück assoziieren kann. Es gibt jedoch diverse weitere Gründe, wie beispielsweise die Minimierung des so genannten „Dreiecks-Routings“ (engl. „triangle routing“), wobei REACH von einer ortsabhängigen Auswahl profitieren kann.

7.1.1 Kurze Kommunikationswege

Für den Betrieb von REACH reicht theoretisch ein einziger REACH-Proxyserver mit installiertem REACH-Server aus, solange jedes mobile Endgerät diesen unabhängig von seinem Aufenthaltsort kontaktieren kann. Nachvollziehbar hierbei ist, dass die Qualität der Verbindung zwischen mobilem Endgerät und REACH-Proxyserver einen Einfluss auf die sich ergebende Dienstgüte hat. Befindet sich ein Nutzer beispielsweise im Ausland, und steht der einzige ihm zugängliche REACH-Proxyserver sehr weit von ihm entfernt,

werden seine Daten einen großen Umweg nehmen müssen. Selbst wenn sich der von einer Anwendung auf dem mobilen Endgerät kontaktierte Server in topologischer Nähe befindet, muss der Datenverkehr immer erst zum REACH-Proxyserver geleitet werden, woraufhin dieser die Daten dann in Richtung des ursprünglich adressierten Servers versendet. Der Kommunikationsweg beschreibt dabei ein Dreieck, was diesem Phänomen die Bezeichnung als „Dreiecks-Routing“ eingebracht hat (siehe Abbildung 7.1).

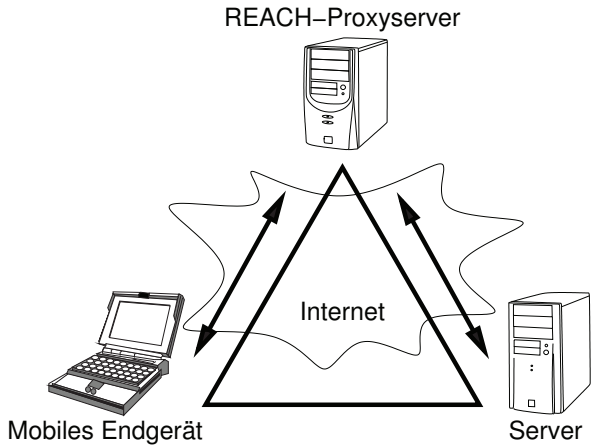

Abbildung 7.1: Beim „Dreiecks-Routing“ werden sämtliche Daten *immer* über einen „Ankerpunkt“ (hier der REACH-Proxyserver) geleitet. Der resultiere Weg ist länger als der direkte Weg zwischen mobilem Endgerät und Server.

Bezug zu Mobile IP: „Dreiecks-Routing“ ist ein Konzept, welches ebenfalls bei „Mobile IP“ [Perk02] eine zentrale Rolle spielt. Hierbei kommen so genannte „Agenten“ zum Einsatz, und ein bestimmter Agent übernimmt dabei die besondere Rolle des „Heimatagenten“. Dieser bildet den für Mobilitätsunterstützungen wichtigen „Ankerpunkt“, über den sämtlicher Datenverkehr verschickt wird. Solange sich ein mobiles Endgerät in seinem „Heimatnetz“ befindet, ist sein Heimatagent unmittelbar erreichbar und es findet eine direkte Kommunikation ohne Umwege statt.

Verlässt ein mobiles Endgerät jedoch sein Heimatnetz und betritt ein Fremdnetz, wird der dortige Agent für dieses ein „Fremdagent“, welcher mit Hilfe eines „Tunnels“ sämtliche Datenströme von und zum Heimatagenten weiterleiten muss. Es findet also ebenfalls ein „Dreiecks-Routing“ statt, da sämtliche Daten immer über den Heimatagenten laufen. Da sich ein solcher Umweg negativ auf die erreichbare Datenrate und auch auf die Latenz-

zeit auswirken kann, wurden bereits Weiterentwicklungen zu Mobile IP vorgeschlagen, welche sich eine Verkürzung der Kommunikationswege zum Ziel gesetzt haben.

Bezug zu REACH: Im Gegensatz zu Mobile IP verwendet REACH keine „Agenten“, sondern Proxyserver. Diese sind ortsfest installiert, sodass ihre IP-Adressen über längere Zeit hinweg unverändert bleiben können. Mit Hilfe dieser statischen IP-Adressen kann die Mobilität der Endgeräte vor den Servern im Internet verborgen werden. Ein jeder REACH-Proxyserver kann hierbei als ein solcher „Ankerpunkt“ dienen, wie es der Heimatagent bei Mobile IP als Aufgabe hatte. Dieser Ansatz weist allerdings ebenfalls die Problematik des „Dreiecks-Routings“ auf, da ein REACH-Proxyserver bezüglich einer bestehenden Datenübertragung nicht gewechselt werden kann. in bestehender Datenstrom muss immer über ein und denselben REACH-Proxyserver versendet werden, auch wenn sich das mobile Endgerät topologisch gesehen gar nicht mehr in seiner Nähe aufhält.

Stattdessen wäre es wünschenswert, wenn es eine Vielzahl an REACH-Proxyservern geben würde und sich ein mobiles Endgerät abhängig von seiner aktuellen Position von REACH-Proxyserver zu REACH-Proxyserver „hangeln“ könnte. Wäre dies möglich, könnte den negativen Auswirkungen des „Dreiecks-Routings“ entgegen gewirkt werden. Welcher REACH-Proxyserver zu einem gegebenen Zeitpunkt „topologisch nahe“ ist, müsste über ein „Umgebungsbewusstsein“ durch die mobilen Endgeräte ermittelt werden. Leider kann diese Idee so nicht umgesetzt werden, da, wie es bereits in Kapitel 4.3.1.2 beschrieben wurde, eine Sitzung nicht von einem REACH-Proxyserver zu einem anderen REACH-Proxyserver „migriert“ werden kann.

Es spricht allerdings nichts dagegen, dass ein mobiles Endgerät mehrere REACH-Proxyserver gleichzeitig mit in die Übertragung einbezieht. So kann beispielsweise tagsüber an der Arbeitsstätte des Benutzers der REACH-Proxyserver der Firma benutzt werden, und Abends nach Feierabend der REACH-Proxyserver in räumlicher wie topologischer Nähe des Nutzers bei sich zuhause. Da die tagsüber aufgebauten Datenströme nicht migriert werden können, muss Abends ein „Dreiecks-Routing“ bezüglich des REACH-Proxyservers der Arbeitsstätte durchgeführt werden. Allerdings können alle von diesem Zeitpunkt an neu aufgebauten Datenströme den REACH-Proxyserver des Benutzers in seiner Wohnung involvieren.

So kann, wie es Abbildung 7.2 zeigt, tagsüber ein zeitaufwendiger Download (Datenfluss **1**) gestartet werden, welcher auch dann noch fortgesetzt werden kann, wenn der Nutzer längst zuhause sitzt und sich sein mobiles Endgerät wieder mit dem Internet verbunden hat. Für alle neuen Datenströme (**2** und **3**) baut der REACH-Client seines mobilen Endgeräts eine Sitzung zum heimischen REACH-Proxyserver auf, und alle zukünftigen Datenströme werden von nun an über diesen Proxyserver geleitet. Die Da-

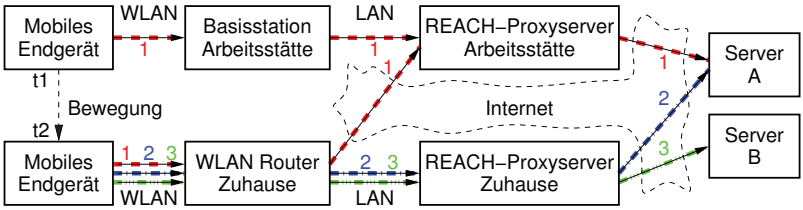

Abbildung 7.2: Ein mobiles Endgerät kann eine Vielzahl an gleichzeitigen „Sitzungen“ zu mehreren REACH-Proxyserver pflegen. Dabei kann zwar das „Dreiecks-Routing“ nicht unterbunden, jedoch dessen negative Auswirkungen gemildert werden.

ten des zeitaufwendigen Downloads laufen im Hintergrund weiterhin über den REACH-Proxyserver seiner Arbeitsstätte. Selbst dann, wenn der Nutzer zuhause erneuten Kontakt mit einem Server wünscht, zu dem er bereits eine Verbindung über einen mittlerweile nicht länger präferierten REACH-Proxyserver hat, wird hierfür der jetzt topologisch nähere REACH-Proxyserver verwendet (Datenstrom **2** zu Server **A**).

Sobald zu einem späteren Zeitpunkt *alle* Datenströme „des ersten“ REACH-Proxyservers beendet wurden (der beispielhafte Download sei abgeschlossen), wird die Sitzung zu diesem beendet. Da Verbindungen seitens vieler Anwendungen *häufig* kurzlebiger Natur sind (beim Websurfen überdauert eine TCP-Verbindung oft nur den Download einer einzigen Webseite), treten hierbei die negativen Auswirkungen des „Dreiecks-Routings“ in den Hintergrund. Es bleibt jedoch für langlebige Übertragungen bestehen.

Da sich ein REACH-Client allerdings mit beliebig vielen REACH-Proxyservern verbinden kann, sind Szenarien mit einer großen Anzahl an geographisch verstreuten REACH-Proxyservern denkbar. So könnte es beispielsweise für die Automobilindustrie von Interesse sein, dass Fahrzeuge dank flächendeckend verteilter REACH-Proxyserver mit einer hohen Kommunikationsgüte an das Internet angebunden werden können.

7.1.2 Lastverteilung

Ein weiterer Grund für den Einsatz mehrerer REACH-Proxyserver wäre der Wunsch nach Lastverteilungsmechanismen. Immerhin stellt jeder REACH-Proxyserver einen „Flaschenhals“ dar, besonders wenn er von mehreren mobilen Endgeräten als „Ankerpunkt“ ausgewählt wird.

Eine Lastverteilung könnte beispielsweise durch eine geographische Streuung mehrerer REACH-Proxyserver realisiert werden. Ein möglicher Anwendungsfall wäre ein Firmengebäude, in welchem in jeder Etage ein REACH-Proxyserver installiert ist. Die mobilen

Geräte der Mitarbeiter, welche von ihren Besitzern durch das Gebäude getragen werden, würden dabei immer den REACH-Proxyserver der aktuellen Etage bevorzugen. Da sich die Mitarbeiter dabei im Regelfall über mehrere Etagen verstreuen, könnte der Datenverkehr somit auf mehrere REACH-Proxyserver aufgeteilt werden.

Problematisch wird es hierbei allerdings, wenn sich zeitweise alle Mitarbeiter samt ihrer Geräte in derselben Etage befinden, beispielsweise auf Grund einer Dienstbesprechung. Würden jetzt alle Mitarbeiter gleichzeitig anfangen Datenverkehr zu verursachen (weil der Chef beispielsweise seine Angestellten dazu auffordert, die neue Webpräsenz der Firma zu bewundern), würde der in dieser Etage arbeitende Proxyserver mit einem hohen Datenaufkommen konfrontiert werden. Mit Hilfe eines rein geographischen oder topologischen „Bewusstseins“ der Endgeräte lässt sich dieses Problem nicht lösen, da sich ja alle mobilen Endgeräte im selben Raum befinden.

Stattdessen wäre ein alternativer Mechanismus zur Lastverteilung wünschenswert, welcher den Datenverkehr auf andere, momentan weniger ausgelastete REACH-Proxyserver leitet. Oder ein Endgerät erfragt beim Aufbau einer Sitzung den Status eines REACH-Proxyservers (zum Beispiel die aktuelle Last oder die Anzahl der momentan angemeldeten Geräte), um sich dann im Fall einer Überlastungssituation für einen anderen REACH-Proxyserver zu entscheiden.

Die letztgenannte Form der Lastverteilung, bei welcher der aktuelle Status eines REACH-Proxyservers mit in die Auswahl einbezogen wird, stellt eine sehr effektive Variante dar. Allerdings wurde sie im Rahmen dieser Dissertation nicht implementiert. Dazu wäre es notwendig gewesen, die Problematik der „Last“ genauer zu beleuchten und schlussendlich eine Metrik für die Eignung eines REACH-Proxyservers herzuleiten. Diese Metrik müsste es erlauben, einen topologisch nahen aber überlasteten REACH-Proxyserver mit einem topologisch entfernten, aber wenig belasteten REACH-Proxyserver vergleichen zu können. Da sich Lastzustände jedoch schnell ändern können, könnte hierzu ein signifikanter zusätzlicher Übertragungsaufwand notwendig werden. Zudem ist es denkbar, dass „Schwingungen“ entstehen, wenn mehrere mobile Endgeräte in Fällen hoher Last von REACH-Proxyserver zu REACH-Proxyserver „springen“.

Weiterhin wäre eine Betrachtung der Verkehrscharakteristik eines jeden „Relay Plugins“ sinnvoll, da beispielsweise Sprachübertragung andere Anforderungen aufweist als der Abruf von Webseiten. Da eine solche Modellierung als sehr komplex erscheint, fließt momentan der Lastzustand nicht mit in die Auswahlentscheidung ein. Für zukünftige Entwicklungen wäre ein solches Schema allerdings lohnenswert.

7.2 Ortsabhängige Serverauswahl

Die Auswahl der zu kontaktierenden REACH-Proxyserver erfolgt durch den REACH-Client auf dem mobilen Endgerät, wobei die aktuelle Aufenthaltsposition mit in die Entscheidungsfindung einbezogen wird. Ein typischer Anwendungsfall hierfür ist die bereits in Abschnitt 7.1.1 diskutierte Minimierung des „Dreiecks-Routings“.

7.2.1 Methoden der Positionsbestimmung

Es gibt unterschiedliche Ansätze, um die Position eines mobilen Endgeräts zu ermitteln. Dies kann durch das mobile Endgerät selbst, durch die Infrastruktur, oder durch beide geschehen. Die dazu nutzbaren Ansätze werden im Folgenden exemplarisch vorgestellt, um darauf aufbauend die in REACH umgesetzte Form der Positionsbestimmung zu beleuchten.

7.2.1.1 Seitens des mobilen Endgeräts

Bei GSM kann ein Handy erkennen, an welcher Basisstation es angemeldet ist. Somit können besondere Tarife angeboten werden, mit denen der Benutzer in einem „Heimbereich“ zu festnetzüblichen Tarifen telefonieren kann [Wiki09d]. Dem Nutzer wird die Verfügbarkeit des „Heimbereichs“ auf dem Display angezeigt; das Mobiltelefon ist sich also seiner Position bewusst. Für diesen Anwendungsfall ist eine Genauigkeit in der Größenordnung der Ausdehnung der Mobilfunkzellen ausreichend.

Ebenfalls angeboten werden Ortungsdienste, mit Hilfe derer die Position eines Mobiltelefons ermittelt werden kann. Die Genauigkeit beschränkt sich hierbei wieder auf die jeweilige Zellgröße. Im ländlichen Bereich ist daher eine geringere Genauigkeit zu erwarten, als beispielsweise im Bereich von Innenstädten. Laut [Stau08] können dabei zwei Möglichkeiten unterschieden werden, um die Genauigkeit zu verbessern: Als „außerhalb des Netzes“ operierendes Verfahren wird eine Ortung anhand der sichtbaren Basisstationen erwähnt. Dabei ermittelt das mobile Endgerät alle sichtbaren Basisstationen, gegebenenfalls noch in Kombination mit der jeweiligen Signalstärke. Diese Daten werden dann mit Hilfe einer zentralen Datenbank in eine Positionsangabe abgebildet. Dabei wird die Notwendigkeit eines Rückkanals erwähnt, da die Datenbank nicht unbedingt auf dem mobilen Endgerät installiert sein muss. Angewendet werden kann dieses Schema auch auf die Lokalisierung mittels WLAN-basierter Netze, wie in [Wint03] geschehen. Alternativ kann eine Ortung aber auch „innerhalb des Netzes“ erfolgen, was im nächsten Abschnitt erläutert wird.

Generell sollte man sich die Frage stellen, wozu die Positionsangabe überhaupt benötigt wird. Für die Erkennung des „Heimbereichs“ seitens eines Mobiltelefons wird eine verbesserte Genauigkeit nicht benötigt. In Notfällen wäre es dagegen vorteilhaft, wenn den Rettungskräften eine möglichst genaue Positionsangabe zur Verfügung stehen würde.

7.2.1.2 Seitens der Infrastruktur

Im Telefoniebereich ist eine netzseitige Positionsangabe bezüglich eines jeden aktiven Mobiltelefons unabdingbar. Schließlich ist es erforderlich, ein Mobiltelefon bei einem eingehenden Anruf ausfindig machen zu können. Dabei genügt allerdings wiederum die Kenntnis der aktuellen Funkzelle. Die netzinternen Positionsinformationen sind bei GSM jedoch noch weitaus ungenauer, denn sie werden erst bei Bedarf durch ein so genanntes „Paging“ konkretisiert [Wiki09c].

Laut [Tele09] soll in naher Zukunft die netzseitige Ortung bei Mobiltelefonen durch Peilungsmechanismen verbessert werden. So kann beispielsweise durch Laufzeitmessungen und Triangulation unter Zuhilfenahme mehrerer Basisstationen die Position eines Mobiltelefons genauer bestimmt werden. Anwendungsfall hierfür sind so genannte „standortbezogene Dienste“, welche als kommende „Killer-Applikationen“ genannt werden.

7.2.1.3 Mit Hilfe spezialisierter Werkzeuge

Weitaus bessere Ergebnisse lassen sich beispielsweise durch Einsatz des amerikanischen „Global Positioning System“ (GPS) [Wiki09b] erreichen, oder in Zukunft durch dessen europäisches Pendant „Galileo“ [Wiki09a]. Hierbei sind Positionsangaben mit einer Genauigkeit im Meterbereich erreichbar, welche dann dem mobilen Endgerät zur Verfügung stehen.

7.2.2 Positionsbewusstsein bei REACH

Der aktuelle Aufenthaltsort eines mobilen Endgeräts ist für REACH durchaus wichtig und auch erforderlich, jedoch werden keine *genauen geographischen* Positionsangaben benötigt. Für ein mobiles Endgerät mit installiertem REACH-Client ist die aktuelle Position dahingehend von Interesse, dass ein „naher“ REACH-Proxyserver ausgewählt werden kann. Ziel ist es, die Kommunikationswege kurz zu halten. Der Begriff „Nähe“ bezieht sich hierbei auf die Topologie des Netzes, nicht auf einen geographischen Abstand. Denn, um die Position innerhalb eines Netzwerks bestimmen zu können, ist eine geographische Positionsangabe wenig hilfreich. Lediglich im großen Maßstab ist ein

Anwendungsfall erkennbar: Falls beispielsweise pro Stadt, pro Landkreis oder pro Bundesland eigene REACH-Proxyserver angeboten werden, wäre die geographische Position wiederum ein probates Entscheidungskriterium.

Bei REACH fallen Positionsangaben demnach bezüglich der Topologie an. So ist sich ein mobiles Endgerät beispielsweise bewusst, dass es sich per WLAN mit *dem* DSL-Router im heimischen Wohnzimmer assoziiert hat, oder, dass es per Ethernetkabel an einem Switch in der Firma angeschlossen worden ist. Anders sieht es allerdings aus, wenn das mobile Endgerät gezwungen ist, sich mangels Alternativen per GPRS mit dem Internet zu verbinden. Dann stehen keine Informationen bezüglich der aktuellen topologischen Position mehr zur Verfügung. Es herrscht lediglich die Erkenntnis, dass sich das mobile Endgerät eben *nicht* am DSL-Router daheim befindet oder am genannten Switch an der Arbeitsstätte seines Besitzers angeschlossen worden ist.

Das mobile Endgerät kennt bei diesem Ansatz lediglich in Ausnahmefällen seine aktuelle Position. Nur wenn es sich in vorab definierten Bereichen (hier: zuhause oder in der Firma) befindet, erkennt es seine Umgebung wieder und kann auf seine Position schließen. Im Folgenden wird gezeigt, dass dies für REACH bereits ausreichend ist. REACH setzt hierfür auf das bereits im Rahmen des „Zugangskonzepts“ vorgestellte „Providerkonzept“ auf, welches für die Verwaltung von Netzzugängen entwickelt worden ist. Dabei ist das Positionsbewusstsein von REACH dahingehend zu verstehen, dass als Information lediglich vorliegt, mit Hilfe welcher „Provider“, also welcher Netzzugangspunkte, das mobile Endgerät mit dem Internet verbunden ist.

Ein mobiles Endgerät mit mehreren Netzzugangsgeräten kann demnach über ein vielfältig ausgeprägtes Positionsbewusstsein verfügen, wenn es sich anhand mehrerer „Provider“ gleichzeitig lokalisieren kann. Aus Sicht eines Menschen kann diese Information durchaus widersprüchlich sein: Was bedeutet es, wenn ein mobiles Endgerät per Ethernet auf Arbeit angebunden ist und sich dabei gleichzeitig per WLAN am heimischen „DSL-Router“ eingebucht hat? Wenn der Benutzer im Nachbarhaus wohnt, kann dies durchaus geschehen, jedoch erschwert dies die Ableitung einer geographischen Position. Aus Sicht von REACH sind diese Informationen jedoch nicht widersprüchlich, da pro Netzzugangsgerät eine eindeutige topologische Positionsangabe vorliegt, und an einer akkumulierten Positionsangabe kein Bedarf besteht.

7.2.2.1 Zonen als Einzugsgebiete für REACH-Proxyserver

REACH benutzt so genannte „Zonen“, mit deren Hilfe eine topologische Position relativ zu einem REACH-Proxyserver erkannt werden kann. Ein jeder REACH-Proxyserver „spannt“ dazu „Einzugsgebiete“ in Form mehrerer solcher Zonen auf, in welchen sich mobi-

le Endgeräte befinden können (siehe Abbildung 7.3). Befindet sich ein mobiles Endgerät dabei in einer als „nahe“ geltenden Zone, gilt der betroffene REACH-Proxyserver als besonders geeignet und kann bevorzugt ausgewählt werden. So können die Kommunikationswege kurz gehalten werden, was ein wesentliches Ziel der ortsabhängigen Auswahl von REACH-Proxyservern darstellt.

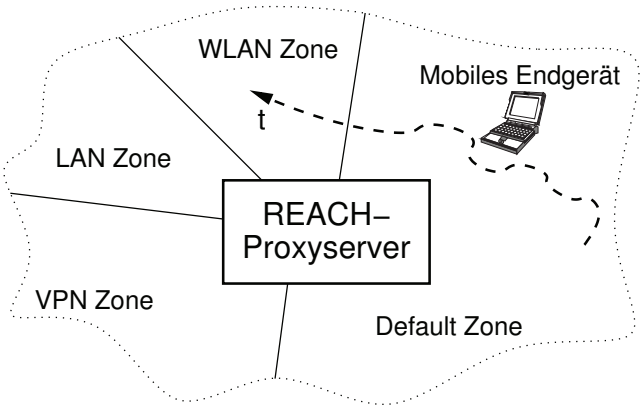

Abbildung 7.3: Jeder REACH-Proxyserver spannt Einzugsgebiete auf, welche hier als Zonen bezeichnet werden. Zu jeder aktiven Netzzugangstechnik kann das mobile Endgerät eindeutig ermitteln, in welcher Zone es sich bezüglich eines bestimmten REACH-Proxyservers befindet.

Solche Zonen werden für jeden REACH-Proxyserver gesondert definiert. Zur Beantwortung der Frage, in welcher Zone sich ein mobilen Endgerät relativ zu einem bestimmten REACH-Proxyserver befindet, kommt das „Providerkonzept“ mit seinen „Network Entries“ zum Einsatz. Zur Verdeutlichung dieses Sachverhalts eignet sich eine beispielhafte Konfigurationsdatei, welche in Abbildung 7.4 – aus Gründen der Übersichtlichkeit – reduziert worden ist.

Diese Konfigurationsdatei `connections.xml` listet alle dem REACH-Client bekannten REACH-Server auf (Schachtelungen `<ReachServer>`). REACH-Server, welche nicht in dieser Datei verzeichnet sind, können nicht berücksichtigt werden, was in Abschnitt 7.2.2.3 begründet wird. Für jeden REACH-Server müssen ein Identifikator `id`, die Zugangsdaten in Form eines Nutzernamens `login` und eines Passworts `password` (zur Authentifizierung während des Aufbaus von Kommunikationskanälen über TCP und UDP, siehe Abschnitt 5.5.2) sowie eine Gerätekennung `device` angegeben werden. Der Identifikator `id` wird benötigt, wenn für eine „Relay Plugin“-Instanz ein bestimmter REACH-Server festgelegt werden soll (siehe VPN-basierter Dienst in Abschnitt 4.3.2.5). Der letzt-

```
<Connections>
  <ReachServer id="Daheim" login="florian" password="secret"
               device="Laptop">
    <Zone id="default">
      [...]
    </Zone>
    <Zone id="wohnung">
      <NetworkEntry id="WLAN@home" />
      <NetworkEntry id="ETHERNET@home" />
      [...]
    </Zone>
  </ReachServer>
</Connections>
```

Abbildung 7.4: Diese vereinfachte Konfigurationsdatei `connections.xml` listet die Zonen aller dem mobilen Endgerät bekannten REACH-Proxyserver auf. Pro Zone können die Identifikatoren von „Network Entries“ (Netzzugänge) benannt werden.

genannte `device`-Parameter hat die Aufgabe, mehrere Sitzungen am REACH-Server pro Benutzer unterscheidbar zu machen. So kann ein Benutzer mehrere mobile Endgeräte, welche jeweils mit einem REACH-Client ausgestattet worden sind, gleichzeitig benutzen.

7.2.2.2 Definition von Zonen

Zu jeden REACH-Server müssen dessen Zonen spezifiziert werden. Jede Zone hat wiederum einen Identifikator `id`, welcher pro REACH-Server eindeutig sein muss. Zudem muss es eine Zone mit dem Identifikator `"default"` geben; für die restlichen Zonen können beliebige Bezeichner ausgewählt werden. Die `"default"`-Zone spezifiziert dabei einen *unbekannten Aufenthaltsort*, wenn bezüglich der topologischen Nähe zum REACH-Proxyserver keine Aussagen abgeleitet werden können. Alle anderen Definitionen von Zonen sind als Ausnahmeregeln zur `"default"`-Zone zu verstehen. Ein REACH-Server, welcher aus einer solchen genauer spezifizierten Zone heraus erreicht werden kann, gilt als „topologisch nahe“ und wird bevorzugt ausgewählt. Um die Verknüpfungen zwischen Zonen und Positionen herzustellen, werden in jeder `<Zone>`-Schachtelung die Identifikatoren von „Network Entries“ aufgelistet, also von in der Datei `networks.xml` aufgelisteten Netzzugangspunkten (siehe Abschnitt 6.3.2). Konnte sich in diesem Beispiel also das mobile Endgerät über die Netzzugänge `"WLAN@home"` oder `"Ethernet@home"` mit dem Internet verbinden, gilt für den betroffenen Netzzugang die Zone `"wohnung"`. Der REACH-Server mit dem Identifikator `"Daheim"` würde als besonders gut geeignet gelten.

Der Leser stellt sich vielleicht die Frage, wozu überhaupt eine `"default"`-Zone spezifiziert werden muss. Schließlich ergibt sich eine solche ja bereits implizit, wenn keine der durch „Network Entry“-`id`s beschriebenen Zonen zutrifft. Das stimmt soweit, allerdings müssen für jede Zone noch weitere Parameter spezifiziert werden, welche hier aus Vereinfachungsgründen weggelassen wurden. Sie werden in Abschnitt 7.3.1 „Ortsabhängige Adressierung“ beschrieben.

7.2.2.3 Vorab bekannte REACH-Proxyserver

Durch die Konfigurationsdatei `connections.xml` wird die Tatsache verdeutlicht, dass dem REACH-Client ein jeder relevante REACH-Proxyserver vorab bekannt sein muss. Nur die hier hinterlegten REACH-Proxyserver werden berücksichtigt. Dies bringt diverse Nachteile mit sich, sorgt aber für eine wesentlich vereinfachte Architektur, da alternativ ein zentraler Auskunftsdienst erforderlich gewesen wäre. Für die gewählte Vereinfachung spricht:

- Ein Anwender benötigt pro REACH-Server einen freigeschalteten Zugang. Da ein Benutzer nicht automatisch für *jeden* REACH-Server einen gültigen Zugang besitzen kann, ist es ein sinnvoll, die „erlaubten“ REACH-Server explizit zu benennen.

- Alternativ könnte eine zentrale Nutzerverwaltung implementiert werden, sodass ein mobiles Endgerät nicht länger Zugangsdaten für mehrere REACH-Server vermerken muss. Ein solcher Dienst müsste implementiert und in Form eines Serverdienstes bereit gestellt werden. Dieser Dienst könnte jedoch nicht „handoversicher“ in Anspruch genommen werden, da anfangs ja noch keine Sitzung mit einem REACH-Server besteht.

- REACH-Proxyserver werden in der Praxis zu getrennten „Verwaltungseinheiten“ gehören. Der persönliche REACH-Proxyserver eines Nutzers zuhause wird nicht denselben Nutzerverwaltungsserver benutzen können wie der REACH-Proxyserver des Arbeitgebers. Daraus folgt, dass auch hier eine Vielfalt an Adress- und Zugangsdaten vorab bekannt sein müssten, damit der „erste Kontakt“ eines mobilen Endgeräts funktionieren könnte.

Daher wurde auf die Implementierung eines zentral verfügbaren Anmeldedienstes verzichtet, und der Anwender muss vorab alle relevanten REACH-Server mit ihren Zonen in die Datei `connections.xml` aufnehmen. Damit werden die folgenden Einschränkungen bewusst in Kauf genommen:

- Für einen vollwertigen Lastverteilungsmechanismus müssten eine Vielzahl an REACH-Proxyservern involviert werden können. Es ist aber unpraktisch, wenn diese alle vorab dem mobilen Endgerät bekannt sein müssen.

- REACH-Proxyserver können wegfallen, ersetzt werden oder ihre Adressen ändern. Es wäre sinnvoll, wenn sich das mobile Endgerät diesbezüglich keinen Status zu merken bräuchte, sondern sich zyklisch eine aktuelle Liste von einem Auskunftsdienst beschaffen könnte.

- Ein solcher Auskunftsdienst könnte ebenfalls über das Diensteangebot eines jeden REACH-Servers informiert sein, und damit eine zentral organisierte Dienstever-waltung und -suche ermöglichen.

Im Zuge einer Weiterentwicklung, spätestens wenn diese mit Blick auf eine umfassende Installation vieler REACH-Proxyserver erfolgt, bietet sich eine nachträgliche Erweiterung um das vorgestellte Schema an. Auf die *eigentliche* Funktion von REACH, nämlich die handoversichere Kommunikation mittels eines Diensteangebots, hat das Fehlen dieser Funktionalität keinen Einfluss.

7.2.2.4 Auswahl bei gleichwertigen REACH-Proxyservern

Dadurch, dass noch keine Bewertungsmechanismen bezüglich der Auswahl von REACH-Proxyservern implementiert worden sind, kann es passieren, dass keine eindeutige Wahl getroffen werden kann. Damit ist gemeint, dass der REACH-Client mehrere aus seiner Sicht gleichwertige REACH-Proxyserver kennt, und sich entscheiden muss, welchen er schlussendlich kontaktiert.

Dieses Problem tritt besonders dann zu Tage, wenn sich das mobile Endgerät bezüglich aller relevanten REACH-Server in keiner „besonderen“ Zone befindet, sondern mangels Informationen bezüglich der topologischen Entfernung diese immer aus der `"default"`-Zone heraus ansprechen würde. Dieser Fall tritt bereits dann ein, wenn sich das mobile Endgerät mit Hilfe von GPRS/GSM oder UMTS eingebucht hat. In diesem Fall wäre jeder bekannte REACH-Proxyserver gleich gut geeignet, zumindest wenn nur das Wissen über die topologische Nähe eine Rolle spielt.

Da im aktuellen Stand der Implementierung keine Bewertung der REACH-Proxyserver (z.B. anhand deren aktueller Auslastung) durchgeführt wird, zieht der REACH-Client als „Notlösung“ die gemessene Umlaufzeit heran. Die Umlaufzeit besitzt nur eine eingeschränke Aussagekraft, verdeutlicht jedoch, dass REACH auch Informationen bezüglich der Übertragungsstrecke mit in die Einscheidungsfindung einfließen lässt.

Es muss noch ein weiterer Schwachpunkt der aktuellen Implementierung genannt werden. In der Praxis können sich einzelne REACH-Proxyserver als extrem ungeeignet erweisen, wenn sie aus der `"default"`-Zone heraus kontaktiert werden. Das wäre beispielsweise der Fall, wenn ein REACH-Proxyserver über DSL angebunden ist, und diesem nur ein „Uplink“ mit geringer Bitrate zur Verfügung steht. Ein solcher Proxyserver würde einen schlechten „Ankerpunkt“ darstellen, da der gesamte ins Internet gerichtete Datenverkehr sich seinen Weg durch den Uplink bahnen müsste.

Diese Problematik ließe sich aber zukünftig durch den bereits angesprochenen Bewertungsmechanismus mit lösen, welcher bereits im Rahmen der Lastverteilungsproblematik erwähnt wurde. Da dieser für REACH bislang nicht umgesetzt wurde, kann es in der Praxis zu derartigen Fehlentscheidungen kommen.

7.3 Erreichbarkeit von REACH-Proxyservern

Während im letzten Unterkapitel die ortsabhängige Auswahl eines REACH-Proxyservers im Vordergrund stand, geht es in diesem Unterkapitel um die ortsabhängige Adressierung eines gegebenen REACH-Proxyservers, nachdem dieser bereits ausgewählt worden ist.

In Abschnitt 5.5.5 wurde erläutert, dass ein REACH-Proxyserver mit mehreren IP-Adressen erreichbar sein sollte, wenn mobile Endgeräte mit zieladressbasierter Wegewahl eingesetzt werden sollen. Nur dann können diese auch auf „weichere Handover“ sowie Kanalbündelungsszenarien zurückgreifen, da bezüglich der unterschiedlichen IP-Adressen eindeutige Einträge in der Routingtabelle hinterlegt werden können.

In der Praxis zeigt sich jedoch noch ein weiteres Problem. Unter Umständen muss ein REACH-Proxyserver abhängig vom Standort des mobilen Endgeräts mit unterschiedlichen IP-Adressen angesprochen werden, damit dieser überhaupt erreicht werden kann.

7.3.1 Ortsabhängige Adressierung

Im einfachsten Fall ist ein bestimmter REACH-Proxyserver nur mit einer einzigen IP-Adresse ausgestattet. Steht einem mobilen Endgerät lediglich die zieladressbasierte Wegewahl zur Verfügung, können zwar nur „harte Handover“ durchgeführt werden, aber eine Kommunikation wäre trotz dieser Einschränkung möglich.

Was passiert aber, wenn die Topologie wie in Abbildung 7.5 gezeigt aussieht? Dargestellt wird die Installation in der Wohnung des Autors. Für Ethernet und für WLAN werden getrennte Netze aufgespannt, welche jeweils einem eigenen IP-Adressbereich aufweisen. Seitens des Internets kann der REACH-Proxyserver nur über die globale IP-Adresse des NAT-Gateways erreicht werden. Hierfür wurde im DSL-Router der DynDNS-

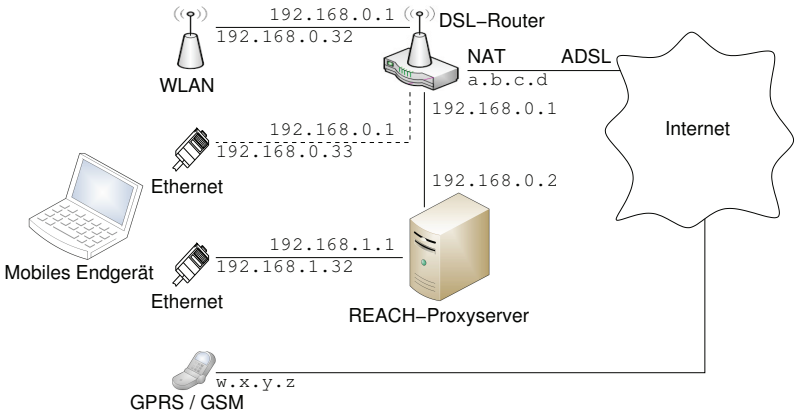

Abbildung 7.5: In der Wohnung des Autors stehen ein Zugang über WLAN sowie ein Zugang mit Hilfe von Ethernet zur Verfügung. Zusätzlich bestünde die Möglichkeit, ein mobiles Endgerät direkt per Ethernet am DSL-Router anzuschließen (gestrichelt dargestellt). An der NAT-Grenze wurden Weiterleitungsregeln geschaltet, damit der REACH-Proxyserver auch mit Hilfe von GPRS/GSM erreicht werden kann.

Dienst [VTRB97] konfiguriert, welcher die sich bei jeder Einwahl ändernde IP-Adresse mit Hilfe eines DNS-Namens maskiert.[1] Auf dem DSL-Router wurden „Port Forwarding Regeln“ für die Transportprotokolle von REACH hinterlegt, sodass eine „nach innen“ gerichtete Weiterleitung zum eigentlichen REACH-Proxyserver über die NAT-Grenze hinweg gewährleistet ist.

Problematisch wird es in diesem Beispiel, wenn sich das mobile Endgerät in einem der beiden lokalen Netze befindet, sich also genauso wie der REACH-Proxyserver hinter dem NAT-Gateway aufhält. In diesem Fall kann das mobile Endgerät den REACH-Proxyserver nicht mehr erreichen, jedenfalls nicht unter Anwendung des DNS-Namens des Gateways. Dieser würde zwar noch korrekterweise in die öffentliche IP-Adresse des Gateways aufgelöst werden, aber beim Kontaktieren des Gateways würde die Weiterleitungsregel zum internen REACH-Proxyserver nicht durchlaufen werden können! Diese kommt nur dann zum Tragen, wenn TCP-Verbindungen oder UDP-Assoziationen *von außen* an die NAT-Grenze herantreten, nicht aber, wenn das Gateway *von innen* angesprochen wird. Das Resultat ist, dass der REACH-Client in diesem Fall keine Verbindung zum REACH-Proxyserver aufbauen kann.

[1] Man erkennt, dass in diesem Fall nur eine einzige IP-Adresse ermittelt wird, hier repräsentiert durch den dauerhaft gültigen DNS-Namen des NAT-Gateways.

Die Lösung besteht darin, dass der REACH-Client in diesem Fall eine andere IP-Adresse benutzen muss, nämlich eine, die innerhalb des LANs gültig ist. Der REACH-Client muss dann aber wissen, *wann* er den REACH-Proxyserver über seinen DNS-Namen ansprechen muss, und *wann* er stattdessen auf eine seiner lokalen IP-Adressen zurückgreifen muss. Hierbei kommt wieder das Konzept der Ortsabhängigkeit ins Spiel, da für den Bereich der beiden LANs jeweils eine eigene Zone definiert werden kann. Bezüglich dieser Zonen können dann unterschiedliche IP-Adressen für ein und denselben REACH-Proxyserver benannt werden. Abbildung 7.6 zeigt die im Demonstrator getestete Konfigurationsdatei.

```
<Connections>
  <ReachServer id="Daheim" login="florian" password="secret"
               device="Laptop">
    <Zone id="default">
      <Address ip="powerstation.homelinux.org"
               tcpport="10000" udpport="10000" />
    </Zone>
    <Zone id="wlanrouter">
      <Address ip="192.168.0.2"
               tcpport="10000" udpport="10000" />
      <Address ip="192.168.0.3"
               tcpport="10000" udpport="10000" />
      <Address ip="192.168.0.4"
               tcpport="10000" udpport="10000" />
      <NetworkEntry id="WLAN@home" />
    </Zone>
    <Zone id="ethernetport">
      <Address ip="192.168.1.32"
               tcpport="10000" udpport="10000" />
      <Address ip="192.168.1.33"
               tcpport="10000" udpport="10000" />
      <Address ip="192.168.1.34"
               tcpport="10000" udpport="10000" />
      <NetworkEntry id="Ethernet@home" />
    </Zone>
  </ReachServer>
</Connections>
```

Abbildung 7.6: Dieser Auszug aus der Konfigurationsdatei `connections.xml` beschreibt den REACH-Proxyserver in der Wohnung des Autors. Zu jeder definierten Zone muss mindestens ein `<Address>`-Parameter angegeben werden, damit der REACH-Proxyserver aus dieser Zone heraus erreicht werden kann.

Hierbei wurden für den REACH-Server `"Daheim"` insgesamt drei Zonen definiert. Die `"default"`-Zone deckt als Einzugsgebiet das gesamte Internet jenseits des NAT-Gateways ab. Neu sind in dieser Darstellung die `<Address>`-Schachtelungen, welche eine beliebige Menge an IP-Adressen (bzw. DNS-Namen) und Portnummern für jeweils TCP und UDP pro Zone spezifizieren. Ist in diesem Beispiel die `"default"`-Zone gültig, muss der REACH-Proxyserver mit Hilfe des DNS-Namens `"powerstation.homelinux.org"` angesprochen werden. Hinter diesem Namen verbirgt sich das NAT-Gateway mit seinen zwei Weiterleitungsregeln, welche bezüglich TCP und UDP die Portnummer `10000` in Richtung des internen REACH-Proxyservers weiterleiten.

Die zweite Zone `"wlanrouter"` wird als erste Spezialisierung relevant, sobald das „Network Entry“ `"WLAN@home"` aktiviert werden konnte. Die dritte Zone `"ethernetport"` deckt den Fall der Ankopplung über Ethernet ab. Für beide Zonen wurden bezüglich des REACH-Proxyservers drei lokale IP-Adressen festgelegt.

Ein mobiles Endgerät, welches sich in der Wohnung des Benutzers befindet, kann durchaus *alle* verfügbaren Netzzugangsgeräte gleichzeitig aktivieren. Bezüglich Ethernet und WLAN würde es Bestandteil der beiden LANs werden, sich also bezüglich des heimischen REACH-Proxyservers in den Zonen `"wlanrouter"` und `"ethernetport"` befinden. Wurde zudem UMTS aktiviert, befindet sich das mobile Endgerät diesbezüglich in der `"default"`-Zone, und es muss den REACH-Proxyserver mit Hilfe des DNS-Namens des NAT-Gateways ansprechen.

Anhand dieses Aufbaus kann allerdings nicht verdeutlicht werden, wieso für den REACH-Proxyserver in den beiden internen Zonen jeweils mehrere IP-Adressen definiert worden sind. Diese sind eigentlich unnötig, da es selbst bei Aktivierung *aller* genannten Netzzugangsgeräte zu keinen Mehrdeutigkeiten bezüglich der Wegewahl kommt. Würde man allerdings das mobile Endgerät direkt per Ethernet an das DSL-Modem anschließen (und dies dem REACH-Client durch Definition eines weiteren „Network Entries“ bekannt machen), wäre die Mehrdeutigkeit bezüglich der Wegewahl sichtbar. Dann wäre eine zeitgleiche Verbindung über Ethernet und WLAN nur noch möglich, wenn der REACH-Proxyserver in dieser Zone (`id="wlanrouter"`) mit mindestens zwei IP-Adressen bekannt gemacht wurde. In diesem Fall wären auch weiterhin „weichere Handover“ und Kanalbündelungen möglich.

7.3.2 Überschreitung von NAT-Grenzen

Nachdem im eben erläuterten Szenario bereits eine NAT-Grenze überschritten wurde, soll auf diesen Fall hier nochmals gesondert eingegangen werden. NAT ist im heutigen Internet häufig anzutreffen. Ein wichtiger Anwendungsfall ermöglicht es, dass trotz nur

einer einzigen verfügbaren öffentlichen IP-Adresse eine Vielzahl an Systemen auf das Internet zugreifen können. Das so genannte „NAT-Gateway“ stellt dabei das einzige System dar, welches über eine „echte“ Anbindung an das Internet verfügt und mit einer öffentlichen IP-Adresse Bestandteil des Internet wird. Zusätzlich ist an das NAT-Gateway ein internes Netzwerk angeschlossen, welches als *privates Netz* bezeichnet wird. Hierbei kommen üblicherweise private IP-Adressen nach RFC 1918 [RMKG+96] zum Einsatz. Da diese *nicht routbar* sind, muss am NAT-Gateway eine Adressumsetzung stattfinden. Die Server im Internet dürfen keine Anfragen seitens einer privaten IP-Adresse erhalten, sondern nur Anfragen seitens einer öffentlichen IP-Adresse bekommen. In diesem Fall muss dies die IP-Adresse des NAT-Gateways sein. Die hierfür an der NAT-Grenze zum Einsatz kommenden Mechanismen heißen „Connection Tracking“ (frei übersetzt: „Verbindungsverfolgung“) und „Masquerading“ (Maskierung), siehe RFC 3022 [SrEg01].

NAT wird häufig in Wohnungsinstallationen angewendet. Auf Grund der Knappheit an IPv4-Adressen wird durch die Provider häufig nur eine einzige IP-Adresse an die Geräte der Kunden vergeben. Ein durch den Telekommunikationsanbieter bereit gestellter „DSL-Router“ verbindet sich mit dem Internet und leitet unter Anwendung von NAT ein privates Netzwerk in das Internet.

NAT kommt mittlerweile sogar schon bei Mobilfunkanbietern zum Einsatz. So werden den Endgeräten der Anwender bereits private IP-Adressen zugewiesen, wobei seitens des Providers NAT durchgeführt wird. Dem Anwender bleibt es bei diesem Konzept leider von vornherein verwehrt, eigene Serverdienste auf mobilen Endgeräten anzubieten.

Im Folgenden wird unterschieden, in welche Richtung eine NAT-Grenze überschritten wird. Dies kann „von innen nach außen“, aber auch „von außen nach innen“ erfolgen.

Von innen nach außen

Der REACH-Proxyserver soll als „Ankerpunkt“ fungieren und benötigt dafür eine feste IP-Adresse. Daher bietet es sich an, die REACH-Proxyserver ortsfest im Internet zu platzieren, wobei diese mit festen öffentlichen IP-Adressen ausgestattet werden können. Ein REACH-Client auf einem mobilen Endgerät erfährt keine Adressierungsprobleme, wenn er sich mit einem solchen REACH-Proxyserver assoziieren möchte.

Es kann allerdings passieren, dass sich ein mobiles Endgerät in einem privaten Natz „hinter“ einer NAT-Grenze befindet. Gerade bei Heiminstallationen kommt häufig NAT zum Einsatz. Der sich dabei ergebende Aufbau ist in Abbildung 7.7 dargestellt. Da der Zugriff auf das Internet über NAT-Grenzen hinweg Probleme verursachen kann, müssen die Übertragungsprotokolle von REACH in Bezug auf potentielle Probleme von NAT-Grenzen untersucht werden. Es ist aber anzumerken, dass es ausreichend ist, lediglich

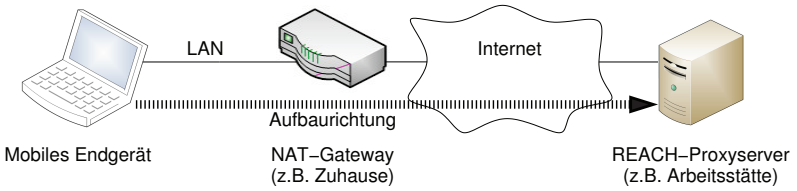

Abbildung 7.7: In der Praxis wird es unvermeidlich sein, dass die Datenstöme zwischen REACH-Client und REACH-Proxyserver eine NAT-Grenze überschreiten müssen. Daher ist es erforderlich, dass die in REACH eingesetzten Protokolle mit NAT zurecht kommen.

die Transportprotokolle von REACH zu betrachten. Alle Datenstöme der Anwendungen werden bereits vorher durch REACH abgefangen und können somit durch eine NAT-Grenze nicht länger negativ beeinflusst werden. Die durch NAT hervorgerufenen Probleme und ihr jeweiliger Bezug zu den Protokollen von REACH lauten:

Connection Tracking: Da die Absenderadresse ausgehender Datenpakete an der NAT-Grenze mit der öffentlichen IP-Adresse des NAT-Gateways *maskiert* wird, ist ein Mechanismus notwendig, welcher die rücklaufenden Datenpakete wieder zu einem *internen* Rechner hinter der NAT-Grenze zuordnen kann. Hierfür werden zusätzliche Informationen aus der Transportschicht mit einbezogen, wie die Art des Transportschichtprotokolls und die verwendeten Portnummern. Es wird ein „Connection Tracking“ durchgeführt, wobei zu jedem IP-Paket ein Eintrag in der so genannten NAT-Tabelle gesucht wird. Für verbindungsorientierte Protokolle wie TCP bleibt ein solcher Eintrag normalerweise so lange bestehen, wie die Verbindung überdauert. Bei UDP, welches verbindungslos arbeitet, werden Gültigkeitsdauern („Timeouts“) benutzt, wodurch inaktive Einträge einem Alterungsprozess unterworfen sind. Bei einem aktuellen LINUX-Kernel liegt diese Zeitspanne bei 30 bzw. 180 Sekunden, je nachdem, ob das ausgehende UDP-Paket mit einem eingehenden UDP-Paket beantwortet wurde oder nicht. Hierfür sei auf RFC 4787 [AuJe07] sowie auf Abschnitt 4.3.2.4, wo diese Problematik bereits diskutiert worden ist, verwiesen.

Um der ungewollten Löschung von UDP-Assoziationen aus den NAT-Tabellen entgegen zu wirken, müssen „Keep-Alive“-Pakete versendet werden. Bei REACH geschieht dies durch eine beidseitige zyklische Versendung von `PCPR_KEEPALIVE`-PDUs, welche im Rahmen der Abrisserkennung bereits in Abschnitt 5.5.3 vorgestellt worden ist.

Bezüglich TCP greift an NAT-Grenzen ebenfalls eine Zeitspanne. TCP-Verbindungen, welche für 24 Stunden inaktiv waren, werden ebenfalls aus der NAT-Tabelle entfernt. Ein

Grund hierfür ist, dass TCP-Verbindungen nicht immer sauber beendet werden, wenn beispielsweise ein Kommunikationspartner „hart“ abgeschaltet wird (Stromausfall) und die aktive Gegenstelle von sich aus keine Daten zu versenden hat, sondern lediglich auf Daten wartet. Hierbei entstehen so genannte „verwaiste“ TCP-Verbindungen, welche durch eine TCP-Instanz selbst nicht erkannt werden können.[2] Dies kann besonders bei stark frequentierten Servern problematisch werden, vor allem, wenn diese eine lange Zeit ohne Unterbrechung laufen. Aber auch an NAT-Grenzen würde eine verwaiste Verbindung einen Eintrag in der NAT-Tabelle belegt halten, wodurch die Tabelle überlaufen könnte. Daher wird an NAT-Grenzen bezüglich TCP eine Inaktivitätserkennung durchgeführt, welche aber im Einzelfall Probleme verursachen kann.

Bezüglich REACH gibt es zwei Möglichkeiten, um mit dem Abbruch inaktiver TCP-Verbindungen umzugehen. Zum einen könnten zyklische „Keep-Alive“-Nachrichten in den Nutzdatenstrom der TCP-Verbindungen eingebracht werden, beispielsweise nach 12 Stunden Inaktivität. Oder, man sieht alternativ *keinen* solchen Mechanismen vor und nimmt die Abrisse bewusst in Kauf: Bricht dann irgendwann an der NAT-Grenze eine TCP-Verbindung weg, bekommen dies anfangs weder der REACH-Client noch der REACH-Server mit. Aber wenn eine der beiden Instanzen Daten über die TCP-Verbindung zu versenden versucht, wird das NAT-Gateway mangels gültigem Eintrag einen Verbindungsabriss signalisieren. Zu diesem Zeitpunkt wird der Verbindungsabbruch einseitig erkannt, wodurch aber der gesamte REACH-Kommunikationskanal bestehend aus zwei TCP-Verbindungen und einer UDP-Assoziation (vorgestellt in Kapitel 5.5.2) heruntergefahren wird. Damit werden auch keine `PCPR_KEEPALIVE`-PDUs mehr über die UDP-Assoziation versendet, sodass der Verbindungsabriss mit kurzer Verzögerung auch an der Partnerinstanz bemerkt wird. Schlussendlich konnte somit auf die Implementierung von „Keep-Alive“-Nachrichten im TCP-Datenstrom verzichtet werden.

Rückkanäle und eingebettete Adressen: Es ist problematisch, Protokolle mit „nach innen“ gerichteten Begleitverbindungen über eine NAT-Grenze hinweg einzusetzen (siehe RFC 3027 [HoSr01]). Ein Beispiel hierfür ist das „File Transfer Protocol“ (FTP, RFC 959 [PoRe85]), welches allerdings dank zweier Mechanismen auch über NAT-Grenzen hinweg eingesetzt werden kann. Zum einen kann auf den „passiven Betriebsmodus“ zurückgegriffen werden, bei welchem auch die Begleitverbindungen von innen nach außen aufgebaut werden. Das funktioniert allerdings nur, wenn sich der FTP-Server nicht ebenfalls hinter einer NAT-Grenze befindet.

Eine leistungsfähige Alternative besteht in der Verwendung so genannter „Applicati-

[2] Es gibt Implementierungen von TCP, bei welchen optional ein „Keep-Alive“-Mechanismus aktiviert werden kann. Dieser ist aber in der Standardeinstellung *immer* ausgeschaltet.

on Layer Gateways“ (ALGs), welche auf der NAT-Grenze angesiedelt werden und für die Verbindungsverfolgung auch die Anwendungsschicht mit einbeziehen. Im LINUX-Kernel sind eine ganze Reihe solcher ALGs aktivierbar, unter anderem für die Protokolle SCTP, FTP und SIP. Diese ALGs beherrschen es zudem, die in die Protokolldatenströme eingebetteten Adressen zu erkennen und zu ändern. Werden beispielsweise Rückkanäle benötigt, kann dass ALG selbstständig Annahmestellen öffnen und deren Adressen den Kommunikationspartnern im öffentlichen Internet mitteilen.

Für REACH ist diese Problematik nicht relevant, da im geschilderten Anwendungsfall ausschließlich TCP-Verbindungen und UDP-Assoziationen seitens des REACH-Clients initiiert werden. Das bedeutet, die NAT-Grenze wird immer nur von innen nach außen überquert, und es werden keine nach innen gerichteten Begleitverbindungen benötigt. Zudem bettet REACH keine IP-Adressen oder Portnummern in die betroffenen Protokolle (SPPR und PPPR/PCPR) ein, sodass es den REACH-Server nicht interessiert, ob die Absenderadresse der eingehenden Daten schlussendlich die eines REACH-Clients ist oder ob diese an einer NAT-Grenze maskiert wurde.

Fazit: Die Überquerung einer NAT-Grenze von innen nach außen ist für REACH unproblematisch. Verbesserungsvorschläge für zukünftige Entwicklungen wären ein „Keep-Alive“-Mechanismus für die beiden TCP-Kanäle der REACH-Kommunikationskanäle.

Von außen nach innen

Es gibt allerdings Szenarien, in denen eine NAT-Grenze in umgekehrter Richtung, also von außen nach innen, überquert werden muss. Das bedeutet, dass sich ein REACH-Proxyserver in einem privaten Netz hinter einer NAT-Grenze befindet, und das mobile Endgerät diesen aus dem Internet heraus kontaktieren möchte. Dieser Anwendungsfall ist zwar weniger praxisrelevant als der im letzten Abschnitt beschriebene, kann aber durchaus vorkommen.

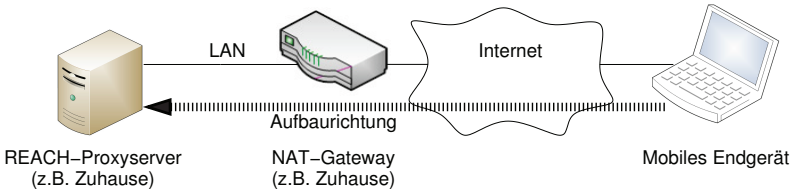

Abbildung 7.8: Soll ein REACH-Proxyserver in einem privaten LAN *hinter* einer NAT-Grenze installiert werden, ist die Erreichbarkeit seitens der mobilen Endgeräte aus dem öffentlichen Internet heraus erschwert.

Eine solche Konstellation, wie sie in Abbildung 7.8 dargestellt ist, kommt vor allem dann vor, wenn ein Nutzer zuhause bei sich einen eigenen REACH-Proxyserver betreiben möchte. Die Anbindung an das Internet erfolgt dabei beispielsweise mit Hilfe von DSL, wobei eine einzige öffentliche IP-Adresse vergeben wird. Auf dem „DSL-Router“ muss daher NAT durchgeführt werden. Der REACH-Server wäre dann auf einem der internen Systeme installiert, also im privaten LAN hinter der NAT-Grenze.

Es stellt sich dabei die Frage, ob eine solche Konstellation überhaupt Sinn ergibt. Schließlich bekommt man bei marktüblichen DSL-Tarifen keine feste IP-Adresse zugewiesen und hat zudem mit zyklischen Zwangstrennungen zu rechnen. Dadurch kann der REACH-Proxyserver seine Funktion als dauerhaften „Ankerpunkt“ nicht erfüllen, jedenfalls nicht in Bezug auf einen Zugriff auf das öffentliche Internet. Sinnvoller wäre es, wenn sich der private Anwender einen Server in einem Rechenzentrum mieten würde. Dieser wäre mit einer festen öffentlichen IP-Adresse ausgestattet, kann jedoch unter Umständen höhere Kosten verursachen als ein System zuhause, vor allem, wenn es sowieso schon vorhanden ist, wie im Fall der Heiminstallation des Autors.

REACH verfehlt hier seinen Sinn, allerdings nur, wenn die sich ändernde öffentliche IP-Adresse wirklich relevant wird. Es wäre ungeschickt, diesen REACH-Proxyserver als *den* „Ankerpunkt“ zum Internetzugriff zu benennen, da dann zyklisch alle ins Internet gerichteten Transportschichtverbindungen abreißen würden. Allerdings macht die Installation durchaus Sinn, wenn beispielsweise nur auf Zielsysteme innerhalb der Wohnung (also auf Rechner im LAN hinter der NAT-Grenze) zugegriffen werden soll. Hiermit ließe sich beispielsweise ein Wartungszugang für Systeme im LAN anbieten, welcher auch den Fall einer spontanen „Zwangstrennung“ mit Zuweisung einer abweichenden IP-Adresse überdauern würde. Außerdem weist diese Topologie den besonderen Vorteil auf, dass das serverseitige „Relay Plugin“ für SIP-basierte Sprachübertragung (siehe Abschnitt 4.3.2.6) die „NAT- und Firewall-Problematik“ von SIP und RTP löst. So können beispielsweise Daheim eingehende Anrufe am REACH-Proxyserver angenommen werden und auf das mobile Endgerät des Anwenders weitergeleitet werden, unabhängig von dessen Aufenthaltsort und zudem handoversicher.

Durch die sich ändernde IP-Adresse des NAT-Gateways tritt allerdings noch ein weiteres Problem zu Tage: Der REACH-Client kann den REACH-Proxyserver nicht anhand einer IP-Adresse vermerken, sondern muss auf seinen DNS-Namen zurückgreifen, welcher beispielsweise über den DynDNS-Dienst mit der jeweils gültigen IP-Adresse verknüpft werden kann. Dieser Mechanismus ist in REACH verfügbar und wurde bereits in Abschnitt 7.3.1 vorgestellt.

Konfiguration: Weiterleitung zweier Ports: Bleibt die Frage zu klären, wie überhaupt ein REACH-Client aus dem öffentlichen Internet einen Zugriff auf einen REACH-Proxyserver hinter einer NAT-Grenze erhalten kann. Dem REACH-Client sei dabei die öffentliche IP-Adresse des NAT-Gateways bereits bekannt. Bezüglich REACH müssen pro Kommunikationskanal zwei TCP-Verbindungen und eine UDP-Assoziation aufgebaut werden. Beide TCP-Verbindungen münden an derselben Annahmestelle am REACH-Proxyserver, welche in den hier dargestellten Beispielen die Portnummer 10000 ist. Bezüglich UDP wird ebenfalls nur eine einzige Annahmestelle benötigt, welche hier ebenfalls beispielhaft auf Port 10000 gebunden wurde. Es gilt also, an der NAT-Grenze zwei Weiterleitungsregeln zu aktivieren, welche für TCP und UDP jeweils Anfragen an Port 10000 „nach innen“ zum REACH-Proxyserver weiterleiten.

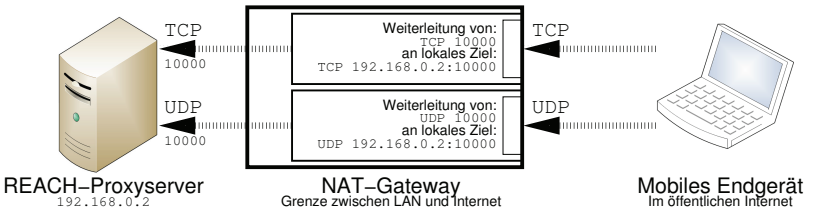

Abbildung 7.9: Damit ein REACH-Client aus dem öffentlichen Internet heraus einen REACH-Proxyserver erreichen kann, welcher sich hinter einer NAT-Grenze in einem privaten Netz befindet, werden zwei „nach innen“ gerichtete Weiterleitungsregeln an der NAT-Grenze benötigt.

Die sich ergebenden Weiterleitungsregeln würden wie in Abbildung 7.9 schematisch dargestellt aussehen. Die rechts seitens der mobilen Endgeräte im Internet initiierten Verbindungen und Assoziationen adressieren das NAT-Gateway, welches hierfür zwei Weiterleitungsregeln befolgt. Diese leiten die angenommenen Daten an den REACH-Proxyserver im angeschlossenen LAN weiter.

Fazit: Auch eine Installation eines REACH-Proxyservers hinter einer NAT-Grenze kann Sinn ergeben und ist bereits durch zwei Weiterleitungsregeln an der NAT-Grenze realisierbar. Dazu wird allerdings Zugriff auf das NAT-Gateway benötigt, was beispielsweise nicht möglich ist, wenn NAT bereits seitens des Providers durchgeführt wird. In diesem Fall wäre der Zugriff nicht möglich.

Entscheidet sich jedoch ein Anwender für den vorgestellten Aufbau, kommt es zu keinen weiteren Beeinträchtigungen bezüglich der Überschreitung der NAT-Grenze. REACH funktioniert auch in diesen Einsatzszenarien.

7.4 Dienstsuche

Das Ziel bei der Begutachtung und Auswahl von REACH-Proxyservern ist die Sicherstellung der Funktionsfähigkeit der „Relay Plugin“-Instanzen des mobilen Endgeräts. Hierzu muss jede Instanz mit mindestens einer Partnerinstanz assoziiert werden, sonst ist eine Weiterleitung der abgefangenen Datenströme nicht möglich. Die Auswahl hat unter Berücksichtigung wenigstens zweier Kriterien zu erfolgen: Welche REACH-Proxyserver haben die benötigten „Relay Plugins“ instantiiert, und welcher dieser REACH-Proxyserver weist momentan eine „herausragende Eignung“ auf? Diese Systeme gilt es ausfindig zu machen und als „Ankerpunkte“ zu verwenden. Dabei soll zu späteren Zeitpunkten auf dann besser geeignete REACH-Proxyserver umgeschwenkt werden können. Andererseits kann bezüglich bereits bestehender Datenübertragungen der „Ankerpunkt“ nicht geändert werden, was parallele Sitzungen mit einer Vielzahl an REACH-Proxyservern erforderlich macht.

Im Rahmen der studentischen Arbeit [Dell08] wurden Verfahren zur Dienstsuche in weltweiten Netzen recherchiert. Die gefundenen Verfahren stammten aus dem Umfeld der verteilten Datenbanken und der „Tauschbörsen“, und erwiesen sich für eine Dienstsuche bei REACH als ungeeignet. So wurde entschieden, die für REACH notwendigen Dienstsuchemechanismen selbst zu implementieren. Hierbei entstand ein Algorithmus, welcher die Auswahl geeigneter REACH-Proxyserver zweistufig durchführt:

1. **Ermittlung gut erreichbarer REACH-Proxyserver:**
   Ein REACH-Proxyserver gilt dann als besonders gut geeignet, wenn er im Sinne des „Zonenkonzepts“ von REACH als „nahe“ identifiziert werden konnte. Bei dieser Klassifizierung werden nur erfolgreich assoziierte Netzzugänge, welche sich zudem im „Neue Daten“-Modus befinden (siehe Abschnitt 5.5.1), berücksichtigt.

   Der REACH-Client kann somit seine (relative) Aufenthaltsposition bestimmen. Alle bekannten REACH-Proxyserver werden anhand ihrer Zonen bewertet. Als Ergebnis erhält der REACH-Client eine Liste an momentan besonders gut erreichbaren REACH-Proxyservern.

2. **Abgleich des Diensteangebots unter Berücksichtigung der Eignung:**
   Jede am REACH-Client bestehende „Relay Plugin“-Instanz weist einen numerischen „Service-Identifikator“, der dessen Typ beschreibt, auf. Weiterhin ist zu jedem REACH-Proxyserver die Menge dessen unterstützter Dienste (welche „Relay Plugins“ wurden instantiiert?) bekannt. Diese Information wurde vorab mittels einer „Servicemap“ (siehe Abschnitt 4.4) an den REACH-Client kommuniziert.

Jede „Relay Plugin“-Instanz am REACH-Client benötigt eine *primäre* Assoziation mit einer Partnerinstanz (ein *Linkbündel*), kann jedoch noch mit einer Vielzahl weiterer Partnerinstanzen sekundäre *Linkbündel* pflegen. Letzteres besagt, dass mehrere REACH-Proxyserver involviert werden, wovon einer der „primäre“ ist und damit den „Ankerpunkt“ für alle zukünftigen Datenströme bilden wird. Mit diesem Konzept wird in REACH das Problem des „Dreiecks-Routings“ behandelt.

Bei jeder für REACH relevanten Positionsänderung des mobilen Endgeräts werden die bestehenden primären *Linkbündel* erneut bewertet und gegebenenfalls durch *Linkbündel* zu besser geeigneten REACH-Proxyservern ersetzt. Die bisherigen primären *Linkbündel* werden zu sekundären *Linkbündeln* und noch solange gepflegt, bis alle enthaltenen *logischen Verbindungen* geschlossen worden sind.

Kann zu einem „Service-Identifikator“ kein herausragender REACH-Proxyserver identifiziert werden, wird ein beliebiger REACH-Proxyserver, welcher den Dienst anbietet, ausgewählt. Sind mehrere Kandidaten verfügbar, werden die gemessenen Umlaufzeiten der relevanten Übertragungsstrecken miteinander verglichen.

Auf Wunsch kann ein einmal gewählter „Ankerpunkt“ auch dauerhaft beibehalten werden. Dies ist für „Relay Plugins“ sinnvoll, welche von einem festen „Ankerpunkt“ profitieren, beispielsweise wenn Rückkanäle benötigt werden (bei FTP beispielsweise). Zusätzlich kann die automatische Wahl des „Ankerpunkts“ umgangen werden, indem der Nutzer einer „Relay Plugin“-Instanz die Verwendung eines bestimmten REACH-Proxyservers vorschreiben kann. So lassen sich ein Zugriff auf Ressourcen in abgeschotteten Netzen realisieren (der „Ankerpunkt“ dient als „Sprungbrett“) oder ein Rückkanal für Fernwartungszwecke schalten. Die hierfür notwendigen Konfigurationsparameter wurden in Abschnitt 4.3.1.3 vorgestellt. In diesen Fällen muss man jedoch die Nachteile des „Dreiecks-Routings“ in Kauf nehmen.

7.5 Kapitelzusammenfassung

Dieses Kapitel beleuchtete das Suchkonzept von REACH, welches die Auswahl und die Erreichbarkeit von REACH-Proxyservern umfasst. Da sich jeder REACH-Client mit einer Vielzahl an REACH-Servern assoziieren kann, wurden die Möglichkeiten einer ortsabhängigen Lastverteilung sowie der Vermeidung von „Dreiecks-Routing“ diskutiert.

Um diese Konzepte zu realisieren, werden zu jedem REACH-Proxyserver „Einzugsgebiete“ (Zonen) definiert. Ein REACH-Client kann somit erkennen, ob ein bestimmter REACH-Proxyserver momentan einen lohnenswerten „Ankerpunkt“ darstellt. Die Aus-

wahl geeigneter REACH-Proxyserver hat jedoch primär unter dem Aspekt des geforderten Diensteangebotes zu erfolgen. Ziel ist es, wenigstens zu jeder „Relay Plugin“-Instanz am REACH-Client eine passende Partnerinstanz an einem REACH-Server zu finden.

Eng verknüpft mit dem „Zonenkonzept“ ist die Adressierungsproblematik, da jeder REACH-Proxyserver über eine Vielzahl an IP-Adressen verfügen kann. Zudem können sich sowohl REACH-Client als auch REACH-Server hinter NAT-Grenzen befinden, sodass die Auswirkungen von NAT auf die Protokolle von REACH diskutiert werden mussten.

8 Validierung und Demonstrator

In diesem Kapitel wird anhand von aufgenommenen Messungen belegt, dass REACH die in Abschnitt 2.1 vorgestellten Anforderungen erfüllt. Dazu werden zwei Messungen vorgestellt, welche die Funktion von REACH in praxisrelevanten Anwendungsszenarien nachweisen. Drei weitere Messungen widmen sich ausgewählten Leistungsmerkmalen von REACH, deren Auswirkungen in angepassten Testumgebungen untersucht worden sind. Anschließend wird beleuchtet, dass sich REACH besonders gut für Demonstrationen vor Publikum eignet. Die dabei zum Einsatz gekommenen Testumgebungen werden jeweils vorgestellt und die erhaltenen Ergebnisse diskutiert.

8.1 Test einzelner Szenarien

Die folgenden Messungen spiegeln praxisrelevante Nutzungsschemata wider, welche mit besonderem Augenmerk auf die Untersuchung von Mobilität ausgewählt worden sind. Zuerst soll jedoch der verwendete Testaufbau vorgestellt werden.

Testaufbau

Als mobiles Endgerät kam ein Laptop mit einem Prozessor des Typs „Intel Pentium 4 Mobile mit 2.2 GHz“ und 768 MiB Arbeitsspeicher zum Einsatz. Als Betriebssystem wurde ein aktuelles Gentoo Linux verwendet. Der Laptop verfügte über eine eingebaute Ethernetschnittstelle mit 100 MBit/s und wurde zusätzlich mit einem USB-WLAN-Adapter des Typs „ZyXEL G-200v2“ mit ZD1211-Chipsatz ausgestattet. Da die verbauten USB-Schnittstellen zu langsam waren, wurden zwei USB 2.0 Schnittstellen mit Hilfe einer „Personal Computer Memory Card International Association“-Karte (PCMCIA-Karte) nachgerüstet. Auf dem Laptop war der REACH-Client für die Verwaltung der beiden Netzzugangsgeräte verantwortlich.

Als WLAN-Basisstationen wurde die vorhandene Infrastruktur der TU Ilmenau benutzt, wobei das verschlüsselte Netz mit dem „Extended Service Set Identifier“ (ESSID) „`802.1X`“ zum Einsatz kam. Als zusätzliche Basisstationen wurden dem Laptop zwei

WLAN-Router des Typs „Asus WL-500g Premium“ angeboten. Diese verfügten außerdem über Ethernetschnittstellen, und konnten dem Laptop damit sowohl über WLAN als auch über Ethernet einen Netzzugang bereit stellen. Durch Ein- und Ausschalten der beiden Basisstationen konnte die Verfügbarkeit der Funknetze beeinflusst werden. Beide Basisstationen wurden über eigenständige Ethernetkabel mit dem Internet verbunden, welches im dargestellten Testaufbau durch den Arbeitsplatzrechner des Autors bereit gestellt wurde. Dieser war mit insgesamt drei Netzwerkkarten ausgestattet, und besaß über die dritte Netzwerkkarte einen Zugang zum Internet. Als Betriebssystem kam ebenfalls Gentoo Linux zum Einsatz, allerdings hatte dieser Rechner keine besonderen Aufgaben zu erfüllen.

Der REACH-Proxyserver stand in einem Rechnerraum und war damit ebenfalls Bestandteil des Campusnetzes. Er besaß einen Prozessor des Typs „Intel Pentium 4 mit 2.4 GHz“, war mit 1 GiB Arbeitsspeicher ausgestattet und wies als Betriebssystem abermals Gentoo Linux auf. Auf diesem System wurde ein REACH-Server gestartet.

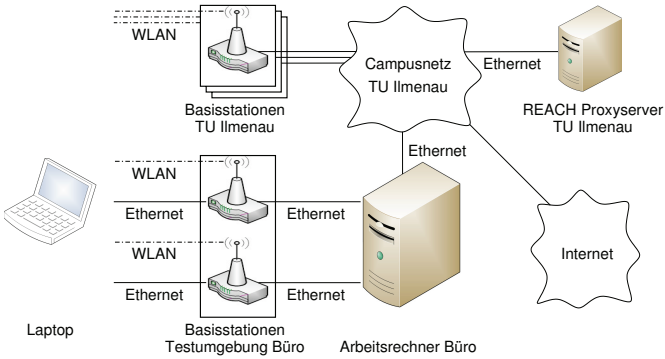

Abbildung 8.1: Testaufbau im Fachgebiet Kommunikationsnetze

Die Testumgebung, welche in Abbildung 8.1 dargestellt wird, eignete sich besonders für praxisorientierte Tests. Da die Basisstationen des Campusnetzes mit einbezogen wurden, waren campusweite Tests möglich. Sowohl „horizontale“ als auch „vertikale Handover“ konnten untersucht werden, was mit den nachfolgend vorgestellten Messungen auch erfolgte.

8.1.1 Rundgang durch den Helmholtzbau

REACH ist in der Lage, unterschiedlichste Netzzugangstechnologien eines mobilen Endgeräts zu benutzen und mit ihnen „horizontale“ und „vertikale Handover“ zu realisieren.

Da der Einsatz von REACH nicht auf eine „künstliche Testumgebung“ beschränkt ist, sollte für praxisorientierte Tests die Kommunikationsinfrastruktur der TU Ilmenau mit einbezogen werden. Als Szenario diente ein Download, welchen der Benutzer an seinem Arbeitsplatz startete. Anschließend nahm er seinen Laptop in die Hand, setzte sich in Bewegung und ging eine Runde durch das Gebäude. Dabei boten die Basisstationen der Campusinstallation die einzige Möglichkeit zum Zugriff auf das Internet.

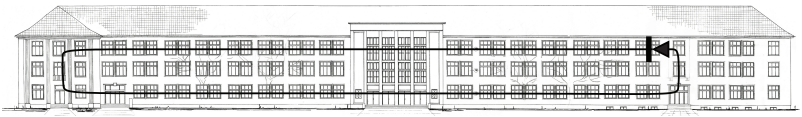

Abbildung 8.2: Die Marschroute durch den Helmholtzbau (das Diagramm stammt aus Beständen der TU Ilmenau, eingescannt und modifiziert)

Aufbau und Ablauf des Tests

Der Laptop befand sich anfangs im Büro H3505 in seiner Dockingstation, wobei eine Kommunikation mittels Ethernet möglich war. Der Laptop wurde wenige Sekunden nach Start der Messung entnommen und durch den Helmholtzbau getragen (siehe Abbildung 8.2). Es ging dabei den Gang entlang, an dessen Ende die Treppe herunter bis zum Erdgeschoss, dort den Gang zurück bis zum entgegengesetzten Ende und dann wieder zurück ins zweite Obergeschoss. Dort wurde der Laptop wieder ins Büro zurückgebracht und in seine Dockingstation gelegt.

Der Handoverentscheider wählte bei Bedarf die Basisstation mit der besten Signalstärke aus, wobei gemäß Einstellung ausschließlich „harte Handover“ durchgeführt worden sind. Ein „horizontaler Handover“ zwischen zwei Basisstationen des Campusnetzes fand nur dann statt, wenn die linear gefilterte Signalstärke der aktuellen Basisstation unter die Abrissschwelle fiel. Die Signalstärke der nachfolgenden Basisstation musste mindestens der Eignungsschwelle entsprechen.

Als Datenstrom wurde ein „Download“, also eine Nutzdatenübertragung mittels TCP in Richtung des Laptops, durchgeführt. Direkt auf dem REACH-Proxyserver wurde ein selbst implementierter TCP-Server gestartet, welcher so schnell es geht Daten in angenommene TCP-Verbindungen schrieb. Auf dem mobilen Endgerät kam der entsprechende TCP-Client zum Einsatz, welcher sich zu besagtem TCP-Server verband, die eingehende Datenrate erfasste und sie periodisch ausgab. Die TCP-Verbindung wurde auf dem Laptop durch den Mechanismus des „transparenten Proxyservers“ abgefangen. Grafik 8.3 visualisiert die sekündlichen Messwerte des TCP-Clients.

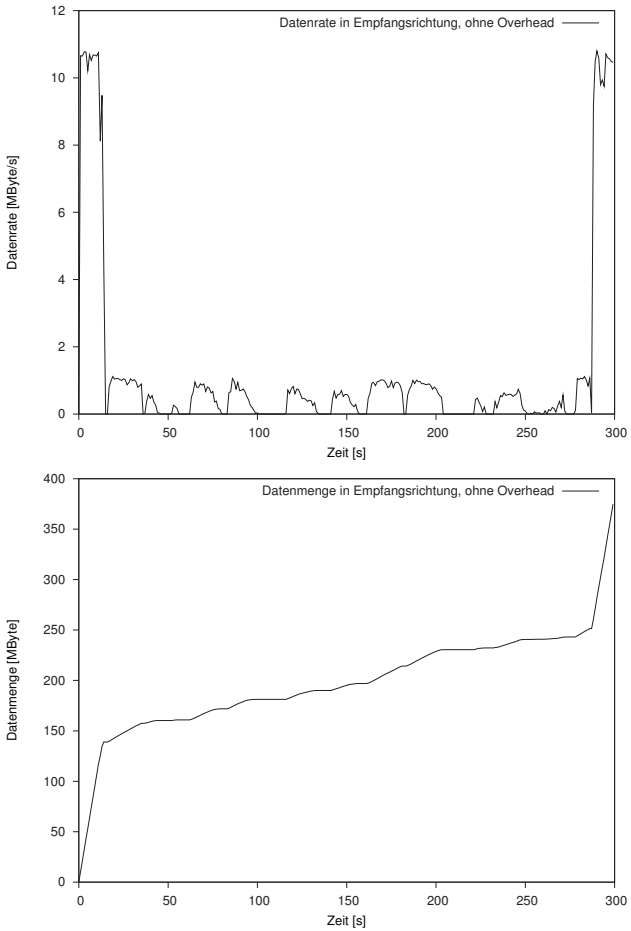

Abbildung 8.3: Empfangene Nutzdatenrate und Nutzdatenmenge

Diskussion

Das obere Diagramm in Abbildung 8.3 zeigt die Nutzdatenrate der abgefangenen TCP-Verbindung. Es ist deutlich erkennbar, dass anfangs Ethernet verfügbar war und somit eine hohe Datenrate von über 10 MByte/s erreicht wurde. Nach Entnahme des Laptops aus seiner Dockingstation brach die Nutzdatenrate kurzzeitig zusammen, da erst der Ersatzlink über WLAN aufgebaut und dann die beim Abriss verloren gegangene

Fragmente erneut übertragen werden mussten („harter vertikaler Handover“).

Deutlich lässt sich erkennen, wie sich der Laptop von Basisstation zu Basisstation „hangelte“. Immer dann, wenn die gefilterte Signalstärke die Abrissschwelle unterschritt, wurde ein erneuter „Scan“ durchgeführt und eine Basisstation für die Neuassoziation ausgewählt. Anschließend mussten jedoch erst noch alle nicht bestätigten Fragmente erneut übertragen werden, was die gezeigten Aussetzer im Nutzdatenfluss verlängerte. Das untere Diagramm von Abbildung 8.3 verdeutlicht, dass die TCP-Verbindung nicht abriss, und es in Zeiten von Isolationssituationen lediglich zu einer Ausbildung von „Plateaus“, also einem Stocken des Datenflusses, kam. Die Basisstationen des Helmholtzbaus bieten selbst keine Mobilitätsunterstützung an, aber in Verbindung mit REACH kann man diese Infrastruktur auch mit mobilen Endgeräten benutzen.

8.1.2 Längerfristige Isolationssituation

Die kritische Zeitspanne von TCP-Verbindungen liegt bei ca. 9 Minuten („Retransmission Timeout“). REACH ist jedoch in der Lage, mit längerfristigen Isolationssituationen umgehen zu können. Es galt daher zu testen, wie sich eine durch REACH abgefangene und gesicherte TCP-Verbindung verhält, wenn eine Isolationssituation länger als diese kritische Zeitspanne anhält.

Aufbau und Ablauf des Tests

Der grundsätzliche Testaufbau und die Art der zu übertragenden Daten war identisch mit der Beschreibung aus Abschnitt 8.1.1. Allerdings wurde der Laptop in diesem Fall nicht bewegt. Als einzige Netzzugangstechnologie stand ihm der USB-WLAN-Adapter zur Verfügung. Da sich die Basisstationen des Testaufbaus in unmittelbarer Nähe des Laptops befanden, wurden ausschließlich diese als Netzzugang ausgewählt.

Als der Datenstrom gestartet wurde, hatte der Handoverentscheider bereits die Assoziation mit der WLAN-Basisstation hergestellt. Die Datenübertragung verlief anfangs ungestört, sie wurde aber nach ca. 6 Minuten durch spontanes Abziehen des USB-WLAN-Adapters unterbrochen. Der Laptop war damit vom Internet isoliert, ein Zustand der für ca. 20 Minuten aufrecht erhalten wurde. Anschließend wurde der Adapter wieder eingesteckt. Die Messung wurde für weitere 4 Minuten fortgeführt und schlussendlich nach insgesamt 30 Minuten abgeschlossen. Die sekündlichen Ausgaben des TCP-Clients werden in Abbildung 8.4 grafisch dargestellt.

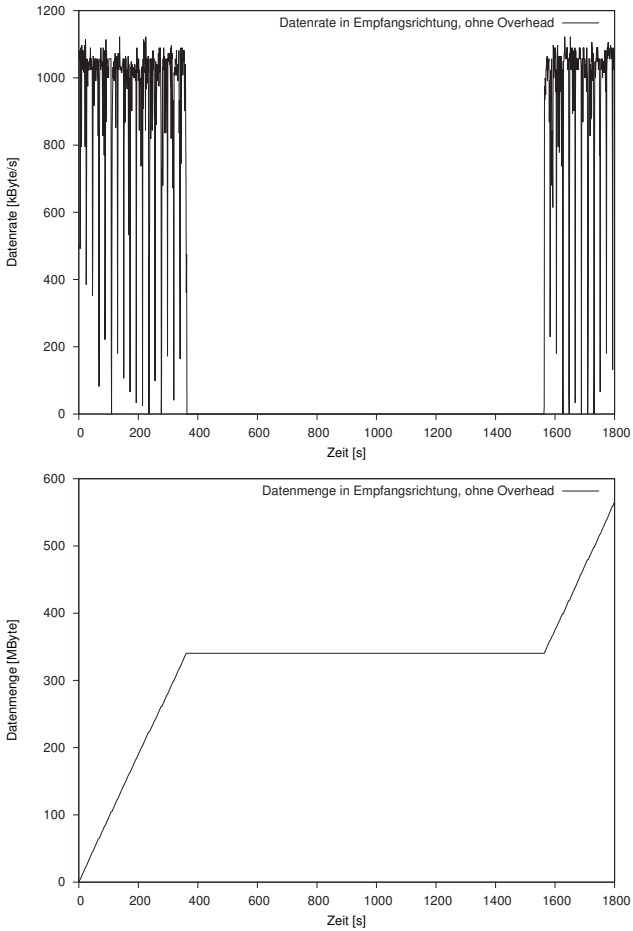

Abbildung 8.4: Empfangene Nutzdatenrate und Nutzdatenmenge

Diskussion

Die Diagramme aus Abbildung 8.4 zeigen deutlich, dass während der 20 minütigen Isolationssituation keine Daten übertragen werden konnten. Es ist allerdings der Verdienst von REACH, dass es dabei zu keinem Abriss der abgefangenen TCP-Verbindung kam. Nachdem der WLAN-Adapter wieder eingesteckt wurde, konnte die Datenübertragung problemlos fortgesetzt werden.

Im oberen Diagramm ist ein zyklisch wiederkehrender Einbruch der Datenrate erkennbar. Ursache war die wiederkehrende Suche nach alternativen Basisstationen, welche alle 20 Sekunden durchgeführt wurde und die Datenrate negativ beeinflusste. Insgesamt wurden ca. 565 Megabyte an Daten aus der TCP-Verbindung entnommen, aber kein einziges Byte an Nutzdaten versendet. Für REACH ist es allerdings unerheblich, ob Nutzdaten im „Up-“ oder im „Downstream“ übertragen werden. Der Datenstrom hätte genauso auch in entgegengesetzter Richtung oder in beide Richtungen gleichermaßen durchgeführt werden können.

8.2 Test verschiedener Betriebsmodi

Die folgenden Messungen sollten den messbaren Unterschied zwischen den drei Betriebsmodi „harter Handover“, „maximale Redundanz“ und „maximale Datenrate“ zeigen. Von besonderem Interesse waren die Auswirkungen auf den Nutzdatenstrom. Es war zu klären, ob eine Kanalbündelung auch wirklich zu einer höheren Datenrate führt, und ob ein redundantes Übertragungsschema vor „Aussetzern“ im Datenfluss schützen kann. Aus den hierbei gewonnenen Ergebnissen lassen sich bei Bedarf Aussagen zu anderen Betriebsmodi, wie den „weichen“ und den „weicheren Handovern“, ableiten.

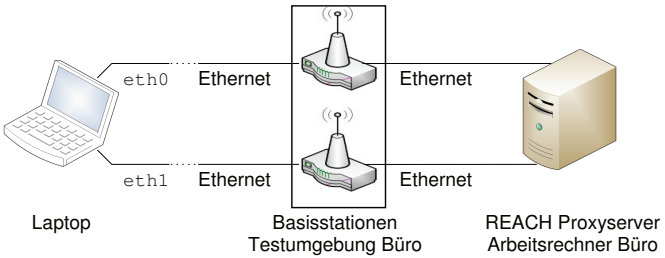

Abbildung 8.5: Auf Ethernet zugeschnittene Testumgebung

Alle drei Messungen wurden mit derselben Testumgebung durchgeführt (dargestellt in Abbildung 8.5). Der Laptop wurde mit einer zweiten Ethernetschnittstelle (eine PCMCIA-Karte) ausgestattet, sodass er über zwei Kabel gleichzeitig angebunden werden konnte. Als problematisch erwies es sich, dass bei einem Kanalbündelungsszenario eine höhere Datenrate erwartet wurde, als eine einzige Ethernetschnittstelle mit 100 MBit/s bewältigen könnte. Daher musste der REACH-Proxyserver mit zwei eigenständigen Kabeln erreicht werden können. Außerdem musste die „Datenquelle“, also der in den vorherigen Messungen verwendete TCP-Server, unbedingt direkt auf dem REACH-Proxyserver

gestartet werden. Andernfalls hätte die Ethernetschnittstelle in Richtung der Datenquelle die Datenrate begrenzt.

Alle Messungen wurden so gestaltet, dass anfangs bereits eine „Ethernetverbindung“ über Schnittstelle `eth0` des Laptops verfügbar war, sodass sich der REACH-Client und der REACH-Server vorab ungestört miteinander assoziieren konnten. Die Messung begann mit dem Start des TCP-Clients auf dem Laptop, welcher wieder als reine Datensenke fungierte. Nach ca. 30 Sekunden wurde das zweite Ethernetkabel in Interface `eth1` eingesteckt, sodass fortan zwei Kommunikationswege zur Verfügung standen. Der zweite Kommunikationsweg wies jedoch eine höhere Priorität auf als der erste Kommunikationsweg, was je nach Betriebsmodus einen Wechsel provozieren sollte. Nach insgesamt 60 Sekunden wurde das erste Ethernetkabel abgezogen, sodass nur noch der „neue“ Kommunikationsweg genutzt werden konnte. Die Messung wurde nach 90 Sekunden beendet.

Die Messergebnisse wurden bei diesen Tests *nicht* durch den TCP-Client (die Datensenke) ermittelt, sondern direkt am REACH-Client abgegriffen. Damit standen auch Informationen bezüglich der einzelnen Kommunikationswege und des Overheads in Sende- sowie in Empfangsrichtung zur Verfügung. Dabei sind diverse weitere Diagramme entstanden, welche in Anhang E nachgeschlagen werden können.

8.2.1 „Harter Handover“

Für diese Messung wurde der Handoverentscheider in den Betriebsmodus des „harten Handovers“ versetzt. In diesem Modus wird immer nur maximal ein Netzzugang verwendet und ein alternativer Netzzugang frühestens dann aufgebaut, wenn der bisherige Netzzugang getrennt worden ist. Dieser Modus ist besonders bei „horizontalen Handovern“ anzutreffen, oder dann, wenn ein bestehender Netzzugang spontan abreißt, aber bislang kein Ersatzlink aufgebaut worden ist.

Diskussion

Wie in Abbildung 8.6 gezeigt wird, erfolgte zum Zeitpunkt 30s ein Wechsel auf den soeben verfügbar gewordenen Netzzugang, was in dessen höherer Priorität begründet lag. Da hierbei ein „harter Handover“ durchgeführt wurde, war kurzzeitig kein Netzzugang verfügbar, und kam es zu einem deutlichen Einbruch der Datenrate. Der Einbruch spiegelt sich im unteren Diagramm in einem Plateau wider. Es kam jedoch zu keiner nachhaltigen negativen Beeinflussung der Nutzdatenrate. Der anschließende Abriss zum Zeitpunkt 60s verlief ohne erkennbare Auswirkungen, da der betroffene Netzzugang nicht mehr verwendet wurde.

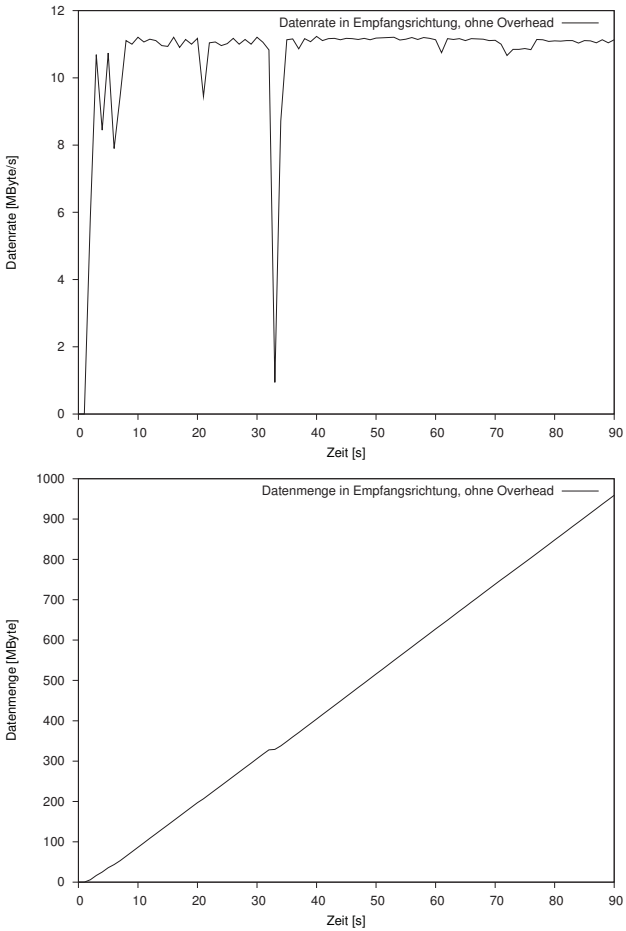

Abbildung 8.6: Empfangene Nutzdatenrate und Nutzdatenmenge

Die Datenverluste verursachten eine deutliche Beeinflussung der Nutzdatenrate, jedoch war in diesem Betriebsmodus die Ressourcenbelegung minimal. Es wurde zu jedem Zeitpunkt maximal ein Kommunikationsweg benutzt. In Anhang E.1 wird dies anhand zusätzlicher Diagramme diskutiert.

8.2.2 Redundanzerhöhung

REACH ist in der Lage, mehrere Netzzugänge gleichzeitig zu benutzen, und die unterschiedlichen Übertragungswege für eine Sicherung gegenüber Verzögerungen des Nutzdatenstroms zu benutzen. Dieser Betriebsmodus ist Bestandteil des „weicheren Handovers“, welcher bei REACH darin besteht, dass ein Ersatzlink noch vor Abriss des Hauptlinks etabliert *und* zur Datenübertragung herangezogen wird. In dieser Zeitspanne, während beide Netzzugänge gleichzeitig verfügbar sind, wird eine redundante Versendung angestrebt. Damit soll einer negativen Beeinflussung des Nutzdatenstroms entgegen gewirkt werden.

Diskussion

Wie es in Abbildung 8.7 verdeutlicht ist, wird ab Zeitpunkt 30s der der „frisch“ etablierte „Zweitlink“ mit in die Datenübertragung einbezogen. Dabei ist kein kurzzeitig tiefes Absinken der Datenrate, welche auf einen Datenverlust schließen ließe, erkennbar. Dies gilt besonders für den Zeitpunkt 60s, zu welchem der erste Netzzugang spontan getrennt wird. Am unteren Diagramm ist ein annähernd lineares Wachstum der empfangenen Nutzdatenmenge erkennbar, welche ohne Ausbildung eines Plateaus erfolgte.

Über den Zeitraum zwischen 30s und 60s hinweg ist ein kleiner Einbruch der Datenrate erkennbar, für den der Autor eine erhöhte Prozessorlast verantwortlich machte. Betrachtet man den anfallenden „Overhead“, welcher in Anhang E.2 grafisch dargestellt wird, erkennt man den Preis der „flüssigen“ Datenübertragung: der Übertragungsaufwand und damit die Belegung von Ressourcen steigt durch die Mehrfachübertragung stark an.

8.2.3 Kanalbündelung

Sind mehrere Netzzugänge gleichzeitig verfügbar, können diese auch im Sinne einer Kanalbündelung verwendet werden. Damit wird in der Summe eine höhere Datenrate ermöglicht, als einer der beteiligten Netzzugänge für sich alleine gesehen erlauben würde. Dabei ist darauf zu achten, dass der sich ergebende Summendatenstrom auch innerhalb des Netzes zum REACH-Proxyserver hin übertragen, und, dass der Nutzdatenstrom anschließend mit ausreichend Übertragungskapazität in Richtung der eigentlichen Server weitergeleitet werden kann.

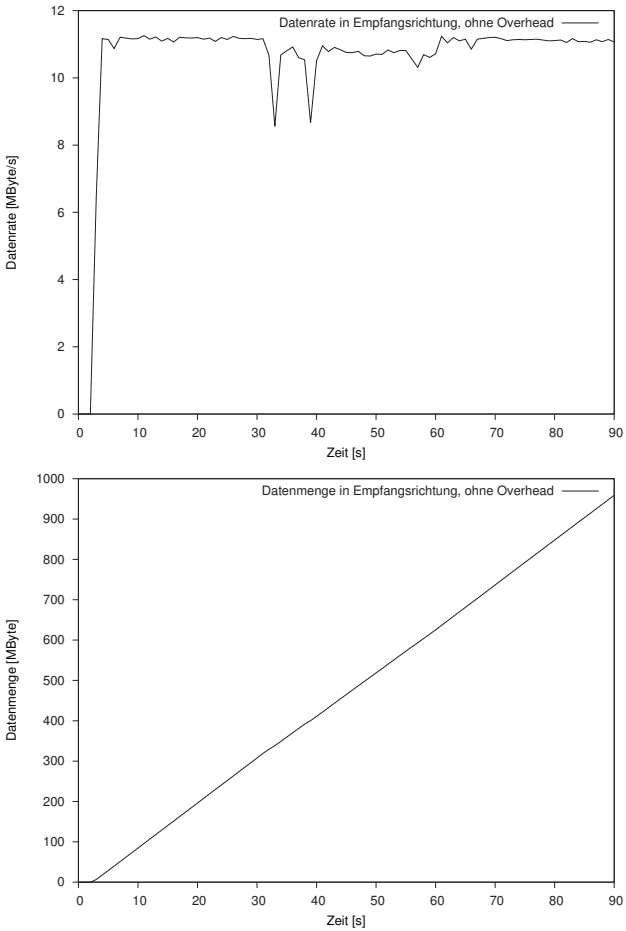

Abbildung 8.7: Empfangene Nutzdatenrate und Nutzdatenmenge

Diskussion

Die aufgenommenen Messergebnisse werden in Abbildung 8.8 dargestellt. Zum Zeitpunkt 30s wurde der „Zweitlink“ verfügbar, welcher sofort mit in die Datenübertragung einbezogen wurde. Da sich hierbei die erreichbare Datenrate erhöhte und während des „Umschaltens“ keine Daten verloren gingen, war erwartungsgemäß kein kurzzeitiger Einbruch der Datenrate messbar. Nach Schaltung der Kanalbündelung wurde eine deutlich

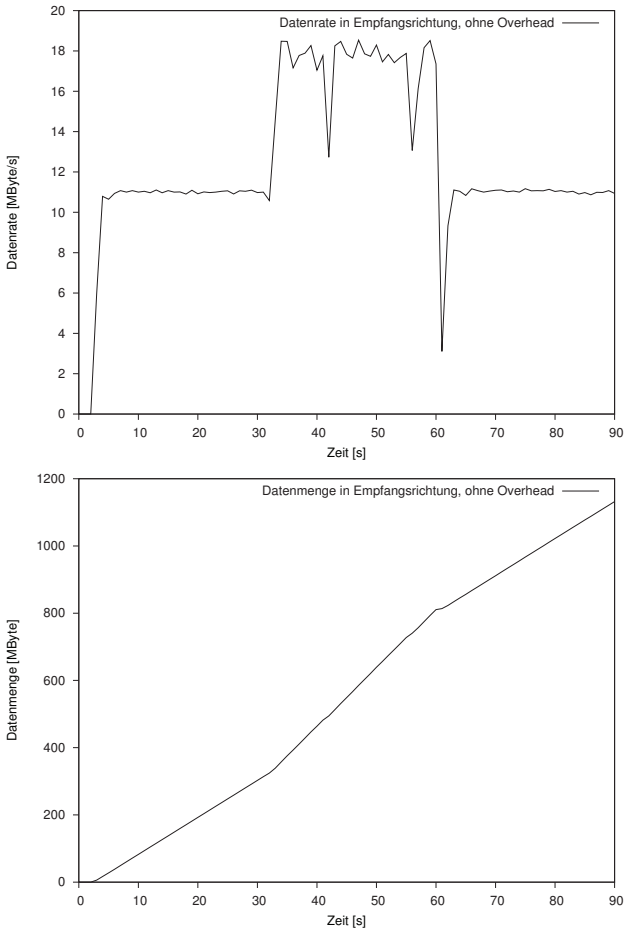

Abbildung 8.8: Empfangene Nutzdatenrate und Nutzdatenmenge

erhöhte Nutzdatenrate verzeichnet, welche allerdings das erwartete Maximum von ca. 2×11 MByte/s nicht erreichte. Die Ursachen hierfür konnten bislang nicht abschließend geklärt werden; in Frage kommen eine erhöhte Prozessorbelastung sowie Laufzeiteffekte bezüglich des Flusssteuerungsmechanismus der Datenstromsicherung. Die erkennbaren spitzen Einbrüche deuten auf kurzzeitige Blockierungen hin, wobei vermutlich die Sende- und Empfangspuffer zu klein waren oder die Umlaufzeit für den Bestätigungsmechanis-

mus nicht korrekt berechnet worden ist (Stichwort: Datenraten-Umlaufzeit-Produkt). Im unteren Diagramm ist dennoch der Anstieg der Datenrate erkennbar.

Eine Kanalbündelung liefert jedoch keinen Schutz vor spontanen Abrissereignissen. Dies erklärt den spitzen Einbruch nach Abziehen des Kabels zum Zeitpunkt 60s. Hierbei trat ein Datenverlust auf, und es kam zu einem Timeout mit anschließender Neuversendung der bislang nicht bestätigten Daten. Der Empfangsdatenstrom kam sichtbar ins Stocken. Betrachtet man den Overhead in Anhang E.3, lässt sich erkennen, dass der Übertragungsaufwand nur geringfügig höher war als der beim Betriebsmodus des „harten Handovers“. Die Ursache für den höheren Overhead lag an der Art der verwendeten Bestätigungen, da beim Kanalbündelungsszenario häufig Lücken im Empfangspuffer entstanden, welche durch „größere“ Bestätigungen selektiv bestätigt werden mussten.

8.3 Systemtests und Demonstrationen

Die bislang gezeigten Komponententests eigneten sich besonders dafür, einzelne Leistungsmerkmale von REACH hervorzuheben. Sie waren allerdings weniger dafür geeignet, um unbedarftem Publikum – das eine Zielgruppe für REACH darstellt – die Problematik samt gebotener Lösung zu vermitteln. Dafür sind funktionale Tests zu bevorzugen, welche die von REACH übernommenen Aufgaben verdeutlichen. Da bereits eine Vielzahl an „Relay Plugins“ zur Verfügung standen und somit eine Vielzahl an Anwendungen unterstützt wurden, konnte REACH dem Publikum anhand gewohnter Anwendungen vorgeführt werden.

Aufbau

Die praktischen Vorführungen fanden selten im Büro des Autors, dem „gewohnten Testumfeld“, statt. Stattdessen musste der Testaufbau an beliebigen Orten aufgestellt werden können. Es ergab sich beispielsweise die in Abbildung 8.9 gezeigte Anordnung.

Für das Publikum sichtbar waren dabei lediglich der Laptop (das mobile Endgerät) sowie die beiden WLAN-Basisstationen, welche formschön auf einem Brett montiert worden waren. Die beiden Basisstationen benötigten Zugriff auf das Internet, was beispielsweise mittels „offener Netzwerkdosen“ realisiert werden konnte. Zudem musste den Basisstationen jeweils eine IP-Adresse zugewiesen werden. Der im Testaufbau im Büro (Abbildung 8.1) vorhandene Arbeitsplatzrechner war in diesen Umgebungen nicht mehr vorgesehen. Als weiterer Unterschied wurden bei den Vorstellungen nicht nur ein, sondern zwei REACH-Proxyserver involviert. Während der erste unverändert seine Aufgaben erfüllte, wurde der zweite ausschließlich für die SIP-basierte Sprachkommunikati-

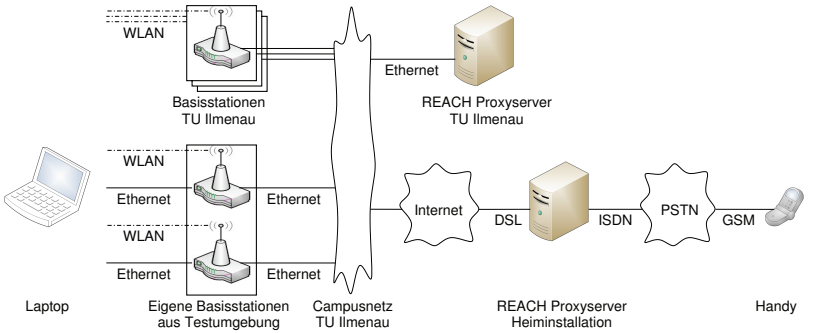

Abbildung 8.9: Aufbau der Testumgebung zu Demonstrationszwecken

on herangezogen. Diese zusätzliche Maschine war Teil der Heiminstallation des Autors und verfügte über mehrere ISDN-Karten samt Anbindung an das öffentliche Telefonnetz („Public Switched Telephone Network“ (PSTN)). Die Vermittlung der Gespräche übernahm die softwarebasierte Vermittlungsstelle Asterisk. Damit war ein Bindeglied zwischen der paketvermittelten Internettelefonie und dem leitungsvermittelten Telefonnetz gegeben. Für das SIP-Telefon auf dem mobilen Endgerät war sogar eine eigene ISDN-Rufnummer im Ilmenauer Ortsnetz verfügbar, sodass Telefongespräche in beide Richtungen aufgebaut werden konnten.

Ablauf der Demonstration

Bei den Vorstellungen des Demonstrators hatte sich eine zweistufige Herangehensweise bewährt. In einem ersten Schritt wurde gezeigt, wie der REACH-Client die verfügbaren Netzzugangsgeräte verwaltet und welche Auswirkungen die unterschiedlichen Zielstellungen auf das Ergebnis der Handoverentscheidung hatten.

Anschließend wurde erklärt, dass die vorgestellten Verwaltungsmechanismen zwar unabdingbar waren, jedoch alleine nicht ausreichen, um eine vollwertige Mobilitätsunterstützung darzustellen. Normalerweise reißen bei einem Wechsel des aktiven Netzzugangs die Transportschichtverbindungen ab. Den Interessenten wurde an dieser Stelle nicht erklärt *wie* REACH dies verhinderte, sondern nur darauf hingewiesen, *dass* das Problem gelöst wurde, was anhand der folgenden Beispiele untermauert werden konnte:

- **Beschallung mit Webradio:** In Wartezeiten, während gerade kein interessierter Besucher vor Ort war, wurde ein Webradio-Datenstrom mit Hilfe des Programms Xine abgespielt. Dieses holte sich Audiodaten mittels HTTP von einem Server

im Internet, wobei die HTTP-Verbindung durch REACH mit Hilfe des Mechanismus der „transparenten Proxyserver“ abgefangen wurde. Die hörbare Musik in Verbindung mit den bunten Visualisierungseffekten eigneten sich dazu, Besucher neugierig zu machen. Die Auswirkung von Handovern und Isolationssituationen auf Webradio konnten ebenfalls vorgeführt werden.

- **Abspielen von YouTube-Videos:** Waren interessierte Besucher anwesend, wurde die Unterbrechungsfreiheit von REACH anhand von YouTube-Videos vorgestellt. Als sehr praktisch erwies sich die Tatsache, dass bei der Wiedergabe der Videos der aktuelle Füllstand des Puffers mit angezeigt wird. Damit waren Isolationssituationen augenblicklich erkennbar, da hierbei der Füllstand des Puffers nicht länger anwuchs. Zudem wurden immer nur wenige Sekunden an Videomaterial gepuffert, sodass man nicht lange zu warten brauchte, bis während einer Isolationssituation das Video einfror. Dabei ließen sich alle Formen von Handovern demonstrieren.

- **Unterbrechungsfreien Telefongespräche mit Wartemelodie:** Im Anschluss an die Videos wurden die Schutzmechanismen von REACH bezüglich SIP-basierter Telefonie vorgeführt. Dazu wurde auf dem Laptop das SIP-Telefon Twinkle gestartet. Der Rufaufbau erfolgte wahlweise seitens eines Handys (aus dem „Festnetz“ heraus) oder ging vom SIP-Telefon auf dem Laptop aus. Die nachfolgend erzeugte Isolationssituation wurde immer mit dem Hinweis kommentiert, dass der mobile Nutzer jetzt mit seinem internetfähigen Telefon in einen Tunnel einfuhr, und damit plötzlich kein Funknetz mehr verfügbar war. Der Gesprächsteilnehmer im „Festnetz“ (das Handy) bekam in weniger als zwei Sekunden eine Wartemelodie zu hören. Anschließend wurde meist in Verbindung mit einem Technologiewechsel wieder „Netz“ zur Verfügung gestellt, wobei die beidseitige Wartemelodie unterbrochen wurde, und man wieder seinen Gesprächpartner hören konnte.

- **Messung der Umlaufzeit:** Erwiesen sich die Besucher als technisch versiert, wurde noch eine Messung der Umlaufzeit mit Hilfe des Kommandos `ping` vorgestellt. Die hierbei ausgetauschten `ICMP`-Pakete wurden durch den VPN-basierten Mechanismus abgefangen. Hierbei konnten besonders die Unterschiede zwischen „harten“ und „weicheren Handovern“ demonstriert werden. Außerdem konnte anhand der Umlaufzeiten zugeordnet werden, welche Netzzugangstechnologien gerade aktiv waren.

Veranstaltungen, bei denen REACH bereits vorgeführt wurde

REACH ist bereits mehrfach öffentlich vorgeführt worden. Die Vorstellung erfolgte dabei meistens nach dem vorgestellten Präsentationsschema, allerdings bestand die Herausforderung darin, auf die unterschiedlichen Wissensstände und Interessen des Publikums einzugehen. So wurde der vorgestellte Demonstrator bereits zu den folgenden Anlässen präsentiert:

- Auf einem Messestand auf der CeBIT 2009 in Hannover,
- beim Tag der offenen Tür der Technischen Universität Ilmenau im Frühjahr 2009 und 2010,
- im Fachgebiet „Integrierte Kommunikationssysteme“ der Technischen Universität Ilmenau im Sommer 2009,
- an der Arbeitsgruppe „Technische Informatik“ der Freien Universität Berlin im Sommer 2009,
- bei der „Summer Academy“ der Technischen Universität Ilmenau im Sommer 2009 und 2010 sowie
- bei einer Vorstellung des Fachgebiets „Kommunikationsnetze“ der Technischen Universität Ilmenau gegenüber der Firma „Hörmann Funkwerk Kölleda“ im Winter 2009/10.

Besonders wertvoll waren die Denkanstöße, welche der Autor auf der CeBIT 2009 erhalten hatte. Hier entstand beispielsweise die Idee der REACH-Box, da diese eine Lösung auf die geschilderten Probleme mehrerer Messebesucher bot. Zudem sah sich der Autor auf der Messe erstmals einem breit gefächertem Publikum gegenüber, wodurch jede einzelne Vorführung anders verlief. Nach und nach stellte sich heraus, auf welche Formulierung und Fakten die Zuschauer mit dem größten Interesse reagierten.

Beim Tag der offenen Tür 2010 wurde das Exponat erstmals einem Kind vorgestellt, wofür sich das Telefonie-Beispiel besonders gut eignete. Sogar Kinder sind sich bewusst, dass Telefongespräche abreißen, wenn plötzlich kein Netz mehr verfügbar ist. Die eingespielte Wartemelodie und die später fortgesetzte Sprachübertragung zeigen eine Funktion, welche die heutzutage verfügbaren Telefonsysteme nicht bieten.

8.4 Kapitelzusammenfassung

In diesem Kapitel wurde anhand real aufgenommener Messungen gezeigt, dass REACH seine besonderen Fähigkeiten auch in praxisorientierten Szenarien unter Beweis stellen konnte. So wurde gezeigt, dass „horizontale“ und „vertikale Handover“ in einer realen Umgebung – dem Ilmenauer Campusnetz – problemlos durchgeführt werden können. Zudem wurde belegt, dass REACH mit längerfristig anhaltenden Isolationssituationen zurecht kommen kann. Anhand weiterer Messungen konnte das Verhalten von REACH bei einem „harten Handover“, bei einer redundanten Datenübertragung sowie bei einem Kanalbündelungsszenario verdeutlicht werden.

Es sind noch eine Vielzahl weiterer Einzeltests denkbar. In [EvSe06] wurden beispielsweise die VPN-basierten „Relay Plugins“ untersucht, wobei ein besonderes Augenmerk auf den Overhead und auf die Paketlaufzeiten gelegt wurde. Die in dieser Ausarbeitung gezeigten fünf Messreihen untermauern jedoch bereits deutlich die Fähigkeiten von REACH. Weiterhin wären Messungen bezüglich der benötigten Assoziations- und Dissoziationszeiten der drei implementierten „Device Backends“ sinnvoll. Rein aus Platzgründen wurde aber auf weitere Tests verzichtet.

Das Besondere an REACH ist jedoch die Tatsache, dass es sich um eine vollständige Lösung handelt, deren zu Grunde liegende Anforderungen sich an denen „normaler Benutzer“ orientieren. REACH eignet sich damit, auch unbedarften Interessenten mobilitätsspezifische Probleme zu verdeutlichen und ihnen gleichzeitig eine funktionierende Lösung zu präsentieren. Bei den zahlreichen praktischen Vorführungen konnte für jeden Wissensstand, vom Kind bis hin zum Netzwerkspezialisten, eine zugeschnittene Einführung samt Demonstration angeboten werden.

9 Ausblick

In der vorliegenden Ausarbeitung zeigen sich weiterführende Themenstellungen, welche an dieser Stelle nochmals vorgestellt werden sollen.

Unterstützung weiterer Netzzugangstechnologien Für die Zukunft deutet sich mit IEEE 802.21 ein einheitlicher Verwaltungsmechanismus für verschiedene Netzzugangstechnologien an. Da hiervon nicht alle erdenklichen Varianten unterstützt werden sollen, stellt IEEE 802.21 kein Ersatz für das Konzept der „Device Backends“ von REACH dar. Es wäre jedoch sinnvoll, ein spezielles „Device Backend“ für IEEE 802.21 zu erstellen.

Untersuchenswert wäre zudem, ob sich HAL und DBUS zur plattformunabhängigen Ansteuerung von Netzzugangsschnittstellen eignen. Falls dies lohnenswert erscheint, sollte ein dediziertes „Device Backend“ erstellt werden.

Weiterhin sind „Device Backends“ denkbar, welche bislang nicht berücksichtigte Netzzugangstechnologien wie Analogmodems oder ISDN verwalten können. Von besonderem Interesse wäre ein „Device Backend“ für einen UMTS-fähigen USB-Adapter, mit dessen Hilfe die REACH-Box zu einem mobilen „UMTS-Gateway“ gemacht werden könnte.

Spezielle „Relay Plugins“ für die REACH-Box Bezüglich der REACH-Box bietet sich die Implementierung eines „transparenten Proxyservers“ für UDP an, was erst dank „junger“ Erweiterungen des Linux-Kernels möglich geworden ist. Darauf aufbauend kann der vorgestellte „Session Border Controller“ (SBC) für SIP-basierte Sprachkommunikation „transparent“ gemacht werden, sodass in den mobilen SIP-Telefonen keine Änderung der Konfigurationsdaten mehr nötig sind. Der Nutzer wählt sich seine PBX dann wie üblich über sein SIP-Telefon aus, und bekommt von der Mobilitätsunterstützung im Normalbetrieb nichts mit.

Weiterhin wäre REACH in der Lage, transaktionsorientierte Dienste ähnlich denen der „Drive-thru“-Architektur [OtKu04] anzubieten. Diesbezüglich könnten „Relay Plugins“ für HTTP, SMTP und POP3 erstellt werden. Ob eine derartige Unterstützung für HTTP jedoch geeignet wäre, auch beispielsweise „YouTube-Videos“ anzuschauen, müsste untersucht werden.

Erweiterung des Handoverentscheiders Der Handoverentscheider bezieht in seiner jetzigen Form noch nicht alle sinnvollen Parameter mit in die Entscheidungsfindung ein. So sollte die Bewertungsfunktion (VHDF) erweitert werden, was aber einen größeren Parametersatz zur Beschreibung der Netzzugänge erforderlich macht. Hierfür bietet es sich an, bezüglich bestimmter Eigenschaften nur eine vordefinierte Auswahl an Möglichkeiten anzubieten.

Es sollte untersucht werden, in welchem Umfang der Handoverentscheider zu einem „lernenden“ Entscheider erweitert werden könnte. Möglich wäre, dass die Entscheidungsschwellen bezüglich drahtlos arbeitender Netzzugangstechnologien automatisch ermittelt werden, oder, dass beispielsweise die für eine Einwahlprozedur benötigte Zeit gemessen und vermerkt wird.

Was ebenfalls noch fehlt, ist eine „normierende“ erste Abbildung der Signalstärke- und Linkgütewerte drahtloser Netzzugangstechnologien. Die exponentielle sowie die inversquadratische Regressionsrechnung dürfen nur mit positiven Werten größer Null „gefüttert“ werden. Für den Fall, dass eine Technologie beispielsweise Werte von -100 bis 100 liefert, sollte hierfür eine lineare Abbildung mittels Korrekturfaktur und -offset auf einen positiven Bereich angeboten werden. Weiterhin kann es sein, dass Signalstärkewerte in dBm geliefert werden. Solche Werte müssen in einem ersten Schritt delogarithmiert werden, bevor anschließend die inversquadratische Regressionsrechnung Sinn ergeben würde. Für diese erste Abbildung sollte daher ein modularer Ansatz gewählt werden, welcher eine Auswahl unterschiedlicher Verfahren erlaubt.

Lastzustand eines REACH-Proxyservers Momentan werden REACH-Proxyserver lediglich anhand ihrer topologischen Nähe, der Menge ihrer angebotenen Dienste sowie der gemessenen Umlaufzeit ausgewählt. Eine Betrachtung des Lastzustands, welcher eine hohe Aussagekraft bezüglich der Eignung als zukünftiger „Ankerpunkt“ aufweist, wird momentan noch nicht durchgeführt. Dazu müssen Bewertungsmetriken und idealerweise schwingungsfreie Auswahlmechanismen erforscht werden.

Angepasste Bedienschnittstellen Die vorgestellten Bedienschnittstellen wurden nicht mit Blick auf unbedarfte Nutzer entwickelt. Für diese sollten spezielle, auf Nutzerfreundlichkeit getrimmte Programme erstellt werden. Auch minimalistische Steuerungsprogramme für die Taskleiste und so genannte „Widgets“ sind möglich.

Lohnenswert wäre es, eine „browserbasierte“ Bedienschnittstelle anzubieten. Mit ihrer Hilfe könnten bei Einsatz der REACH-Box auch solche Endgeräte auf den REACH-Client einwirken, welche über keine REACH-spezifischen Zusatzprogramme verfügen.

Zudem sollten Konzepte zur Authentisierung und Authorisierung für die Wartungs-

schnittstellen vorgesehen werden. Momentan gibt es keinerlei Anmeldemechanismen. Zudem sind noch keine Zugriffsrechte vorgesehen, sodass jeder Nutzer alle Parameter verändern darf.

Nutzerverwaltung und Zugriffsberechtigungen Die für einen Praxiseinsatz wichtige Nutzer- und Endgeräteverwaltung am REACH-Server wurde bislang noch nicht implementiert. So findet momentan kein Abgleich von Endgeräte-Identifikatoren statt, sodass jeder Benutzer eine beliebige Menge an Endgeräten an einem REACH-Server anmelden kann.

Zudem unterscheiden die REACH-Server momentan nur zwei Benutzer, deren Nutzername und Passwort im Quelltext „hart verdrahtet“ worden sind. Hier sollte wenigstens ein auf Konfigurationsdateien basierender Mechanismus implementiert werden, welcher mehrere Benutzer samt Passwörter und erlaubter Gerätekennungen verwaltet. In Szenarien, welche mehrere REACH-Proxyserver involvieren, wäre sogar eine zentrale Nutzerverwaltung sinnvoll.

Dass die Anmeldeinformationen momentan im Klartext über TCP übertragen werden, ist für einen Prototypen noch vertretbar. Für einen Praxiseinsatz sollte stattdessen besser ein „Challenge-Response“-Verfahren involviert werden.

Abspeichern geänderter Konfigurationsdaten Eine weiteres Defizit der aktuellen Implementierung besteht darin, dass zwar die Konfigurationen des REACH-Clients und des REACH-Servers über die Bedienschnittstellen geändert werden können, aber die geänderten Parameter nicht in die Konfigurationsdateien geschrieben werden. So gehen momentan bei einem Neustart alle Änderungen verloren. Besser wäre es, wenn die Dateien nach jeder Parameteränderung gespeichert werden.

Kompatibilität mit IPv6 Eine berechtigte Frage ist, ob REACH auch dann noch funktioniert, wenn das Internet auf IPv6 „umgeschaltet“ wird. Der Kern von REACH ist darauf bereits in weiten Teilen vorbereitet, da die Adressen bezüglich eines jeden REACH-Proxyservers mittels `getaddrinfo()`-Funktion aufgelöst werden, was eine adresstypunabhängige Handhabung ermöglicht. Kritisch sind dagegen die IP-Adressen aus Konfigurationsdateien, da hier teilweise IPv4-Adressen angenommen werden, sodass Änderungen am Quelltext erforderlich werden.

Aufwendiger wird es beim Blick auf die „Device Backends“ und das „Routing Backend“. Erstere müssen die dem jeweiligen Interface zugewiesenen Adressinformationen herauszufinden und an den Kern melden. Hierzu wird der Ausgabedatenstrom diverser Systemkommandos untersucht, wobei bislang nur auf IPv4-Adressen geachtet wird. Das

„Routing Backend“ erfordert ebenfalls Anpassungen, da statt mit Subnetzmasken bei IPv6 mit Präfixlängen gearbeitet werden muss, und zudem IPv6-Adressen eine andere Struktur aufweisen.

Umfassende Anpassungen sind hingegen bezüglich der „Relay Plugins“ zu erwarten. Einzig das SOCKSv5-basierte Schema ist bereits IPv6-tauglich und erfordert keine weiteren Änderungen. Alle anderen Mechanismen verwenden für die Signalisierungsdatenblöcke (beim Aufbau *logischer Verbindungen*) immer vier Byte für ein Adressfeld, was für IPv6 nicht ausreichend ist. Diesbezügliche Änderungen betreffen allerdings nur die „Relay Plugins“, nicht den transportorientierten Kern von REACH.

Die Erweiterung auf Kompatibilität mit IPv6 ist damit möglich, erfordert aber umfassende Erweiterungen und Tests. Damit kann allerdings gewartet werden, bis ein Wechsel auf IPv6 „akut“ wird.

Migration von Sitzungen Ein REACH-Proxyserver tritt als „Ankerpunkt“ in Erscheinung, was zu der Problematik des „Dreiecks-Routings“ führt. Bei SIP-basierter Sprachübertragung könnte jedoch mit Hilfe von SIP-re-`INVITE`-Nachrichten auch ein solcher „Ankerpunkt“ geändert werden. Für REACH könnte dieser Mechanismus adaptiert werden, sodass ein Telefonat bei Mobilität nacheinander über verschiedene REACH-Proxyserver geleitet wird. Hierfür müssen die entsprechenden serverseitigen „Relay Plugins“ miteinander kommunizieren, sich gegenseitig *logische Verbindungen* übergeben und damit eine Sitzungsmigration durchführen können.

Portierung auf andere Plattformen Momentan wurde REACH nur für GNU/Linux-basierte Systeme getestet, was in erster Linie daran liegt, dass nur für diese Zielplattform die erforderlichen „Device Backends“ und das „Routing Backend“ erstellt worden sind. Der Kern von REACH wurde bereits mit Blick auf Plattformunabhängigkeit entwickelt, sodass er weitestgehend auf GNU/Linux-spezifischen Mechanismen verzichtet. Die von REACH benötigten externen Bibliotheken sind für eine Vielzahl an Betriebssystemen verfügbar.

Probleme können in Bezug auf einzelne „Relay Plugins“ auftreten, da beispielsweise Microsoft Windows den Mechanismus der „transparenten Proxyserver“ nicht anbietet.

Bislang wurde REACH noch nicht auf einem „64 Bit-System“ getestet. Zwar sind keine prinzipiellen Probleme zu erwarten, aber es ist wahrscheinlich, dass sich Programmierfehler zeigen, wenn stellenweise ein „32 Bit-System“ angenommen worden ist.

10 Zusammenfassung

In dieser Dissertation wurde das Ziel verfolgt, eine praktikable Mobilitätsunterstützung für den Internetzugang mobiler Endgeräte zu entwickeln. Dabei galt die wesentliche Einschränkung, dass keine Modifikationen der Anwendungen, der Betriebssysteme oder der Netzwerkkomponenten, welche sich heutzutage im Einsatz befinden, benötigt werden dürfen. Zudem sollte die gesuchte Lösung mit einer Vielzahl an Anwendungen harmonieren, und wenn möglich, sogar die Benutzer mit in die Mobilitätsunterstützung einbeziehen.

In Kapitel 2 wurden Anforderungen definiert, welche die gesuchte Mobilitätsunterstützung zu erfüllen hatte. Weiterhin wurde festgestellt, dass Mechanismen zum Schutz von Datenströmen gegenüber Handoverereignissen sowie ein integrierter Handoverentscheider und eine vollständige Kontrolle über die Netzzugangsgeräte des mobilen Endgeräts erforderlich sind. Die Benutzer benötigen eine Bedienschnittstelle, welche allerdings optional sein musste, da die Komponenten der Mobilitätsunterstützung im Sinne eines „Systemdienstes“ im Hintergrund laufen und auch auf „eingebetteten Systemen“ ohne Bildschirm einsetzbar sein sollten. Nach Vorstellung der Anforderungen wurde auf bereits existierende Mobilitätsunterstützungen eingegangen, wobei sich gezeigt hatte, dass lediglich die Middleware-basierten Varianten ohne Modifikationen der Anwendungen und der Betriebssysteme auskommen können.

Da zwar keine der verfügbaren Architekturen alle gestellten Anforderungen unterstützte, es sich jedoch abzeichnete, dass ein Middleware-basierter Ansatz dazu in der Lage wäre, wurde die „Roaming-Enabled Architecture“ (REACH) entwickelt und in Kapitel 3 vorgestellt. REACH setzt Proxyserver ein, welche als „Ankerpunkte“ für die Datenübertragung in Richtung der Server im Internet in Erscheinung treten. Die Middleware wird durch zwei Softwarekomponenten „aufgespannt“, wobei auf den mobilen Endgeräten der REACH-Client und auf den Proxyservern der REACH-Server installiert und gestartet wird. Mit REACH ist es sogar möglich, dass die mobilen Endgeräte keinerlei Änderungen erfordern, was hier als „Netzmobilität“ eingeführt worden ist. Hierbei übernimmt eine externe REACH-Box die Verwaltung von Netzzugängen sowie der Sicherung der Datenströme und Sitzungen.

Ein zentrale Fragestellung bei Middleware-basierten Mobilitätsunterstützungen be-

zieht sich auf das Zusammenspiel zwischen Komponenten der Middleware und den Anwendungen, was umfassend in Kapitel 4 beleuchtet worden ist. Die Anwendungsdatenströme müssen dazu abgefangen werden, wofür unterschiedliche Mechanismen verwendet werden können, von denen manche sogar eine Unterstützung auf Ebene der Benutzer ermöglichen. Um die Eignung der jeweiligen Mechanismen bewerten zu können, wurden typische Nutzungsszenarien herausgearbeitet. Wie sich herausstellte, gibt es keinen *besten* Mechanismus, sondern jeder weist spezielle Vor-, aber immer auch Nachteile auf. Auf dieser Erkenntnis fußt das Dienstekonzept von REACH, welches ein kombiniertes Angebot unterschiedlicher Dienste vorsieht, sodass sich die Vorteile der einzelnen Mechanismen ergänzen und sich ihre Nachteile gegenseitig überdecken. Für die folgenden Mechanismen wurden bereits funktionierende „Relay Plugins“ (Dienste für REACH) erstellt, welche umfassend erläutert worden sind:

- Die Proxyprotokolle SOCKSv4 und SOCKSv5
- „Transparente Proxyserver“ für TCP (UDP denkbar)
- Weiterleitung von Ports, ein- und ausgehend für TCP und UDP
- VPN-basiert, der Netzzugangs- oder Vermittlungsschicht zugeordnet
- SIP-basierte Telefonie (transparenter Betriebsmodus denkbar)

Diese „Relay Plugins“ sind dynamisch durch den Benutzer auswähl- und ladbar, und setzen auf den transportorientierten Kern von REACH auf, dessen Aufgaben in Kapitel 5 beschrieben worden sind. REACH unterscheidet stromorientierte und paketorientierte *logische Verbindungen*, welche mit TCP und UDP vergleichbar sind, jedoch keine Probleme mit mobilitätsspezifischen Gegebenheiten aufweisen. Solche *logischen Verbindungen* werden zu *Linkbündeln* gruppiert, welche wiederum Sitzungen zwischen jeweils zwei miteinander assoziierten „Relay Plugin“-Instanzen darstellen. Da jedes „Relay Plugin“ mit einer Vielzahl solcher *Linkbündel* umgehen kann, können auch mehrere REACH-Proxyserver gleichzeitig angesprochen werden, was den negativen Effekten von „Dreiecks-Routing“ entgegen wirkt. Anschließend wurden die Multiplexer für beide Formen der *logischen Verbindungen* vorgestellt und die beim Auf- und Abbau aufgetretenen Herausforderungen beleuchtet. Es wurden „opportunistische Handshakes“ vorgestellt und auf die Notwendigkeit von Flusssteuerungsmechanismen eingegangen. Unterhalb der Multiplexer befinden sich die Sicherungsinstanzen, welche den negativen Auswirkungen auf der mobilen Teilstrecke entgegen wirken. Hierbei mussten Datenverluste, Duplikate und

Reihenfolgevertauschungen berücksichtigt werden. Isolationssituationen stellten eine besondere Herausforderung dar, da sich die beteiligten Instanzen anschließend wieder synchronisieren müssen. Es wurde beschrieben, wie die „gemultiplexten“ und gesicherten Datenströme schlussendlich mittels nicht modifiziertem TCP und UDP übertragen werden. Dabei stand das Wegewahlproblem zur Debatte, da REACH eine simultane Übertragung von Nutzdaten über mehrere Netzzugänge unterstützen sollte. Hierfür wurden quelladressbasierte und zieladressbasierte Wegewahl beleuchtet, wobei bei letzterer noch die Möglichkeit bestehen konnte, dass senderseitig Sockets an Netzzugangsgeräte gebunden werden.

Das Zugangskonzept von REACH, welches die Handoverentscheidung, die Manipulation von Routingtabellen und die Verwaltung von Netzzugangsgeräten und Netzzugangspunkten umfasst, war Schwerpunkt von Kapitel 6. Zuallererst wurden Kriterien vorgestellt, anhand derer sich Netzzugänge unterscheiden und bewerten lassen. Da für die Entscheidungsfindung eine Zielstellung benötigt wird, folgte die Vorstellung einer denkbaren Auswahl. Das vorgestellte „Providerkonzept“ erlaubt es, dass der Handoverentscheider eine kombinierte Betrachtung von Netzzugangsgeräten und Netzzugangspunkten durchführen kann. Anschließend wurden verschiedene Formen von Handoverentscheidern und deren Betriebsmodi vorgestellt, wobei REACH einen prädiktiven Handoverentscheider mit der Möglichkeit „weicherer Handover“ und Kanalbündelungsszenarien anbietet. Dafür wurde ein auf linearer Regressionsrechnung basierender Vorhersagemechanismus vorgestellt, dessen Ausreißerbehandlung „Fadingeffekte“ herausrechnen kann, wobei sowohl innerhalb als auch außerhalb von Gebäuden überzeugende Ergebnisse bezüglich WLAN erzielt werden konnten. Es wurde begründet, dass für die Ansteuerung von Netzzugangsgeräten und die Manipulation von Routingtabellen plattformspezifische Mechanismen involviert werden müssen. Diese gliedert REACH in so genannte „Backends“ aus, um einen plattformunabhängigen Kern zu erhalten. Es wurden „Device Backends“ für Ethernet, WLAN und für mittels Bluetooth angebundene Mobiltelefone vorgestellt. Das präsentierte „Routing Backend“ erlaubt unter GNU/Linux alle drei Varianten der Wegewahl.

Damit mobile Endgeräte zu mehreren REACH-Proxyservern gleichzeitig Sitzungen aufbauen können, wurde in Kapitel 7 das Suchkonzept von REACH vorgestellt. Es wurde diskutiert, dass sich mit einer ortsabhängigen Proxyserverwahl die negativen Effekte von „Dreiecks-Routing“ behandeln lassen und eine „geographische Lastverteilung“ möglich wird. Für die dafür notwendige Positionsbestimmung wurden Möglichkeiten aufgezeigt, wobei sich für REACH eine Betrachtung der gerade involvierten Netzzugangspunkte als geeignet erwiesen hat. Ein jeder REACH-Proxyserver spannt Einzugsgebiete (Zonen) auf, anhand derer ein mobiles Endgerät erkennen kann, ob ein bestimmter Proxyserver gerade als „nah“ gilt und bevorzugt verwendet werden sollte. Dabei wurde die Erreich-

barkeit von REACH-Proxyservern diskutiert, denn abhängig vom Wegewahlschema und dem Aufenthaltsort des mobilen Endgeräts müssen unterschiedliche IP-Adressen involviert oder NAT-Grenzen überschritten werden. Es wurde begründet, dass die Protokolle von REACH keine Probleme bei der Überschreitung von NAT-Grenzen aufweisen. Schlussendlich wurde der Dienstsuchemechanismus von REACH vorgestellt, welcher das Dienstangebot mehrerer REACH-Proxyserver mit einbeziehen kann.

Kapitel 8 stellte die Testumgebung von REACH vor, welche gleichzeitig auch als Demonstrator für Präsentationen herangezogen werden konnte. So wurde REACH bereits zu diversen Anlässen vorgestellt, wobei sich die CeBIT 2009 als am publikumswirksamsten erwiesen hatte. Da REACH eine umfassende Mobilitätsunterstützung anbietet, erwiesen sich Systemtests anschaulicher als Komponententests. Es wurde jedoch auch das beobachtbare Verhalten abgefangener TCP- und UDP-basierter Datenübertragungen untersucht, wobei besonders der Einfluss von Handoverereignissen im Vordergrund stand.

Mit Kapitel 9 wurde ein Ausblick auf weiterführende Problemstellungen, welche durch die vorliegende Dissertation zu Tage getreten sind, aufgeführt.

Der nachfolgende Anhang widmet sich einer vertiefenden Darstellung einzelner Aspekte, welche nicht mit in den Hauptteil integriert werden sollten. So befindet sich hier eine Beschreibung der entwickelten Kommunikationsprotokolle, welche die handoverresistente Kommunikation zwischen REACH-Client und REACH-Server ermöglichen. Es folgen Details zur Ansteuerung und Funktionsweise der drei erstellten „Device Backends“ und des „Routing Backends“, gefolgt von einer umfassenden Diskussion der Messreihen, welche zur Parameterfindung des prädiktiv arbeitenden Handoverentscheiders beigetragen haben. Anschließend wird anhand weiterer Messreihen der durch REACH verursachte zusätzliche Übertragungsaufwand (Overhead) diskutiert. Die Ausarbeitung schließt mit einer Vorstellung der im Demonstrator verwendeten Komponenten.

In Tabelle 10.1 wird ein abschließender Vergleich zwischen REACH und anderen Middleware-basierten Mobilitätsunterstützungen aufgeführt. REACH erfüllt als einziges alle gestellten Anforderungen und bietet eine Mobilitätsunterstützung an, welche mit heutigen Anwendungen in heutigen Netzen funktioniert. Der entstandene Demonstrator eignet sich sowohl für ein unbedarftes Publikum als auch für Experten, wobei beiden Gruppen die wesentlichen Aspekte von REACH anschaulich vermittelt werden können.

Ansatz:	Bewertungskriterium:	Keine Einschränkungen und Modifikationen:	der Anwendungen, Betriebssysteme und Server	der Netzzugangsgeräte, -punkte und des Kernnetzes	„Blackbox“-Szenario („Netzmobilität“)	Flexibles und kombinierbares Dienstangebot	Sicherungsmechanismen:	Minimierung von Ausfallzeiten	Kanalbündelungsszenario	Isolationsunterstützung	Betrachtung der Nutzerebene	Integrierte Verwaltungsmechanismen:	Netzzugangsgeräte	Handoverentscheider	Bedienschnittstelle verfügbar	Kontextabhängiges Verhalten, Ortsabhängigkeit
MSOCKS			−	+	−	−		−	−	−	−		s	s	s	−
USHA			+	+	s	−		+	−	−	−		+	−	s	−
iMASH			−	+	−	−		s	s	s	−		s	+	s	s
PLASTIC			−	+	−	−		+	−	s	−		+	+	s	+
Drive-thru			+	+	+	+		−	−	+	−		+	+	s	+
MUM			−	+	−	−		+	−	+	−		+	+	s	+
REACH			+	+	+	+		+	+	+	+		+	+	+	+

(+) unterstützt, (−) nicht unterstützt
(s) spekulativ: möglich, keine Erwähnung bekannt

Tabelle 10.1: Beim Vergleich mit den bestehenden Middleware-basierten Mobilitätsunterstützungen zeigte sich, dass nur REACH sämtliche formulierten Anforderungen gleichzeitig erfüllt.

Anhang

A Protokolle des Transportkonzepts

In diesem Kapitel werden die PDUs der Protokolle des transportorientierten Kerns von REACH vorgestellt (siehe Kapitel 5). Begonnen wird mit den Protokollen der beiden Multiplexer, wobei das „Stream Control Protocol for REACH“ (SCPR) für die stromorientierten und das „Packet Control Protocol for REACH“ (PCPR) für die paketorientierten *logischen Verbindungen* von REACH zuständig ist. Anschließend wird auf die Protokolle der beiden Sicherungsinstanzen eingegangen. Hier ist das „Stream Protection Protocol for REACH“ (SPPR) für die Sicherung des stromorientierten Summendatenstroms zuständig, während das „Packet Protection Protocol for REACH“ (PPPR) für die Sicherung des paketorientierten Summendatenstroms entwickelt worden ist.

A.1 Protokolle der Multiplexer

Während ein stromorientierter Multiplexer eine SCPR-Protokollinstanz darstellt, entspricht ein paketorientierter Multiplexer einer PCPR-Protokollinstanz. Beide Protokolle werden jetzt nacheinander vorgestellt.

A.1.1 Das „Stream Control Protocol for REACH“ (SCPR)

Das SCPR ist für die Verwaltung und die Datenübertragung der stromorientierten *logischen Verbindungen* zuständig. Zwei miteinander assoziierte Multiplexer sind mittels eines einzigen Summendatenstroms miteinander verbunden.

A.1.1.1 Übersicht aller SCPR-PDUs

Mit Hilfe des ersten Bytes einer jeden SCPR-PDU kann der genaue PDU-Typ ermittelt werden. Er wird gemäß Abbildung A.1 anhand der drei höchstwertigen Bits bestimmt. Die folgenden fünf Bits stellen das Initiator-Bit sowie die ersten vier Bits des Identifikators (ID) der *logischen Verbindung* dar.

In Tabelle A.1 sind alle SCPR-PDUs aufgeführt. Nur die drei `SCPR_DATA`-PDUs können Nutzdaten der höheren Schicht transportieren. Eine `SCPR_SYN`-PDU signalisiert eine

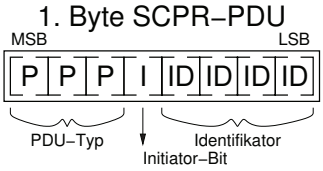

Abbildung A.1: Felder des ersten Bytes einer SCPR-PDU

neue stromorientierte *logische Verbindung*. Hierbei kann ein Signalisierungsdatenblock übergeben werden, welcher unter anderem den Identifikator des übergeordneten *Linkbündels* benennt. So kann der gerufene Multiplexer die eingehende *logische Verbindung* an das zugehörige *Linkbündel* und damit an die korrekte „Relay Plugin"-Instanz übergeben.

P-Bits	SCPR-PDU	ID-Bits	SCPR-Headerlänge	Nutzdaten
000	SCPR_SYN	unbenutzt	10-265 Byte	Nein
001	SCPR_CLOSE	unbenutzt	4 Byte	Nein
010	SCPR_DATA4	4/4 Bits ID	3 Byte	Ja
011	SCPR_DATA12	4/12 Bits ID	4 Byte	Ja
100	SCPR_DATA20	4/20 Bits ID	5 Byte	Ja
101	SCPR_FLOW4	4/4 Bits ID	2 Byte	Nein
110	SCPR_FLOW12	4/12 Bits ID	3 Byte	Nein
111	SCPR_FLOW20	4/20 Bits ID	4 Byte	Nein

Tabelle A.1: Typen an SCPR-PDUs

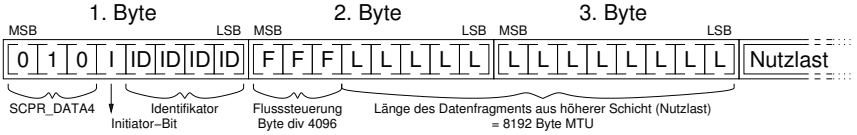

Abbildung A.2: Der Protokollkopf einer SCPR_DATA4-PDU besteht aus mehreren Feldern. Die „Maximum Transmission Unit" (MTU) beträgt 8192 Byte.

SCPR_SYN

Anzeige einer neuen *logischen Verbindung*.

- 1 Byte SCPR-PDU-Typ
- 3 Byte Verbindungsidentifikator (ID)
- 1 Byte Länge des nachfolgenden Signalisierungsblocks (variabler Anteil)

- 5-260 Byte Signalisierungsblock

`SCPR_CLOSE`

Beendigungsanzeige einer *logischen Verbindung*

- 1 Byte SCPR-PDU-Typ
- 3 Byte Verbindungsidentifikator (ID)

`SCPR_DATA4`

Daten-PDU für die Verbindungsidentifikatoren 0-15.

- 1 Byte SCPR-PDU-Typ, incl. Initiator-Bit und 4 Bits ID
- 2 Byte Flusssteuerung und Längenfeld Nutzdaten:
  - 3 Bits Flusssteuerung ($\mathtt{F} \in [0 \ldots 7]$, $\times 4096$ Byte)
  - 13 Bits Längenfeld Nutzdaten ($\mathtt{L} \in [1 \ldots 8192]$ Byte)
- (L) Byte Datenfragment aus höherer Schicht

`SCPR_DATA12`

Daten-PDU für die Verbindungsidentifikatoren 16-4095.

- 1 Byte SCPR-PDU-Typ, incl. Initiator-Bit und 4 Bits ID
- 1 Byte mit weiteren 8 Bits ID
- 2 Byte Flusssteuerung und Längenfeld Nutzdaten:
  - 3 Bits Flusssteuerung ($\mathtt{F} \in [0 \ldots 7]$, $\times 4096$ Byte)
  - 13 Bits Längenfeld Nutzdaten ($\mathtt{L} \in [1 \ldots 8192]$ Byte)
- (L) Byte Datenfragment aus höherer Schicht

`SCPR_DATA20`

Daten-PDU für die Verbindungsidentifikatoren 4096-1048575.

- 1 Byte SCPR-PDU-Typ, incl. Initiator-Bit und 4 Bits ID
- 2 Byte mit weiteren 16 Bits ID
- 2 Byte Flusssteuerung und Längenfeld Nutzdaten:
  - 3 Bits Flusssteuerung ($\mathtt{F} \in [0 \ldots 7]$, $\times 4096$ Byte)
  - 13 Bits Längenfeld Nutzdaten ($\mathtt{L} \in [1 \ldots 8192]$ Byte)
- (L) Byte Datenfragment aus höherer Schicht

`SCPR_FLOW4`
: Reine Flusssteuerungs-PDU für die Verbindungsidentifikatoren 0-15.

- 1 Byte SCPR-PDU-Typ, incl. Initiator-Bit und 4 Bits ID
- 1 Byte Flusssteuerung ($\texttt{F} \in [1 \ldots 256]$, $\times 4096$ Byte)

`SCPR_FLOW12`
: Reine Flusssteuerungs-PDU für die Verbindungsidentifikatoren 16-4095.

- 1 Byte SCPR-PDU-Typ, incl. Initiator-Bit und 4 Bits ID
- 1 Byte mit weiteren 8 Bits ID
- 1 Byte Flusssteuerung ($\texttt{F} \in [1 \ldots 256]$, $\times 4096$ Byte)

`SCPR_FLOW20`
: Reine Flusssteuerungs-PDU für die Verbindungsidentifikatoren 4096-1048575.

- 1 Byte SCPR-PDU-Typ, incl. Initiator-Bit und 4 Bits ID
- 2 Byte mit weiteren 16 Bits ID
- 1 Byte Flusssteuerung ($\texttt{F} \in [1 \ldots 256]$, $\times 4096$ Byte)

A.1.1.2 Automatengraph SCPR

Der Automatengraph einer SCPR-Protokollinstanz wird in Abbildung A.3 dargestellt. Sowohl die Rolle als Initiator als auch als Gerufener wird modelliert.

A.1.2 Das „Packet Control Protocol for REACH“ (PCPR)

Das PCPR ist für die Verwaltung und die Datenübertragung der paketorientierten *logischen Verbindungen* zuständig. Zwei assoziierte Multiplexer sind mittels eines Paketstroms miteinander „verbunden“.

A.1.2.1 Übersicht aller PCPR-PDUs

Auch beim PCPR kann bereits anhand des ersten Bytes einer jeden PCPR-PDU der genaue PDU-Typ ermittelt werden. Der primäre PDU-Typ wird gemäß Abbildung A.4 durch die zwei höchstwertigen Bits bestimmt. Sind beide Bits gesetzt, bestimmen die vier `ID`-Bits den sekundären PDU-Typ. Das dargestellte „Protection Indication“-Bit (PI) ist für den Multiplexer ohne Belang, da es für die paketorientiert arbeitende Sicherungsinstanz reserviert worden ist.

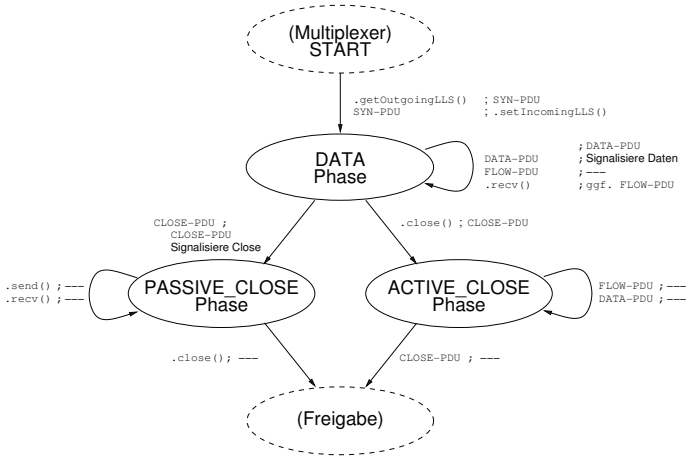

Abbildung A.3: Dieser Automatengraph repräsentiert eine SCPR-Protokollinstanz. Er ist jedoch *unvollständig*, da aus Gründen der Übersichtlichkeit nicht jeder implementierte Zustandsübergang dargestellt wird.

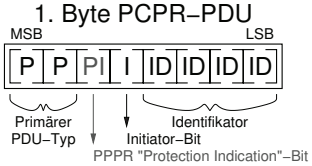

Abbildung A.4: Felder des PCPR-Protokollkopfes

In Tabelle A.2 sind die vier Möglichkeiten primärer PCPR-PDUs aufgeführt. Die drei `PCPR_DATA`-PDUs dienen dem Transport von Nutzdaten der höheren Schicht und fügen zu jedem Paket einen geringen Overhead von nur 1-3 Byte hinzu, abhängig vom Identifikator der *logischen Verbindung* (ID). Sind beide `P`-Bits gesetzt, muss Tabelle A.3 zur Bestimmung des sekundären PCPR-PDU-Typs herangezogen werden. Im Vergleich zum SCPR werden hier viele PDUs unterschieden, da der paketorientierte Übertragungsweg noch für weitere Aufgaben herangezogen wird. So gibt es PDUs für die Durchführung der Isolationserkennung, „Keep-Alive“-PDUs für die Abrisserkennung sowie PDUs zur Bestimmung der Umlaufzeiten. Obschon der paketorientierte Multiplexer als PCPR-Instanz angesehen werden kann, implementiert er nicht alle der hier aufgeführten PDU-Typen. Einige werden, wie genannt, ausschließlich von Protokollinstanzen unterhalb des Multiplexers erstellt und ausgewertet.

P-Bits	PCPR-PDU	ID-Bits	PCPR-Headerlänge	Nutzdaten
00	PCPR_DATA4	4/4 Bits ID	1 Byte	Ja
01	PCPR_DATA12	4/12 Bits ID	2 Byte	Ja
10	PCPR_DATA20	4/20 Bits ID	3 Byte	Ja
11	siehe sekundärer Typ	Sekundärer Typ	siehe sekundärer Typ	evtl.

Tabelle A.2: Primäre PCPR-Pakettypen

ID-Bits	PCPR-PDU	PCPR-Headerlänge	Nutzdaten
0000	PCPR_AUTH	1 Byte	Anmeldung
0001	PCPR_SYN	14-269 Byte	Nein
0010	PCPR_SYNDATA	14-269 Byte	Ja
0011	PCPR_SYNACK	8 Byte	Nein
0100	PCPR_SYNACKDATA	8 Byte	Ja
0101	PCPR_SYNACKACK	8 Byte	Nein
0110	PCPR_CLOSE	8 Byte	Nein
0111	PCPR_CLOSEACK	8 Byte	Nein
1000	PCPR_ISOPROBE	1 Byte	Nein
1001	PCPR_RTTREQUEST	17 Byte	Nein
1010	PCPR_RTTREPLY	13 Byte	Nein
1011	PCPR_KEEPALIVE	1 Byte	Nein
1100	nicht vergeben	–	–
1101	nicht vergeben	–	–
1110	nicht vergeben	–	–
1111	nicht vergeben	–	–

Tabelle A.3: Sekundäre PCPR-Pakettypen

PCPR_DATA4

Daten-PDU für die Verbindungsidentifikatoren 0-15. Nur ein Byte Overhead pro Nutzlast.

- 1 Byte PCPR-PDU-Typ, incl. Initiator-Bit und 4 Bits ID
- Rest: Gekapseltes Paket aus höherer Schicht

PCPR_DATA12

Daten-PDU für die Verbindungsidentifikatoren 16-4095. Zwei Byte Overhead pro Nutzlast.

- 1 Byte PCPR-PDU-Typ, incl. Initiator-Bit und 4 Bits ID
- 1 Byte mit weiteren 8 Bits ID
- Rest: Gekapseltes Paket aus höherer Schicht

PCPR_DATA20

Daten-PDU für die Verbindungsidentifikatoren 4096-1048575. Drei Byte Overhead pro Nutzlast.

- 1 Byte PCPR-PDU-Typ, incl. Initiator-Bit und 4 Bits ID
- 2 Byte mit weiteren 16 Bits ID
- Rest: Gekapseltes Paket aus höherer Schicht

`PCPR_AUTH`

Authentifizierung einer Sitzung, Aufbau des Kommunikationskanals.

- 1 Byte PCPR-PDU-Typ
- Rest: Daten gemäß Authentifizierungsmechanismus

`PCPR_SYN`

Anzeige einer neuen *logischen Verbindung*.

- 1 Byte PCPR-PDU-Typ
- 3 Byte Verbindungsidentifikator (ID)
- 4 Byte Referenzfeld
- 1 Byte Länge des nachfolgenden Signalisierungsblocks (variabler Anteil)
- 5-260 Byte Signalisierungsblock

`PCPR_SYNDATA`

Anzeige einer neuen *logischen Verbindung* zzgl. Nutzdaten.

- 1 Byte PCPR-PDU-Typ
- 3 Byte Verbindungsidentifikator (ID)
- 4 Byte Referenzfeld
- 1 Byte Länge des nachfolgenden Signalisierungsblocks (variabler Anteil)
- 5-260 Byte Signalisierungsblock
- Rest: Gekapseltes Paket aus höherer Schicht

`PCPR_SYNACK`

Bestätigung einer Verbindungsanzeige.

- 1 Byte PCPR-PDU-Typ
- 3 Byte Verbindungsidentifikator (ID)
- 4 Byte Referenzfeld

`PCPR_SYNACKDATA`

Bestätigung einer Verbindungsanzeige zzgl. Nutzdaten.

- 1 Byte PCPR-PDU-Typ
- 3 Byte Verbindungsidentifikator (ID)
- 4 Byte Referenzfeld
- Rest: Gekapseltes Paket aus höherer Schicht

`PCPR_SYNACKACK`

Bestätigung der Bestätigung einer Verbindungsanzeige.

- 1 Byte PCPR-PDU-Typ
- 3 Byte Verbindungsidentifikator (ID)
- 4 Byte Referenzfeld

`PCPR_CLOSE`

Beendigungsanzeige einer *logischen Verbindung*.

- 1 Byte PCPR-PDU-Typ, incl. Initiator-Bit
- 3 Byte Verbindungsidentifikator (ID)
- 4 Byte Referenzfeld

`PCPR_CLOSEACK`

Bestätigung einer Beendigungsanzeige einer *logischen Verbindung*.

- 1 Byte PCPR-PDU-Typ, incl. Initiator-Bit
- 3 Byte Verbindungsidentifikator (ID)
- 4 Byte Referenzfeld

`PCPR_ISOPROBE`

Erkennung von Isolationssituationen

- 1 Byte PCPR-PDU-Typ

`PCPR_RTTREQUEST`

Bestimmung der „Round Trip Time“ (RTT), wird nur vom REACH-Client verschickt

- 1 Byte PCPR-PDU-Typ
- 4 Byte Sequenznummer

- 4 Byte RTT-Messergebnis aus vorherigem Zyklus, in Millisekunden
- 4 Byte Zeitstempel, Anteil in Sekunden
- 4 Byte Zeitstempel, Anteil in Nanosekunden

`PCPR_RTTREPLY`

Beantwortung einer `PCPR_RTTREQUEST`-PDU, wird nur vom REACH-Server verschickt

- 1 Byte PCPR-PDU-Typ
- 4 Byte Sequenznummer
- 4 Byte Zeitstempel, Anteil in Sekunden
- 4 Byte Zeitstempel, Anteil in Nanosekunden

`PCPR_KEEPALIVE`

„Keep-Alive“-PDU. Keine Nutzlast.

- 1 Byte PCPR-PDU-Typ

A.1.2.2 Automatengraphen PCPR

Wird eine ausgehende *logische Verbindung* angefordert, agiert die lokale PCPR-Instanz als Initiator, wobei der Automatengraph aus Abbildung A.5 gilt. Im Falle einer eintreffenden Verbindung (Rolle des Gerufenen) gilt dagegen der Automatengraph aus Abbildung A.6. Zum Verbindungsabbau wird jedoch in beiden Fällen der Teilgraph aus Abbildung A.7 benutzt.

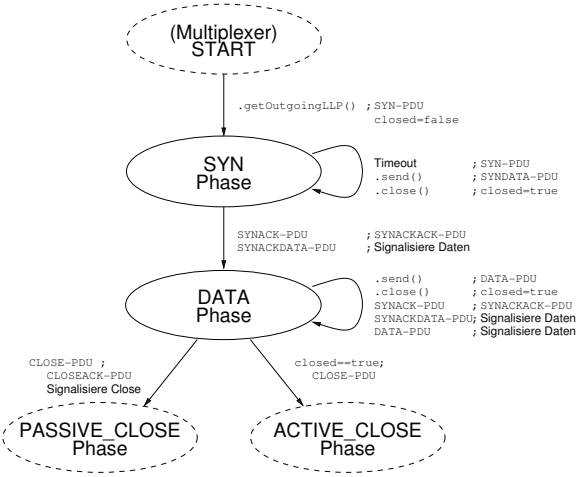

Abbildung A.5: Dieser Automatengraph repräsentiert eine PCPR-Protokollinstanz, wobei hier nur der für den Initiator relevante Teil dargestellt wird. Er ist jedoch *unvollständig*, da aus Gründen der Übersichtlichkeit nicht jeder implementierte Zustandsübergang dargestellt wird.

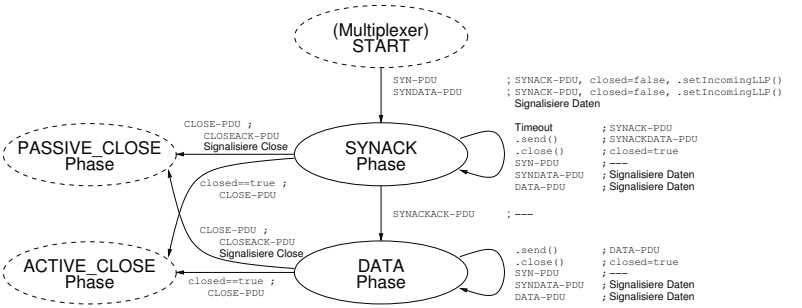

Abbildung A.6: Dieser Automatengraph repräsentiert eine PCPR-Protokollinstanz, wobei hier nur der für den Gerufenen relevante Teil dargestellt wird. Er ist jedoch *unvollständig*, da aus Gründen der Übersichtlichkeit nicht jeder implementierte Zustandsübergang dargestellt wird.

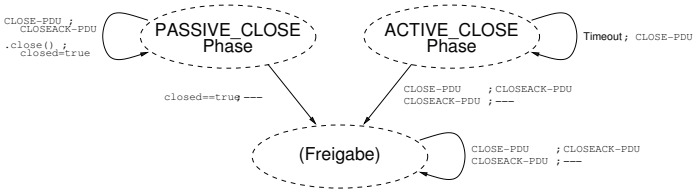

Abbildung A.7: Der unvollständige Automatengraph bezüglich des Abbaus paketorientierter *logischer Verbindungen*. Die Probleme, welche durch eine schnelle Neuvergabe des Identifikators verursacht werden und das zusätzliche Referenzfeld erforderlich gemacht haben, werden hier nicht dargestellt.

A.1.3 Struktur des Signalisierungsblocks

Eintreffende *logische Verbindungen*, welche von der Partnerinstanz angefordert worden sind, müssen einem *Linkbündel* und damit einer lokalen „Relay Plugin“-Instanz zugeordnet werden. Zudem benötigen manche „Relay Plugin“-Instanzen bereits bei der Annahme *logischer Verbindungen* Zusatzinformationen, welche vorliegen müssen, bevor die ersten Nutzdaten aus der *logischen Verbindungen* entnommen werden.

Zu diesem Zweck führen die SYN-PDUs einen Signalisierungsblock mit sich, welcher immer wenigstens das *Linkbündel* sowie den Dienst-Typ des zugehörigen „Relay Plugins“ benennt. Weiterhin kann ein bis zu 255 Byte großer Datenblock mitgegeben werden, welcher von der annehmenden „Relay Plugin“-Instanz ausgewertet werden kann. Es handelt sich um „Out-of-band“ Signalisierungsdaten, da sie losgelöst vom eigentlichen Datenstrom der *logischen Verbindungen* behandelt werden.

Struktur des Signalisierungsblocks:

- 4 Byte Identifikator des zugehörigen *Linkbündels*
- 1 Byte Identifikator des Dienstes (Service-Typ) für *Linkbündel*aufbau
- 0-255 Byte Signalisierungsblock für Dienstnehmer („Relay Plugin“)

A.2 Protokolle der Sicherungsinstanzen

Stromorientiert arbeitende Sicherungsinstanzen implementieren das „Stream Protection Protocol for REACH“ (SPPR), während paketorientiert arbeitende Sicherungsinstanzen das „Packet Protection Protocol for REACH“ (PPPR) umsetzen. Beide Protokolle werden jetzt nacheinander vorgestellt.

A.2.1 Das „Stream Protection Protocol for REACH“ (SPPR)

Anhand des ersten Bytes einer jeden SPPR-PDU kann der genaue PDU-Typ ermittelt werden. Er wird gemäß Abbildung A.8 anhand der zwei höchstwertigen Bits bestimmt. Die restlichen sechs Bits bestimmen die Anzahl an enthaltenen selektiven Blockbestätigungen („Selective ACK“ (SACK)).

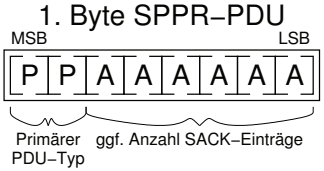

Abbildung A.8: Felder des SPPR-Protokollkopfes

Tabelle A.4 führt die vier Möglichkeiten primärer SPPR-PDUs auf. Sind beide P-Bits gesetzt, muss Tabelle A.5 zur Bestimmung des sekundären SPPR-PDU-Typs herangezogen werden. Bestätigungsinformationen werden grundsätzlich redundant übertragen, also im Fall mehrerer Übertragungswege mehrfach versendet. Quittungsinformationen werden allerdings pro Übertragungsweg nur einmalig übertragen, da SPPR-PDUs mittels TCP-Verbindungen gesichert übertragen werden. Dadurch wird der Übertragungsaufwand für Bestätigungsinformationen gering gehalten.

P-Bits	SPPR-PDU	A-Bits	SPPR-Headerlänge	Nutzdaten
00	SPPR_ACK	Anzahl SACKs	$(5 + \mathtt{A} * 8)$ Byte	Nein
01	SPPR_DATA	Anzahl SACKs	$(7 + \mathtt{A} * 8)$ Byte	Ja
10	SPPR_DATAACK	Anzahl SACKs	$(11 + \mathtt{A} * 8)$ Byte	Ja
11	siehe sekundärer Typ	Sekundärer Typ	siehe sekundärer Typ	Nein

Tabelle A.4: Primäre SPPR-Pakettypen

A-Bits	SPPR-PDU	SPPR-Headerlänge
000000	SPPR_SYN	(6 + 4) Byte
000001	SPPR_SYNACK	1 Byte
000010	SPPR_FIN	1 Byte
000011	nicht vergeben	–
...	...	...
111111	nicht vergeben	–

Tabelle A.5: Sekundäre SPPR-Pakettypen

SPPR_ACK

Bestätigung erhaltener Nutzdaten. Es wird ein lückenlos empfangener Datenbereich quittiert, wobei die enthaltene Sequenznummer das obere Ende benennt. Zusätzlich können zwischen 0 und 63 eigenständige Sequenznummernblöcke selektiv bestätigt werden („Selective ACK“ (SACK)).

- 1 Byte SPPR-PDU-Typ, incl. 6 Bits für Anzahl SACK-Blöcke (A)
- 4 Byte höchste lückenlos empfangene Sequenznummer
- (A) SACK-Blöcke, bestehend aus jeweils:
  - 4 Byte Sequenznummer Blockanfang
  - 4 Byte Sequenznummer Blockende

SPPR_DATA

Transport eines Nutzdatenblocks ohne „lückenlose“ Quittierungsinformationen. Zusätzlich können zwischen 0 und 63 eigenständige Sequenznummernblöcke selektiv bestätigt werden.

- 1 Byte SPPR-PDU-Typ, incl. 6 Bits für Anzahl SACK-Blöcke (A)
- 4 Byte Sequenznummer
- 2 Byte Längenfeld Nutzdaten ($L \in [1 \ldots 32768]$ Byte)
- (A) SACK-Blöcke, bestehend aus jeweils:
  - 4 Byte Sequenznummer Blockanfang
  - 4 Byte Sequenznummer Blockende
- (L) Byte Datenfragment aus höherer Schicht

SPPR_DATAACK

Transport eines Nutzdatenblocks mit zusätzlichen „lückenlosen“ Quittierungsinformationen. Zusätzlich können zwischen 0 und 63 eigenständige Sequenznummernblöcke selektiv bestätigt werden.

- 1 Byte Byte SPPR-PDU-Typ, incl. 6 Bits für Anzahl SACK-Blöcke (A)
- 4 Byte Sequenznummer
- 2 Byte Längenfeld Nutzdaten (L $\in [1 \ldots 32768]$ Byte)
- 4 Byte höchste lückenlos empfangene Sequenznummer
- (A) SACK-Blöcke, bestehend aus jeweils:
  - 4 Byte Sequenznummer Blockanfang
  - 4 Byte Sequenznummer Blockende
- (L) Byte Datenfragment aus höherer Schicht

SPPR_SYN

Initiiert eine Sitzung zwischen zwei stromorientierten Sicherungsinstanzen. Die PDU wird immer beidseitig versendet, zwecks einmaligem Austausch der Empfangspuffergröße und eines frei definierbaren Signalisierungsdatenblocks für die höhere Schicht (Austausch der Empfangspuffergröße stromorientierter *logischer Verbindungen*, zur Initialisierung der Flusssteuerung im Multiplexer).

- 1 Byte SPPR-PDU-Typ
- 4 Byte Empfangspuffergröße des Absenders für Flusssteuerungszwecke
- 1 Byte Länge der nachfolgenden Signalisierungsdaten, variabler Anteil
- 0-255 Byte Signalisierungsdaten der höheren Schicht (momentan: 4 Byte)

SPPR_SYNACK

Bestätigung einer SPPR_SYN-PDU.

- 1 Byte SPPR-PDU-Typ

SPPR_FIN

Kommando zur sofortigen Beendigung einer Sitzung. Kann beidseitig verschickt werden. Es werden keine weiteren PDUs versendet oder ausgewertet. Beide Sicherungsinstanzen und Multiplexer sind augenblicklich ungültig, und alle zugehörigen *logischen Verbindungen* und *Linkbündel* gelten ab sofort als geschlossen.

- 1 Byte SPPR-PDU-Typ

A.2.2 Das „Packet Protection Protocol for REACH“ (PPPR)

Die Aufgabe der paketorientiert arbeitenden Sicherungsinstanzen besteht darin, alle speziell durch REACH verursachten Duplikate am Empfänger zu erkennen und zu verwerfen.

Die PDUs des PCPR wurden bereits in Abschnitt A.1.2.1 vorgestellt, wobei das im ersten Byte des Protokollkopfes (Header) aufgeführte „Protection Indication“-Bit bislang nicht erläutert worden ist. Dieses Bit (siehe Abbildung A.9) ist Teil des ‘Packet Protection Protocol for REACH“ (PPPR), und wird durch die Sicherungsinstanz gesetzt bzw. ausgewertet.

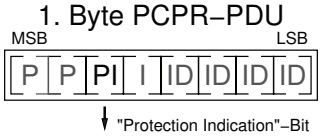

Abbildung A.9: Feld des PPPR-Protokollkopfes im PCPR-Protokollkopf

Soll eine PCPR-PDU mehrfach versendet werden, wird das „Protection Indication“-Bit gesetzt und gleich nach dem ersten Byte ein Erweiterungspaketkopf eingefügt (siehe Abbildung A.10). Dieser enthält die Sequenznummer der PPPR-PDU, sodass der Empfänger Duplikate erkennen und filtern kann.

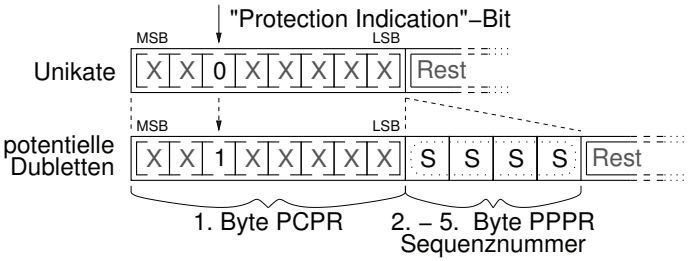

Abbildung A.10: Das PPPR setzt bei allen mehrfach versendeten PCPR-PDUs im ersten Byte des Protokollkopfes das „Protection Indication“-Bit und fügt einen 4 Byte großen Erweiterungsheader ein. Dieser enthält eine Sequenznummer.

B Implementierungsdetails zu den „Device Backends“

In diesem Kapitel werden Details bezüglich der Implementierung der „Device Backends“ von REACH geliefert. Als erstes wird auf das Kommunikationsprotokoll zwischen REACH-Client und den „Device Backends“ eingegangen. Anschließend werden die internen Mechanismen der drei für den Prototyp implementierten „Device Backends“, welche zur Verwaltung der Netzzugangstechnologien „Ethernet“, „WLAN“ und „GPRS“ entwickelt worden sind, vorgestellt.

B.1 Kommunikation zwischen REACH-Client und den „Device Backends“

Im Folgenden wird ein exemplarischer Protokollablauf zwischen REACH-Client und einem „Device Backend“ vorgestellt. Hierbei kommt beispielhaft das „Device Backend“ für WLAN zum Einsatz. Zuerst wird jedoch eine Übersicht über die existierenden Nachrichtentypen geliefert und auf deren Bedeutung eingegangen.

B.1.1 Protokollfunktionen

Die Kommunikation erfolgt anhand von Nachrichten im XML-Format, welche über die Standardein- und ausgabe der gestarteten „Device Backends“ *bidirektional* ausgetauscht werden. Dabei werden seitens des REACH-Clients andere Nachrichten verschickt als seitens eines „Device Backends“. Es handelt sich hierbei um ein Protokoll, welches zur Umsetzung des „Providerkonzepts“ von REACH erforderlich war (siehe Abschnitt 6.3.2).

B.1.1.1 Nachrichtentypen in Richtung eines „Device Backends“

Dem REACH-Client stehen die folgenden Nachrichtentypen zur Verfügung, welche an ein „Device Backend“ verschickt werden können:

- `<Parameters>`: Dem „Device Backend“ wird mitgeteilt, welches Netzzugangsgerät es verwalten soll. Dafür muss wenigstens ein Gerätename, beispielsweise `wlan0`, spezifiziert werden. Je nach „Device Backend“ können noch weitere Parameter erforderlich sein.

- `<AddNetwork>`: Ein Netzzugangspunkt wird bekannt gemacht. Von diesem Moment an kann das „Device Backend“ seine Umgebung mit Blick auf diesen Netzzugangspunkt hin untersuchen, und gegebenenfalls seine Verfügbarkeit an den Handoverentscheider melden. Es können eine Vielzahl an Netzzugangspunkten bekannt gemacht werden. Dabei werden Detektions- und Assoziationsparameter unterschieden.

- `<ModNetwork>`: Die Parameter eines bereits bekannt gemachten Netzzugangspunkts können nachträglich geändert werden.

- `<DelNetwork>`: Ein bekannter Netzzuganspunkt soll „vergessen“ werden. Nach Erhalt dieser Nachricht wird er nicht länger berücksichtigt, also beispielsweise als verfügbar gemeldet.

- `<Associate>`: Der REACH-Client kann mit diesem Befehl einen Assoziationsprozess zu einem der momentan als verfügbar gemeldeten Netzzugangspunkte anstoßen.

- `<Dissociate>`: Um eine bereits aufgebaute oder eine sich im Aufbau befindliche Assoziation abzubauen, schickt der REACH-Client diese Nachricht.

- `<ShutDown>`: Mit Hilfe dieses Kommandos kann ein „Device Backend“ beendet werden. Es wird daraufhin einen geordneten Zustand der überwachten Hardware herstellen und sich beenden.

B.1.1.2 Nachrichtentypen in Richtung des REACH-Clients

Ein „Device Backend“ verschickt keine Befehlsnachrichten an den REACH-Client, sondern agiert ausschließlich als Befehlsempfänger. Es werden lediglich Nachrichten versendet, welche Statusänderungen anzeigen.

- `<Attach>`: Direkt nach seinem Start meldet sich das „Device Backend“ beim REACH-Client an. In dieser Nachricht wird ein Zeitstempel mitgeliefert, welcher als Referenzzeitpunkt für alle weiteren (relativen) Zeitstempel herangezogen wird.

- `<DataTemplate scope="device">`: Ein jedes „Device Backend“ erwartet einen spezifischen Parametersatz bezüglich des zu überwachenden Netzzugangsgeräts. Dafür enthält diese Nachricht eine „Schablone“, welche die unterstützen Parameter beschreibt. Sie wird einmalig nach dem Start des „Device Backends“ an den REACH-Client verschickt.

- `<DataTemplate scope="network">`: Die Bekanntmachung von Netzzugangspunkten erfordert ebenfalls einen Satz an Parametern, welcher durch diese Nachricht als Parameterschablone beschrieben wird. Sie wird einmalig nach dem Start des Backends gesendet.

- `<Parameters scope="device">`: Eine solche Nachricht zeigt Änderungen bezüglich des überwachten Netzzugangsgeräts an. Momentan sind die folgenden Datenpunkte definiert, welche in Form von Schlüssel-Werte-Paaren übertragen werden:

  - `initialized`: Der zugehörige Wert gibt an, ob das „Device Backend“ bereits ausreichend mit Daten parametriert worden ist. Diese Nachricht wird im Normalfall einmalig nach Erhalt des hardwarespezifischen Konfigurationsblocks verschickt.

  - `measurements`: Hiermit signalisiert das „Device Backend“ einmalig, ob das verwaltete Netzzugangsgerät Gütewerte liefern wird. Dieser Parameter kann nicht beeinflusst werden, da er technologieabhängig ist und „hart“ im Quelltext des „Device Backends“ definiert werden muss.

  - `measurement_ms`: Falls `measurements` eine Lieferung von Gütewerten anzeigt, weist dieser Parameter auf das zu erwartende Erfassungsintervall in Millisekunden hin. Dieser Parameter gilt allerdings nur bezüglich des Netzzugangspunkts einer bestehenden Assoziation, nicht bezüglich möglicher weiterer verfügbarer („sichtbarer“) Netzzugangspunkte.

  - `measurement_value` und `measurement_time`: Diese Schlüssel deuten auf einen zeitstempelbehafteten Signalstärke- oder Linkgütewert hin, welche periodisch bezüglich einer bestehenden Assoziation anfallen können. Sie eignen sich für Regressionszwecke, da hohe Erfassungsraten möglich sind.

- `<Parameters scope="network">`: Wenn sich Parameter bezüglich eines Netzzugangspunkts ändern, wird dies anhand einer Liste an Schlüssel-Werte-Paaren kommuniziert. Dabei können eine Vielzahl an Netzzugangspunkten unterschieden werden.

  - `visibility`: Der betroffene Netzzugangspunkt gilt als „sichtbar“ oder als „unsichtbar“. Im ersten Fall kann ein Assoziationsprozess angestoßen werden.
  - `measurement_value` und `measurement_time`: Diese Schlüssel deuten auf einen zeitstempelbehafteten Signalstärke- oder Linkgütewert hin, welcher periodisch pro „sichtbarem“ Netzzugangspunkt anfallen kann. Da die Erfassungsrate unbekannt ist, eignen sich diese Messwerte nicht für Regressionszwecke. Sie reichen allerdings aus, um die Eignung des jeweiligen Netzzugangspunkts abschätzen zu können. Bezüglich WLAN werden hiermit die Ergebnisse periodischer „Scans“, welche beispielsweise alle 20 Sekunden durchgeführt werden, mitgeteilt.

- `<HardwareIsAvailable>`: Wird das überwachte Netzzugangsgerät verfügbar, signalisiert das „Device Backend“ dies anhand einer solchen Nachricht.

- `<HardwareIsNotAvailable>`: Wird das überwachte Netzzugangsgerät entfernt oder deaktiviert, erfährt der REACH-Client dies durch eine Nachricht diesen Typs.

- `<Associated>`: Eine angeforderte Assoziation konnte erfolgreich etabliert werden. Das mobile Endgerät ist damit über einen weiteren Netzzugang mit dem Internet verbunden. Als Parameter werden die der Schnittstelle zugewiesene IP-Adresse, die Netzmaske sowie die IP-Adresse des Gateways mitgeliefert.

- `<Dissociated>`: Eine bestehende oder sich im Aufbau befindliche Assoziation wurde getrennt. Dies kann die Folge eines vorausgehenden Trennungskommandos sein, oder einen spontanen Linkabriss bedeuten.

B.1.2 Beispielhafter Ablauf

Man kann die Herkunft einer Nachricht an ihrer äußersten XML-Schachtelung erkennen: Eine Nachricht seitens eines „Device Backends“ weist die Schachtelung `<DeviceBackend>` auf, während Nachrichten des REACH-Clients immer in eine `<ReachClient>`-Schachtelung eingefasst sind.

B.1.2.1 Initialisierungsphase

Nachdem ein „Device Backend“ gestartet worden ist, meldet es sich mit der in Abbildung B.1 dargestellten `<Attach>`-Nachricht am REACH-Client an. Diese Nachricht enthält einen Zeitstempel, welcher für relative Zeitangaben in späteren Nachrichten herangezogen wird. Zeitangaben relativ zum Start des jeweiligen „Device Backends“ sind

für die Zwecke von REACH besser geeignet als die Systemzeit. Die so erhaltenen Zahlen sind „kleiner“, was für die Regressionsrechnung von Vorteil ist, da die Gefahr von Rundungsfehlern verringert wird.

```
<DeviceBackend>
  <Attach rel_secs="1255944297"
          rel_msecs="697" />
</DeviceBackend>
```

Abbildung B.1: Die Anmeldenachricht eines „Device Backends“ enthält einen Zeitstempel. Relativ zu diesem erfolgen alle späteren Zeitangaben.

Unmittelbar im Anschluss sendet das „Device Backend“ noch eine zweite Nachricht mit zwei Parameterschablonen an den REACH-Client (siehe Abbildung B.2). Diese Schablonen definieren, welche Parameter das „Device Backend“ bezüglich des zu überwachenden Netzzugangsgeräts sowie der möglichen Netzzugangspunkte erwartet.

```
<DeviceBackend>
  <DataTemplate scope="device">
    <Item key="interface"
          value="string" />
    <Item key="driver"
          value="string" />
    <Item key="measurement_type"
          value="string" />
    <Item key="measurement_ms"
          value="uint" />
  </DataTemplate>
  <DataTemplate scope="network">
    <Item key="SSID"
          value="string" />
    <Item key="supplicant_configfile"
          value="string" />
  </DataTemplate>
</DeviceBackend>
```

Abbildung B.2: Ein „Device Backend“ schickt nach seinem Start zwei Parameterschablonen an den REACH-Client.

Nach Erhalt dieser beiden Nachrichten ist sich der REACH-Client sicher, dass das betroffene „Device Backend“ erfolgreich gestartet werden konnte. Zudem ist bekannt, welche Parameter vom „Device Backend“ akzeptiert werden.

B.1.2.2 Parametrierung des Netzzugangsgeräts

Zuerst muss dem „Device Backend“ durch den REACH-Client ein Netzzugangsgerät zugewiesen werden. Eine solches wird unter GNU/Linux durch den Namen eines Interfaces (`interface`) spezifiziert. Es können jedoch noch diverse weitere Parameter erforderlich sein, wie der in Abbildung B.3 gezeigte Treibername (`driver`), welcher für die in diesem Beispiel konfigurierte WLAN-Karte die „Wireless Extensions“ (`wext`) des Linux-Kernel involviert. Für Regressionszwecke werden Signalstärkewerte (`signalstrength`) mit einem Erfassungsintervall (`measurement_ms`) von 100 Millisekunden angefordert. Die Parameter müssen der Schablone des „Device Backends“ entsprechen (Abbildung B.2).

```
<ReachClient>
  <Parameters>
    <Item key="interface"
          value="wlan0" />
    <Item key="driver"
          value="wext" />
    <Item key="measurement_type"
          value="signalstrength" />
    <Item key="measurement_ms"
          value="100" />
  </Parameters>
</ReachClient>
```

Abbildung B.3: Jedes gestartete „Device Backend“ ist genau für ein Netzzugangsgerät verantwortlich. Diese Nachricht an das „Device Backend“ benennt das zu überwachende Interface und gemäß Schablone diverse weitere Parameter.

Konnte das „Device Backend“ die in Abbildung B.3 aufgeführte Nachricht auswerten, und wurden zudem auch alle erforderlichen Parameter gesetzt, wird eine Statusnachricht (siehe Abbildung B.4) formuliert und an den REACH-Client geschickt. Sie signalisiert, dass das überwachte Netzzugangsgerät vollständig und ordnungsgemäß initialisiert worden ist. Von nun an kann der REACH-Client mit Nachrichten rechnen, welche eine Änderung der Verfügbarkeit des zu überwachenden Netzzugangsgeräts anzeigen. Im Initialzustand gilt ein Netzzugangsgerät als „nicht verfügbar“, also wird später die erste diesbezügliche Änderungsnachricht „die Verfügbarkeit“ melden.

Des Weiteren wird in dieser Nachricht mitgeteilt, ob das betroffene „Device Backend“ Messwerte (Signalstärke- bzw. Linkgütewerte) liefern wird. Da es sich im vorliegenden Beispiel um das „Device Backend“ für WLAN handelt, nennt dieser Parameter ein `true`, also „wahr“. Zudem wird die Größe des Messintervalls in Millisekunden enannt.

```
<DeviceBackend>
  <Parameters scope="device">
    <Item key="initialized"
          value="true" />
    <Item key="measurements"
          value="true" />
    <Item key="interface"
          value="wlan0" />
    <Item key="measurement_ms"
          value="100" />
  </Parameters>
</DeviceBackend>
```

Abbildung B.4: Nachdem ein „Device Backend“ mit ausreichend Parametern bezüglich des zu überwachenden Netzzugangsgeräts versorgt wurde, deklariert es sich mit Hilfe einer solchen Nachricht als „initialisiert“. Zudem wird angezeigt, ob und wie häufig Messwerte geliefert werden und wie der Name des verwalteten Interfaces lautet.

B.1.2.3 Bekanntmachung der Netzzugangspunkte

Das „Device Backend“ „kennt“ somit das zu überwachende Netzzugangsgerät, aber es liegen noch keine Informationen bezüglich möglicher Netzzugangspunkte vor. Diese Daten werden allerdings benötigt, damit das „Device Backend“ seine Umgebung auf geeignete Netzzugangspunkte hin beobachten kann.

Für die Bekanntmachung von Netzzugangspunkten werden `<AddNetwork>`-Nachrichten (siehe Abbildung B.5) verwendet. In diesem Beispiel werden zwei Netzzugangspunkte (hier: WLAN-Basisstationen) spezifiziert, welche durch jeweils eine eigene `<AddNetwork>`-Schachtelung repräsentiert werden.

Zu jedem Netzzugangspunkt müssen Detektions- und Assoziationsparameter hinterlegt werden. In diesem Fall gilt der Parameter `"SSID"` als Detektionsparameter, während die mit `"supplicant_configfile"` referenzierte Konfigurationsdatei für die Anmeldung und die Verschlüsselung bezüglich der zugehörigen WLAN-Basisstation benötigt wird. Die Parameter müssen dabei der Schablone, welche das „Device Backend“ vorgegeben hat (siehe Abbildung B.2), entsprechen.

B.1.2.4 Verfügbarkeit des Netzzugangsgeräts und der Netzzugangspunkte

Nachdem dem „Device Backend“ sowohl das Netzzugangsgerät als auch eine Menge an Netzzugangspunkten bekannt gemacht worden sind, kann es deren Erkennung durchführen. Hierbei können jederzeit weitere Netzzugangspunkte mit Hilfe von `<AddNetwork>`-

```
<ReachClient>
  <AddNetwork label="WLAN AP 1">
    <Item key="SSID"
          value="REACH Access Point 1" />
    <Item key="supplicant_configfile"
          value="supplicant_reach1.conf" />
  </AddNetwork>
  <AddNetwork label="WLAN AP 2">
    <Item key="SSID"
          value="REACH Access Point 2" />
    <Item key="supplicant_configfile"
          value="supplicant_reach2.conf" />
  </AddNetwork>
</ReachClient>
```

Abbildung B.5: Dem „Device Backend“ für WLAN werden mit Hilfe dieser Nachricht zwei Netzzugangspunkte bekannt gemacht. Es gibt hierbei Parameter, welche der Detektion von Netzzugangspunkten dienen, während andere Parameter für die Assoziationsprozedur benötigt werden.

Nachrichten hinzugefügt oder diese mit Hilfe von `<ModNetwork>`- oder `<DelNetwork>`-Nachrichten geändert beziehungsweise gelöscht werden.

```
<DeviceBackend>
  <HardwareIsAvailable interface="wlan0" />
</DeviceBackend>
```

Abbildung B.6: Wenn ein „Device Backend“ das überwachte Netzzugangsgerät erkennt, meldet es dies zusammen mit dem Gerätenamen an den REACH-Client.

In diesem Beispielablauf wird jetzt das Vorhandensein des überwachten Netzzugangsgeräts durch das „Device Backend“ erkannt. Es handelte sich hierbei um einen „WLAN-USB-Stick“ mit USB-Anschluss, welcher vom Autor spontan in den Rechner gesteckt worden ist. Dem REACH-Client wird die Verfügbarkeit durch die in Abbildung B.6 gezeigte Nachricht mitgeteilt. Dabei wird der Gerätename mitgeliefert, welcher für hier nicht genannte Folgeschritte innerhalb des REACH-Clients benötigt wird. Wird das Netzzugangsgerät zu einem späteren Zeitpunkt wieder entfernt, reagiert das „Device Backend“ darauf umgehend mit einer `<HardwareIsNotAvailable>`-Nachricht.

Wurde das Netzzugangsgerät gefunden, wird es von nun an vom „Device Backend“ zur Detektion möglicher von Netzzugangspunkte herangezogen. Bei drahtlosen Netzen kann dies durch die Beobachtung der Umgebung geschehen. Bei drahtgebundenen Netzen wie

```
<DeviceBackend>
  <Parameters scope="network" label="WLAN AP 1">
    <Item key="visibility"
          value="true" />
  </Parameters>
</DeviceBackend>
```

Abbildung B.7: Wird ein Netzzugangspunkt verfügbar, meldet das „Device Backend“ dieses an den REACH-Client. Der Handoverentscheider kann den referenzierten Netzzugangspunkt von nun an für eine Assoziation heranziehen.

Ethernet bietet sich ein Abgleich der MAC-Adresse eines per DHCP zugewiesenen Gateways mit einer Liste an hinterlegten Einträgen an. Ein erkannter und damit verfügbarer Netzzugangspunkt wird durch eine `visibility`-Nachricht, wie sie Abbildung B.7 zeigt, gemeldet. Für jeden als „sichtbar“ geltenden Netzzugangspunkt können Signalstärke- oder Linkgütewerte anfallen, was aber vom „Device Backend“ abhängig ist.

```
<DeviceBackend>
  <Parameters scope="network" label="WLAN AP 1">
    <Item key="measurement_value"
          value="5.200000e+01" />
    <Item key="measurement_time"
          value="13064" />
  </Parameters>
</DeviceBackend>
```

Abbildung B.8: Bezüglich aller „sichtbaren“ Netzzugangspunkte können Signalstärke- bzw. Linkgütewerte anfallen, was allerdings von der Netzzugangstechnologie abhängig ist. Zudem ist an dieser Stelle nicht erkennbar, was durch den Messwert ausgedrückt wird. Es handelt sich um einen Wert, dessen Wertebereich und Einheit unbekannt sind.

Falls möglich, werden Messwerte zyklisch wie in Abbildung B.8 dargestellt übertragen. Ein solcher Messwert gilt immer für einen bestimmten Netzzugangspunkt, welcher vorab als „sichtbar“ gemeldet worden ist. Man erkennt an der dargestellten Nachricht, dass keine absoluten, sondern relative Zeitstempel verwendet werden. Diese beziehen sich auf den in der Anmeldenachricht genannten Zeitpunkt, und verweist hier auf eine Zeitspanne von 13064 Millisekunden seit der Anmeldung des „Device Backends“. Im weiteren Verlauf können weitere Netzzugangspunkte „sichtbar“ werden, und bereits „sichtbare“ Netzzugangspunkte wieder verschwinden.

B.1.2.5 Assoziation mit einem Netzzugangspunkt

Wird dem REACH-Client ein Netzzugangspunkt als „sichtbar“ gemeldet, gilt dieser für den Handoverentscheider als *assoziierbar*. Deuten zudem die Messwerte auf eine ausreichend gute Funkverbindung hin, kann der REACH-Client den Befehl äußern, dass das „Device Backend“ eine Assoziation mit diesem Netzzugangspunkt aufbauen soll.

```
<ReachClient>
  <Associate label="WLAN AP 1" />
</ReachClient>
```

Abbildung B.9: Wünscht der Handoverentscheider eine Internetverbindung über einen bestimmten Netzzugangspunkt, wird eine `<Associate>`-Nachricht an das zugehörige „Device Backend“ geschickt. Anhand des `label`-Parameters wird der Bezug zu einem per `<AddNetwork>`-Nachricht hinterlegten Datensatz hergestellt.

Das dafür zuständige `<Associate>`-Kommando aus Abbildung B.9 benennt einen Identifikator (`label`), welcher dem „Device Backend“ bereits aus einer `<AddNetwork>`-Nachricht bekannt sein muss. Dieser Identifikator ist mit Assoziationsinformationen verknüpft, welche für den „Einwahlprozess“ benötigt werden. Gemäß des „Providerkonzepts“ kann ein „Device Backend“ immer nur zu einem Netzzugangspunkt gleichzeitig eine Assoziation pflegen.

```
<DeviceBackend>
  <Associated label="WLAN AP 1"
              interfaceIpAddress="192.168.1.32"
              interfaceNetmask="255.255.255.0"
              gatewayIpAddress="192.168.1.1" />
</DeviceBackend>
```

Abbildung B.10: War das „Device Backend“ mit dem Assoziationsprozess erfolgreich, meldet es die erhaltenen IP-Adressinformationen an den REACH-Client.

Konnte das Netzzugangsgerät erfolgreich mit einem Netzzugangspunkt assoziiert werden, schickt das „Device Backend“ eine `<Associated>`-Nachricht. Diese enthält, wie in Abbildung B.10 dargestellt, neben dem Identifikator des Netzzugangspunkts noch die dem Interface zugewiesene IP-Adresse, die geltende IP-Netzmaske sowie die IP-Adresse des Gateways. Es ist damit Aufgabe der „Device Backends“, die beim Assoziationsprozess erhaltenen Adressinformationen zu ermitteln und sie dem REACH-Client mitzuteilen.

Die zur Ermittlung notwendigen Mechanismen können für jede Netzzugangstechnologie unterschiedlich ausfallen.

```
<DeviceBackend>
  <Parameters scope="device">
    <Item key="measurement_value"
          value="6.100000e+01" />
    <Item key="measurement_time"
          value="2154" />
  </Parameters>
</DeviceBackend>
```

Abbildung B.11: Bezüglich einer bestehenden Assoziation liefert das „Device Backend“ für WLAN einen hochratigen Strom an Messwerten, welcher sich für Regressionszewecke eignet.

Konnte eine Assoziation aufgebaut werden, besteht bei WLAN die Möglichkeit, durch Auslesen der Datei `/proc/net/wireless` sehr häufig Messwerte bezüglich der Funkstrecke zu erhalten. Eine diesbezügliche Nachricht ist in Abbildung B.11 dargestellt. Sie kann entsprechend der ausgehandelten Parameter mehrmals pro Sekunde übertragen werden.

```
<ReachClient>
  <Dissociate />
</ReachClient>
```

Abbildung B.12: Um eine bestehende oder angeforderte Assoziation zu beenden, schickt der REACH-Client eine `<Dissociate>`-Nachricht an das „Device Backend“.

Wird eine Assoziation nicht länger benötigt, weil beispielsweise ein als besser geeignet erscheinendes Netzzugangsgerät erfolgreich aktiviert werden konnte („weicher vertikaler Handover“) oder weil *dieses* Netzzugangsgerät mit einem anderen Netzzugangspunkt assoziiert werden soll („harter horizontaler Handover“), wird ein Dissoziationskommando (siehe Abbildung B.12) an das „Device Backend“ übergeben.

Wurde eine Assoziation beendet, weil entweder ein Trennungsbefehl voraus ging oder weil ein Abriss stattfand, meldet das „Device Backend“ diesen Zustandswechsel durch eine `<Dissociated>`-Meldung (in Abbildung B.13 dargestellt).

Nachdem das Trennungsereignis an den REACH-Client kommuniziert wurde, wird der Handoverentscheider aktiv und kann das „Device Backend“ für einen erneuten Assoziationsprozess heranziehen. Anzumerken ist, dass die Nachrichten ereignisorientiert anfallen.

```
<DeviceBackend>
  <Dissociated label="WLAN AP 1" />
</DeviceBackend>
```

Abbildung B.13: Wenn eine Assoziation auf Wunsch getrennt wurde oder abgerissen ist, signalisiert eine solche `<Dissociated>`-Nachricht dem REACH-Client, dass das vom „Device Backend“ verwaltete Netzzugangsgerät für eine erneute Assoziationsprozedur zur Verfügung steht.

Es werden also lediglich Änderungen kommuniziert, mit Ausnahme von Signalstärke- und Linkgütewerten, welche zyklisch[1] anfallen. Alle Änderungen bezüglich der Verfügbarkeit des überwachten Netzzugangsgeräts, der Sichtbarkeit von Netzzugangspunkten und dem Status einer angeforderten Assoziation werden wie dargestellt an den REACH-Client gemeldet. Das geschieht so lange, bis das „Device Backend“ heruntergefahren wird.

B.1.2.6 Beendigungsphase

Wird der REACH-Client beendet, schickt er ein Beendigungskommando an alle gestarteten „Device Backends“. Eine solche `<ShutDown>`-Nachricht wird in Abbildung B.14 gezeigt. Das „Device Backend“ wird sich daraufhin beenden, aber vorher noch für einen geordneten Zustand des überwachten Netzzugangsgeräts sorgen.

```
<ReachClient>
  <ShutDown />
</ReachClient>
```

Abbildung B.14: Mit Hilfe dieser `<ShutDown>`-Nachricht befiehlt der REACH-Client einem gestarteten „Device Backend“, sich geordnet zu beenden.

B.2 Die implementierten „Device Backends“

Im Folgenden werden die drei für den Demonstrator implementierten „Device Backends“ vorgestellt. Sie basieren allesamt darauf, dass auf dem mobilen Endgerät diverse Systemkommandos verfügbar sind, welche auch ein Administrator benutzen würde, wenn er einen Internetzugang manuell in der Kommandozeile konfigurieren wollte. Die „Device Backends“ wurden so entwickelt, dass sie selbige Kommandos in ähnlicher Reihenfolge

[1] Auch wenn es keine Änderung der Signalstärke bzw. Linkgüte gab, fällt dennoch zyklisch ein neuer zeitstempelbehafteter Messwert an.

automatisch aufrufen. Dabei wird jeweils die Ausgabe der Kommandos untersucht, um einerseits Fehlerfälle feststellen zu können, aber auch um beispielsweise benötigte Adressinformationen zu erhalten.

B.2.1 Ethernet

Ethernet ist eine weit verbreitete drahtgebundene Netzzugangstechnologie, welche in vielen modernen mobilen Endgeräten (besonders in Laptops) vertreten ist. Die Unterstützung von Ethernet durch REACH bietet sich damit an.

Ethernet bietet dabei keine „Einwahlprozeduren“ an. Bezüglich einer Ethernetschnittstelle kann allenfalls erkannt werden, ob sie über ein Kabel mit einem anderen Gerät verbunden worden ist. Dazu wird das Vorhandensein eines Trägersignals überprüft, was allerdings bereits durch die Hardware und den dazugehörigen Treiber seitens des Betriebssystems erfolgt.

Es liegt somit lediglich die Information vor, *dass* eine physikalische Verbindung vorliegt, jedoch nicht, an welchen Netzzugangspunkt das Gerät angeschlossen worden ist. Um das herauszufinden, sind weitere Schritte notwendig.

Detektion des Netzzugangsgeräts und des Mediums

Die erste Aufgabe des „Device Backends“ für Ethernet besteht damit darin, zu erkennen, ob die überwachte Ethernetschnittstelle verfügbar ist und ob momentan eine physikalische Verbindung besteht. Diese Überprüfung muss zyklisch erfolgen, um beispielsweise den Verlust des Trägersignals bei einem Abziehen des Netzwerkkabels zeitnah erkennen zu können.

```
neelix ~ # ifplugstatus --auto eth0
eth0: unplugged
neelix ~ # ifplugstatus --auto eth1
eth1: link beat detected
neelix ~ # ifplugstatus --auto eth2
eth2: not supported
Failure (No such device)
```

Abbildung B.15: Der Status einer Ethernetschnittstelle lässt sich mit Hilfe des Kommandos `ifplugstatus` ermitteln. Hier sind drei mögliche Ausgaben des Kommandos aufgeführt.

Unter GNU/Linux bietet sich dazu das Kommando `ifplugstatus` an, welche eine Ethernetschnittstelle auf das Vorhandensein eines Trägersignals hin untersuchen kann.

In Abbildung B.15 sind die Ausgaben dieses Kommandos für drei Szenarien dargestellt. Im ersten Fall konnte kein Trägersignal erkannt werden, im zweiten Fall dagegen schon. Im dritten Fall ist das geprüfte Interface `eth2` nicht vorhanden gewesen, da beispielsweise die zugehörige Steckkarte nicht verfügbar war. Das Kommando sollte mit dem Parameter `--auto` aufgerufen werden, da man sonst eine Fehlermeldung erhalten würde, sollte das Interface vorab nicht anderweitig aktiviert worden sein.

Das „Device Backend“ für Ethernet ruft das `ifplugstatus`-Kommando zyklisch auf und untersucht dessen Ausgabe auf das Vorhandensein bestimmter Zeichenketten. Konnte „`unplugged`“ gefunden werden, gilt das Netzzugangsgerät als verfügbar, aber ohne ein angeschlossenes Kabel. Die Zeichenkette „`link beat detected`“ zeigt das Vorhandensein des Geräts *und* einer Gegenstation an, während bei „`No such device`“ nicht einmal das Netzzugangsgerät als verfügbar gilt.

Somit kann das „Device Backend“ den Verfügbarkeitsstatus der zu überwachenden Ethernetschnittstelle erkennen und dies dem REACH-Client mitteilen. Dazu wird eine `<HardwareIsAvailable>`- bzw. eine `<HardwareIsNotAvailable>`-Nachricht verschickt.

Erkennung des Netzzugangspunkts

Wird ein Trägersignal erkannt, gilt es herauszufinden, an welchen Netzzugangspunkt das mobile Endgerät angeschlossen worden ist. Diese Information wird dazu benötigt, den erkannten Netzzugangspunkt als „sichtbar“ melden zu können, damit der Handoverentscheider des REACH-Clients über die Nutzbarkeit des Netzzugangspunkts befinden kann.

Die Erkennung des aktuellen Aufenthaltsorts erfordert bei Ethernet den Einsatz des „Dynamic Host Configuration Protocols“ (DHCP). Dieses ermöglicht eine automatische Adresskonfiguration. Die Nutzung von DHCP seitens des mobilen Endgeräts setzt allerdings voraus, dass das Netzwerk, an welches es angeschlossen wird, auch einen DHCP-Server anbietet. Ist das nicht der Fall, werden keine Adressinformationen zugewiesen, und es kann auch keine Positionsangabe abgeleitet werden. Der Netzzugang gilt in solchen Fällen als nicht benutzbar.

Konnten der Ethernetschnittstelle erfolgreich Adressinformationen zugewiesen werden, fehlt immer noch die für REACH wichtige Positionsangabe. Das mobile Endgerät muss seine Umgebung „erkennen“, um einen der hinterlegten Netzzugangspunkte als „sichtbar“ melden zu können.

Dafür kann jedoch nicht die dem Interface zugewiesene IP-Adresse herangezogen werden. Zum einen kann sich diese von Mal zu Mal ändern, sodass ein eigentlich bekannter Netzzugangspunkt fälschlicherweise nicht wieder erkannt werden würde. Zudem werden

die verwendeten IP-Adressbereiche in der Praxis nicht eindeutig sein. DSL-Router für den Heimgebrauch sind häufig so eingestellt, dass der eingebaute DHCP-Server IP-Adressen aus dem Adressbereich `192.168.0.1/24` vergibt. Das mobile Endgerät kann hierbei nicht mehr unterscheiden, an welchen DSL-Router es angeschlossen worden ist, wenn identische Adressbereiche mehrfach verwendet werden.

Bevor die Frage der Positionserkennung geklärt wird, soll zunächst auf die Adresszuweisung mit Hilfe von DHCP eingegangen werden. Dieser Schritt ist notwendig, da er der Positionserkennung vorausgeht.

```
neelix ~ # /sbin/dhcpcd -d -B -A -t 30 eth0
dhcpcd: version 5.2.2 starting
dhcpcd: eth0: executing '/lib/dhcpcd/dhcpcd-run-hooks', reason PREINIT
dhcpcd: eth0: executing '/lib/dhcpcd/dhcpcd-run-hooks', reason CARRIER
dhcpcd: eth0: reading lease '/var/lib/dhcpcd/dhcpcd-eth0.lease'
dhcpcd: eth0: discarding expired lease
dhcpcd: eth0: broadcasting for a lease
dhcpcd: eth0: sending DHCP_DISCOVER (xid 0xf7264b20), next in 4.19
dhcpcd: eth0: offered 192.168.1.32 from 192.168.1.1
dhcpcd: eth0: sending DHCP_REQUEST (xid 0xf7264b20), next in 3.22
dhcpcd: eth0: acknowledged 192.168.1.32 from 192.168.1.1
dhcpcd: eth0: leased 192.168.1.32 for 86400 seconds
dhcpcd: eth0: adding IP address 192.168.1.32/24
dhcpcd: eth0: adding route to 192.168.1.0/24
dhcpcd: eth0: adding default route via 192.168.1.1
dhcpcd: eth0: writing lease '/var/lib/dhcpcd/dhcpcd-eth0.lease'
dhcpcd: eth0: executing '/lib/dhcpcd/dhcpcd-run-hooks', reason BOUND
```

Abbildung B.16: Für eine Adresszuweisung mit Hilfe von DHCP bietet sich das `dhcpcd`-Kommando an. Dieses stellt alle relevanten Adressinformationen über die Standardausgabe bereit, wo sie abgefangen werden können.

Für die Zuweisung einer IP-Adresse setzt das implementierte „Device Backend“ das `dhcpcd`-Kommando ein. Alternativ wurde vom Autor versucht, das `dhclient`-Kommando zu benutzen, was allerdings scheiterte. Lediglich der `dhcpcd` liefert alle erforderlichen Informationen über seine Ausgabe (siehe Abbildung B.16), welche vom „Device Backend“ ausgewertet werden. Zu diesem Zweck muss der `dhcpcd` mit dem Kommandozeilenparameter `-d` in den „Debug-Modus“ geschaltet werden, da nur dann auch die IP-Adresse des Gateways ausgegeben wird. Der Parameter `-B` sorgt dafür, dass sich das Programm nicht „dämonisiert“, also dass es blockierend „im Vordergrund“ verbleibt. Der Schalter `-A` sorgt dafür, dass die Adresszuweisungsprozedur beschleunigt wird, da nach erfolgreicher Zuweisung einer IP-Adresse auf eine Kollisionsprüfung mit Hilfe des „Address

Resolution Protocols“ (ARP) verzichtet wird. Mit Hilfe des Parameters `-t 30` wird eine Zeitschranke von 30 Sekunden vorgegeben, sodass das Kommando in Netzwerken ohne DHCP-Server nach einer definierten Zeitspanne eine Fehlermeldung liefern wird.

Wichtig ist für REACH, dass während einer solchen Assoziationsprozedur keine *wirksamen* „Default Routen“ hinzugefügt werden, da es sonst zu einem Fehlverhalten der „transparenten Proxyserver“ kommen würde (siehe Abschnitt 4.3.2.2). Dies ließe sich hier mit Hilfe des Parameters `-nogateway` effektiv verhindern, jedoch würde dann in der Ausgabe des `dhcpcd`-Kommandos die „`adding route via GATEWAY`“-Zeile nicht mehr auftauchen. Diese Information wird allerdings unbedingt benötigt. Leider ist es mit Hilfe des `dhcpcd`-Kommandos nicht möglich, die mittels DHCP zugewiesene IP-Adresse des Gateways zu erhalten, ohne dass dabei auch eine Route gesetzt wird. Dieses Problem ließ sich mit der Installation einer aktuelleren Version des `dhcpcd`-Kommandos lösen (Version 5.2.2 statt der als stabil gekennzeichneten Version 4.0.15). Die neuere Version überführt zwar weiterhin besagte Einträge in die Routingtabelle, versieht sie allerdings mit einem positiven Wert für die Metrik. Da nur der Weg mit der niedrigsten Metrik verwendet wird, kam es zu keinen Wechselwirkungen mit den „transparenten Proxies“ mehr, da diese eine priorisierte Route mit der Metrik `0` involvieren.

Die Ausgabe von `dhcpcd` wird gezielt auf das Vorhandensein der im Folgenden aufgeführten Zeichenketten hin untersucht. Die dabei ermittelten Informationen werden vom „Device Backend“ vermerkt und bei Bedarf dem REACH-Client mitgeteilt.

- „`adding IP address 192.168.1.32/24`“: Dem Interface wurde die IP-Adresse `192.168.1.32` zugewiesen. Die IP-Netzmaske liegt als Anzahl der 1-Bits (`/24`) vor und muss anschließend noch umgerechnet werden. Schlussendlich liegt sie in diesem Beispiel als `255.255.255.0` vor.

- „`adding default route via 192.168.1.1`“: Die nach der Zeichenkette `via` angegebene IP-Adresse spezifiziert das zugewiesene „Default Gateway“.

- „`BOUND`“: Die Zeile mit dieser Zeichenkette kommt *nach allen* Zeilen, welche für das „Device Backend“ relevante Informationen enthalten. Sie signalisiert eine erfolgreiche Adresszuweisung.

- „`timed out`“: Es konnte keine Adresszuweisung durchgeführt werden. Wahrscheinlich ist kein DHCP-Server verfügbar.

- „`carrier lost`“: Das Netzwerkkabel wurde entfernt. Eine bestehende Adressbindung wurde wieder aufgelöst, oder die Adresszuweisungsprozedur wurde mittendrin abgebrochen.

Konnte die Adresszuweisungsprozedur erfolgreich durchgeführt werden, verfügt das Interface über eine IP-Adresse und eine Netzmaske. Zudem ist die IP-Adresse des zugewiesenen Gateways bekannt.

Die IP-Adresse des Gateways wird im nächsten Schritt dazu benutzt, um die Positionserkennung durchzuführen. Zwar ist die IP-Adresse des Gateways für sich alleine gesehen nicht eindeutig, allerdings besitzt das Gateway eine eindeutige MAC-Adresse. Diese MAC-Adresse gilt es herauszufinden. Sind dem REACH-Client die MAC-Adressen der Gateways aller in Frage kommenden Netzzugangspunkte bekannt, kann so auf den Aufenthaltsort geschlossen werden.

Um die verfügbare IP-Adresse des Gateways in eine MAC-Adresse aufzulösen, wird das Kommando `arp` eingesetzt. Das führt allerdings nur dann zum Erfolg, wenn bereits ein diesbezüglicher Eintrag in der ARP-Tabelle existiert. Nach einer „frischen“ Adresszuweisung über DHCP ist der zugehörige Eintrag noch nicht vorhanden, sodass erst das „Address Resolution Protocol“ (ARP) zur Adressauflösung benutzt werden muss. ARP lässt sich jedoch nur indirekt anstoßen, was beispielsweise durch einen Datentransfer zu der geforderten IP-Adresse erfolgen kann. Das „Device Backend“ bedient sich hierfür des Kommandos `ping`.

```
neelix ~ # arp --device eth0 192.168.1.1
192.168.1.1 (192.168.1.1) -- no entry
neelix ~ #
neelix ~ # ping -r -c 1 -W 1 -I eth0 192.168.1.1
PING 192.168.1.1 from 192.168.1.32 eth0: 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=1.16 ms

--- 192.168.1.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.160/1.160/1.160/0.000 ms
neelix ~ #
neelix ~ # arp --device eth0 192.168.1.1
Address       HWtype  HWaddress           Flags Mask   Iface
my.router     ether   00:17:31:dc:04:fb   C            eth0
neelix ~ #
```

Abbildung B.17: Die MAC-Adresse des zugewiesenen Gateways lässt sich mit Hilfe des Kommandos `arp` aus der ARP-Tabelle auslesen, wenn diese vorher gefüllt worden ist. Dafür wird hier das `ping`-Kommando benutzt.

In Abbildung B.17 ist zweimal die Ausgabe des Kommandos `arp` aufgeführt. Diesem muss explizit das zu nutzende Interface (hier: `eth0`) mitgeteilt werden, damit die Wegewahl umgangen wird. Schließlich ist nicht auszuschließen, dass die hier genannte

IP-Adresse des Gateways `192.168.1.1` bereits über ein anderes Interface angesprochen werden kann, wodurch die MAC-Adresse eines „falschen“ Systems zurückgeliefert werden könnte.

Im ersten Fall war die ARP-Tabelle noch leer, und das Kommando reagierte mit der Zeichenkette „`no entry`“. Das „Device Backend“ führt dieses „erste“ `arp`-Kommando jedoch niemals aus, sondern startet vorab immer das genannte `ping`-Kommando zur Füllung der ARP-Tabelle. Aber auch `ping` muss dazu veranlasst werden, die Routingtabelle zu ignorieren, was mit Hilfe des Parameters `-r` befohlen wird. Dieser Parameter erfordert die explizite Angabe eines Interfaces, was hier der Zusatzparameter `-I eth0` übernimmt. Mit `-c 1` wird das `ping`-Kommando dazu angehalten, nur einen einzigen `ICMP-ECHO-REQUEST` zu versenden. Zudem soll das Programm nach spätestens einer Sekunde (`-W 1`) abbrechen.

Die Ausgaben des `ping`-Kommandos sind für das „Device Backend“ nicht von Interesse. Es dienst lediglich dazu, die ARP-Tabelle zu füllen, was im Beispiel in Abbildung B.17 erfolgreich war: das zweite `arp`-Kommando lieferte schließlich die gesuchte MAC-Adresse des Gateways. In der Ausgabe muss jedoch die korrekte Zeile gefunden werden, was anhand der `Iface`-Spalte zu erfolgen hat. Liegt die korrekte Zeile vor, kann die zugehörige MAC-Adresse ausgelesen werden. Diese kann jedoch auch „`(incomplete)`“ als Inhalt haben, was auf ein zwischenzeitlich abgezogenes Netzwerkkabel schließen lässt.

Schlussendlich steht im Erfolgsfall eine „vollständig aufgebaute Verbindung“ zum Internet bereit, obwohl der Handoverentscheider zu diesem Zeitpunkt noch nicht einmal über die Sichtbarkeit eines Netzzugangspunkts informiert worden ist. Das erfolgt im nächsten Schritt, nachdem das „Device Backend“ die jetzt bekannte MAC-Adresse mit allen hinterlegten MAC-Adressen verglichen hat. Ist die MAC-Adresse bekannt, wird die „Sichtbarkeit“ des zugehörigen Netzzugangspunkts an den REACH-Client gemeldet.

Sollte sich der Handoverentscheider dazu entschließen, diesen „sichtbaren“ Netzzugangspunkt zu benutzen, wird er eine `<Associate>`-Nachricht an das „Device Backend“ versenden. Dieses muss nun nichts weiter unternehmen, da „die Internetverbindung“ bereits besteht. Es wird lediglich vermerkt, dass die Assoziation *angefordert* worden ist, und sofort mit einer `<Associated>`-Nachricht geantwortet. Diese enthält die vorab vom `dhcpcd`-Kommando „abgegriffenen“ IP-Adresse des Interfaces, die IP-Netzmaske sowie die IP-Adresse des Gateways. Sollte es im Folgenden zu einem Verlust des Trägersignals kommen, wird die unterbrochene Assoziation anhand einer `<Dissociated>`-Nachricht signalisiert.

Vom „Device Backend“ benötigte Parameter

Das „Device Backend“ für Ethernet benötigt zwei Parameter, um seine Aufgabe zu erfüllen. Ersterer gilt als hardwarespezifischer Parameter, welcher einmalig angegeben werden muss. Letzterer Parameter stellt einen Detektionsparameter dar, welcher bezüglich eines jeden Netzzugangspunkts spezifiziert werden muss.

- Hardwarespezifische Parameter:
  - „`device`“: benennt die zu überwachende Ethernetschnittstelle, in den gezeigten Beispielen durchgängig das Interface `eth0`.
- Detektionsparameter:
  - „`gateway mac address`“: benennt die MAC-Adresse des Gateways bezüglich dieses Netzzugangspunkts.

Das implementierte „Device Backend“ für Ethernet wurde damit beschrieben. Die genannten Mechanismen setzen das Vorhandensein bestimmter Systemkommandos, wie beispielsweise `ifplugstatus` und `dhcpcd`, voraus. Das vorgestellte „Device Backend“ ist somit hochgradig auf die GNU/Linux-Plattform zugeschnitten.

B.2.2 WLAN nach IEEE 802.11

WLAN stellt eine populäre, drahtlos arbeitende Netzzugangstechnologie dar. WLAN zeichnet sich dadurch aus, dass es mehrere Varianten gibt, welche in verschiedenen Frequenzbereichen arbeiten und dabei unterschiedliche Datenraten erreichen können. Zudem lassen sich diverse Sicherheitsmechanismen auswählen. WLAN-Funknetze können verschlüsselt oder unverschlüsselt arbeiten, wobei im ersten Fall eine Vielzahl an Schemata angeboten werden. Die Herausforderung bei der Erstellung eines „Device Backends“ für WLAN bestand darin, für alle möglichen Szenarien eine Lösung anbieten zu können, also Assoziationen zu ermöglichen.

Das vorliegende „Device Backend“ sieht für einen WLAN-Adapter des mobilen Endgeräts lediglich den „Station mode“ vor, also die Teilnahme an Infrastrukturnetzen. Der so genannte „Ad-hoc mode“ wird nicht unterstützt, da das Augenmerk von REACH auf der Erreichbarkeit von REACH-Proxyservern liegt, was in der aktuellen Implementierung nur beim Infrastrukturmodus gewährleistet werden kann. Der „Master mode“, welcher aus einem WLAN-Adapter des mobilen Endgeräts eine Basisstation macht, wird aus denselben Gründen ebenfalls nicht unterstützt.

WLAN arbeitet „assoziationsorientiert“, da sich ein mobiles Endgerät mit einer Basisstation *assoziieren* muss. Dabei kann ein mobiles Endgerät eine Vielzahl nutzbarer Basisstationen in seiner Umgebung vorfinden, muss davon jedoch eine auswählen und sich mit dieser *assoziieren*. Dabei können Anmelde- und Verschlüsselungsschemata durchlaufen werden.

Detektion des Netzzugangsgeräts

WLAN-Adapter werden unter GNU/Linux über Interfaces angesprochen. Deren Namen folgen je nach zu Grunde liegendem Treiber dem Benennungsschema `wlanX`, `athX` oder `ethX`, wobei `X` eine natürliche Zahl darstellt. Die im Demonstrator verwendeten WLAN-Adapter des Typs „ZyXEL G-220 v2“ basieren auf einem ZD1211-Chip und werden vom Linux-Kernel durch den gleichnamigen Treiber über den „Universal Serial Bus“ (USB) angesprochen. Die Interfaces werden hierbei beginnend mit `wlan0` benannt.

Die erste Aufgabe des „Device Backends“ ist die Erkennung des Netzzugangsgeräts, also des WLAN-Adapters. Der USB-Stick kann schließlich durch den Nutzer spontan eingesteckt und wieder abgezogen werden, was automatisch durch das „Device Backend“ erkannt werden muss.

Der Linux-Kernel erkennt einen verfügbar gewordenen WLAN-Adapter augenblicklich; er aktiviert daraufhin den zugehörigen Treiber und vergibt einen Gerätenamen. Dieser Gerätename ist persistent, wird also pro „gesehenem“ Adapter dauerhaft vermerkt. Dies wird bei aktuellen Linux-Installationen durch den „Device Manager“ `udev` [Wiki09g] sichergestellt. Dadurch ist es beispielsweise egal, an welchen USB-Port ein WLAN-Adapter angeschlossen wird, oder ob vorher noch ein anderer WLAN-Adapter angeschlossen worden war. Ist der Interfacename eines bestimmten Adapters einmal bekannt, kann sich der Nutzer sicher sein, dass beide dauerhaft miteinander verknüpft bleiben werden.

Ist ein Adapter verfügbar und konnte der zugehörige Treiber aktiviert werden, taucht eine zugehörige Zeile in der Pseudo-Datei `/proc/net/wireless`, welche durch den Linux-Kernel bereit gestellt wird, auf. Da jeder verfügbare WLAN-Adapter aufgeführt wird, muss die relevante Zeile herausgesucht werden. Da die erste Spalte das Interface benennt, ist eine Zuordnung problemlos möglich (siehe Abbildung B.18).

Wird ein WLAN-Adapter deaktiviert oder entfernt, verschwindet die zugehörige Zeile aus der in Abbildung B.18 gezeigten Pseudodatei. Es bietet sich damit an, sie zyklisch auf Änderungen hin zu untersuchen. Das beschriebene „Device Backend“ liest die Datei hierfür sekündlich ein und generiert bei einem Zustandswechsel eine Änderungsnachricht an den REACH-Client. Wird ein beobachteter Adapter erstmalig gefunden, generiert das „Device Backend“ eine `<HardwareIsAvailable>`-Nachricht. Wird er hingegen nicht

```
neelix ~ # cat /proc/net/wireless
Inter-| sta-|   Quality        |   Discarded packets
 face | tus | link level noise |  nwid  crypt   frag  retry   misc
 wlan0: 0000    0     0     0        0      0      0      0      0
```

Abbildung B.18: In der Pseudodatei `/proc/net/wireless` werden alle momentan verfügbaren WLAN-Adapter aufgeführt. Jedes Interface wird durch eine eigene Zeile beschrieben (die Darstellung wurde aus Platzgründen rechtsseitig gekürzt).

länger gelistet, ist eine `<HardwareIsNotAvailable>`-Nachricht die Folge.

In Abbildung B.18 sind in den Spalten drei, vier und fünf jeweils noch Statusinformationen bezüglich der Linkgüte („`Quality`“) in Bezug auf „`link`“, „`level`“ und „`noise`“ aufgeführt. Diese Werte sind hier jedoch alle mit Null angegeben, da sie nur dann gültig sind, wenn der WLAN-Adapter mit einer Basisstation assoziiert ist. Im dargestellten Fall bestand jedoch keine Assoziation, und damit waren keine Gütewerte verfügbar. Der Vorteil bei der Ermittlung von Gütewerten über diese Pseudodatei liegt darin, dass die Werte augenblicklich aktualisiert werden, wenn Daten von der zugehörigen Basisstation empfangen werden. Demnach können Gütewerte mit einer hohen Rate ermittelt werden, was sich der Handoverentscheider beim Einsatz von Regressionsrechnung zu Nutze macht.

Linkgütewerte werden jedoch nicht nur bezüglich einer aktiven Assoziation benötigt, sondern auch für alle „sichtbaren“ Basisstations-Kandidaten. Deren Gütewerte stehen allerdings nicht über *diesen* Mechanismus bereit, sondern müssen über aktives „Scannen“ ermittelt werden, was ab Seite 298 beschrieben wird.

Aktivierung eines WLAN-Adapters

Ist ein WLAN-Adapter verfügbar, kann er zum „Scannen“ herangezogen werden. Dazu muss er vorab aktiviert worden sein, was mit Hilfe des Kommandos `ifconfig` erfolgt.

Ein vom Kernel erkannter WLAN-Adapter ist anfangs noch inaktiv. Die Ausgabe des Kommandos `ifconfig` sieht dann so aus wie im oberen Teil von Abbildung B.19. Mit Hilfe des Parameters `up` kann der Adapter aktiviert werden, wonach die Ausgabe von `ifconfig` dem unteren Teil der Ausgabe in Abbildung B.19 entspricht. Der Unterschied zwischen einem inaktiven und einem aktivierten Adapter manifestiert sich in der zweiten Zeile der Ausgabe. Ist hier das Schlüsselwort „`UP`“ zu finden, gilt der Adapter als aktiv und es kann zum „Scannen“ herangezogen werden. Sollte das Schlüsselwort fehlen, muss der Adapter erst aktiviert werden.

```
neelix ~ # /sbin/ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr 00:13:49:71:80:fc
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

neelix ~ # /sbin/ifconfig wlan0 up
neelix ~ # /sbin/ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr 00:13:49:71:80:fc
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Abbildung B.19: Mit Hilfe des Kommandos `ifconfig` kann überprüft werden, ob ein WLAN-Adapter aktiviert worden ist. Sollte dies nicht der Fall sein, kann dies mit Hilfe des Parameters `up` nachgeholt werden.

Suche nach Netzzugangspunkten

Ist ein WLAN-Adapter verfügbar und aktiv, wird er vom „Device Backend“ zum „Scannen“ nach Basisstationen herangezogen. Relevante Basisstationen müssen dem „Device Backend“ vorab bekannt gemacht worden sein, da sich nicht jede gefundene Basisstation für eine Assoziation eignet: nur zu bekannten Basisstationen können auch Authentifikationsparameter (Assoziationsparameter) hinterlegt worden sein, welche zum Beitritt zu verschlüsselten Funknetzen unumgänglich sind.

Der eigentliche Suchmechanismus ist im Kernel implementiert und kann durch das Kommando „`/sbin/iwlist wlan0 scan`“ angestoßen werden. Der Gerätename `wlan0` ist entsprechend anzupassen.

Abbildung B.20 zeigt hierbei die Ausgabe des „Scan“-Kommandos, wenn das Interface vorab nicht aktiviert wurde. Da in diesem Fall keine Ergebnisse zurückgeliefert werden können, ist eine Aktivierung des Interfaces unumgänglich.

War der Adapter bereits aktiv, wird ein Suchvorgang angestoßen. Dieser kann je nach Hardware und Treiber mehrere Sekunden in Anspruch nehmen, da hierbei aktiv alle relevanten Frequenzen auf das Vorhandensein von WLAN-Funknetzen hin untersucht werden. Die Ausgabe des Kommandos ähnelt derjenigen aus Abbildung B.21, welche jedoch aus Gründen der Übersichtlichkeit vom Autor gekürzt wurde. Jede erkannte Basisstati-

```
neelix ~ # /sbin/iwlist wlan0 scan
wlan0     Interface doesn't support scanning : Network is down
```

Abbildung B.20: Die in der Umgebung des mobilen Endgeräts verfügbaren WLAN-Basisstationen können durch das `iwlist`-Kommando „ge`scan`nt“ werden. Hier wird ein Fehlerfall gargestellt: der Adapter war nicht aktiv.

on wird durch mehrere Zeilen beschrieben. Ein solcher „Block“ beginnt immer mit dem Schlüsselwort `Cell` gefolgt von der Zellennummer, welche die gefundenen Basisstationen aufsteigend nummeriert. Die zu jeder gefundenen Basisstation relevanten Informationen sind der „Effective Service Set Identifier“ („`ESSID`“), die Linkgüte („`Quality`“) und die Signalstärke („`Signal level`“). Alle weiteren Ausgaben sind für das „Device Backend“ nicht von Interesse.

```
neelix ~ # /sbin/iwlist wlan0 scan
wlan0     Scan completed :
          Cell 01 - Address: 00:17:31:DC:04:FB
                    Channel:11
                    Frequency:2.462 GHz (Channel 11)
                    Quality=69/100  Signal level=69/100
                    Encryption key:on
                    ESSID:"REACH Access Point 1"
```

Abbildung B.21: Im Erfolgsfall liefert das „Scannen“ eine Liste an verfügbaren Basisstationen. Aus Gründen der Übersichtlichkeit ist die hier dargestellte Ausgabe vom Autor reduziert worden.

Der REACH-Client macht potentielle Netzzugangspunkte anhand der ESSID des Funknetzes bekannt. Die ESSID stellt damit den Detektionsparameter dar, welcher Aussagen über die Sichtbarkeit eines Netzzugangspunkts ermöglicht. Ist im Ergebnis eines Suchvorgangs eine hinterlegte ESSID erstmalig aufgeführt, wird dem REACH-Client deren „Sichtbarkeit“ mitgeteilt. Fehlt der betroffene Eintrag dagegen erstmalig, wird der zugehörige Netzzugangspunkt als „unsichtbar“ kommuniziert.

Für jeden „sichtbaren“ Netzzugangspunkt liegen gemäß des Suchergebnisses ein Linkgüte- und ein Signalstärkewert vor. Es konnte jedoch nicht herausgefunden werden, was diese Werte genau bedeuten. Bei einem Linux-Kernel der Version 2.6.30 waren beim `zd1211`-Treiber beide Werte immer gleich. Bei älteren Kerneln unterschieden sich beide Werte noch, wobei sich der Signalstärkewert einen höheren Dynamikbereich aufwies als der Linkgütewert. Anscheinend ist die Bedeutung dieses Werts sowohl von Kernelversion zu Kernelversion als auch von Treiber zu Treiber unterschiedlich.

Assoziation mit einem Netzzugangspunkt

Ein „sichtbarer“ Netzzugangspunkt mit „ausreichender“ Signalstärke bzw. Linkgüte gilt für den Handoverentscheider als benutzbar. Er kann den Netzzugangspunkt für einen Assoziationsprozess heranziehen, was auf Ebene des Betriebssystems durch Benutzung eines „Supplikanten“ („Bittsteller“) erfolgt. Der hierfür unter GNU/Linux verfügbare `wpa_supplicant` unterstützt sowohl unverschlüsselte Funknetze sowie mit den Methoden „Wired Equivalent Privacy“ (WEP), „Wi-Fi Protected Access“ (WPA) oder „Wi-Fi Protected Access 2“ (WPA2) verschlüsselte WLAN-Netze. Das macht ihn zu einem universell einsetzbaren Werkzeug, wenn sich ein mobiles Endgerät mit einer WLAN-Basisstation assoziieren soll.

Das `wpa_supplicant`-Kommando wird dabei mit vier „Schaltern“ aufgerufen (siehe Abbildung B.22). Mit `-d` werden zusätzliche Debug-Ausgaben aktiviert, welche die Erkennung bestimmter Fehlerfälle erlauben. Zudem müssen der Treiber („Driver“, `-D`) zum Zugriff auf den Kernel, das Interface („Interface“, `-i`) sowie der Dateiname einer Konfigurationsdatei („Configuration“, `-c`) angegeben werden.

```
neelix ~ # wpa_supplicant -d -D wext -i wlan0 -c ./supplicant_reach1.conf
CTRL-EVENT-SCAN-RESULTS
Trying to associate with 00:17:31:dc:04:fb (SSID='REACH Access Point 1' freq=2462 MHz)
Associated with 00:17:31:dc:04:fb
WPA: Key negotiation completed with 00:17:31:dc:04:fb [PTK=CCMP GTK=CCMP]
CTRL-EVENT-CONNECTED - Connection to 00:17:31:dc:04:fb completed (auth) [id=0 id_str=]
```

Abbildung B.22: Mit Hilfe des `wpa_supplicant`-Kommandos wird eine Assoziation mit einer Basisstation eingerichtet. In diesem Beispiel wurde aus Gründen der Übersichtlichkeit auf den Debug-Parameter verzichtet.

Der `wpa_supplicant` gibt nach seinem Start diverse Meldungen aus, wovon die meisten für das „Device Backend“ ohne Belang sind. Es geht lediglich darum, zu erkennen, ob die Assoziation erfolgreich hergestellt werden konnte. Eine gültige Assoziation liegt vor, wenn die Zeichenkette „`CTRL-EVENT-CONNECTED`“ gefunden werden konnte. Werden dagegen die Zeichenketten „`Network is down`“ (Netzzugangsgerät nicht mehr verfügbar), „`timed out`“ (Fehlschlag bei Anmeldung) oder „`Associated with 00:00:00:00:00:00`“ (tritt vor allem bei Treiberproblemen auf) entdeckt, wird das `wpa_supplicant`-Kommando augenblicklich vom „Device Backend“ beendet.

Das, was von Netzzugangspunkt zu Netzzugangspunkt unterschiedlich sein wird, sind die Konfigurationsdateien bezüglich des `wpa_supplicant`-Kommandos. Diese müssen speziell auf das jeweils eingesetzte Authentifizierungsschema zugeschnitten werden und enthalten beispielsweise Nutzernamen und Passwörter.

Der in Abbildung B.22 gezeigte Aufruf des `wpa_supplicant`-Kommandos referen-

```
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
ap_scan=1

network={
  scan_ssid=1
  proto=WPA2
  key_mgmt=WPA-PSK
  ssid="REACH Access Point 1"
  #psk="PasswortZuAccessPoint"
  psk=ERRECHNETER_SCHLÜSSEL
}
```

Abbildung B.23: Hier wird eine Konfigurationsdatei für das `wpa_supplicant`-Kommando gezeigt, welche auf dem WPA-PSK-Verfahren basiert.

ziert die Konfigurationsdatei `supplicant_reach1.conf`, deren Inhalt in Abbildung B.23 (mit unkenntlich gemachten Passwörtern) dargestellt ist. Sie benennt als Betriebsmodus WPA2 und enthält zudem einen gemeinsam genutzten Schlüssel „Pre-shared Key“ (PSK). Zur Erstellung dieser Datei kann das Kommandozeilenprogramm `wpa_passphrase` benutzt werden, welches aus einer gegebenen SSID und einem Passwort (im Klartext) den `psk`-Parameter errechnet.

Der `wpa_supplicant` kann jedoch auch mit aufwendigeren Schemata benutzt werden, wie beispielsweise dem „Extensible Authentication Protocol“ (EAP) über IEEE 802.1X. Dieses wird im Campusnetz der TU Ilmenau eingesetzt, da sie am „Roaming-Verbund“ des „Deutschen Forschungsnetzes“ (DFN) teilnimmt. Die Anmeldung erfolgt hierbei nicht länger auf vorab hinterlegten Schlüsseln (PSK) sondern auf der Basis von Zertifikaten, Nutzername und Nutzerpasswort (WPA-EAP).

Die Konfigurationsdatei des `wpa_supplicant`-Kommandos sieht im Fall von WPA-EAP komplexer aus. Das Rechenzentrum der TU Ilmenau stellt hierfür die in Abbildung B.24 dargestellte Schablone [Tui8] bereit, in welche lediglich der Nutzername und das Nutzerpasswort eingetragen werden müssen. Zudem muss vorab eine Datei mit Zertifikaten heruntergeladen werden, sodass die Referenz in der Konfigurationsdatei auf diese Datei verweist. Es bietet sich an, die Datei mit den Zertifikaten in denselben Ordner abzulegen, in dem auch alle anderen Konfigurationsdateien des REACH-Clients liegen.

Es ist damit erkennbar, dass die Konfigurationsdatei je nach Anwendungsfall unterschiedlich aussehen wird und zudem Zertifikate erforderlich sein können. Bezüglich des „Device Backends“ wurde daher beschlossen, dass die Konfigurationsdateien des `wpa_supplicant`-Kommandos vorab vom Nutzer erstellt werden müssen und diese nicht

```
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
eapol_version=1
ap_scan=1
fast_reauth=1

network={
  ssid="802.1X"
  proto=WPA
  key_mgmt=WPA-EAP
  pairwise=TKIP
  group=TKIP
  eap=TTLS
  anonymous_identity="anonymous@tu-ilmenau.de"
  identity="IhrBenutzername@tu-ilmenau.de"
  password="IhrGanzGeheimesRZPasswort"
  ca_cert="/etc/cert/TUIlmenau-CACert.pem"
  phase2="auth=PAP"
  priority=2
}
```

Abbildung B.24: Diese Schablone einer Konfigurationsdatei für den `wpa_supplicant` wird vom Rechenzentrum der TU Ilmenau zum Download angeboten. Bei dem hierbei eingesetzten Verfahren WPA-EAP werden ein Nutzername, ein Passwort und eine Datei mit Zertifikaten benötigt.

durch den REACH-Client modifiziert werden. Die Alternative hieße, dass die Dateien bei Bedarf aus Konfigurationsparametern zusammengesetzt werden müssten, was jedoch einen umfangreichen Parametersatz erfordern würde. Dabei bestünde die Gefahr, dass ein bislang nicht berücksichtigtes Nutzungsschema nicht unterstützt werden könnte, wenn ein bestimmter Parameter vergessen worden wäre.

Da alle Netzzugangspunkte bereits im Vorfeld bekannt sein müssen, ist es nach Ansicht des Autors im aktuellen Stadium ein vertretbarer Aufwand, dass zu allen WLAN-Basisstationen (genauer: WLAN-Funknetzen) vorab eine solche Konfigurationsdatei angelegt und erforderliche Zertifikate beschafft werden müssen.

Zuweisung von IP-Adressinformationen

Nachdem die Assoziationsprozedur erfolgreich durchlaufen worden ist, müssen Einstellungen bezüglich der IP-Adresskonfiguration durchgeführt werden. Das ist jedoch nicht mehr die Aufgabe des `wpa_supplicant`-Kommandos, da dieses lediglich einen „Ethernet-ähnlichen“ Netzzugang herstellt.

Für die Ermittlung einer IP-Adresse für das Interface, dessen IP-Netzmaske, der IP-Adresse diverser DNS-Server und der IP-Adresse des Gateways wird auf DHCP gesetzt. Der Mechanismus basiert wiederum auf dem Kommando `dhcpcd`, welches bereits in Abschnitt B.2.1 in Bezug auf Ethernet beschrieben worden ist. Die Implementierung der beiden „Device Backends“ ist in Bezug auf die Verwendung des `dhcpcd`-Kommandos identisch.

Nachdem die IP-Adressinformationen zugewiesen worden sind, gilt der Netzzugang als aufgebaut. Das „Device Backend“ signalisiert dies mit Hilfe einer `<Associated>`-Nachricht. Im Gegensatz zu Ethernet ist zu diesem Zeitpunkt bereits klar, mit welchem Netzzugangspunkt eine Assoziation besteht, sodass die MAC-Adresse des Gateways nicht ermittelt zu werden braucht. Bei WLAN wäre ein solches Schema sowieso nicht zu empfehlen, da mehrere Basisstationen zu einem gemeinsamen „Roaming-Verbund“ zusammengeschlossen sein können, und MAC-Adressen im Vorfeld nicht bekannt sein können (oder nur mit einem hohen Aufwand).

Vom „Device Backend“ benötigte Parameter

Auch das „Device Backend“ für WLAN unterscheidet zwei Arten von Parametern:

- Hardwarespezifische Parameter:
  - „`device`“: Benennt das zu überwachende WLAN-Interface, hier `wlan0`.
  - „`driver`“: Der Treiber, der vom `wpa_supplicant`-Kommando zur Ansteuerung des WLAN-Interfaces verwendet werden soll.
  - „`measurement_type`“: Sollen Signalstärke- (`signalstrength`) oder Linkgütewerte (`quality`) ermittelt werden?
  - „`measurement_ms`“: Der Abstand zwischen zwei Messungen kann bezüglich aktiver Assoziationen in Millisekunden spezifiziert werden.
- Detektions- und Assoziationsparameter:
  - „`SSID`“: Hiermit wird der „Service Set Identifier“ (SSID) bzw. der „Effective Service Set Identifier“ (ESSID) des beschriebenen Netzzugangspunkts benannt. Von einem ESSID spricht man, wenn mehrere Basisstationen zu einem gemeinsamen Funknetz zusammengefasst worden sind. Der Parameter lautet in beiden Fällen `SSID` und wird zur Erkennung verfügbarer Basisstationen herangezogen.

– „`supplicant_configfile`“: Hiermit wird eine bereits existierende Konfigurationsdatei referenziert. Diese wird benötigt, wenn eine Assoziation mit einer Basisstation hergestellt werden soll.

Das soeben beschriebene „Device Backend“ für WLAN ist hochgradig auf GNU/Linux-basierte Systeme zugeschnitten. Es gilt, dass die genannten Systemkommandos auf dem mobilen Endgerät installiert sein müssen. Ferner ist sicherzustellen, dass neuere Versionen keine abweichenden Ausgaben erzeugen. Sollte dies der Fall sein, wird es zu Fehlfunktionen kommen, da die jeweiligen Schlüsselwörter nicht mehr erkannt werden oder die Extraktion von Adressinformationen nicht länger funktioniert. In diesen Fällen muss das „Device Backend“ angepasst oder es muss auf eine ältere Version der Hilfsprogramme zurückgegriffen werden.

Fallstricke bei der Messwertermittlung

Für eine sinnvolle Regressionsrechnung muss eine häufige Ermittlung der Signalstärke- bzw. Linkgütewerte stattfinden. Die Regressionsrechnung ist schließlich dazu da, zukünftige Abrisse durch eine Beobachtung vergangener und aktueller Messwerte vorauszusagen. Das funktioniert allerdings nur dann, wenn nicht seltener als ca. einmal pro Sekunde ein neuer Wert ermittelt werden kann. Diese Problemstellung wurde bereits in Abschnitt 6.2.4 diskutiert.

Bei WLAN stehen zwei Mechanismen bereit, um Signalstärke- und Linkgütewerte zu erhalten. Zum einen kann die Pseudodatei `/proc/net/wireless` ausgelesen werden, was zwar hohe Erfassungsraten erlaubt, jedoch nur bezüglich einer bestehenden Assoziation Werte liefert. Möchte man hingegen auch die Eignung alternativer Basisstationen in Erfahrung bringen, muss eine Suche angestoßen werden. Diese liefert Signalstärke- und Linkgütewerte für alle „sichtbaren“ Basisstationen in Reichweite, allerdings mit einer für Regressionszewecke zu geringen Erfassungsrate. Eine Regressionsrechnung kann damit ausschließlich bezüglich einer bestehenden Assoziation durchgeführt werden, und nicht in Bezug auf Basisstations-Kandidaten in der Umgebung des mobilen Endgeräts.

Der Grund liegt darin, dass für einen solchen „Scanvorgang“ alle vom WLAN-Adapter unterstützten Frequenzen untersucht werden müssen, was eine gewisse Zeitspanne erfordert. Für die genutzten USB-basierten „Zyxel G-220 v2“ beträgt die Dauer ca. 800ms, was durchaus noch vertretbar wäre. Für die in der REACH-Box verbauten WLAN-Adapter des Typs „Wistron DCMA-81“ mit ihren 802.11 a/b/g-Transceivern beträgt die gemessene Dauer bereits 4200ms, was *deutlich* zu viel ist, um noch eine sinnvolle Regressionsrechnung durchführen zu können.

Das Problem erweist sich auf den zweiten Blick noch als weitaus gravierender. Bei einem „Scan“ muss der Transceiver seine Frequenz ändern, sodass er keine Daten mehr mit der Basisstation, mit welcher er assoziiert ist, austauschen kann. Die Folge ist, dass man mit einem solchen „Scanvorgang“ eine bestehende Assoziation stören und sogar unterbrechen kann. Eine Suche ist nur dann unproblematisch, wenn keine Daten übertragen werden sollen. Während einer laufenden Datenübertragung sind die Chancen eines Abrisses allerdings hoch.

Das bedeutet leider nicht nur, dass keine Gütewerte mit ausreichender Rate ermittelt werden können, sondern auch, dass während einer bestehenden Assoziation nicht nach alternativen Basisstationen gesucht werden darf. Man müsste warten, bis die Signalstärke der aktuell assoziierten Basisstation derart niedrig geworden ist, dass das „Device Backend“ einen Linkabriss riskiert und seine Umgebung nach anderen Basisstationen absucht.

Glücklicherweise ist das Problem bekannt und es steht eine „sehr junge“ Lösung seitens der Kernelentwickler zur Verfügung. In Kernelversion 2.6.32 wurde der Suchmechanismus für kernelinterne Treiber derart geändert, dass ein Suchvorgang eine aktive Assoziation nicht länger negativ beeinflusst. Das wird dadurch erreicht, dass der „Scan“ immer nur in unkritischen Phasen stattfindet, sodass es zu keinen Abrissen durch übersehene Rahmen mehr kommt. Der Preis, den man aus Anwendersicht hierfür zu zahlen hat, ist ein wesentlich längerer Suchvorgang. So wird in [Will09] angegeben, dass mit einer Zeitspanne von 30 Sekunden pro Durchlauf zu rechnen ist. Das ist für eine Suche nach alternativen Basisstationen ausreichend, verhindert jedoch diesbezüglich den sinnvollen Einsatz von Regressionsrechnung.

B.2.3 Mobiltelefone mit Bluetooth-Anbindung

Das dritte für den Demonstrator implementierte „Device Backend“ ist für einen Internetzugang über Bluetooth-fähige Mobiltelefone zuständig. Das Mobiltelefon wird dazu als Modem angesprochen, wobei die Kommunikation zwischen mobilem Endgerät und Mobiltelefon drahtlos über Bluetooth stattfindet. Bluetooth bietet hierfür das serielle Profil an, sodass selbige Mechanismen zum Einsatz kommen können, welche auch für seriell angebundene Modems angewendet werden.

Es spielt dabei keine Rolle, ob das Mobiltelefon mit Hilfe des „General Packet Radio Service“ (GPRS) über das „Global System for Mobile Communications“ (GSM) oder mit Hilfe von „Enhanced Data Rates for GSM Evolution“ (EDGE) arbeitet, oder ob es auf dem „Universal Mobile Telecommunications System“ (UMTS) basiert [GMH0]. Aktuelle Weiterentwicklungen wie „High Speed Downlink Packet Access“ (HSDPA) und

zukünftige Mobilfunksysteme wie „Long Time Evolution“ (LTE) bilden keine Ausnahme. Der hier vorgestellte Mechanismus, bei welchem sich das Mobiltelefon aus Sicht des mobilen Endgeräts wie ein Modem verhält, ist in allen Fällen identisch. Auf der seriellen Schnittstelle kommt hierbei das „Point-to-Point Protocol“ (PPP) zum Einsatz. Es ist dabei egal, ob die serielle Schnittstelle physikalisch vorhanden ist, oder ob diese durch das serielle Profil von USB, einer Infrarotübertragung nach der „Infrared Data Association“ (IrDA) oder mittels Bluetooth bereit gestellt wird. In allen diesen Fällen taucht im `/dev`-Verzeichnis des GNU/Linux-Systems eine Gerätedatei auf, welche eine serielle Schnittstelle repräsentiert. Über diese Gerätedatei wird eine „Einwahl“ mit Hilfe von „Attention“-Kommandos (AT-Kommandos, Hayes-Befehlssatz) angestoßen und bei Erfolg eine PPP-Instanz gestartet.

PPP kommt hierbei nur auf der seriellen Schnittstelle zum Einsatz, wird also vom mobilen Endgerät und vom Mobiltelefon „gesprochen“. Das Mobiltelefon stellt seinerseits einen paketorientierten Netzzugang bereit, beispielsweise IP über GPRS, und wird schlussendlich Datenpakete zwischen Telefonnetz und PPP-Übertragungskanal vermitteln.

Es gibt eine Vielzahl an Anleitungen im Internet, wie GNU/Linux-basierte Endgeräte konfiguriert werden müssen, damit ein Zugang zum Internet über ein Mobiltelefon bereit gestellt werden kann [Post09, Will08]. Die dazu notwendigen Schritte sind prinzipiell immer dieselben, mussten für REACH jedoch in Form eines „Device Backends“ umgesetzt werden.

Diskussion: Netzzugangsgerät und Netzzugangspunkt

Auch dieses „Device Backend“ hat die Aufgabe, das Vorhandensein eines Netzzugangsgeräts zu erkennen und diesbezügliche Zustandswechsel an den REACH-Client zu melden. Unklar ist anfangs allerdings, was überhaupt mit „Netzzugangsgerät“ gemeint ist. So wird beispielsweise ein als USB-Stick ausgeprägter Bluetooth-Adapter verwendet, welcher den Zugriff auf das Mobiltelefon erst ermöglicht. Der Adapter wird dabei nicht dauerhaft am mobilen Endgerät angeschlossen sein, und ein Mobiltelefon kann sich auch außerhalb der Funkreichweite des Bluetooth-Adapters befinden.

Es gibt mehrere Möglichkeiten, was mit „der Verfügbarkeit des Netzzugangsgeräts“ und „der Sichtbarkeit von Netzzugangspunkten“ gemeint sein kann:

1. Mit „dem Netzzugangsgerät“ könnte der Bluetooth-Adapter am mobilen Endgerät gemeint sein. Sobald beispielsweise der Bluetooth-USB-Stick angeschlossen ist, gelte dann „das Netzzugangsgerät“ als verfügbar. Die nächste Frage wäre dann aber, was mit „Netzzugangspunkten“ und deren Sichtbarkeit gemeint wäre.

Ist der Bluetooth-Adapter am mobilen Endgerät verfügbar, kann die Umgebung nach bekannten Bluetooth-Geräten abgesucht werden. Im vorliegenden Fall wäre dies das Mobiltelefon, welches (1.) eingeschaltet sein, (2.) Bluetooth aktiviert haben und (3.) sich in Funkreichweite des mobilen Endgeräts befinden muss.

Wäre damit das Mobiltelefon der Netzzugangspunkt, welcher als „sichtbar“ gemeldet werden soll, sobald eine Verbindung über Bluetooth möglich erscheint? Dagegen spricht, dass „Sichtbarkeit“ die Möglichkeit einer Assoziation impliziert, also dass ein seitens des Handoverentscheiders geäußertes `<Associate>`-Kommando auch zu einem positiven Ergebnis führen kann. Dies ist allerdings alleinig durch die Erreichbarkeit des Mobiltelefons über Bluetooth noch nicht sichergestellt. Schließlich könnte sich der Nutzer samt seiner Geräte gerade in einem Tunnel befinden, in welchem keine Netzabdeckung besteht. Der Handoverentscheider würde dann wieder und wieder versuchen, eine Verbindung zum Internet aufzubauen, da der gewünschte Netzzugangspunkt (das Mobiltelefon) „sichtbar“ wäre. Daher wurde nach einer anderen Aufteilung gesucht.

2. Als „sichtbar“ darf ein Netzzugangspunkt also nur dann gelten, wenn auch eine Assoziation möglich ist. Im Fall von Mobilfunk folgt daraus, dass sich das Mobiltelefon in ein Funknetz eingebucht haben und dass die Signalstärke ausreichend sein muss. *Der* Netzzugangspunkt wäre damit das Mobilfunknetz. Da jedoch ein Mobiltelefon durch sein „Subscriber Identity Module“ (SIM, SIM-Karte) an einen bestimmten Netzanbieter gebunden ist, hat man als Telefonbenutzer keine sinnvolle Auswahlmöglichkeit, was mehrere Netzzugangspunkte im Sinne von REACH angeht. Selbst die Vielzahl an Basisstationen, die einem Mobiltelefon zur Verfügung stehen, sind aus Sicht eines Telefonbenutzers nicht unterscheidbar. Allerdings kann es bei einem Auslandsaufenthalt notwendig sein, einen bestimmten Netzanbieter zu bevorzugen. Die hierfür notwendigen Schritte gehen jedoch über die hier diskutierten AT-Kommandos hinaus und würden zusätzliche Parametersätze samt Fallunterscheidung erfordern.

   Daher wird folgende Vereinfachung angenommen: Ein Mobiltelefon bietet nur zu einem einzigen Netzanbieter eine Assoziationsmöglichkeit. Wenn das Mobiltelefon eingebucht ist, wird der Netzzugangspunkt als „sichtbar“ gemeldet. Ist das Mobiltelefon isoliert, gilt der Netzzugangpunkt als „unsichtbar“. Zu einem „sichtbaren“ Netzzugangspunkt fallen zudem Signalstärke- bzw. Linkgütewerte an.

   Ferner *könnte* noch unterschieden werden, ob beispielsweise gerade die „Homezone“ aktiv ist, oder ob überhaupt der heimische Netzanbieter ausgewählt wurde.

Bei einem Auslandsaufenthalt könnten sonst durch „Roaming“ unerwünscht hohe Rechnungsbeträge provoziert werden. Hierzu wären zusätzliche Detektionsparameter notwendig, um den „eigenen“ Netzanbieter erkennen zu können und um die Verfügbarkeit der „Homezone“ zu prüfen.

Das jetzige „Device Backend“ implementiert eine solche Erkennung nicht. Es wird daher auch kein Detektionsparameter angeboten, welcher die Erkennung des aktuellen Netzanbieters erlaubt. Momentan wird lediglich festgestellt, ob eine „Internetverbindung“ möglich ist.

Bliebe abschließend zu klären, was mit „dem Netzzugangsgerät“ gemeint ist, und welche Randbedingungen erfüllt sein müssen, damit es als verfügbar gelten kann. Hierbei sind wiederum mehrere Möglichkeiten denkbar:

a) „Das Netzzugangsgerät“ ist der Bluetooth-Adapter, welcher als verfügbar gemeldet wird, wenn er am mobilen Endgerät aktiviert werden konnte.

   Von Vorteil hierbei wäre es, dass am Verfügbarkeitsstatus sofort abgelesen werden könnte, ob der Bluetooth-Adapter angeschlossen worden ist. Allerdings würde sich diese Festlegung auf die weiteren Schritte nachteilig auswirken, denn zwischen einem als verfügbar gemeldeten Bluetooth-Adapter und der nächsten Erfolgsmeldung an den REACH-Client, welche die Möglichkeit einer Assoziation signalisieren würde, sind diverse Zwischenschritte notwendig. Im Detail betrachtet bedeutet dies, dass das Mobiltelefon mittels Bluetooth angesprochen werden können muss (eingeschaltet, mit aktiviertem Bluetooth-Transceiver und in Reichweite). Weiterhin muss eine „Bindung“ zwischen beiden Geräten vollzogen worden sein, damit sie sich gegenseitig kennen und eine verschlüsselte Kommunikationsbeziehung besteht. Wurde eine solche Bindung hinterlegt, steht mit dem seriellen Profil ein Übertragungskanal bereit, über welchen auf das GSM-Modem eingewirkt werden kann. Hierbei gilt es nun, mit Hilfe von AT-Kommandos den Status der Netzabdeckung zu ermitteln, um daraus auf „die Sichtbarkeit des Netzzugangs“ zu schließen.

   Die Verfügbarkeitserkennung des Bluetooth-Adapters wäre dabei das kleinste der genannten Probleme. Kritischer ist, ob eine „Bindung“ besteht und ob sich das gewünschte Mobiltelefon mit aktiviertem Bluetooth-Transceiver in Reichweite des mobilen Endgeräts befindet.

b) Als „Netzzugangsgerät“ gilt daher nicht der Bluetooth-Adapter, sondern das Mobiltelefon. Dieses wird dann als „sichtbar“ gemeldet, wenn es über Bluetooth angesprochen werden kann.

Das Vorhandensein des Bluetooth-Adapters, die Bindungsprozedur sowie ein Aufenthalt in Bluetooth-Funkreichweite gelten damit als notwendige Bedingungen, damit das „Device Backend“ das Mobiltelefon als verfügbar melden kann.

Im Fall eines verfügbaren Netzzugangsgeräts wird nach Netzzugangspunkten gesucht. Bezogen auf den Internetzugriff über Mobiltelefone bedeutet dies, dass mit Hilfe von AT-Kommandos die Signalstärke des Mobilfunknetzes ermittelt wird. Dabei fallen Signalstärkewerte an, welche vom Handoverentscheider des REACH-Clients zur Abschätzung der Eignung herangezogen werden können.

Nachteilig ist es hierbei, dass die Signalstärkewerte der Bluetooth-Funkstrecke nicht mit in den Auswahlprozess einbezogen werden. Allerdings kann diese Funkstrecke vereinfacht als „fest“ angesehen werden, da ein mobiler Teilnehmer sein Mobiltelefon zusammen mit seinem Laptop transportieren wird, sodass die Entfernung zwischen beiden Geräten als unkritisch angesehen werden kann. Die Signalstärkewerte der Bluetooth-Funkstrecke werden zwar vom „Device Backend“ ermittelt, jedoch nicht an den REACH-Client kommuniziert. Es findet lediglich eine interne Auswertung statt, welche sich im Verfügbarkeitsstatus niederschlägt. Sollte also die Funkverbindung zwischen mobilem Endgerät und Mobiltelefon schlechter werden, wird ab einem gewissen Punkt das Mobiltelefon nicht länger als verfügbar angezeigt.

Diese Variante war von den vorgestellten Möglichkeiten die Schlüssigste und wurde daher für die Implementierung ausgewählt.

Detektion des Netzzugangsgeräts

Als „das Netzzugangsgerät“ gilt damit das Mobiltelefon. Zusätzliche Komponenten, wie der Bluetooth-Adapter am mobilen Endgerät, welche für die Anbindung des Mobiltelefons an das mobile Endgerät erforderlich sind, werden nicht länger unterschieden. Deren Vorhandensein wird lediglich als notwendige Bedingung für die „Sichtbarkeit“ des Mobiltelefons gewertet.

Im Folgenden wird diskutiert, wie ein Mobiltelefon durch das „Device Backend“ erkannt wird. Im Vorfeld müssen diverse Parameter ermittelt und die folgenden Vorbereitungsschritte durchgeführt worden sein:

- Die Bluetooth-Geräteadresse des Mobiltelefons muss bekannt sein, denn sie muss dem „Device Backend“ als Detektionsparameter übergeben werden.

- Es muss bereits eine „Bindung“ zwischen Mobiltelefon und mobilem Endgerät stattgefunden haben. Bei dieser Bluetooth-spezifischen Prozedur werden beide Geräte miteinander bekannt gemacht und es findet ein Schlüsselaustausch statt.

- Auf dem mobilen Endgerät muss das „serielle Profil“ bezüglich Bluetooth konfiguriert worden sein. Dies geschieht unter GNU/Linux mit Hilfe der Konfigurationsdatei `/etc/bluetooth/rfcomm.conf`. Diese Datei gehört zum Bluez-Softwarepaket und ist für die Zuordnung von Bluetooth-Adaptern zu Gerätedateien im `/dev`-Verzeichnis zuständig. Bei Bluez lauten die Namen dieser Gerätedateien zu den seriellen Schnittstellen `/dev/rfcommX`, mit `X` als natürliche Zahl.

Sind diese Vorbedingungen erfüllt, kann die Suche nach dem Mobiltelefon durchgeführt werden. Damit diese erfolgreich sein kann, müssen der Bluetooth-Adapter am mobilen Endgerät verfügbar sein (angeschlossen) und sich das Mobiltelefon mit aktiviertem Bluetooth-Transceiver in Funkreichweite aufhalten. Dies kann mit Hilfe des `hcitool`-Kommandos überprüft werden. Es ist Bestandteil des Bluez-Programmpakets und wird zur Parametrierung des „Host Controller Interfaces“ (HCI) verwendet. Das HCI stellt eine standardisierte Schnittstelle zur Basisband-Signalaufbereitung eines Bluetooth-Transceivers dar.

Der hierzu verwendete Mechanismus ist in [BTPS07] dokumentiert. Dort wird beschrieben, wie sich eine Arbeitsstation automatisch sperren lässt, wenn sich der Benutzer von ihr entfernt. Hierzu wird die Bluetooth-Signalstärke des Mobiltelefons des Benutzers ausgewertet.

```
neelix ~ # /usr/bin/hcitool -i hci0 cc 00:01:E3:1F:AE:EE
Invalid device: No such device
neelix ~ # /usr/bin/hcitool -i hci0 cc 00:01:E3:1F:AE:EE
Can't create connection: Input/output error
```

Abbildung B.25: Der erste Bluetooth-Verbindungsversuch ist fehlgeschlagen, weil kein Bluetooth-Adapter verfügbar war. Der zweite Versuch schlug fehl, weil das Mobiltelefon mit der genannten Bluetooth-Adresse nicht erreicht werden konnte.

In Abbildung B.25 wird die Anwendung des `hcitool`-Kommandos für den vorgestellten Einsatzzweck gezeigt. Es werden zwei „Connect“-Versuche (`cc`) durchgeführt, wobei jeweils die Bluetooth-Schnittstelle `hci0` sowie das Mobiltelefon mit der Bluetooth-Adresse `00:01:E3:1F:AE:EE` involviert werden. Beide dargestellten Versuche verliefen jedoch erfolglos. Im ersten Fall war der Bluetooth-Adapter nicht eingesteckt und im zweiten

Fall war er zwar verfügbar, aber das Mobiltelefon befand sich nicht in Funkreichweite. Dies ist an den Rückmeldungen erkennbar.

```
neelix ~ # /usr/bin/hcitool -i hci0 cc 00:01:E3:1F:AE:EE
neelix ~ # /usr/bin/hcitool -i hci0 rssi 00:01:E3:1F:AE:EE
RSSI return value: 0
neelix ~ # /usr/bin/hcitool -i hci0 dc 00:01:E3:1F:AE:EE
[...]
neelix ~ # /usr/bin/hcitool -i hci0 cc 00:01:E3:1F:AE:EE
neelix ~ # /usr/bin/hcitool -i hci0 rssi 00:01:E3:1F:AE:EE
RSSI return value: -16
neelix ~ # /usr/bin/hcitool -i hci0 dc 00:01:E3:1F:AE:EE
```

Abbildung B.26: War der Verbindungsaufbau erfolgreich, kann der RSSI-Wert ermittelt werden. Dieser liefert eine Aussage über die signalstärkebezogene Nutzbarkeit der Bluetooth-Funkstrecke.

Zwei erfolgreiche Verbindungsversuche werden in Abbildung B.26 gezeigt: die Verbindungswünsche wurden hierbei nicht mit einer Fehlermeldung quittiert. Anschließend wurde jeweils der „Received Signal Strength Indicator“ (RSSI) mit Hilfe des Parameters `rssi` ausgelesen. Das Ergebnis ist ein numerischer Wert, dessen Bedeutung im Anschluss diskutiert wird. Die Verbindung ist anschließend wieder zu trennen, was mit Hilfe des „Disconnect“-Parameters (`dc`) geschieht. Die gezeigten Kommandos werden zyklisch aufgerufen, beispielsweise alle fünf Sekunden.

Der RSSI-Wert ermöglicht eine Aussage über die Nutzbarkeit der Bluetooth-Funkstrecke. Die eigentliche Bedeutung dieses Werts ist jedoch komplex: Bluetooth stellt ein System mit geregelter Sendeleistung dar, wobei versucht wird, die Signalstärke am entsprechenden Empfänger in einem idealen Bereich zu halten. Dieser als „Golden Receive Power Range“ bezeichnete Bereich sollte weder unter-, noch überschritten werden, damit das Empfangssignal ordnungsgemäß dekodiert werden kann [BTGR]. Der als RSSI gelieferte Wert drückt hierbei den „Abstand“ der aktuellen Signalstärke zur „Golden Receive Power Range“ aus [Blue99]. Idealerweise ist der Wert also Null, was dank der Leistungsregelung bis zu einer bestimmten maximalen Entfernung zwischen beiden Bluetooth-Transceivern eingehalten wird. Ein Wert von Null signalisiert also eine bedenkenlos benutzbare Bluetooth-Funkstrecke.

In Abbildung B.26 ist noch eine zweite RSSI-Ermittlung dargestellt, wobei ein Wert von `-16` vorlag. Dieser Wert kam zu Stande, als der Autor sein Mobiltelefon in eine Schreibtischschublade gelegt hatte. Da der RSSI-Wert hier ungleich Null war, sollte das Mobiltelefon in diesem Fall nicht für einen Internetzugriff herangezogen werden. Das „Device Backend“ würde es als „nicht verfügbar“ melden.

Detektion des Netzzugangspunkts

Besteht bezüglich Bluetooth eine Zugriffsmöglichkeit auf das Mobiltelefon, sollte das Betriebssystem das serielle Profil bereits automatisch aktiviert haben. Damit besteht die Möglichkeit, mit Hilfe der über Bluetooth angebotenen seriellen Schnittstelle auf das GSM-Modem des Mobiltelefons zuzugreifen.

```
Terminal ready
AT+CSQ
+CSQ: 19,99

OK

Thanks for using picocom
```

Abbildung B.27: Die Signalstärke des Mobilfunknetzes kann mit Hilfe des AT-Kommandos `AT+CSQ` ausgelesen werden.

In den hier gezeigten Beispielen wurde die serielle Schnittstelle durchgängig über die Gerätedatei `/dev/rfcomm0` zugänglich gemacht. Auf diese Schnittstelle kann beispielsweise mit einem Terminalprogramm zugegriffen werden, was in Abbildung B.27 demonstriert wird. Der Zugriff auf das Mobiltelefon geschieht mit Hilfe von AT-Kommandos. Ein spezielles AT-Kommando ist `AT+CSQ`, welches die aktuell anliegende Signalstärke des Mobilfunknetzes liefert. Der erste Rückgabewert (hier: `19`) beschreibt die eigentliche Signalstärke, welche gemäß der in [Hütt] aufgeführten Tabelle in dBm umgerechnet werden kann. Der zweite Wert (hier: `99`) benennt die Qualität des Empfangssignals. Dieser Wert liegt jedoch nur dann vor, wenn auch eine Verbindung besteht. Ist das Telefon im Ruhezustand, wie es in diesem Beispiel der Fall war, lautet der Wert `99`.

Je größer der erste Wert ist, desto stärker ist das Signal. Dabei sind Werte zwischen 0 und 31 definiert, wobei nicht jedes Mobiltelefon alle 32 Stufen unterscheiden kann. Man muss sich jedoch für einen Schwellwert entscheiden, unterhalb dessen „das Netz“ als nicht mehr nutzbar gilt. Ist der zyklisch ermittelte Wert ungleich `99`, gilt der Netzzugangspunkt als „sichtbar“. Die eigentliche Bewertung dieses Werts erfolgt jedoch erst durch den Handoverentscheider im REACH-Client. Dieser implementiert zudem eine Hysteresefunktion, um ein „Flattern“ des Nutzbarkeitszustands zu unterbinden. Für die Abrissschwelle wurde ein Wert von `10`, für die Eignungsschwelle ein Wert von `15` und für die Idealschwelle ein Wert von `20` empirisch festgelegt.

Einwahl: Aktivierung eines PDP-Kontexts

Der eigentliche Assoziationsprozess ist dem Einwahlvorgang eines Analogmodems ähnlich. So wird das `ATD`-Kommando zur Einwahl benutzt, allerdings mit einer besonderen Rufnummer. Diese Rufnummer repräsentiert keine real existierende Rufnummer, sondern referenziert einen „Packet Data Protocol“-Kontext (PDP-Kontext), welcher vorab definiert sein muss. Win solcher wird mit Hilfe des `AT+CGDCONT`-Kommandos hinterlegt und kann anschließend mittels des `ATD*99***1#`-Kommandos aktiviert werden.

```
Terminal ready
AT+CGDCONT=1,ip,internet.t-d1.de
OK
ATD*99***1#
CONNECT
~ÿ}#À!}!}#} }9}"}&} }*} } }'}"}(}"}%}
Thanks for using picocom
```

Abbildung B.28: Ein PDP-Kontext kann zu Testzwecken direkt im Terminalprogramm aktiviert werden. Die kryptischen Zeichen stellen den empfangenen PPP-Datenstrom dar.

Die Hinterlegung und Aktivierung eines PDP-Kontexts kann manuell mit Hilfe eines Terminalprogramms ausprobiert werden (siehe Abbildung B.28). Die Parameter des PDP-Kontexts müssen zum Mobilfunkprovider passen. Hier wird der Kontext `1` mit dem Vermittlungsschichtprotokoll `ip` verknüpft, und als „Gateway GPRS Support Node“ (GGSN) der Rechnername `internet.t-d1.de` spezifiziert. Der Kontextidentifikator `1` taucht im Wählkommando `ATD` erneut auf. Konnte der PDP-Kontext erfolgreich aktiviert werden, wird dies vom GSM-Modem mit einer `CONNECT`-Meldung angezeigt. Der erkennbare Zeichensalat mit den vielen geschweiften Klammern ist bereits der PPP-Datenstrom, welcher hier allerdings nur ausgegeben wird, da er vom Terminalprogramm nicht interpretiert werden kann.

Um den PPP-Datenstrom interpretieren zu können, bedarf es des PPP-Dämons `pppd`. Dieser übernimmt die Kontrolle über die serielle Schnittstelle, führt die in Abbildung B.28 dargestellte Einwahl durch und stellt anschließend eine PPP-Instanz dar, welche den PPP-Protokolldatenstrom interpretiert.

Für die eigentliche „Einwahl“ muss ein „Chat-Skript“ erstellt werden (siehe Abbildung B.29). Dieses ist für die „Einwahlprozedur“ mit Hilfe der beschriebenen AT-Kommandos zuständig. Das `pppd`-Kommando bedient sich des externen `chat`-Kommandos, um den PDP-Kontext zu aktivieren. Im Erfolgsfall steht an der seriellen Schnittstelle ein PPP-Datenstrom bereit, welcher vom `pppd`-Kommando interpretiert werden kann.

```
ABORT   'BUSY'
ABORT   'NO CARRIER'
ABORT   'ERROR'
''      AT
OK      AT+CGDCONT=1,ip,internet.t-d1.de
OK      ATD*99***1#
```

Abbildung B.29: Die Hinterlegung und Aktivierung des PDP-Kontexts wird durch ein solches „Chat-Skript“ gesteuert.

```
neelix ~ # /usr/sbin/pppd nodetach /dev/rfcomm0 115200 \
  usepeerdns nodefaultroute noauth connect \
  '/usr/sbin/chat -v -f ./chat-ppp'
Serial connection established.
Using interface ppp0
Connect: ppp0 <--> /dev/rfcomm0
local  IP address 80.187.63.48
remote IP address 192.168.254.254
primary   DNS address 10.74.83.22
secondary DNS address 193.254.160.1
^CTerminating on signal 2
Connect time 0.2 minutes.
Sent 0 bytes, received 0 bytes.
Connection terminated.
```

Abbildung B.30: Das `pppd`-Kommando kann zu Testzwecken manuell in der Kommandozeile gestartet werden. Die gezeigten Parameter entsprechen denen, die auch das „Device Backend“ verwendet.

Das `pppd`-Kommando wird durch das „Device Backend“ gestartet. In Abbildung B.30 wird ein vergleichbarer manueller Start des `pppd`-Kommandos über die Kommandozeile gezeigt. Der Parameter `nodetach` sorgt dafür, dass das Programm „im Vordergrund“ verbleibt, sich also nicht „dämonisiert“. Als zweiter Parameter folgt der Name der Gerätedatei der seriellen Schnittstelle, in diesem Fall `/dev/rfcomm0`. Die Schnittstellengeschwindigkeit wird hier mit `115200` Baud angegeben. Mit `usepeerdns` werden bei Erfolg bis zu zwei IP-Adressen zu DNS-Servern erfragt. Mit `nodefaultroute` wird verhindert, dass eine „Default Route“ vom `pppd`-Kommando gesetzt wird. Da bei GPRS keine weitere Authentifizierung über PPP stattfindet, muss dies per `noauth`-Parameter angezeigt werden. Schlussendlich wird mit Hilfe des `connect`-Parameters das `chat`-Kommando benannt, welches wiederum den Dateinamen eines „Chat-Skripts“ benötigt. Das „Chat-Skript“ muss bereits existieren.

Mit dem in Abbildung B.30 dargestellten Kommando steht im Erfolgsfall eine vollwertige Internetverbindung bereit. Lediglich bezüglich des „Default Gateways“ wurde kein Eintrag in der Routingtabelle hinterlegt, was aber beabsichtigt ist. Das `pppd`-Kommando wurde hier anschließend manuell mit Hilfe der Tastenkombination `STRG-C` abgebrochen.

Die folgenden Ausgaben des `pppd` sind für das „Device Backend“ relevant: Die Zeile mit der Zeichenkette „`local IP address`“ benennt die lokal zugewiesene IP-Adresse, während die Zeichenkette „`remote IP address`“ auf die IP-Adresse des entfernten „Punkts“ hinweist. Letztere IP-Adresse ist als IP-Adresse des Gateways zu interpretieren. Die Netzmaske ist bei PPP mit „`255.255.255.255`“ anzunehmen. Wurden beide IP-Adressen gefunden, gilt „die Einwahl“ als abgeschlossen und die bestehende Assoziation zum Netzzugangspunkt wird dem REACH-Client mitgeteilt.

In der Ausgabe des `pppd`-Kommandos wird auf drei Fehlerfälle geachtet: Die Meldung „`Failed to open`“ lässt auf einen Verlust der seriellen Schnittstelle schließen. Die Zeichenkette „`Modem hangup`“ signalisiert einen netzseitigen Verlust des PDP-Kontexts. Die Meldung „`Terminating on signal`“, welche auch in Abbildung B.30 dargestellt ist, steht für einen angeforderten Abbruch. Diese Meldung wird provoziert, wenn das „Device Backend“ eine Verbindungstrennung wünscht und hierfür das `SIGINT`-Signal an den `pppd` sendet.

Vom „Device Backend“ benötigte Parameter

Die Parameter des hier vorgestellten „Device Backends“ lassen sich in Detektions- und Einwahlparameter gruppieren. Erstere sind für die Erkennung des Netzzugangsgeräts (das Mobiltelefon) zuständig. Für die Erkennung des Netzzugangspunkts gibt es keine dedizierten Parameter, da hierfür die Signalstärke bezüglich des Mobilfunknetzes ausgewertet wird. Für die Assoziation wiederum werden Parameter für das `pppd`-Kommando benötigt.

- Hardwarespezifische Parameter:
  - „`hci device`“: Der Gerätename der verwendeten Bluetooth-Schnittstelle, in diesem Beispiel durchgängig `hci0`.
  - „`rfcomm interface`“: Die Gerätedatei der seriellen Schnittstelle zum Mobiltelefon, welche über Bluetooth bereit gestellt wird. In diesem Beispiel wird von `/dev/rfcomm0` ausgegangen.
  - „`cp bt address`“: Die Bluetooth-Adresse des Mobiltelefons („Cell phone“).
- Assoziationsparameter:

– „`chatscript`“: Der Dateiname des „Chat-Skripts“, welches vorab durch den Nutzer erstellt worden sein muss. Es enthält Mobilfunkanbieterspezifische AT-Kommandos zur Hinterlegung und Aktivierung eines PDP-Kontexts.

B.3 Bereitstellung von Systemverwaltungsrechten

Eine weitere Annehmlichkeit bei der Nutzung von „Device Backends“ rührt aus der Tatsache, dass für die Beeinflussung von Netzzugangsgeräten Systemverwaltungsrechte („`root`-Rechte“) erforderlich sind. Da es sich bei den „Device Backends“ um eigenständige Programme handelt, können diese gezielt mit gehobenen Privilegien gestartet werden. Der REACH-Client selbst benötigt keine Systemverwaltungsrechte und kann daher aus Sicherheitsgründen in einem „normalen“ Nutzerkontext gestartet werden. Unter GNU/Linux bietet sich dazu die Verwendung der Software SUDO („substitute user do“ [Wiki09f]) an. Sie erlaubt es einem Nutzer, bestimmte freigegebene Programme selektiv mit erweiterten Privilegien zu starten. SUDO muss dafür im Vorfeld so parametriert werden, dass die „Device Backends“ mit Systemverwaltungsrechten gestartet werden. Die dazu relevanten Zeilen aus der Konfigurationsdatei `/etc/sudoers` zeigt Abbildung B.31. Hierbei werden drei „Device Backends“ referenziert, welche für die Netzzugangstechnologien „GPRS über mit per Bluetooth angebundenem Mobiltelefon“, „WLAN“ sowie „Ethernet“ verantwortlich sind. Das „Routing Backend“, welches im nächsten Kapitel genauer beleuchtet wird, muss ebenfalls über SUDO mit Systemverwaltungsrechten gestartet werden.

```
reach ALL = NOPASSWD: /usr/local/bin/reachplugin-device-bluetoothgprs
reach ALL = NOPASSWD: /usr/local/bin/reachplugin-device-wlan
reach ALL = NOPASSWD: /usr/local/bin/reachplugin-device-ethernet
reach ALL = NOPASSWD: /usr/local/bin/reachplugin-route
```

Abbildung B.31: In der Konfigurationsdatei `/etc/sudoers` werden die Programmnamen hinterlegt, welche bestimmte Nutzer gezielt mit Systemverwaltungsrechten starten dürfen. In diesem Beispiel wird es dem Benutzer `reach` gestattet, drei „Device Backends“ sowie ein „Routing Backend“ durch SUDO mit `root`-Rechten zu starten. Dabei wird keine erneute Passworteingabe erforderlich (Parameter `NOPASSWD`).

C Implementierungsdetails zum „Routing Backend“

Dieses Kapitel beginnt mit einer Beschreibung des Kommunikationsprotokolls zwischen REACH-Client und dem „Routing Backend“. Anschließend wird auf die Mechanismen eingegangen, welche notwendig sind, um auf einem GNU/Linux-System die Wegewahl zu beeinflussen. Diese Mechanismen wurden für das erstellte „Routing Backend“ adaptiert.

C.1 Kommunikation zwischen REACH-Client und dem „Routing Backend“

In diesem Abschnitt wird ein beispielhafter Protokollablauf zwischen REACH-Client und „Routing Backend“ diskutiert. Dazu wird mit einer Vorstellung der möglichen Nachrichtentypen begonnen.

C.1.1 Protokollfunktionen

Die Kommunikation erfolgt mittels Nachrichten im XML-Format, welche bidirektional über die Standardein- und ausgabe des „Routing Backends“ ausgetauscht werden. Das „Routing Backend“ kann somit zu Testzwecken auch manuell gestartet werden. Der Aufbau der Nachrichten orientiert sich dabei an demselben Schema, welches bereits für die Kommunikation mit den „Device Backends“ entwickelt worden ist (siehe Kapitel B).

C.1.1.1 Nachrichtentypen in Richtung des „Routing Backends“

Dem REACH-Client stehen die folgenden betriebssystemunabhängigen Nachrichtentypen zur Verfügung, welche an das „Routing Backend“ verschickt werden können:

- `<AddInterface>`: Dem „Routing Backend“ wird ein neu hinzugekommenes Netzzugangsgerät bekannt gemacht. Dies passiert immer dann, wenn ein „Device Backend“

gestartet worden ist, und damit ein neues Interface in den Wegewahlmechanismus eingebunden werden soll.

- `<DelInterface>`: Sobald ein „Device Backend“ für ein bestimmtes Netzzugangsgerät nicht länger verantwortlich ist, wird das betroffene Interface mit Hilfe einer solchen Nachricht aus der Wegewahlbetrachtung entfernt.

- `<Enable>`: Konnte ein Netzzugangsgerät erfolgreich assoziiert werden, wird dem „Routing Backend“ hiermit mitgeteilt, dass alle Routen bezüglich dieses Netzzugangsgeräts gesetzt werden sollen.

- `<Disable>`: Eine Assoziation wurde abgebaut, und alle diesbezüglichen Routen sollen wieder aus den Routingtabellen entfernt werden.

- `<AddAddress>`: Für die zieladressbasierte Wegewahl müssen die IP-Adressen der REACH-Proxyserver mit Netzzugangsgeräten des mobilen Endgeräts verknüpft werden. Die Wegewahl erfolgt dann anhand von Zieladressen.

- `<DelAddress>`: Eine per `<AddAddress>`-Nachricht hinterlegte Adressbindung kann hiermit wieder gelöst werden.

- `<ShutDown>`: Soll das „Routing Backend“ beendet werden, sendet ihm der REACH-Client diese Nachricht. Es wird die Routingtabellen wieder in einen geordneten Zustand überführen und sich anschließend beenden.

C.1.1.2 Nachrichtentypen in Richtung des REACH-Clients

Das „Routing Backend“ versendet nur dann Nachrichten, wenn es entweder „frisch“ gestartet worden ist oder wenn der REACH-Client auf eine Bestätigung wartet. Solche Bestätigungsnachrichten dienen dazu, „Laufzeitprobleme“ zu verhindern, denn der REACH-Client kann erst nach Erhalt einer Bestätigungsnachricht von einer abgeschlossenen Manipulation der Routingtabellen ausgehen.

- `<Attach>`: Sofort nachdem das „Routing Backend“ gestartet worden ist, meldet es sich mit dieser Nachricht am REACH-Client an. Diese Nachricht signalisiert zudem, ob sich das Backend für die quell- oder zieladressbasierte Wegewahl entschieden hat.

- `<Enabled>`: Diese Nachricht ist eine Bestätigung zu einer vorausgegangenen `Enable`-Nachricht. Sie zeigt an, dass alle Routen bezüglich eines Netzzugangsgeräts gesetzt worden sind und dass der Netzzugang ab sofort benutzt werden kann.

- `<Disabled>`: Hiermit wird der Erhalt einer `Disable`-Nachricht bestätigt. Sie wird erst versendet, nachdem die Routingtabellen bezüglich einer abgebauten Assoziation gesäubert worden sind.

- `<AddressAdded>`: Eine per `<AddAddress>`-Nachricht angeforderte Adressbindung wurde von „Routing Backend“ vermerkt und wurde je nach Zustand des Netzzugangsgeräts auch bereits in die Routingtabellen übertragen.

C.1.2 Beispielhafter Ablauf

Im Folgenden wird ein detaillierter Einblick in das Kommunikationsprotokoll zwischen REACH-Client und „Routing Backend“ geboten. Anhand eines beispielhaften Protokollablaufs werden zudem die Aufgaben des „Routing Backends“ herausgearbeitet. Das für GNU/Linux erstellte „Routing Backend“ unterstützt alle drei der in den Abschnitten 5.5.5 und 6.4.1 vorgestellten Wegewahlschemata. Diese umfassen:

- Quelladressbasierte Wegewahl sowie

- zieladressbasierte Wegewahl, diese entweder
  - mit der Möglichkeit einer Interface-Wahl pro Socket oder
  - ohne diese Möglichkeit.

Da der REACH-Client als ein betriebssystemunabhängiges Programm implementiert worden ist, werden die Auswahl des Wegewahlschemas sowie die eigentliche Modifikationen der Routingtabellen[1] durch das externe „Routing Backend“ übernommen. Dieses wird vom REACH-Client gestartet und mit einem definierten Protokoll angesprochen, sodass der REACH-Client über eine betriebssystemunabhängige Sichtweise auf die Wegewahl verfügt.

REACH unterscheidet die zwei „Hauptbetriebsmodi“ quelladress- und zieladressbasierte Wegewahl. Die Auswahl erfolgt alleinig durch das „Routing Backend“, da nur dieses die Fähigkeiten des vorliegenden Betriebssystems erkennen und verfügbare Schemata ermitteln kann. Je nach getroffener Wahl unterscheidet sich allerdings der Umfang der auszutauschenden Nachrichten. Daher wird bei der folgenden Vorstellung jeweils genannt, für welchen Modus die jeweiligen Nachrichten relevant sind.

Das „Routing Backend“ erledigt unsichtbar im Hintergrund noch eine zweite Aufgabe. Es ist verantwortlich dafür, dass die Namensauflösung funktioniert, was durch dedizierte

[1] Der Begriff „Routingtabelle“ wird bewusst im Plural verwendet, da bei der quelladressbasierten Wegewahl in der Tat mehrere Routingtabellen zum Einsatz kommen (Stichwort: „Policy Routing“).

Einträge in den Routingtabellen sichergestellt werden muss. Ist nämlich ein zu kontaktierender DNS-Server nicht direkter Bestandteil des Subnetzes des zugeordneten Interfaces, können die DNS-Anfragen nicht direkt zugestellt werden, und die Namensauflösung würde fehlschlagen. Da REACH nicht mit „Default Routen“ arbeitet, müssen in diesen Fällen zu jedem DNS-Server explizit Routen gesetzt werden (siehe Kapitel 6.4.2). An dieser Stelle ist es wichtig zu wissen, dass das „Routing Backend“ diese Aufgabe zwar erfüllt, dies aber nicht anhand der Protokollnachrichten erkennbar ist.

C.1.2.1 Initialisierungsphase

Wenn das „Routing Backend“ gestartet wird, führt es eine interne Initialisierung durch. Es prüft dabei, welche Wegewahl-Schemata vom vorliegenden Betriebssystem unterstützt werden, und welche Systemkommandos für die Manipulation der Routingtabellen zur Verfügung stehen. Konnte die Initialisierungsphase erfolgreich durchlaufen werden, meldet sich das „Routing Backend“ mit der in Abbildung C.1 dargestellten `<Attach>`-Nachricht am REACH-Client an. Die Nachricht benennt zudem den vom „Routing Backend“ gewählten Hauptbetriebsmodus, was in diesem Beispiel die quelladressbasierte Wegewahl ist (`routing="sourcebased"`). Das gewählte Wegewahlschema kann nicht geändert werden und gilt für die gesamte Laufzeit.

```
<RoutingBackend>
  <Attach routing="sourcebased" />
</RoutingBackend>
```

Abbildung C.1: Das „Routing Backend“ meldet sich nach erfolgreicher Initialisierung am REACH-Client an. Dabei wird dem REACH-Client mitgeteilt, ob eine quelladress- oder zieladressbasierte Wegewahl durchgeführt wird.

C.1.2.2 Bekanntmachung eines Netzzugangsgeräts

Die Netzzugangsgeräte (Interfaces) eines mobilen Endgeräts werden durch „Device Backends“ verwaltet, was bereits in Kapitel 6.3.3 umfassend erläutert worden ist. Jedes dieser „Device Backends“ ist für ein bestimmtes Interface des mobilen Endgeräts zuständig und wird durchgängig durch einen Identifikator (`label`) referenziert.

Wurde ein „Device Backend“ gestartet und konnte es mit dem Namen eines Interfaces parametriert werden, wird vom REACH-Client eine `<AddInterface>`-Nachricht an das „Routing Backend“ übergeben (Abbildung C.2). Sie benennt den Identifikator `label` sowie den Namen des Netzzugangsgeräts `interface`. Spätere Nachrichten an das „Routing

Backend“ referenzieren das Netzzugangsgerät nur noch anhand seines `label`s, welches dank der `<AddInterface>`-Nachricht jedoch auch später noch in den Interface-Namen überführt werden kann.

```
<ReachClient>
  <AddInterface label="ZD1211 USB" interface="wlan0" />
</ReachClient>
```

Abbildung C.2: Jedes von REACH verwaltete Netzzugangsgerät muss dem „Routing Backend“ bekannt gemacht werden. Dadurch wird eine spätere Abbildung zwischen `label`- und `interface`-Parameter ermöglicht.

Der Identifikator `label` stellt dieselbe Zeichenkette dar, welche in der Konfigurationsdatei `devices.xml` zur Benennung des „Device Entry“-Objekts angegeben worden ist (siehe „Providerkonzept“ in Abschnitt 6.3.2).

C.1.2.3 Freigabe eines Netzzugangsgeräts

Analog zu der in Abbildung C.2 dargestellten `<AddInterface>`-Nachricht gibt es noch die `<DelInterface>`-Nachricht, welche immer dann an das „Routing Backend“ gesendet wird, wenn ein „Device Backend“ ein Netzzugangsgerät nicht länger kontrolliert. Das ist beispielsweise dann der Fall, wenn ein „Device Backend“ beendet oder wenn diesem ein anderes Netzzugangsgerät zugewiesen wird. Im letzteren Fall wird es im Anschluss eine erneute `<AddInterface>`-Nachricht geben, allerdings mit dem neuen Interface-Namen als Inhalt.

```
<ReachClient>
  <DelInterface label="ZD1211 USB" />
</ReachClient>
```

Abbildung C.3: Wird ein Netzzugangsgerät nicht länger durch REACH verwaltet, muss das „Routing Backend“ alle diesbezüglichen Bindungen vergessen und die Routingtabellen bereinigen.

Die in Abbildung C.3 gezeigte `<DelInterface>`-Nachricht benennt lediglich das `label` eines Netzzugangsgeräts, was jedoch dank der vorausgegangenen `<AddInterface>`-Nachricht ausreichend ist.

C.1.2.4 Bekanntgabe einer neuen Assoziation

Konnte ein Netzzugangsgerät erfolgreich mit einem Netzzugangspunkt assoziiert werden, kann es vom REACH-Client zum Datentransfer herangezogen werden. Vorher müssen

allerdings die Routingtabellen angepasst werden. Die dem Netzzugangsgerät zugewiesene IP-Adresse, die geltende IP-Netzmaske sowie die IP-Adresse des Gateways werden hierzu vom „Device Backend“ ermittelt. Der REACH-Client übergibt die ermittelten Adressen anschließend mit der in Abbildung C.4 dargestellten `<Enable>`-Nachricht.

```
<ReachClient>
  <Enable label="ZD1211 USB"
          interfaceIpAddress="192.168.1.32"
          interfaceNetmask="255.255.255.0"
          bindIpAddress="10.254.254.11"
          gatewayIpAddress="192.168.1.1" />
</ReachClient>
```

Abbildung C.4: Konnte der REACH-Client ein Netzzugangsgerät mit einem Netzzugangspunkt assoziieren, muss dies dem „Routing Backend“ mitgeteilt werden. Dieses hat nun die Aufgabe, die entsprechenden Einträge in den Routingtabellen zu hinterlegen.

Der Identifikator `label` referenziert das Netzzugangsgerät, welches ab sofort als *assoziiert* gilt, also „online“ ist. Neben den drei bereits vorgestellten Adressparametern ist noch der Parameter `bindIpAddress` erkennbar. Diese Adresse wird bei quelladressbasierter Wegewahl benutzt und stellt die von den betroffenen Sockets zu verwendende Absenderadresse dar. Wird ein Socket auf diese IP-Adresse gebunden, werden alle ausgehenden Daten dieses Sockets über das genannte Netzzugangsgerät versendet.

Dem „Routing Backend“ stehen damit im Fall von quelladressbasierter Wegewahl bereits alle Informationen zur Verfügung, welche benötigt werden, um die erforderlichen Einträge in den Routingtabellen zu hinterlegen.

```
<RoutingBackend>
  <Enabled label="ZD1211 USB"
           interfaceIpAddress="192.168.1.32"
           interfaceNetmask="255.255.255.0"
           bindIpAddress="10.254.254.11"
           gatewayIpAddress="192.168.1.1" />
</RoutingBackend>
```

Abbildung C.5: Erst wenn das „Routing Backend“ diese Bestätigungsnachricht an den REACH-Client versendet, ist sichergestellt, dass das Netzzugangsgerät ordnungsgemäß bezüglich der Wegewahl konfiguriert worden ist.

Nach Auswertung der `<Enable>`-Nachricht und der erfolgreichen Manipulation der Routingtabellen schickt das „Routing Backend“ eine wie in Abbildung C.5 dargestellte

<Enabled>-Nachricht an den REACH-Client. Dadurch weiß dieser, dass das Netzzugangsgerät auch bezüglich der Wegewahl parametriert worden ist. Der Netzzugang kann von nun an zur Datenübertragung herangezogen werden.

Im Fall von zieladressbasierter Wegewahl gilt ein kleiner Unterschied: der Inhalt des bindIpAddress-Parameters wird in diesem Fall vom „Routing Backend“ durch die dem Netzzugangsgerät zugewiesene IP-Adresse ersetzt. Statt wie in Abbildung C.5 dargestellt würde der Parameter dann nicht länger die IP-Adresse „10.254.254.11“, sondern stattdessen die IP-Adresse „192.168.1.32“ benennen.

C.1.2.5 Verlust einer Assoziation

Wenn der Handoverentscheider bezüglich einer aktiven Assoziation (Internetverbindung) einen Dissoziationswunsch geäußert hat oder wenn eine solche spontan abreißt, wird dies von zugehörigen „Device Backend“ bemerkt und signalisiert. Alle bezüglich des betroffenen Netzzugangsgeräts in den Routingtabellen hinterlegten Einträge werden nach diesem Ereignis nicht länger gültig sein und müssen entfernt werden. Hierzu muss das „Routing Backend“ über das Trennungsereignis informiert werden, was mit Hilfe einer <Disable>-Nachricht geschieht (siehe Abbildung C.6).

```
<ReachClient>
  <Disable label="ZD1211 USB" />
</ReachClient>
```

Abbildung C.6: Wurde eine Assoziation beendet, egal ob freiwillig oder auf Grund eines spontanen Abrisses, muss das „Routing Backend“ die Routingtabellen bezüglich des betroffenen Netzzugangsgeräts aufräumen.

Diese Nachricht drückt aus, dass zwischen dem Netzzugangsgerät mit label="ZD1211 USB" und einem (hier nicht genannten) Netzzugangspunkt keine Assoziation mehr besteht. Das „Routing Backend“ wird daraufhin alle Einträge, welche dieses Netzzugangsgerät involvieren, aus den Routingtabellen entfernen.

```
<RoutingBackend>
  <Disabled label="ZD1211 USB" />
</RoutingBackend>
```

Abbildung C.7: Mit einer solchen Nachricht wird dem REACH-Client die erfolgreiche Bereinigung der Routingtabellen signalisiert. Erst jetzt darf der REACH-Client das betroffene Netzzugangsgerät zu einem erneuten Assoziationsvorgang heranziehen.

Nachdem das „Routing Backend“ die betroffenen Einträge entfernt hat, antwortet es mit einer `<Disabled>`-Nachricht (siehe Abbildung C.7). Der REACH-Client weiß damit, dass zu dieser Assoziation keine „Spuren“ mehr in den Routingtabellen existieren, sodass zukünftige Assoziationskommandos (über das zugehörige „Device Backend“) nicht durch verwaiste Einträge in den Routingtabellen negativ beeinflusst werden können.

C.1.2.6 Verknüpfung einer Zieladresse mit einem Netzzugangsgerät

Im Betriebsmodus der quelladressbasierten Wegewahl reichen die bereits vorgestellten Nachrichten aus, um alle relevanten Informationen zwischen REACH-Client und „Routing Backend“ auszutauschen. Für die zieladressbasierte Wegewahl werden jedoch noch weitere Nachrichtentypen benötigt, da für die entsprechenden Einträge in der *zentralen* Routingtabelle[2] die Adressen der REACH-Proxyserver kommuniziert werden müssen.

Bei der zieladressbasierten Wegewahl wird eine Zieladresse (die IP-Adresse eines REACH-Proxyservers) mit einem Netzzugangsgerät verknüpft. Gewünscht wird eine deterministische Bindung der Kommunikationsbeziehungen an vorgegebene Netzzugangsgeräte des mobilen Endgeräts. Dazu werden Einträge in der zentralen Routingtabelle des mobilen Endgeräts abgelegt, damit die ausgehenden IP-Pakete den gewünschten Weg über das korrekte Netzzugangsgerät nehmen.

Der REACH-Client unterscheidet bei der zieladressbasierter Wegewahl zwei Betriebsmodi. Beim „einfacheren“ der beiden Verfahren wird die Wahl des Netzzugangsgeräts allein anhand der Zieladresse eines jeden IP-Pakets durchgeführt. Das bedeutet, pro zeitgleich zu nutzendem Netzzugangsgerät pro REACH-Proxyserver eine eigene, eindeutige IP-Adresse benötigt. Kann ein bestimmter REACH-Proxyserver beispielsweise nur über eine einzige IP-Adresse adressiert werden, ist auch nur eine einzige Kommunikationsbeziehung über *ein* Interface in der Routingtabelle spezifizierbar. In diesem Fall sind bezüglich dieses REACH-Proxyservers lediglich „harte Handover“ möglich.

Der „erweiterte“ Betriebsmodus entspricht dem vorgestellten Schema, führt jedoch noch eine „explizite Bindung“ der Sockets an das gewünschte Netzzugangsgerät durch. Der Vorteil dabei ist, dass dann identische Zieladressen eines REACH-Proxyservers auch bezüglich mehrerer ausgehender Wege verwendet werden können. Somit kann über mehrere Netzzugangsgeräte hinweg mit ein und demselben REACH-Proxyserver kommuniziert werden, auch wenn dieser nur über eine einzige öffentliche IP-Adresse verfügt. Dieses Schema ist damit aus Sicht von REACH ähnlich leistungsfähig wie die quelladressbasierte Wegewahl, jedoch werden für die erforderliche `SO_BINDDEVICE`-Socketoption erweiterte Privilegien benötigt. Ein „normaler“ Nutzer besitzt diese Privilegien nicht,

[2] Bei zieladressbasiertem Routing wird lediglich die zentrale Routingtabelle manipuliert.

sodass der „erweiterte“ Modus nur dann zur Verfügung steht, wenn der REACH-Client als Nutzer `root` gestartet worden ist. Der REACH-Client wurde so implementiert, dass er das Vorhandensein der entsprechenden „`CAP_NET_RAW`-Capability“ erkennen kann und dann bei zieladressbasierter Wegewahl automatisch auf den „erweiterten“ Betriebsmodus zurückgreifen kann. Die vorausgehende Frage, ob eine zieladressbasierte Wegewahl durchgeführt wird, beantwortet dem REACH-Client die bereits in Abbildung C.1 dargestellte `<Attach>`-Nachricht.

Der REACH-Client ist bei zieladressbasierter Wegewahl in der Pflicht, die entsprechenden Zieladressen an das „Routing Backend“ zu übergeben. Das geschieht mit Hilfe von `<AddAddress>`-Nachrichten (siehe Abbildung C.8).

```
<ReachClient>
  <AddAddress label="ZD1211 USB"
              ipAddress="141.24.92.184" />
</ReachClient>
```

Abbildung C.8: Bei zieladressbasierter Wegewahl wird dem „Routing Backend“ mitgeteilt, welche Zieladressen mit welchen Netzzugangsgeräten verknüpft werden sollen.

Diese Nachricht benennt den Namen eines Interfaces sowie die IP-Adresse eines REACH-Proxyservers. Damit ist ein Eintrag für die Routingtabelle formulierbar. IP-Pakete mit der genannten IP-Adresse als Ziel werden anschließend über das durch den `label`-Parameter referenzierte Netzzugangsgerät geleitet.

Jedes Netzzugangsgerät kann mit einer Vielzahl an IP-Adressen zu unterschiedlichen REACH-Proxyservern verknüpft werden. Besteht beispielsweise Kontakt zu zehn REACH-Proxyservern gleichzeitig, müssen auch zehn IP-Adressen mit dem aktuell zu nutzenden Netzzugangsgerät (dem „Hauptlink“) verknüpft werden. Bei der „einfacheren“ Variante der zieladressbasierten Wegewahl darf jedoch keine IP-Adresse gleichzeitig in mehreren Regeln vertreten sein, da es sonst zu Mehrdeutigkeiten käme. Beim „erweiterten“ Betriebsmodus gilt diese Einschränkung nicht. Die Einträge in der Routingtabelle werden dann lediglich dazu benötigt, dass die IP-Pakete *überhaupt eine Route* zum REACH-Proxyserver vorfinden. Diese kann bewusst mehrdeutig sein, da die Wegewahl bereits durch die `SO_BINDDEVICE`-Socketoption vorweggenommen worden ist.

Nachdem das „Routing Backend“ die `<AddAddress>`-Nachricht ausgewertet und gegebenenfalls eine Modifikation der Routingtabelle durchgeführt hat, antwortet es dem REACH-Client mit einer `<AddressAdded>`-Nachricht (siehe Abbildung C.9). Der REACH-Client wird damit in diesem Beispiel darüber informiert, dass über das Interface mit dem Label `"ZD1211 USB"` eine Route zu der IP-Adresse `"141.24.92.184"` gesetzt

```
<RoutingBackend>
  <AddressAdded label="ZD1211 USB"
                interface="wlan0"
                ipAddress="141.24.92.184" />
</RoutingBackend>
```

Abbildung C.9: Eine <AddAddress>-Nachricht wird mit einer solchen <AddressAdded>-Nachricht bestätigt. Der REACH-Client ist sich nach Erhalt sicher, dass die diesbezügliche Route hinterlegt worden ist.

worden ist. Steht dem REACH-Client die SO_BINDDEVICE-Socketoption zur Verfügung, weiß er dank des Parameters interface auch, an welches Netzzugangsgerät die entsprechenden Sockets gebunden werden müssen (in diesem Fall an das Interface "wlan0").

<AddAddress>- und <AddressAdded>-Nachrichten werden nur bei zieladressbasierter Wegewahl verwendet. Das gilt auch für die <DelAddress>-Nachrichten, welche im nächsten Abschnitt vorgestellt werden.

C.1.2.7 Aufhebung einer Adressbindung

Um eine per <AddAddress>-Nachricht etablierte Bindung zwischen einem Netzzugangsgerät und einer IP-Adresse wieder aufzuheben, bedient sich der REACH-Client des <DelAddress>-Nachrichtentyps (dargestellt in Abbildung C.10).

```
<ReachClient>
  <DelAddress label="ZD1211 USB"
              ipAddress="141.24.92.184" />
</ReachClient>
```

Abbildung C.10: Hinterlegte Adressbindungen können mit Hilfe von <DelAddress>-Nachrichten aufgehoben werden.

Eine solche Nachricht fordert das „Routing Backend“ dazu auf, eine vorhandene Bindung zwischen einem Netzzugangsgerät (label-Parameter) und einer IP-Adresse eines REACH-Proxyservers zu „vergessen“. Dabei werden auch alle diesbezüglichen Einträge aus der Routingtabelle entfernt.

Zu diesem Nachrichtentyp wird keine entsprechende Bestätigungsnachricht benötigt. Ab dem Moment, wenn der REACH-Client die <DelAddress>-Nachricht absetzt, wird er die betroffene Bindung nicht mehr in Anspruch nehmen. Daher besteht kein Bedarf an einer Rückmeldung, welche signalisieren würde, dass der Eintrag wirklich vergessen wurde. Denn möchte der REACH-Client selbige Bindung erneut hinterlegen, muss

er hierfür wiederum eine `<AddAddress>`-Nachricht absetzen *und* auf die entsprechende `<AddressAdded>`-Bestätigung warten. Durch die serielle Abarbeitung der Nachrichten innerhalb des „Routing Backends“ wird sichergestellt, dass der REACH-Client immer über eine konsistente Sichtweise auf die ordnungsgemäß in der Routingtabelle hinterlegten Adressbindungen verfügt.

Es gibt mehrere Fälle, in denen die Aufhebung einer Adressbindung durch eine solche `<DelNetwork>`-Nachricht erforderlich ist:

- **Nicht länger benötigte Adressbindung**: Besteht zu einem REACH-Proxyserver keine Kommunikationsbeziehung mehr, werden die zugehörigen Einträge aus der Routingtabelle entfernt.

- **Wechsel einer Zone**: REACH implementiert ein ortsabhängiges Adressierungsschema, welches je nach Aufenthaltsort des mobilen Endgeräts einen gewünschten REACH-Proxyserver über abweichende IP-Adressen kontaktieren kann. Ein solches Schema ist elementar wichtig für REACH, was in Abschnitt 7.2.2.1 nachgelesen werden kann.

  Der Grund hierfür ist, dass sich zum Beispiel bei einem „horizontalen Handover“ das „Einzugsgebiet“ eines REACH-Proxyservers ändern kann, also eine neue Zone gültig wird. In diesem Fall ändert sich die IP-Adresse, mit welcher der REACH-Proxyserver über *dieses* Netzzugangsgerät angesprochen werden muss. Dazu erfolgt eine Aufhebung der bestehenden Bindung (mit Hilfe einer `<DelNetwork>`-Nachricht) und anschließend eine neue Bindung zwischen demselben Netzzugangsgerät und der von nun an zu benutzenden IP-Adresse (anhand einer `<AddNetwork>`-Nachricht samt Bestätigung).

- **Wiederverwendung knapper Zieladressen**: Speziell bei der zieladressbasierten Wegewahl *ohne* der Möglichkeit der expliziten Bindung von Sockets an ein Netzzugangsgerät gelten die verfügbaren IP-Adressen eines jeden REACH-Proxyservers als *knapp*.

  Ist ein bestimmter REACH-Proxyserver nur über eine einzige IP-Adresse erreichbar, muss im Rahmen eines „vertikalen Handovers“ die bestehende Adressbindung über das „alte“ Netzzugangsgerät aufgehoben werden, damit die dabei „freigewordene“ IP-Adresse für eine Bindung über das „neue“ Netzzugangsgerät herangezogen werden kann. Es ist erkennbar, dass dabei nur „harte Handover“ möglich sind.

  Aus diesem Grund kann zu jedem REACH-Proxyserver eine Vielzahl an IP-Adressen pro Zone angegeben werden. Auf mobilen Endgeräten, welche lediglich die „einfache“ Variante der zieladressbasierten Wegewahl unterstützen, können dann auch

„weiche“ und „weichere Handover“ sowie Kanalbündelungsszenarien angeboten werden. Die dafür notwendige Adressverwaltung ist im REACH-Client implementiert, aber nur für letztgenanntes Wegewahlschema erforderlich.

C.1.2.8 Gültigkeitsdauer bei zieladressbasierter Wegewahl

Eine `<AddAddress>`-Nachricht wird vom „Routing Backend“ mit einer `<AddressAdded>`-Nachricht beantwortet, unabhängig davon, ob das referenzierte Netzzugangsgerät zu diesem Zeitpunkt mit einem Netzzugangspunkt assoziiert ist oder nicht. Es gilt, dass hinterlegte Adressbindungen *langlebiger Natur* sind, also auch mehrerer Assoziations- und Dissoziationsprozeduren überdauern. Eine Bindung kann nur durch eine `<DelAddress>`-Nachricht wieder aufgelöst werden.

Adressbindungen werden vom „Routing Backend“ entgegen genommen und intern vermerkt. Sie werden jedoch nur bei einer momentan bestehenden Assoziation auch umgehend in die Routingtabelle überführt. Wird eine Assoziation erst später etabliert (signalisiert anhand einer `<Enable>`-Nachricht), wird das „Routing Backend“ alle Einträge bezüglich des „frisch“ assoziierten Netzzugangsgeräts wieder hinzufügen. Erst im Anschluss, nachdem alle Routen gesetzt worden sind, wird die `<Enabled>`-Nachricht an den REACH-Client versendet.

So wird sichergestellt, dass der REACH-Client immer genau weiß, welche Adressbindungen tatsächlich in der Routingtabelle hinterlegt sind und benutzt werden dürfen. Für jedes diesbezügliche Kommando an das „Routing Backend“ wird immer eine Antwort erwartet, bevor der REACH-Client die Adressen benutzen darf: Eine `<AddAddress>`-Nachricht wird durch eine `<AddressAdded>`-Nachricht bestätigt, und nach einer erfolgreichen Assoziation wird die zugehörige `<Enable>`-Nachricht immer durch eine `<Enabled>`-Nachricht bestätigt. Hat der REACH-Client *beide* „Bestätigungen“ vorliegen, gelten die betroffenen Routen als gesetzt und der Netzzugang als benutzbar. Erst dann dürfen Daten an den REACH-Proxyserver versendet werden.

C.1.2.9 Beendigungsphase

Mit Beendigung des REACH-Clients wird auch das „Routing Backend“ beendet. Dazu wird die in Abbildung C.11 gezeigte `<ShutDown>`-Nachricht übergeben. Das „Routing Backend“ wird daraufhin die Routingtabellen in einen geordneten Zustand überführen und sich anschließend beenden.

```
<ReachClient>
  <ShutDown />
</ReachClient>
```

Abbildung C.11: Diese Nachricht befiehlt dem „Routing Backend“, sich geordnet zu beenden.

C.2 Zugriff auf die Routingtabellen

Der Datenübertragungsmechanismus zwischen REACH-Client und REACH-Proxyserver kann mit Hilfe von quelladressbasierter oder zieladressbasierter Wegewahl „gesteuert“ werden. Bei ersterem werden alle Sockets auf bestimmte IP-Adressen aus einem privaten Adressbereich gebunden, wobei jede dieser IP-Adressen einem der verfügbaren Netzzugangsgeräte zugeordnet ist. Es findet also eine Wegewahl statt, welche davon abhängt, auf welche IP-Adresse ein Socket gebunden worden ist. Bei der zieladressbasierten Wegewahl wird das Netzzugangsgerät hingegen anhand der Zieladresse eines jeden IP-Pakets ermittelt. REACH erweitert dieses Schema noch um die explizite Bindung eines Sockets an ein Netzzugangsgerät, was allerdings nicht in jedem Fall erlaubt ist.

Bislang wurde jedoch noch nicht beschrieben, welche Mechanismen notwendig sind, um die Routingtabellen zu verändern. Dies soll nun geschehen.

C.2.1 Auswahl eines Routingschemas

Sobald das „Routing Backend“ gestartet wird, beginnt es mit der Ermittlung der verfügbaren Wegewahlschemata. Ein „Routing Backend“ für Microsoft Windows bräuchte diese Tests nicht durchzuführen, da hier prinzipiell keine quelladressbasierte Wegewahl möglich ist. Zudem gehören bei Microsoft Windows die relevanten Systemkommandos bereits zum Basissystem, sodass eine Verfügbarkeitsprüfung unnötig wäre. Da der Fokus der vorliegenden Arbeit jedoch auf GNU/Linux-basierten mobilen Endgeräten liegt, sind Aussagen bezüglich Microsoft Windows ungetestet und damit spekulativ.

Unter GNU/Linux sind zwei Fragen zu klären: Erstens, ist eine quelladressbasierte Wegewahl seitens des Kernels verfügbar, und wenn ja, steht zudem das iproute2-Softwarepaket zur Verfügung? Letzteres enthält das modernere `ip`-Kommando, welches die Funktionen der „klassischen“ Kommandos `ifconfig` und `route` vereint und zudem mit mehreren Routingtabellen umgehen kann.

Der erste Test bezieht sich auf das `ip`-Kommando, wobei in Abbildung C.12 drei mögliche Resultate dargestellt werden. Das Kommando wird hierzu mit „neutralen Parametern“ gestartet und anschließend dessen Ausgabe interpretiert. Im ersten Fall war

```
system_a ~ # /sbin/ip route add
bash: /sbin/ip: No such file or directory

florian@system_b ~ $ /sbin/ip route add
RTNETLINK answers: Operation not permitted
florian@system_b ~ $ su
Password:
system_b florian # /sbin/ip route add
RTNETLINK answers: No such device
system_b florian #
```

Abbildung C.12: Es muss geprüft werden, ob das `ip`-Kommando verfügbar ist und ob die Nutzerrechte ausreichend sind, um die Routingtabellen zu modifizieren.

das Kommando nicht verfügbar, womit es als nicht benutzbar gilt. Im zweiten Fall wurde es als Nutzer `florian` auf einem anderen System gestartet, wobei es zwar als verfügbar detektiert wurde (die erstgenannte Fehlermeldung blieb aus), jedoch deutet die gelieferte Fehlermeldung auf nicht ausreichende Rechte hin. In diesem Fall kann das Kommando ebenfalls nicht benutzt werden, wobei hier eher ein generelles Konfigurationsproblem zu vermuten ist, da das „Routing Backend“ mit Systemverwaltungsrechten gestartet werden muss. Dies wurde in diesem Beispiel durch das `su`-Kommando („Set User“) nachgeholt, wonach sich die Ausgabe des `ip`-Kommandos abermals geändert hatte: Die Meldung „`No such device`“ ist hierbei als Erfolgsmeldung zu deuten, da es anzeigt, dass sowohl das Kommando selbst auch genügend Rechte zur Manipulation der Routingtabellen vorhanden sind.

```
system_a ~ # /sbin/ip rule
RTNETLINK answers: Operation not supported
Dump terminated

system_b ~ # /sbin/ip rule
0:      from all lookup local
32766:  from all lookup main
32767:  from all lookup default
```

Abbildung C.13: Ist das `ip`-Kommando verfügbar, wird anschließend geprüft, ob seitens des Kernels die quelladressbasierte Wegewahl angeboten wird.

Konnte das `ip`-Kommando gefunden werden und sind zudem ausreichend Rechte verfügbar, wird ein zweiter Test durchgeführt (siehe Abbildung C.13). Hierbei geprüft, ob der Kernel mit „mehreren Routingtabellen“ umgehen kann.

Im ersten Fall deutet die Fehlermeldung „`Operation not supported`“ darauf hin, dass diese Kernelfunktion nicht verfügbar ist. Das „Routing Backend“ wird in diesem Fall auf die zieladressbasierte Wegewahl Routing unter Verwendung des `ip`-Kommandos zurückgreifen.

Im zweiten Fall wird eine Liste von Regeln (`rule`) ausgegeben, was auf eine vorhandene Unterstützung von „Policy Routing“ hindeutet, und gleichbedeutend mit der Verfügbarkeit mehrerer Routingtabellen ist. Das „Routing Backend“ wird in diesem Fall die quelladressbasierte Wegewahl einsetzen.

```
system_a ~ # /sbin/route add default
bash: /sbin/route: No such file or directory

florian@system_b ~ $ /sbin/route add default
SIOCADDRT: Operation not permitted
florian@system_b ~ $ su
Password:
system_b florian # /sbin/route add default
SIOCADDRT: No such device
```

Abbildung C.14: Sollte das `ip`-Kommando nicht verwendet werden können, wird die Nutzbarkeit des `route`-Kommandos überprüft.

Gab es dagegen ein prinzipielles Problem mit dem `ip`-Kommando, wird als Alternative das `route`-Kommando getestet (siehe Abbildung C.14). Die Vorgehensweise ist dabei ähnlich: im ersten Fall war das Kommando nicht verfügbar, im zweiten Fall waren die Rechte nicht ausreichend und im dritten Fall war die Meldung „`No such device`“ als Erfolg zu werten. Nur im letzten Fall kann eine zieladressbasierte Wegewahl auf der Basis des `route`-Kommandos durchgeführt werden. In den beiden anderen Fällen fehlen dem „Routing Backend“ alle Werkzeuge. Es würde sich mit einer Fehlermeldung beenden.

C.2.2 Quelladressbasierte Wegewahl

Die quelladressbasierte Wegewahl basiert auf dem so genannten „Policy Routing“. Hierbei wird ein jedes IP-Paket gegen einen Satz von Regeln (die „Policy“) geprüft, und erst anschließend „geroutet“. Allerdings wird erst durch diese Regeln bestimmt, welche Routingtabelle zum Einsatz kommen wird. Die „zentrale Routingtabelle“, welche beispielsweise mit Hilfe des `route`-Kommandos manipuliert werden kann, entspricht dabei der Tabelle `main` (erkennbar in Abbildung C.13). Zudem existieren bereits in der Basiskonfiguration zwei weitere Routingtabellen mit den Namen `local` und `default`, welche für die vorgestellten Zwecke allerdings ohne Belang sind.

Für die quelladressbasierte Wegewahl macht man sich diese Struktur zu nutze. Die Sockets, deren Datenströme über ein bestimmtes Netzzugangsgerät geleitet werden sollen, werden vom REACH-Client auf eine zugehörige lokale IP-Adresse gebunden. Diese IP-Adressen werden mit Hilfe von „Policy Routing“ mit den Netzzugangsgeräten des mobilen Endgeräts verknüpft. Dazu werden Regeln benutzt, welche die Absenderadresse der IP-Pakete auswerten. Als Ergebnis erhält man eine bestimmte Routingtabelle, welche anschließend zur Weiterleitung des Pakets herangezogen wird.

```
neelix ~ # ip address show lo
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet 10.254.254.10/24 brd 10.254.254.255 scope host lo
    inet 10.254.254.11/24 brd 10.254.254.255 scope host secondary lo
    inet 10.254.254.12/24 brd 10.254.254.255 scope host secondary lo
```

Abbildung C.15: Für die quelladressbasierte Wegewahl werden dem `lo`-Device zusätzliche lokale IP-Adressen zugewiesen.

Die besagten lokalen IP-Adressen werden vorab dem Loopback-Device `lo` zugewiesen, was idealerweise bereits durch die Startskripte des GNU/Linux-Systems erfolgt sein sollte. In Abbildung C.15 werden alle dem `lo`-Device zugeordneten IP-Adressen aufgelistet. Die Adressen `10.254.254.10`, `10.254.254.11` und `10.254.254.12` sind dabei die für den REACH-Client nutzbaren Adressen, was eine simultane Datenübertragung über drei Netzzugangsgeräte gleichzeitig erlaubt.

```
neelix ~ # /sbin/ip rule add from 10.254.254.10 table 100
neelix ~ # /sbin/ip rule add from 10.254.254.11 table 101
neelix ~ #
neelix ~ # /sbin/ip route add default via 192.168.1.1     \
                    dev eth0 table 100
neelix ~ # /sbin/ip route add default via 192.168.1.1     \
                    dev wlan0 table 101
```

Abbildung C.16: Für die quelladressbasierte Wegewahl müssen Regeln (`rule`) und Routen (`route`) gesetzt werden.

Das „Routing Backend“ bedient sich zum Hinterlegen der Regeln und zum Setzen der zugehörigen Routen des `ip`-Kommandos. Abbildung C.16 zeigt, wie diese Kommandos aussehen müssen, wenn man diese Mechanismen manuell über die Kommandozeile konfigurieren möchte.

Eine bestimmte Routingtabelle wird dabei über das Schlüsselwort `table` mit Hilfe eines numerischen Identifikators referenziert. REACH nutzt hierfür die Nummern ab `100` aufsteigend, wobei so viele Identifikatoren benötigt werden, wie das mobile Endgerät mit Netzzugangsgeräten ausgestattet ist. Die quelladressbasierte Wegewahl basiert darauf, dass für jede „besondere“ lokale IP-Adresse (die dem `lo`-Device zugewiesenen Adressen) eine Regel (`rule`) hinzugefügt (`add`) wird, welche die genannte IP-Adresse als Absenderadresse (`from`) bestimmt. Trifft diese Regel für ein bestimmtes IP-Paket zu, gilt die vom `table`-Parameter referenzierte Routingtabelle.

Die genannten Routingtabellen sind anfangs leer. Pro Tabelle wird jedoch nur ein einziger Eintrag benötigt, nämlich eine „Default Route“ über das gewünschte Netzzugangsgerät. Jedes IP-Paket, welches mit Hilfe dieser Routingtabelle weitergeleitet wird, wird demnach über dieses ausgehende Interface versendet. Da die Wahl der Tabelle anhand der Absenderadresse erfolgt, bezeichnet man das vorgestellte Schema als *quelladressbasierte Wegewahl*. Dabei ist es kein Problem, wenn die IP-Adressen zweier „Default Gateways“ zufällig identisch sind. Durch den `dev`-Parameter wird hinsichtlich der Route sichergestellt, dass das gewünschte Netzzugangsgerät („Device“) auch wirklich benutzt wird.

```
neelix ~ # /sbin/ip rule show
0:      from all lookup local
32764:  from 10.254.254.11 lookup 101
32765:  from 10.254.254.10 lookup 100
32766:  from all lookup main
32767:  from all lookup default
neelix ~ #
neelix ~ # /sbin/ip route show table 100
default via 192.168.1.1 dev eth0
neelix ~ # /sbin/ip route show table 101
default via 192.168.1.1 dev wlan0
```

Abbildung C.17: Zwei erfolgreich gesetzte Regeln und Routen bei quelladressbasierter Wegewahl.

In Abbildung C.17 ist dargestellt, wie die Regeltabelle und die Routingtabellen aussehen, nachdem die Kommandos aus Abbildung C.16 eingegeben worden sind.

Der REACH-Client ist damit in der Lage, eine quelladressbasierte Wegewahl durchzuführen. Dazu muss er aber wissen, welches Netzzugangsgerät mit welcher lokalen IP-Adresse verknüpft worden ist. Dieser Zusammenhang wird in der Konfigurationsdatei `routing.xml` hergestellt, welche zudem den Dateinamen des zu startenden „Routing Backends“ benennt. Zu jedem Gerätenamen `deviceentrylabel` wird hier die IP-Adres-

```
<Routing exec="/usr/local/bin/reachplugin-route">
  <masq deviceentrylabel="Internal Ethernet"
        bindip="10.254.254.10" />
  <masq deviceentrylabel="ZD1211 USB"
        bindip="10.254.254.11" />
</Routing>
```

Abbildung C.18: Der REACH-Client erhält seine wegewahlspezifischen Parameter aus der Konfigurationsdatei `routing.xml`. Diese benennt den Dateinamen des „Routing Backends“ und spezifiziert zudem Netzzugangsgerät-Adress-Paare für die quelladressbasierte Wegewahl.

se `bindip` aufgeführt, auf welche der REACH-Client im Fall von quelladressbasierter Wegewahl seine Sockets „binden“ muss, wenn ein bestimmtes Netzzugangsgerät benutzt werden soll.

Es wird jedoch noch ein letzter Schritt benötigt, damit das vorgestellte Schema ordnungsgemäß funktioniert. So wie dargestellt arbeitet zwar bereits die Wegewahl für ausgehende IP-Pakete, jedoch weisen diese als Absenderadresse noch eine private IP-Adresse des `lo`-Devices auf, anhand welcher im Internet nicht „geroutet“ werden kann. Es muss also noch ein „Masquerading“ konfiguriert werden, sodass die Absenderadresse eines jeden ausgehenden IP-Pakets mit der IP-Adresse des passierten Netzzugangsgeräts „maskiert“ wird. Für rücklaufende IP-Pakete muss zudem klar sein, auf welche lokale IP-Adresse die Zieladresse wieder geändert werden muss, damit die Daten schlussendlich an einem Socket des REACH-Clients gelesen werden können („connection tracking“).

```
#INTERFACE              SOURCE
eth0                    10.254.254.10
wlan0                   10.254.254.11
ppp0                    10.254.254.12
```

Abbildung C.19: Bei Nutzung von Shorewall kann das für die quelladressbasierte Wegewahl erforderliche „Masquerading“ in der Konfigurationsdatei `/etc/shorewall/masq` konfiguriert werden.

„Masquerading“ wird bei Netfilter in der `nat`-Tabelle konfiguriert. Da im aufgebauten Demonstrator durchgängig das Frontend Shorewall eingesetzt wird, beschränkt sich der Konfigurationsaufwand auf die Datei `/etc/shorewall/masq` (siehe Abbildung C.19). Hier wird jeweils das Interface genannt, dessen IP-Adresse benutzt werden soll, um ein ausgehendes IP-Paket zu maskieren. In der zweiten Spalte stehen die zugehörigen IP-Adressen des `lo`-Devices, welche ein Filterkriterium darstellen.

Die `masq`-Datei wird von REACH nicht verändert und muss vorab hinterlegt worden sein. Zudem muss das `lo`-Device die genannten IP-Adressen aufweisen und die Konfigurationsdatei des REACH-Clients `routing.xml` muss die Adressbindung benennen.

C.2.3 Zieladressbasierte Wegewahl

Bei der zieladressbasierten Wegewahl wird die Zieladresse eines jeden IP-Pakets ausgewertet. Diese Zieladressen sind die IP-Adressen von REACH-Proxyservern. Jedoch besteht das Problem nicht darin, die IP-Pakete bloß zustellen zu können, sondern es muss eine deterministische Wegewahl stattfinden.

Dies geschieht dadurch, dass die IP-Adressen der REACH-Proxyserver jeweils mit einem bestimmten Netzzugangsgerät verknüpft werden, damit ausgehende IP-Pakete das mobile Endgerät über einen festgelegten Weg verlassen.

```
neelix ~ # /sbin/ip route add to 141.24.92.184  \
                    via 192.168.1.1 dev eth0
neelix ~ #
neelix ~ # /sbin/ip route del to 141.24.92.184  \
                    via 192.168.1.1 dev eth0
```

Abbildung C.20: Für die zieladressbasierte Wegewahl können die Routen mit Hilfe des `ip`-Kommandos gesetzt und entfernt werden.

Zum Setzen und Löschen von Routen kann entweder das `ip`-Kommando (siehe Abbildung C.20) oder das `route`-Kommando (siehe Abbildung C.21) benutzt werden. Die Auswahl des Kommandos erfolgt wie bereits dargestellt durch das „Routing Backend“.

```
neelix ~ # /sbin/route add 141.24.92.184        \
                       gw 192.168.1.1 dev eth0
neelix ~ #
neelix ~ # /sbin/route del 141.24.92.184        \
                       gw 192.168.1.1 dev eth0
```

Abbildung C.21: Auch mit Hilfe des `route`-Kommandos können Routen zu REACH-Proxyservern gesetzt und entfernt werden.

Die in den Abbildungen C.20 und C.21 gezeigten Befehle sind von ihrer Wirkung her identisch: das erste Kommando setzt jeweils eine Route (`add`) zu einer bestimmten IP-Zieladresse (`141.24.92.184`) über ein genanntes Gateway (`192.168.1.1`) durch ein bestimmtes Interface (`dev eth0`). Das zweite Kommando entfernt die entsprechende Route wieder aus der zentralen Routingtabelle.

D Prädiktive Handoverentscheidungen

In diesem Kapitel werden insgesamt zehn Messreihen vorgestellt, welche das Verhalten des prädiktiven Handoverentscheiders von REACH demonstrieren. Dazu kam ein USB-WLAN-Adapter der Firma ZyXEL zum Einsatz (Modell „ZyXEL ZyAIR G-200 v2“), welcher die Standards IEEE 802.11b und g beherrscht. Als Basisstation kam jeweils ein WLAN-Router (Modell „ASUS WL 500G Premium“) zum Einsatz, welcher an das Campusnetzwerk der TU Ilmenau angeschlossen worden ist. Bei allen „Outdoor“-Messungen stand dieser außen auf einem Fensterbrett des Raums H3501 (das Fenster geradeaus beim Betreten des Raums, mit gutem Blick auf den Platz mit Newtonbau, Humboldtbau und rechts den Kirchhoffbau). Bei den beiden „Indoor“-Messungen 7 und 8 befand sich die Basisstation auf dem Schreibtisch des Autors.

In allen Fällen kam ein mit WPA2 verschlüsseltes Netzwerk zum Einsatz. Es wurde jeweils ein TCP-Datenstrom von einem ortsfest aufgebauten PC in Richtung des Laptops verschickt. Der Laptop ermittelte zyklisch die aktuell anliegende Signalstärke sowie die Linkgüte durch Auslesen der Datei `/proc/net/wireless` und schrieb diese Werte zusammen mit einem Zeitstempel sowie der gemessenen TCP-Datenrate in eine Datei. Diese Datei wurde dann später „offline“ mit Hilfe des Handoverentscheiders von REACH ausgewertet, was zu den Diagrammen in diesem Kapitel führte.

Anhand der aufgenommenen Signalstärke- und Linkgütewerte mussten in einem ersten Schritt die drei Entscheidungsschwellen (Abriss-, Eignungs- und Idealschwelle) empirisch bestimmt werden. Um die Schwellen möglichst passend zu wählen, wurde die Datenrate des TCP-Datenstroms betrachtet. Der stetige Strom an eintreffenden Daten sorgte zudem dafür, dass häufig Daten über die Luftschnittstelle empfangen wurden und somit auch sich häufig ändernde Signalstärke- und Linkgütewerte ermittelt werden konnten. Nachdem die Schwellen ausgewählt worden waren, konnte der Handoverentscheider zur Auswertung der Messreihen herangezogen werden. Durch iteratives Ausprobieren wurde zudem versucht, die anfangs getroffene Wahl für die Schwellen sukzessive zu verbessern.

Die zehn Messreihen sind zu unterschiedlichen Zeiten entstanden; die Messreihen 1 und 2 sind etwa 2 Jahre früher aufgenommen worden als die Messreihen 3 bis 10. Bei allen „Outdoor“-Messungen herrschten trockene winterliche Bedingungen vor, das heißt, es gab keine Abschattung durch belaubte Bäume oder durch Niederschlag in Form von

Regen oder Schnee.

Die Messreihen 1 und 2 werden im Folgenden dazu benutzt, die Unterschiede zwischen exponentieller, inversquadratischer und linearer Regressionsrechnung zu untersuchen. Bei beiden Messungen wurde ein Linux-Kernel in der Version 2.6.27 eingesetzt, welcher bezüglich des verwendeten WLAN-Treibers noch „eigenständige“ Signalstärke- *und* Linkgütewerten anbot.

Die Messreihen 3 bis 6 wurden im Winter 2009 / 2010 aufgenommen, um die Funktion des Handoverentscheiders anhand einer größeren Datenbasis untersuchen zu können. Beim hierbei eingesetzten Linux-Kernel 2.6.31 lieferte der verwendete Treiber keine Linkgütewerte mehr, sodass nur noch eine Betrachtung der Signalstärkewerte vorgenommen werden konnte. Zudem stand ein anderer Laptop zur Verfügung, sodass dank USB 2.0 hier höhere Datenübertragungsraten erreicht werden konnten.

Bei den Messreihen 7 und 8 handelt es sich um „Indoor“-Messungen. Es wurde vorab vermutet, dass eine Signalstärkeermittlung mit einem Messwert pro Sekunde nicht ausreichend sein würde, um innerhalb von Gebäuden brauchbare Prädiktionsergebnisse zu erhalten. Daher wurden die Messwerte ab Messreihe 7 mit einer willkürlich festgelegten höheren Rate von 50 Messungen pro Sekunde aufgenommen. Zusätzlich stellte sich die Frage, ob die in den Messreihen 1 bis 6 verwendete lange Fenstergröße von 30 Sekunden innerhalb von Gebäuden noch Sinn ergibt. Um dies zu untersuchen, wurden alle folgenden Messreihen jeweils mit einer Fenstergröße von 12, 20 sowie 30 Sekunden ausgewertet, immer mit einem Mindestfüllstand von 10 Sekunden.

Die Ergebnisse aus den Messreihen 7 und 8 waren im Vergleich weitaus „schöner“, sodass beschlossen wurde, mit den Messreihen 9 und 10 nochmals Außenmessungen ebenfalls mit einer höheren Messwerterfassungsrate durchzuführen. Auch hier kamen wieder die drei Fenstergrößen 12, 20 sowie 30 Sekunden zum Einsatz, wieder jeweils mit einem Mindestfüllstand von 10 Sekunden. Die Idee war es, im Vergleich mit den Messreihen 7 und 8 eine Fenstergröße zu finden, welche sowohl innerhalb von Gebäuden als auch außerhalb sinnvolle Ergebnisse liefert.

Da sich gezeigt hat, dass der Rechenaufwand bei einer Messrate von 50 Werten pro Sekunde nicht länger „online“ durchgeführt werden kann, blieb schlussendlich die Frage zu klären, welche Erfassungsrate sowohl gute Ergebnisse liefert als auch noch zeitnah berechenbar war. Daher wurden die Messreihen 8 und 10 zwar mit 50 Messwerten pro Sekunde aufgenommen, deren Auswertung fand allerdings nur mit jedem fünften Wert statt, was einer Erfassungsrate von 10 Werten pro Sekunde entspricht.

D.1 Messreihe 1

Szenario

Anhand der Messreihe aus Abbildung D.1 sollte untersucht werden, wie sich Signalstärke, Linkgüte und Datenrate verhalten, wenn sich das mobile Endgerät geradlinig von der Basisstation entfernt. Die Messung begann direkt unterhalb der vom Autor aufgestellten Basisstation, welche sich im zweiten Stock des Helmholtzbaus im Raum H3501 befand. Der Laptop wurde mit langsamer Schrittgeschwindigkeit in Richtung des Newtonbaus getragen. Es ging geradlinig weiter den Fußweg entlang, links am Staubecken vorbei in Richtung Mensa. Etwa auf Höhe des Staubeckens wurde die Messung abgebrochen, da die Werte bereits zu weit abgefallen waren. Da der Laptop *vor* dem Nutzer getragen wurde, war die Funkstrecke meist verdeckt.

Die Messdaten wurden einmal pro Sekunde ermittelt. Für die Auswertung kam ein Zeitfenster von 30 Sekunden (30 Messwerte) zum Einsatz, bei einem minimalen Füllstand von 10 Sekunden (10 Messwerte). Da sowohl Signalstärke- als auch Linkgütewerte zur Verfügung standen, werden im Folgenden beide Verläufe ausgewertet und miteinander verglichen.

D.1.1 Auswertung des Signalstärkeverlaufs

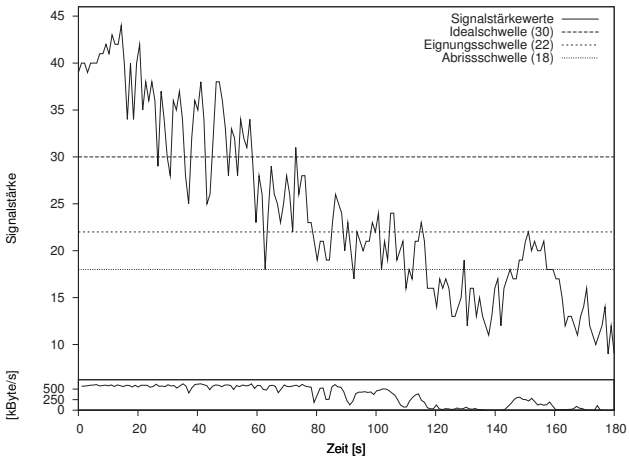

Abbildung D.1: Signalstärkewerte und TCP-Datenrate

Messreihe 1, Signalstärkewerte, exponentielle Regression

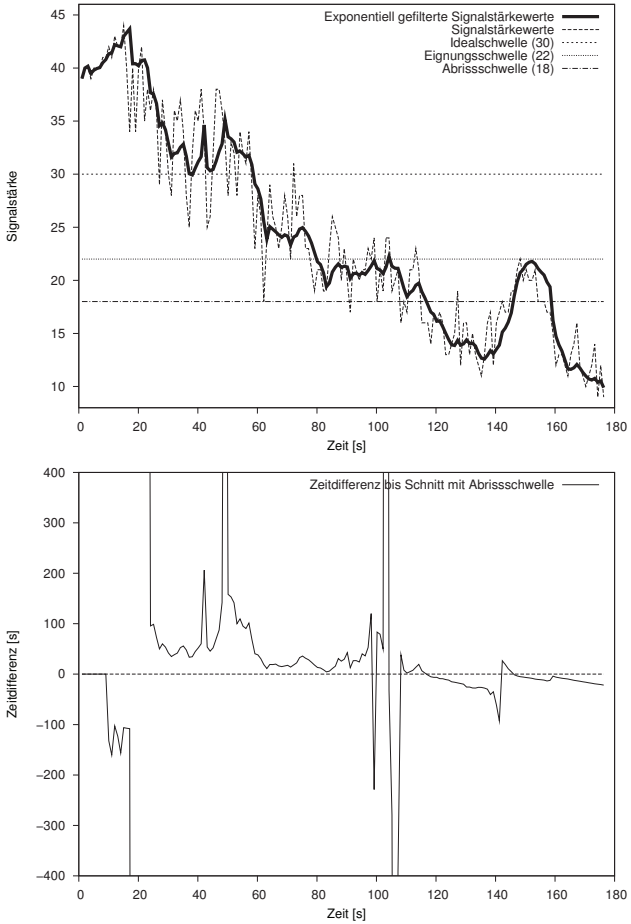

Abbildung D.2: Das obere Diagramm zeigt die exponentiell gefilterten Signalstärkewerte im Vergleich mit den Rohdaten. Das untere Diagramm führt zu jedem Zeitpunkt den zeitlichen Abstand zum errechneten Schnittpunkt zwischen der momentan gültigen Regressionsgeraden und der Abrissschwelle auf. Innerhalb der ersten 10 Sekunden war die Mindestwertemenge für die Regressionsrechnung noch nicht verfügbar, sodass hier noch auf eine Berechnung der Zeitdifferenz verzichtet worden war.

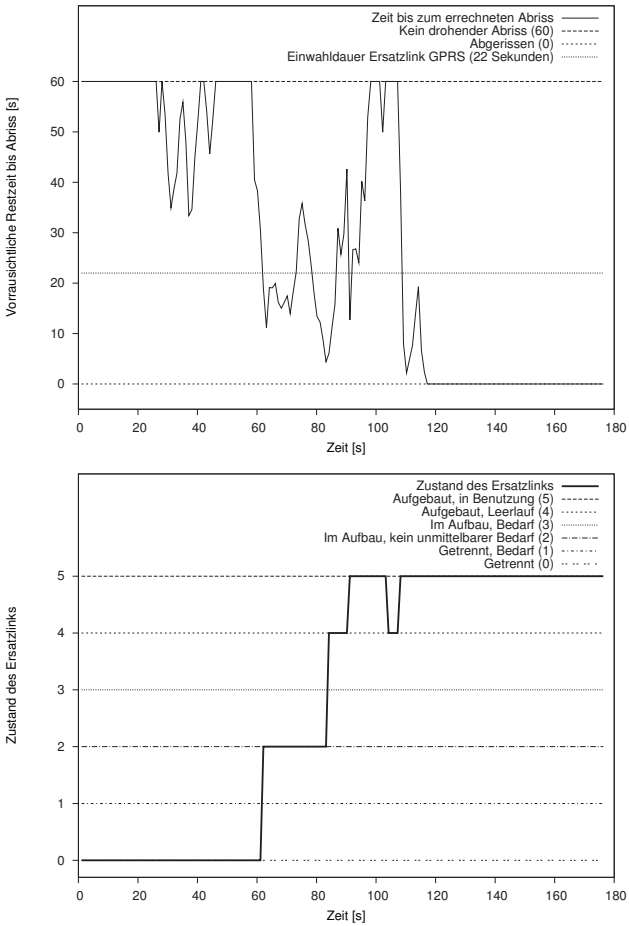

Abbildung D.3: Voraussichtliche Restzeit bis zum Abriss des Hauptlinks (oberes Diagramm) und Ersatzlinknutzung durch den Handoverentscheider (unteres Diagramm)

Messreihe 1, Signalstärkewerte, inversquadratische Regression

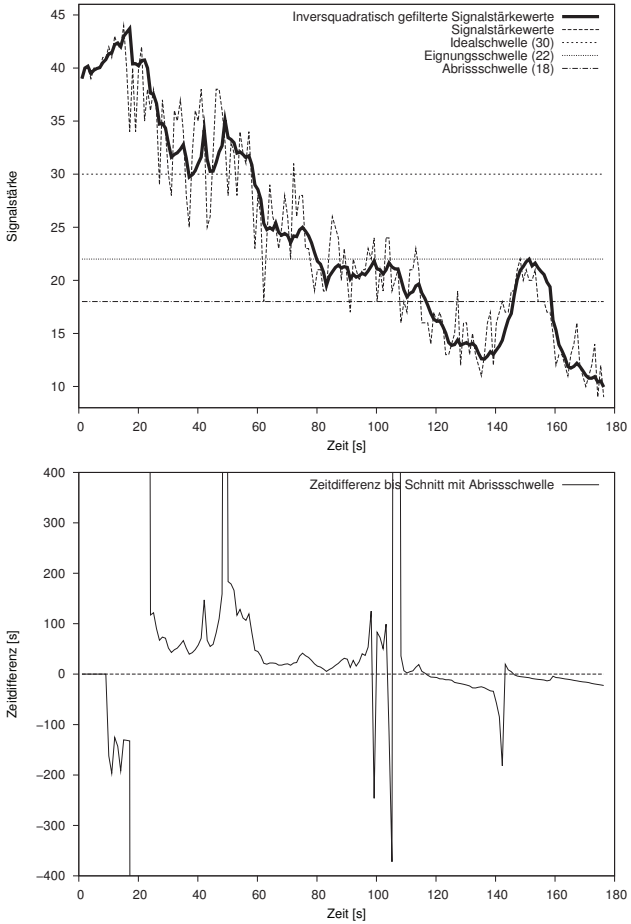

Abbildung D.4: Das obere Diagramm zeigt die inversquadratisch gefilterten Signalstärkewerte im Vergleich mit den Rohdaten. Das untere Diagramm führt zu jedem Zeitpunkt den zeitlichen Abstand zum errechneten Schnittpunkt zwischen der momentan gültigen Regressionsgeraden und der Abrissschwelle auf. Innerhalb der ersten 10 Sekunden war die Mindestwertemenge für die Regressionsrechnung noch nicht verfügbar, sodass hier noch auf eine Berechnung der Zeitdifferenz verzichtet worden war.

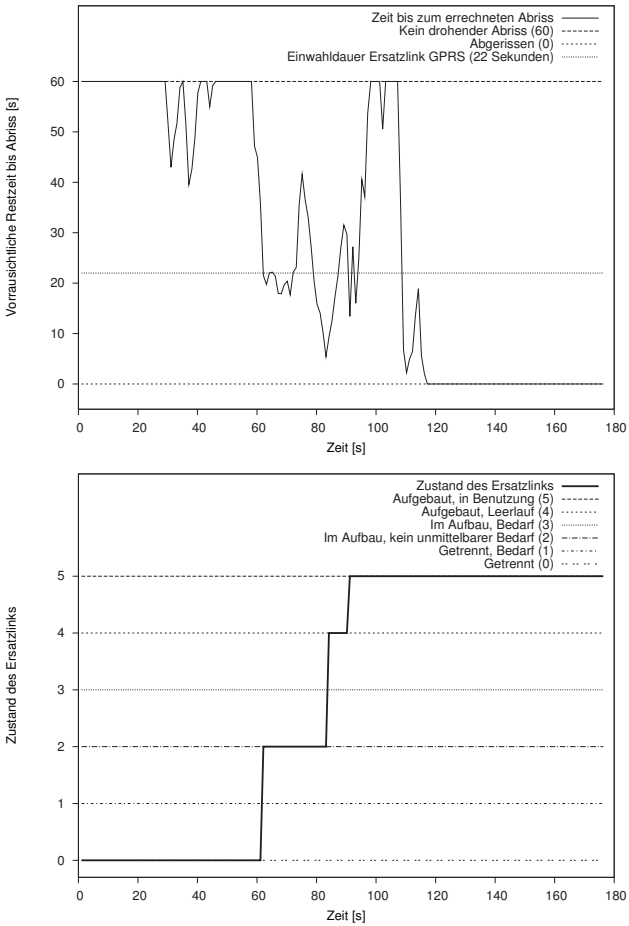

Abbildung D.5: Voraussichtliche Restzeit bis zum Abriss des Hauptlinks (oberes Diagramm) und Ersatzlinknutzung durch den Handoverentscheider (unteres Diagramm)

Messreihe 1, Signalstärkewerte, lineare Regression

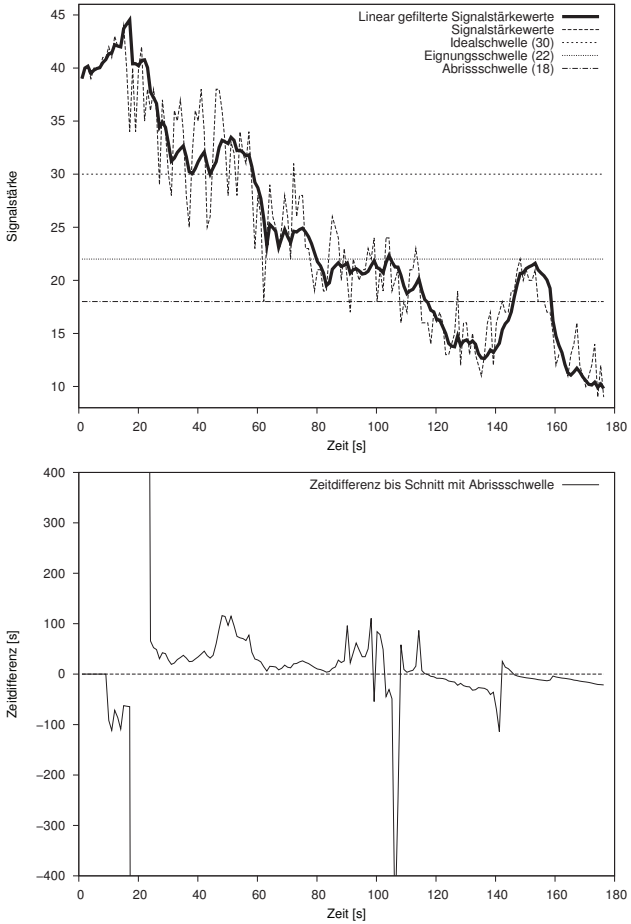

Abbildung D.6: Das obere Diagramm zeigt die linear gefilterten Signalstärkewerte im Vergleich mit den Rohdaten. Das untere Diagramm führt zu jedem Zeitpunkt den zeitlichen Abstand zum errechneten Schnittpunkt zwischen der momentan gültigen Regressionsgeraden und der Abrissschwelle auf. Innerhalb der ersten 10 Sekunden war die Mindestwertemenge für die Regressionsrechnung noch nicht verfügbar, sodass hier noch auf eine Berechnung der Zeitdifferenz verzichtet worden war.

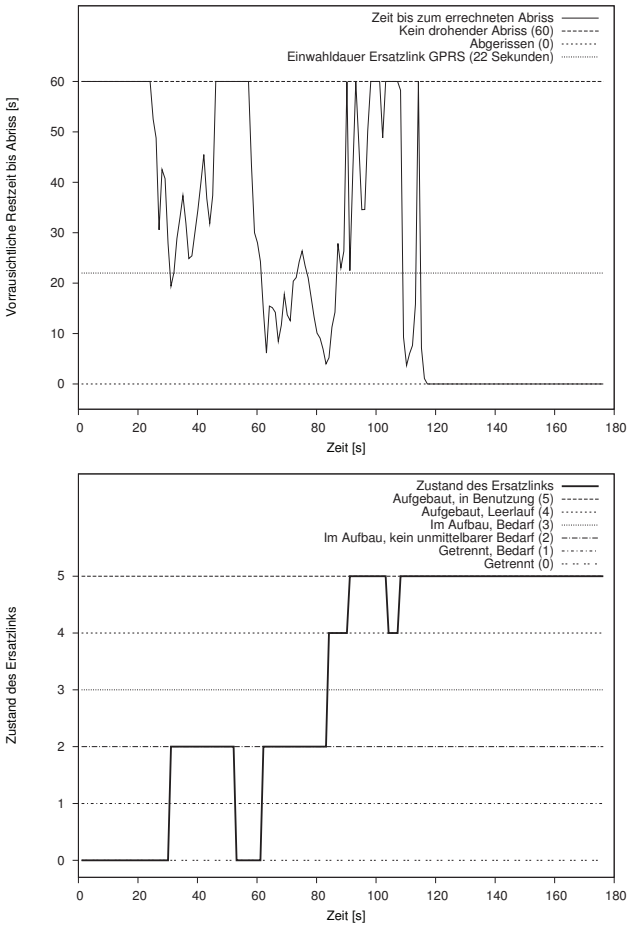

Abbildung D.7: Voraussichtliche Restzeit bis zum Abriss des Hauptlinks (oberes Diagramm) und Ersatzlinknutzung durch den Handoverentscheider (unteres Diagramm)

Verstrichene Zeit [s]	Ereignis Ersatzlink GPRS
31	Assoziiere Ersatzlink
53	Assoziation abgebaut
62	Assoziiere Ersatzlink
84	Assoziation aufgebaut
91	Beginne mit Datenübertragung
104	Stoppe Datenübertragung
108	Beginne mit Datenübertragung

Tabelle D.1: Diese Kommandos und Statusänderungen wurden bezüglich des hier betrachteten Ersatzlinks (GPRS mit einer Einwahlzeit von 22 Sekunden) vom Handoverentscheider abgesetzt bzw. empfangen. Zur Anwendung kam hierbei die lineare Regressionsrechnung.

Diskussion: Auswertung der Signalstärkewerte

Die vorliegende Messreihe bestehend aus Signalstärkewerten wurde sowohl mit Hilfe exponentieller, inversquadratischer als auch mit linearer Regressionsrechnung ausgewertet. In Bezug auf die Nutzung des Ersatzlinks lieferten die exponentielle und die inversquadratische Regressionsrechnung ein vergleichbares Ergebnis (untere Diagramme in den Abbildungen D.3 und D.5), während bei der linearen Regressionsrechnung zum Zeitpunkt 31s der Ersatzlink unnötigerweise angefordert wurde (unteres Diagramm in Abbildung D.7). Der Grund hierfür ist in den Restzeitdiagrammen (selbe Abbildungen, obere Diagramme) auszumachen. Bei der linearen Regressionsrechnung fällt das Restzeitdiagramm bei 31s kurzzeitig unter die Entscheidungsschwelle. Sowohl bei der exponentiellen als auch der inversquadratischen Regressionsrechnung geschieht dies nicht, da deren Prognose auf der Annahme eines konkaven Verlaufs beruht.

Für die exponentielle und die inversquadratische Regressionsrechnung kann festgestellt werden, dass sie in diesem Beispiel einen sehr guten Zeitpunkt zur Assoziation des Ersatzlinks errechnet haben. Der aufgebaute Ersatzlink verweilt bis zu seiner Nutzung nur wenige Sekunden im Leerlauf.

D.1.2 Auswertung des Linkgüteverlaufs

Die in Abbildung D.8 gezeigten Linkgütewerte wurden in einem Schritt zusammen mit den bereits diskutierten Signalstärkewerten aufgenommen. Obwohl sich der Verlauf und die Charakteristik unterschiedlich zu den Signalstärkewerten verhalten, kann dennoch eine ähnliche Entwicklung gemäß des beschriebenen Szenarios erkannt werden.

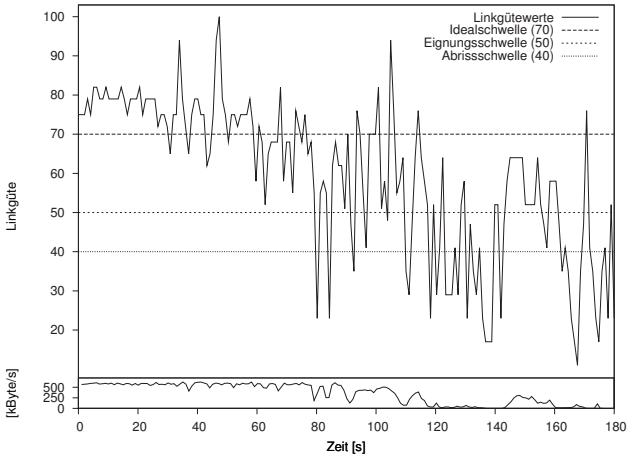

Abbildung D.8: Linkgütewerte und TCP-Datenrate

Messreihe 1, Linkgütewerte, exponentielle Regression

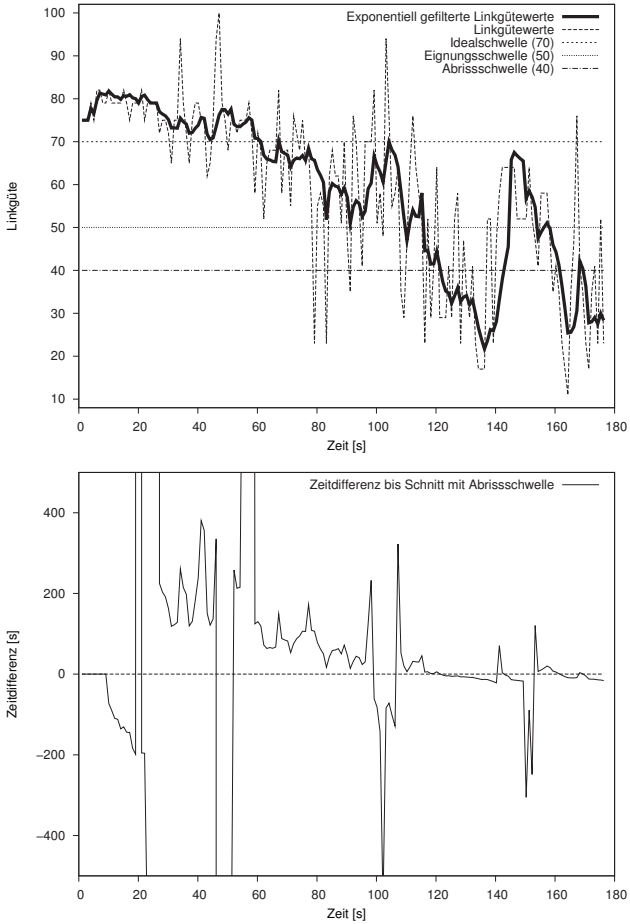

Abbildung D.9: Das obere Diagramm zeigt die exponentiell gefilterten Linkgütewerte im Vergleich mit den Rohdaten. Das untere Diagramm führt zu jedem Zeitpunkt den zeitlichen Abstand zum errechneten Schnittpunkt zwischen der momentan gültigen Regressionsgeraden und der Abrissschwelle auf. Innerhalb der ersten 10 Sekunden war die Mindestwertemenge für die Regressionsrechnung noch nicht verfügbar, sodass hier noch auf eine Berechnung der Zeitdifferenz verzichtet worden war.

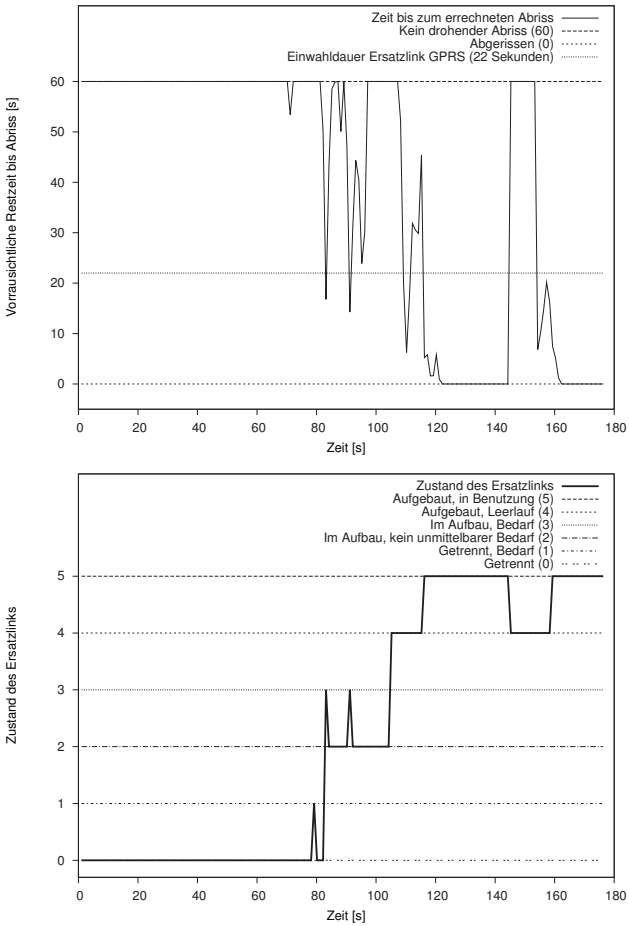

Abbildung D.10: Voraussichtliche Restzeit bis zum Abriss des Hauptlinks (oberes Diagramm) und Ersatzlinknutzung durch den Handoverentscheider (unteres Diagramm)

Messreihe 1, Linkgütewerte, inversquadratische Regression

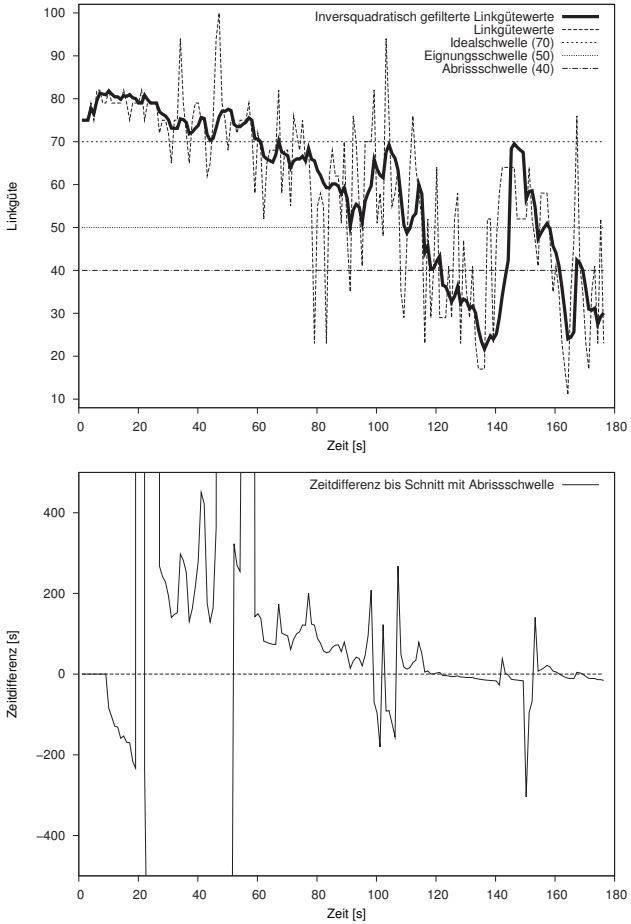

Abbildung D.11: Das obere Diagramm zeigt die inversquadratisch gefilterten Linkgütewerte im Vergleich mit den Rohdaten. Das untere Diagramm führt zu jedem Zeitpunkt den zeitlichen Abstand zum errechneten Schnittpunkt zwischen der momentan gültigen Regressionsgeraden und der Abrissschwelle auf. Innerhalb der ersten 10 Sekunden war die Mindestwertemenge für die Regressionsrechnung noch nicht verfügbar, sodass hier noch auf eine Berechnung der Zeitdifferenz verzichtet worden war.

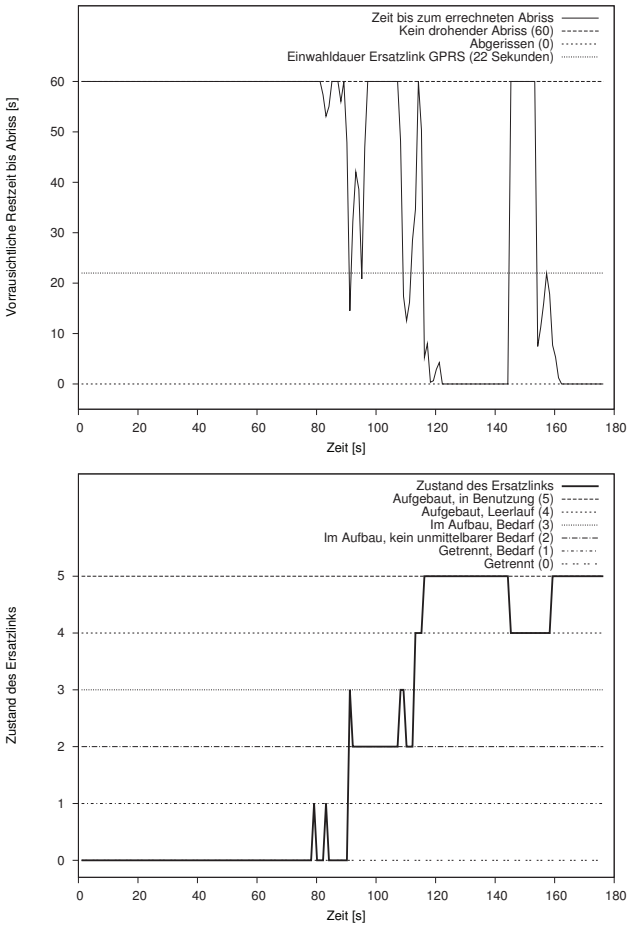

Abbildung D.12: Voraussichtliche Restzeit bis zum Abriss des Hauptlinks (oberes Diagramm) und Ersatzlinknutzung durch den Handoverentscheider (unteres Diagramm)

Messreihe 1, Linkgütewerte, lineare Regression

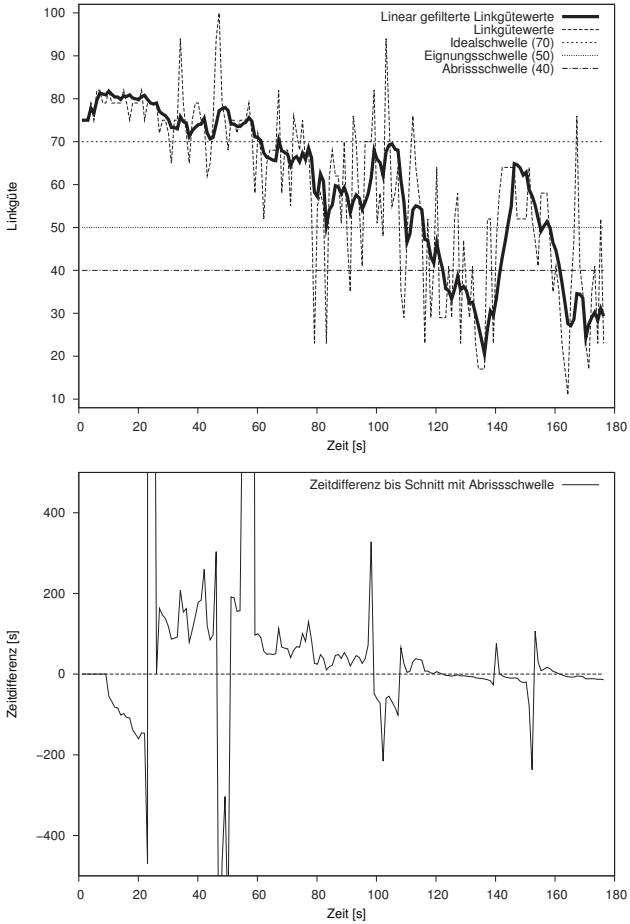

Abbildung D.13: Das obere Diagramm zeigt die linear gefilterten Linkgütewerte im Vergleich mit den Rohdaten. Das untere Diagramm führt zu jedem Zeitpunkt den zeitlichen Abstand zum errechneten Schnittpunkt zwischen der momentan gültigen Regressionsgeraden und der Abrissschwelle auf. Innerhalb der ersten 10 Sekunden war die Mindestwertemenge für die Regressionsrechnung noch nicht verfügbar, sodass hier noch auf eine Berechnung der Zeitdifferenz verzichtet worden war.

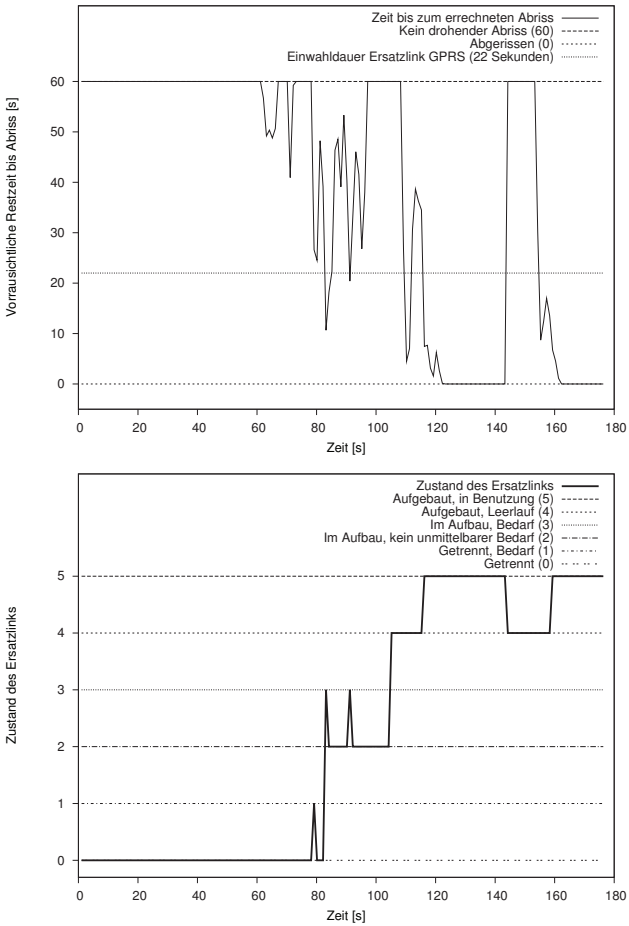

Abbildung D.14: Voraussichtliche Restzeit bis zum Abriss des Hauptlinks (oberes Diagramm) und Ersatzlinknutzung durch den Handoverentscheider (unteres Diagramm)

Verstrichene Zeit [s]	Ereignis Ersatzlink GPRS
83	Assoziiere Ersatzlink
105	Assoziation aufgebaut
116	Beginne mit Datenübertragung
144	Stoppe Datenübertragung
159	Beginne mit Datenübertragung

Tabelle D.2: Diese Kommandos und Statusänderungen wurden bezüglich des hier betrachteten Ersatzlinks (GPRS mit einer Einwahlzeit von 22 Sekunden) vom Handoverentscheider abgesetzt bzw. empfangen. Zur Anwendung kam hierbei die lineare Regressionsrechnung.

Diskussion: Auswertung der Linkgütewerte

Bei Betrachtung der Linkgütewerte liefern alle drei Regressionsrechnungen in Bezug auf die Nutzung des Ersatzlinks ein vergleichbares Ergebnis. Ein kleiner Unterschied ist jedoch bezüglich der inversquadratischen Regressionsrechnung feststellbar: gemäß Abbildung D.12 (unteres Diagramm) wird der Ersatzlink hierbei zeitlich knapp aufgebaut (gut, da pünktlich). Sowohl bei der exponentiellen als auch bei der linearen Regressionsrechnung wird der Ersatzlink ein paar Sekunden früher angefordert.

Es stellt sich die Frage, ob man sich für die Auswertung der Signalstärke- oder der Linkgütewerte entscheiden sollte. Zur Beantwortung dieser Frage ist es jedoch unzweckmäßig, die in den Tabellen D.1 und D.2 gezeigten Ersatzlinkereignisse miteinander zu vergleichen. Die gezeigten Ergebnisse hängen sehr stark von den nutzerseitig festgelegten Entscheidungsschwellen ab, welche hier durch empirisches Ausprobieren und der Methode des „scharfen Hinsehens“ ausgewählt worden waren.

D.2 Messreihe 2

Szenario

Bei Betrachtung der Signalstärkewerte aus Messreihe 1 (Abbildung D.1) zeigten sich zwei Effekte. Zum einen war die maximal erreichbare Reichweite unerwartet gering, was aber daran lag, dass der Benutzer den Laptop vor sich her trug und damit die Basisstation abgeschattet worden war. Zweitens war ab ca. 140s eine kurzfristige Erholung der Signalstärkewerte sowie der TCP-Datenrate erkennbar, für dessen Ursache ein alternativer Ausbreitungspfad vermutet wird. Ein „schön“ abfallender Verlauf war somit leider nicht gegeben. Da es sich hierbei jedoch um reale Messwerte handelt, ist damit davon auszugehen, dass in der Praxis eben keine „schönen“ Verläufe zu erwarten sind.

Nichtsdestotrotz wurde eine erneute Messung (siehe Abbildung D.15) mit identischem Szenario aus Messreihe 1 wiederholt. Der einzige Unterschied bestand darin, dass der Laptop jetzt seitlich getragen wurde, also die Antenne immer eine Sichtverbindung zur Basisstation besaß. Die erhaltenen Verläufe sehen „glatter“ aus, und es konnte auch eine größere Entfernung überbrückt werden. Erst, als der Laptop in Richtung Mensa hinter dem Hügel „abtauchte“, brach die Datenrate zusammen.

D.2.1 Auswertung des Signalstärkeverlaufs

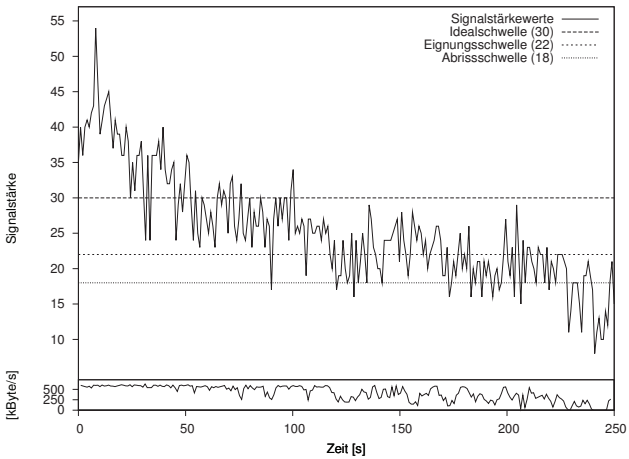

Abbildung D.15: Signalstärkewerte und TCP-Datenrate

Messreihe 2, Signalstärkewerte, exponentielle Regression

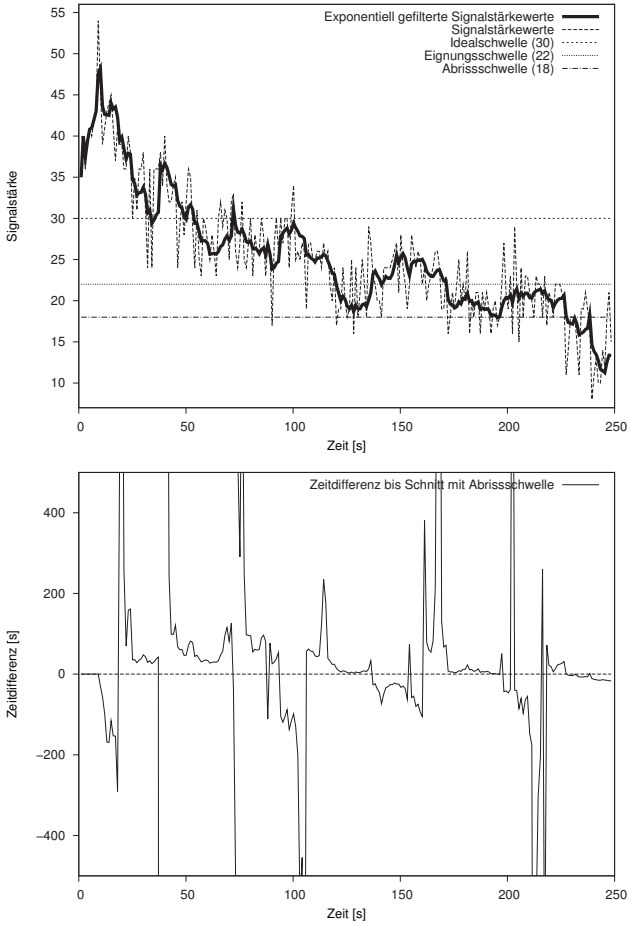

Abbildung D.16: Das obere Diagramm zeigt die exponentiell gefilterten Signalstärkewerte im Vergleich mit den Rohdaten. Das untere Diagramm führt zu jedem Zeitpunkt den zeitlichen Abstand zum errechneten Schnittpunkt zwischen der momentan gültigen Regressionsgeraden und der Abrissschwelle auf. Innerhalb der ersten 10 Sekunden war die Mindestwertemenge für die Regressionsrechnung noch nicht verfügbar, sodass hier noch auf eine Berechnung der Zeitdifferenz verzichtet worden war.

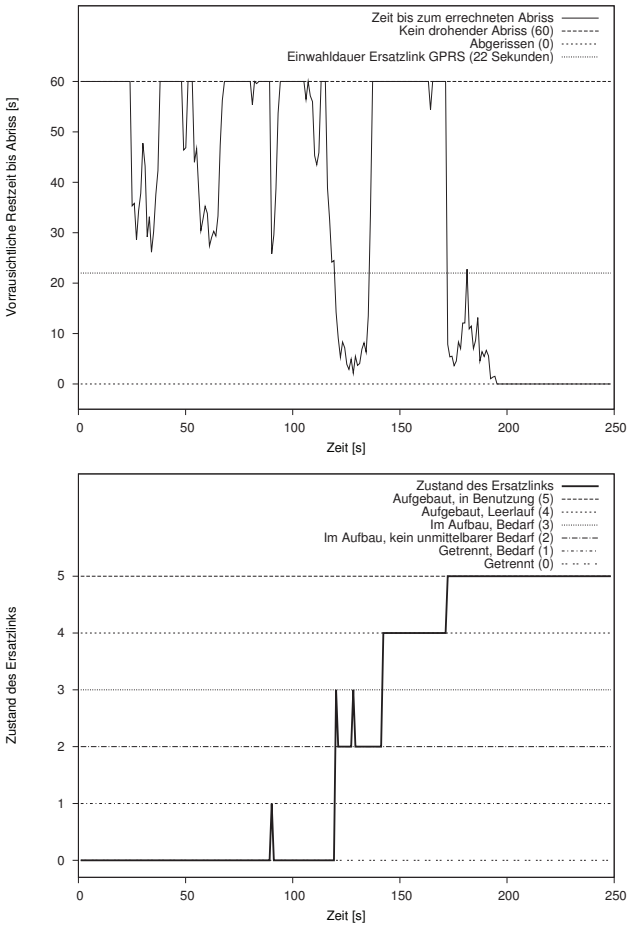

Abbildung D.17: Voraussichtliche Restzeit bis zum Abriss des Hauptlinks (oberes Diagramm) und Ersatzlinknutzung durch den Handoverentscheider (unteres Diagramm)

Messreihe 2, Signalstärkewerte, inversquadratische Regression

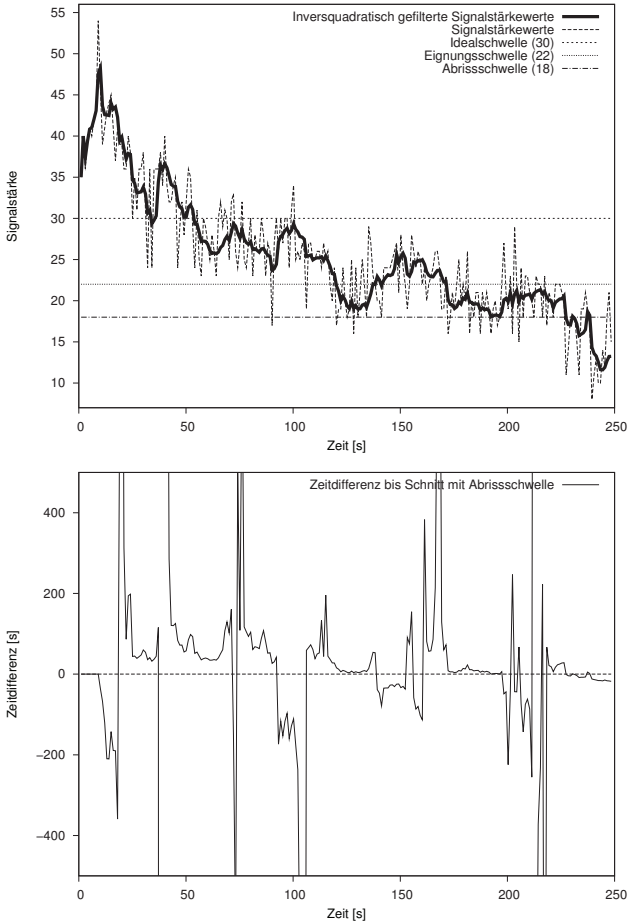

Abbildung D.18: Das obere Diagramm zeigt die inversquadratisch gefilterten Signalstärkewerte im Vergleich mit den Rohdaten. Das untere Diagramm führt zu jedem Zeitpunkt den zeitlichen Abstand zum errechneten Schnittpunkt zwischen der momentan gültigen Regressionsgeraden und der Abrissschwelle auf. Innerhalb der ersten 10 Sekunden war die Mindestwertemenge für die Regressionsrechnung noch nicht verfügbar, sodass hier noch auf eine Berechnung der Zeitdifferenz verzichtet worden war.

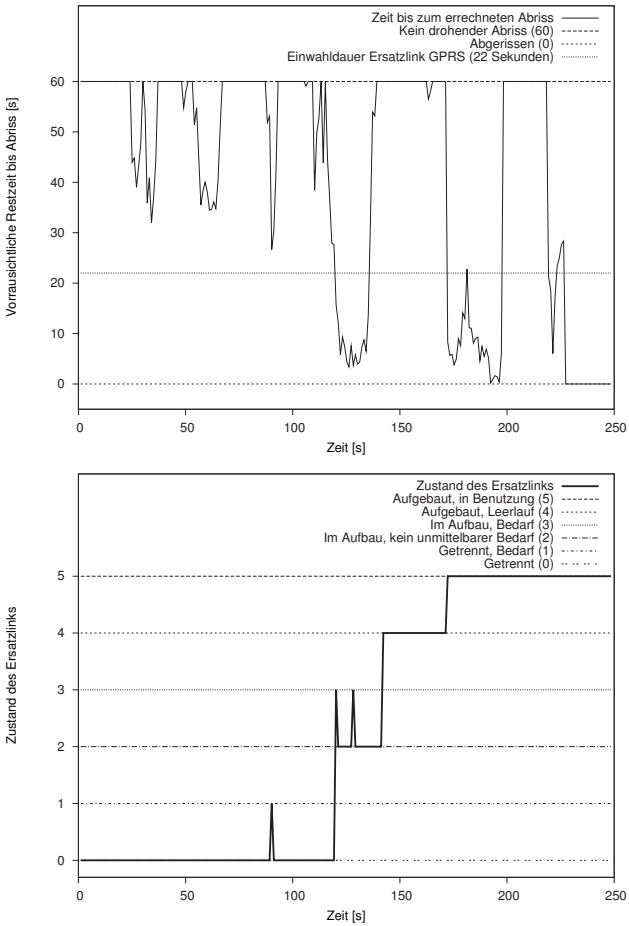

Abbildung D.19: Voraussichtliche Restzeit bis zum Abriss des Hauptlinks (oberes Diagramm) und Ersatzlinknutzung durch den Handoverentscheider (unteres Diagramm)

Messreihe 2, Signalstärkewerte, lineare Regression

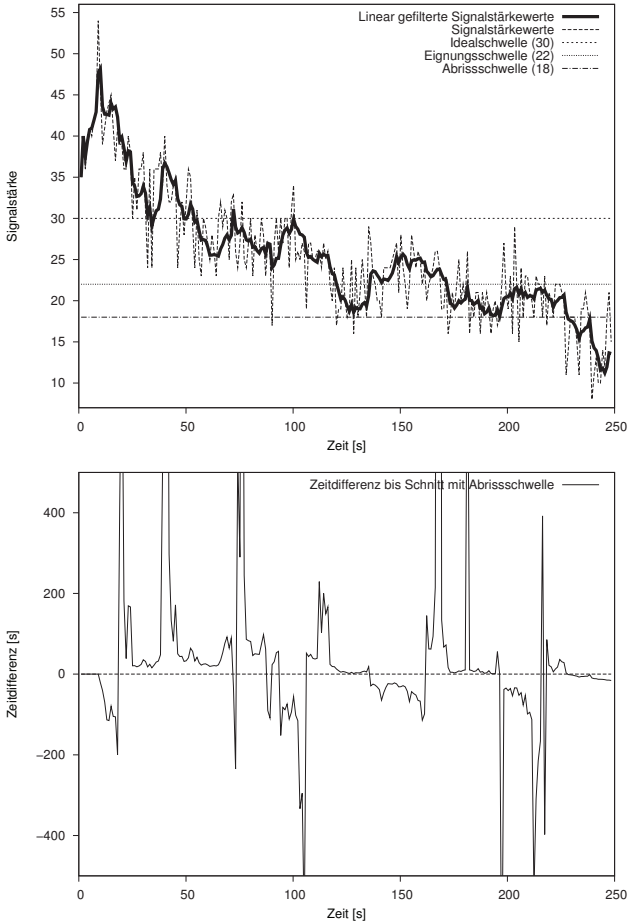

Abbildung D.20: Das obere Diagramm zeigt die linear gefilterten Signalstärkewerte im Vergleich mit den Rohdaten. Das untere Diagramm führt zu jedem Zeitpunkt den zeitlichen Abstand zum errechneten Schnittpunkt zwischen der momentan gültigen Regressionsgeraden und der Abrissschwelle auf. Innerhalb der ersten 10 Sekunden war die Mindestwertemenge für die Regressionsrechnung noch nicht verfügbar, sodass hier noch auf eine Berechnung der Zeitdifferenz verzichtet worden war.

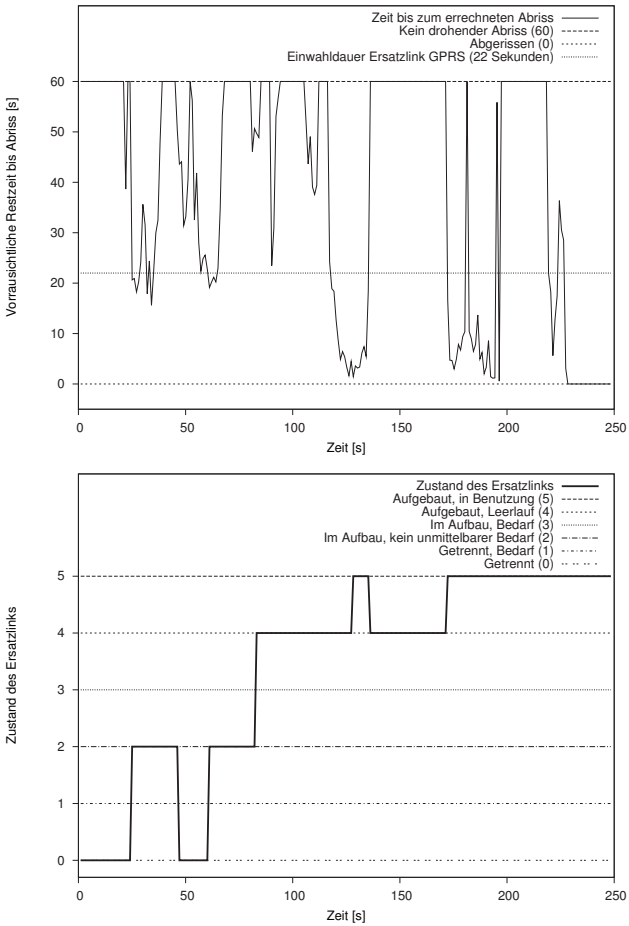

Abbildung D.21: Voraussichtliche Restzeit bis zum Abriss des Hauptlinks (oberes Diagramm) und Ersatzlinknutzung durch den Handoverentscheider (unteres Diagramm)

Verstrichene Zeit [s]	Ereignis Ersatzlink GPRS
25	Assoziiere Ersatzlink
47	Assoziation abgebaut
61	Assoziiere Ersatzlink
83	Assoziation aufgebaut
128	Beginne mit Datenübertragung
136	Stoppe Datenübertragung
172	Beginne mit Datenübertragung

Tabelle D.3: Diese Kommandos und Statusänderungen wurden bezüglich des hier betrachteten Ersatzlinks (GPRS mit einer Einwahlzeit von 22 Sekunden) vom Handoverentscheider abgesetzt bzw. empfangen. Zur Anwendung kam hierbei die lineare Regressionsrechnung.

Diskussion: Auswertung der Signalstärkewerte

Die exponentielle und die inversquadratische Regressionsrechnung lieferten wie in Messreihe 1 (ähnliches Szenario) in Bezug auf die Inanspruchnahme des Ersatzlinks ein vergleichbares Ergebnis. Bei Verwendung der linearen Regressionsrechnung wurde der Ersatzlink jedoch unnötigerweise angefordert (siehe Abbildung D.21 und Tabelle D.3), was an der ausgeprägten „hängenden Nase“ zum Zeitpunkt 35s im Restzeitdiagramm D.21 erkennbar ist. Die lineare Regressionsrechnung scheint den stark fallenden Verlauf zwischen 10s und 30s am getreuesten nachzubilden, was aber auch zu erwarten war, da sowohl die exponentielle als auch die inversquadratische Regressionsrechnung einen „gebogenen“ (konkaven) Verlauf prognostizieren und damit tendenziell spätere Abrisse errechnen.

Alle drei Verfahren lieferten ein typisches und brauchbares Ergebnis: so wurde weder der Ersatzlink für die gesamte Dauer angefordert (wie im Fall eines „gierigen“ Entscheiders), noch wurde der Ersatzlink erst zeitnah zum Zeitpunkt des Abrisses aufgebaut (reaktiver Entscheider). Stattdessen schaffte es der proaktive Entscheider in allen drei Fällen, den Ersatzlink immer „rechtzeitig genug“ aufzubauen. Im Fall der linearen Regressionsrechnung ist jedoch wie in Messreihe 1 ein erhöhter Ressourcenverbrauch erkennbar.

D.2.2 Auswertung des Linkgüteverlaufs

Bei Betrachtung der Linkgütewerte fallen vereinzelte, stark gedämpfte Messwerte auf, welche an „Deep Fading“ erinnern. Allerdings handelt es sich hierbei nicht um Signalstärkewerte (wo „Deep Fading“ zu erwarten wäre) sondern um errechnete Linkgütewerte, in welche eine unbekannte Anzahl an Informationen mit eingeflossen sind.

Dieser stark durch Ausreißer geprägte Verlauf (Abbildung D.22) war mit dafür verantwortlich, dass im Rahmen dieser Ausarbeitung ein besonderes Augenmerk auf eine Ausreißerbehandlung gelegt wurde.

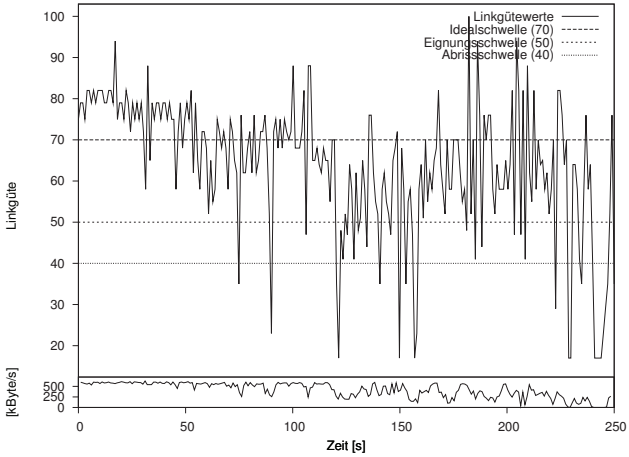

Abbildung D.22: Linkgütewerte und TCP-Datenrate

Messreihe 2, Linkgütewerte, exponentielle Regression

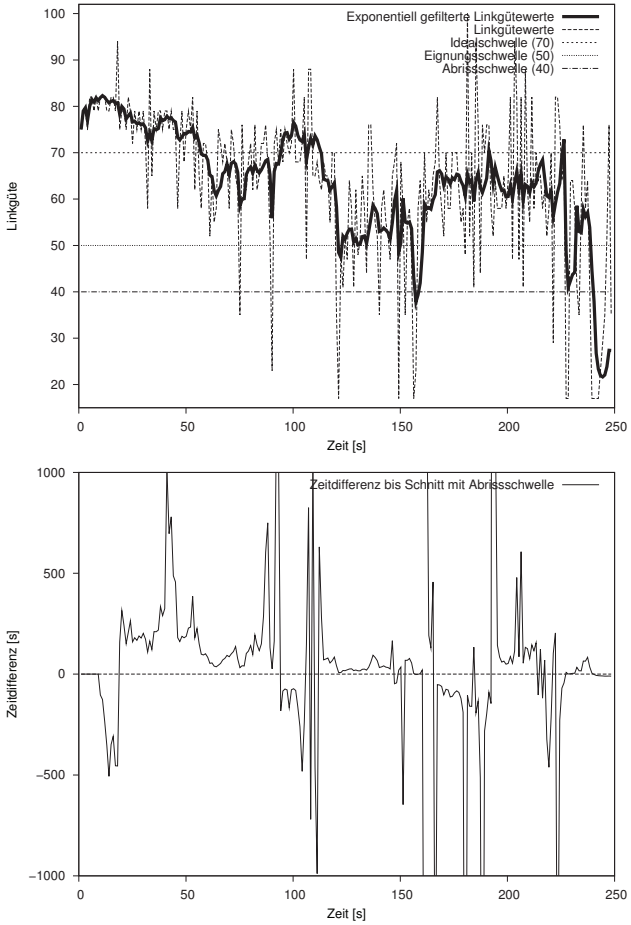

Abbildung D.23: Das obere Diagramm zeigt die exponentiell gefilterten Linkgütewerte im Vergleich mit den Rohdaten. Das untere Diagramm führt zu jedem Zeitpunkt den zeitlichen Abstand zum errechneten Schnittpunkt zwischen der momentan gültigen Regressionsgeraden und der Abrissschwelle auf. Innerhalb der ersten 10 Sekunden war die Mindestwertemenge für die Regressionsrechnung noch nicht verfügbar, sodass hier noch auf eine Berechnung der Zeitdifferenz verzichtet worden war.

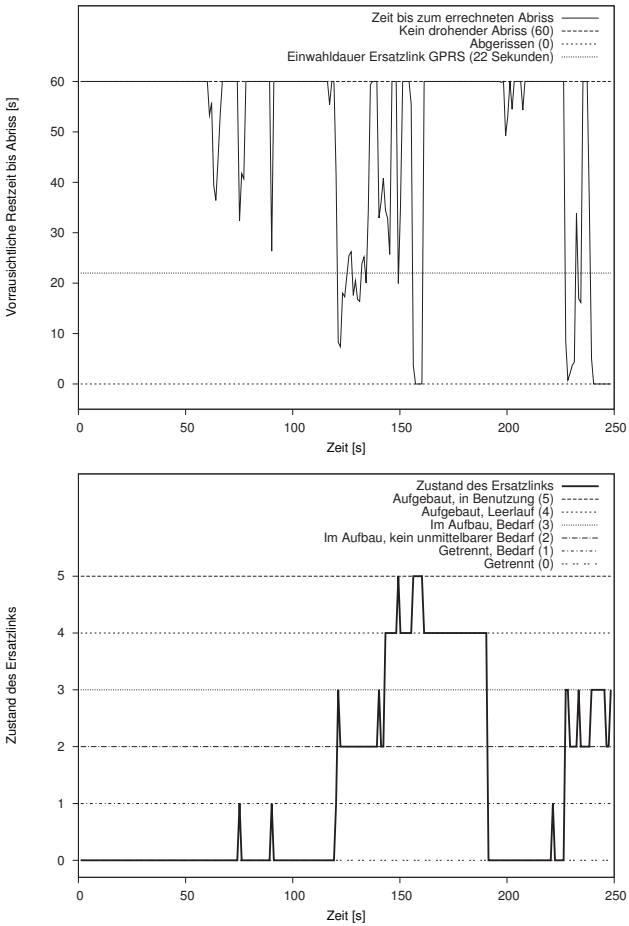

Abbildung D.24: Voraussichtliche Restzeit bis zum Abriss des Hauptlinks (oberes Diagramm) und Ersatzlinknutzung durch den Handoverentscheider (unteres Diagramm)

Messreihe 2, Linkgütewerte, inversquadratische Regression

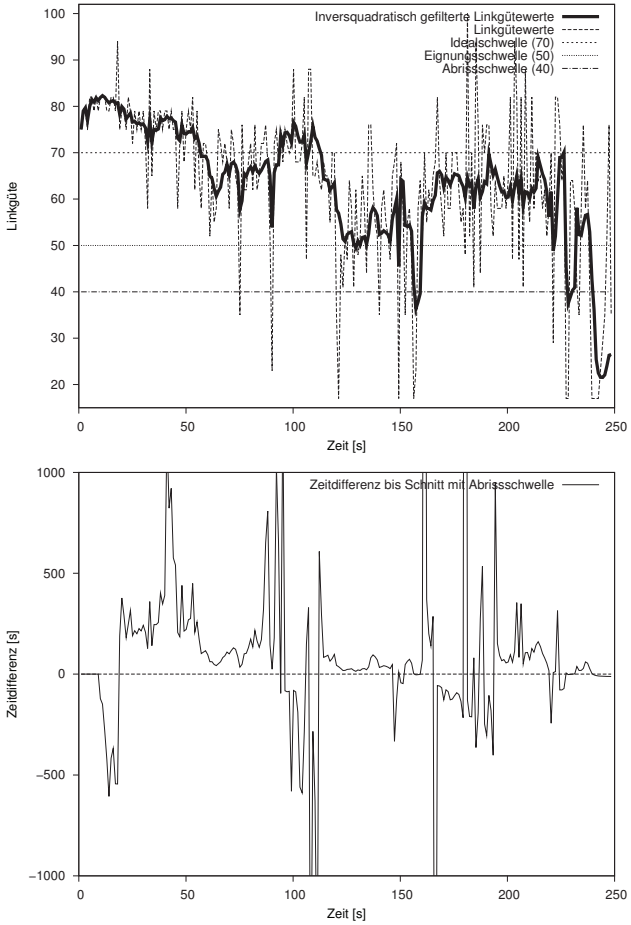

Abbildung D.25: Das obere Diagramm zeigt die inversquadratisch gefilterten Linkgütewerte im Vergleich mit den Rohdaten. Das untere Diagramm führt zu jedem Zeitpunkt den zeitlichen Abstand zum errechneten Schnittpunkt zwischen der momentan gültigen Regressionsgeraden und der Abrissschwelle auf. Innerhalb der ersten 10 Sekunden war die Mindestwertemenge für die Regressionsrechnung noch nicht verfügbar, sodass hier noch auf eine Berechnung der Zeitdifferenz verzichtet worden war.

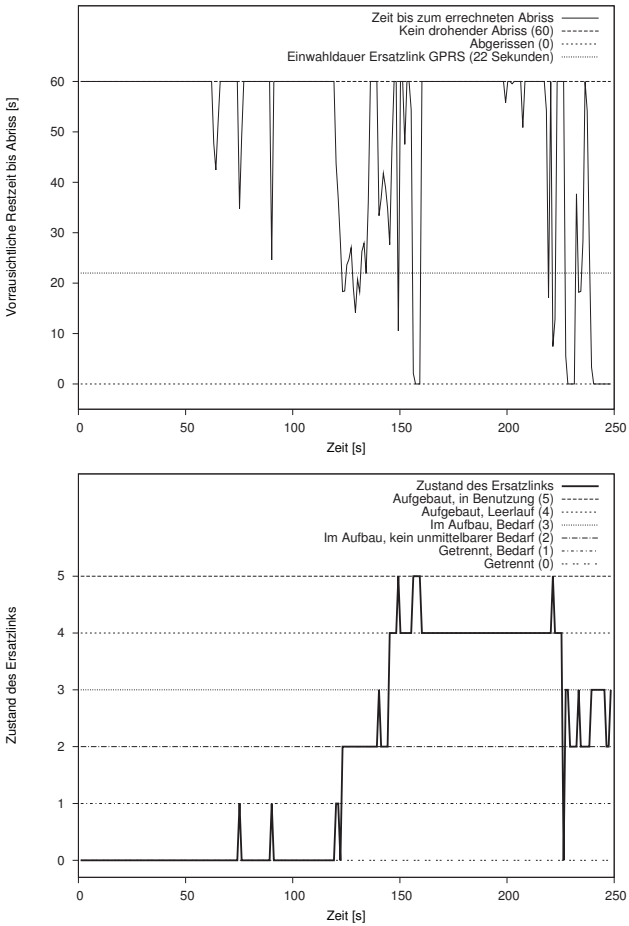

Abbildung D.26: Voraussichtliche Restzeit bis zum Abriss des Hauptlinks (oberes Diagramm) und Ersatzlinknutzung durch den Handoverentscheider (unteres Diagramm)

Messreihe 2, Linkgütewerte, lineare Regression

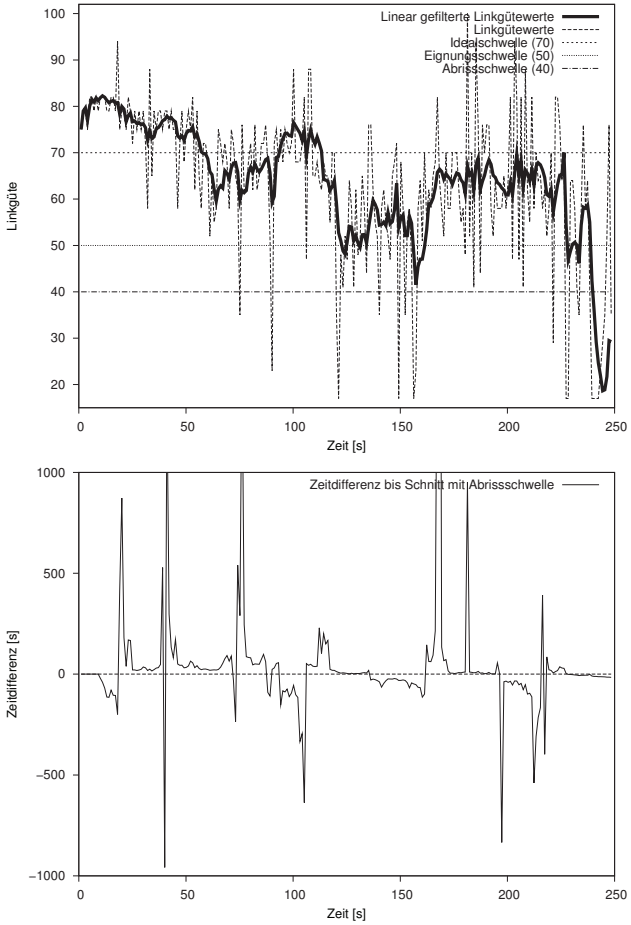

Abbildung D.27: Das obere Diagramm zeigt die linear gefilterten Linkgütewerte im Vergleich mit den Rohdaten. Das untere Diagramm führt zu jedem Zeitpunkt den zeitlichen Abstand zum errechneten Schnittpunkt zwischen der momentan gültigen Regressionsgeraden und der Abrissschwelle auf. Innerhalb der ersten 10 Sekunden war die Mindestwertemenge für die Regressionsrechnung noch nicht verfügbar, sodass hier noch auf eine Berechnung der Zeitdifferenz verzichtet worden war.

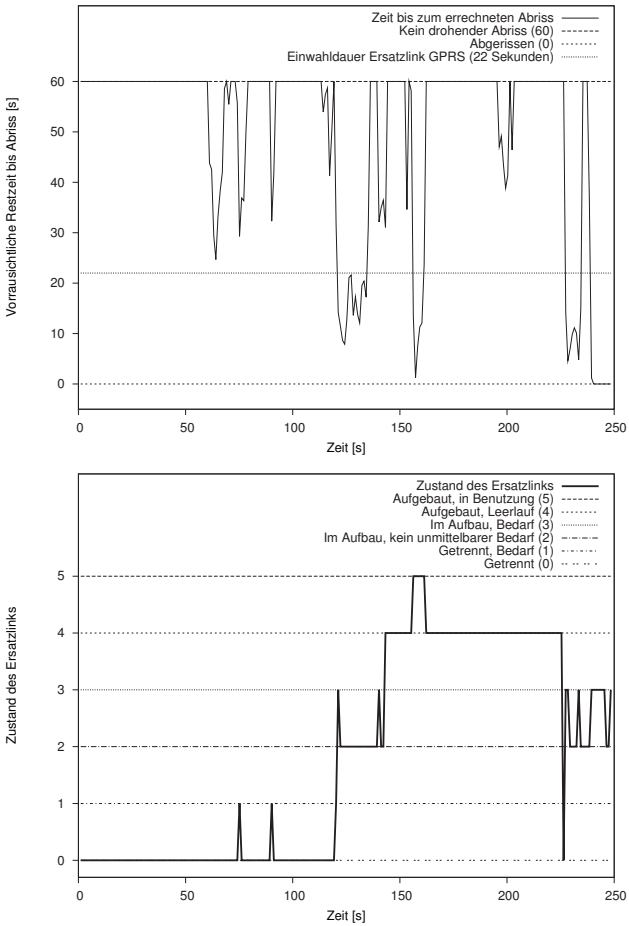

Abbildung D.28: Voraussichtliche Restzeit bis zum Abriss des Hauptlinks (oberes Diagramm) und Ersatzlinknutzung durch den Handoverentscheider (unteres Diagramm)

Verstrichene Zeit [s]	Ereignis Ersatzlink GPRS
121	Assoziiere Ersatzlink
143	Assoziation aufgebaut
156	Beginne mit Datenübertragung
162	Stoppe Datenübertragung
226	Assoziation abgebaut
227	Assoziiere Ersatzlink

Tabelle D.4: Diese Kommandos und Statusänderungen wurden bezüglich des hier betrachteten Ersatzlinks (GPRS mit einer Einwahlzeit von 22 Sekunden) vom Handoverentscheider abgesetzt bzw. empfangen. Zur Anwendung kam hierbei die lineare Regressionsrechnung.

Diskussion: Auswertung der Linkgütewerte

Im Bereich von 0s bis 190s zeigen alle drei Berechnungsverfahren ein vergleichbares Ergebnis: das Restzeitdiagramm fiel in allen drei Fällen bei ca. 120s unter die Marke von 22s, was zu einem Aufbau des Ersatzlinks führte. Ab ca. 190s werden allerdings Unterschiede deutlich.

Interessanterweise wurde bei der exponentiellen Regressionsrechnung der Ersatzlink zum Zeitpunkt 195s getrennt (unteres Diagramm in Abbildung D.24). Es stellte sich heraus, dass dies eine Fehlentscheidung war, da es sich nur um eine kurzfristige „Erholung“ handelte, während der langfristige Trend weiter in Richtung eines Abrisses verlief. Ursache dieses Verhaltens war eine augenscheinlich zu niedrig angesetzte Idealschwelle, da diese vom gefilterten Wert überschritten wurde, was schließlich zum Abbau des Ersatzlinks führte. Allerdings ergäbe hierbei eine weitere Anhebung der Idealschwelle keinen Sinn, da diese im Vergleich zu den Linkgütewerten „im guten Bereich“ (Abbildung D.22, erste 50 Sekunden) bereits hoch angesetzt war. Scheinbar sind die Verläufe von Linkgütewerten weniger gut geeignet als die Verläufe von Signalstärkewerten, da bei Linkgütewerten eine Vielzahl an Parametern mit eingerechnet werden, und sich somit ein vergleichsweise „unruhiger“ Verlauf ergibt. Daher würde sich der Autor nach Sichtung der Messreihen 1 und 2 für eine zukünftige Auswertung von Signalstärkewerten entscheiden. Diese Entscheidung musste jedoch nicht aktiv getroffen werden, da beginnend ab Messreihe 3 vom Treiber des verwendeten WLAN-Chipsatzes keine Linkgütewerte mehr geliefert wurden und sich die Wahl damit erübrigt hatte.

D.3 Messreihe 3

Szenario

Die Basisstation befand sich wiederum im Obergeschoss des Helmholtzbaus auf der äußeren Fensterbank des Raums H3501. Die Messung (siehe Abbildung D.29) begann auf dem Fußweg nahe der Hausecke unterhalb der Basisstation. Der aufzeichnende Laptop wurde mit betont langsamer Schrittgeschwindigkeit in Richtung des Newtonbaus bewegt. Vor der Treppe wurde links abgebogen, und der Träger bewegte sich links entlang des Newtonbaus in Richtung des Antennenlabors. Dabei wurde die Basisstation durch die Hausecke des Helmholtzbaus abgeschattet, sodass sich die Signalstärke stetig verschlechterte bis schlussendlich kein Datentransfer mehr festgestellt werden konnte.

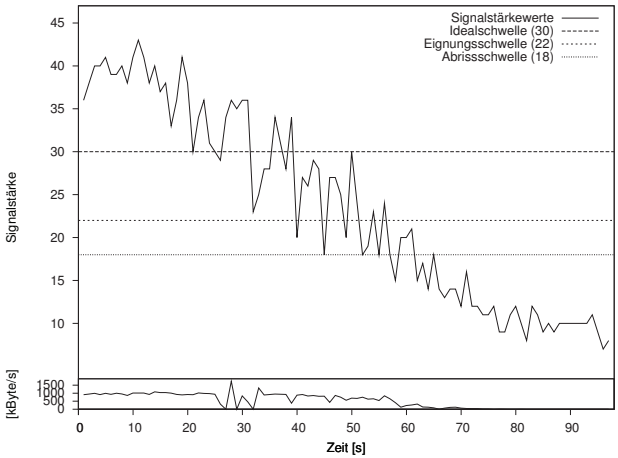

Abbildung D.29: Signalstärkewerte und TCP-Datenrate

Es wurde immer ein Signalstärkewert pro Sekunde ermittelt. Die Fenstergröße betrug 30 Sekunden (30 Messwerte) bei einem minimalen Füllstand von 10 Sekunden (10 Messwerte). Die gezeigten Diagramme beruhen durchgängig auf linearer Regressionsrechnung.

Messreihe 3, Signalstärkewerte, lineare Regression

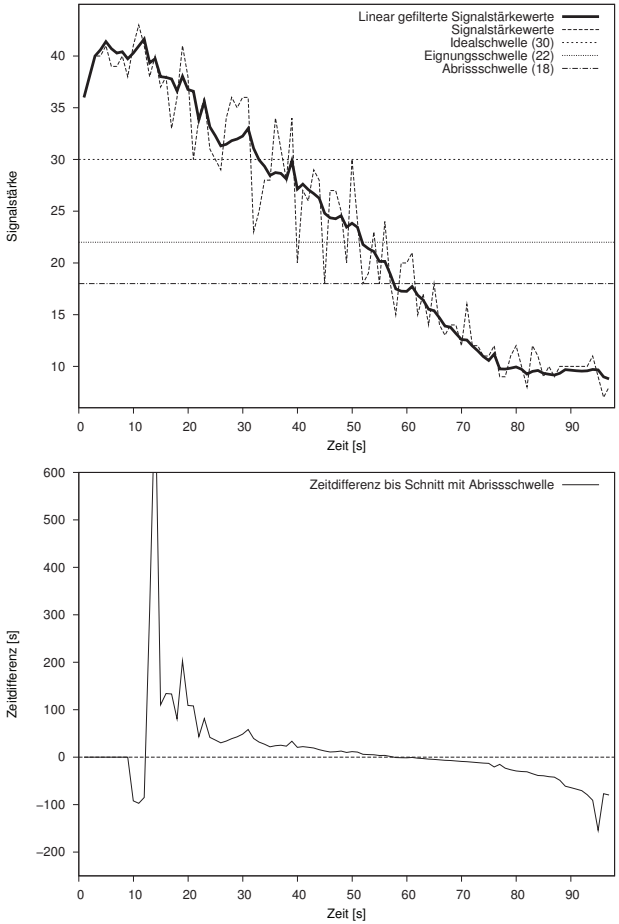

Abbildung D.30: Das obere Diagramm zeigt die linear gefilterten Signalstärkewerte im Vergleich mit den Rohdaten. Das untere Diagramm führt zu jedem Zeitpunkt den zeitlichen Abstand zum errechneten Schnittpunkt zwischen der momentan gültigen Regressionsgeraden und der Abrissschwelle auf. Innerhalb der ersten 10 Sekunden war die Mindestwertemenge für die Regressionsrechnung noch nicht verfügbar, sodass hier noch auf eine Berechnung der Zeitdifferenz verzichtet worden war.

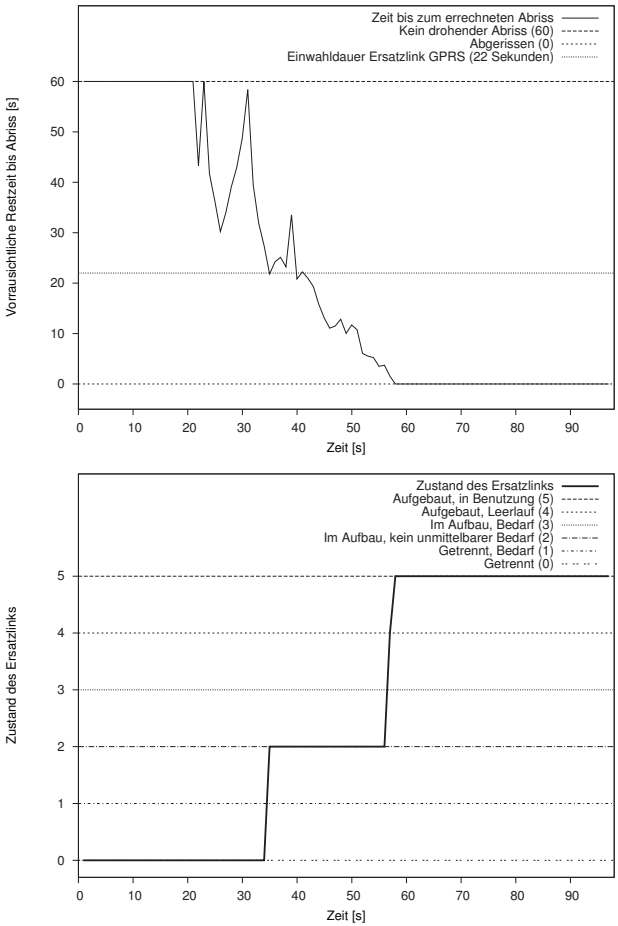

Abbildung D.31: Voraussichtliche Restzeit bis zum Abriss des Hauptlinks (oberes Diagramm) und Ersatzlinknutzung durch den Handoverentscheider (unteres Diagramm)

Verstrichene Zeit [s]	Ereignis Ersatzlink GPRS
35	Assoziiere Ersatzlink
57	Assoziation aufgebaut
58	Beginne mit Datenübertragung

Tabelle D.5: Diese Kommandos und Statusänderungen wurden bezüglich des hier betrachteten Ersatzlinks (GPRS mit einer Einwahlzeit von 22 Sekunden) vom Handoverentscheider abgesetzt bzw. empfangen.

Diskussion

In diesem Szenario befand sich das mobile Endgerät anfangs in unmittelbarer Nähe zur Basisstation, und entfernte sich dann von ihr. Die linear gefilterten Signalstärkewerte verliefen beinahe entlang einer Geraden (oberes Diagramm in Abbildung D.30), was ein gutes Ergebnis der linearen Regressionsrechnung vermuten ließ. Das untere Diagramm zeigt einen ansehnlichen Verlauf mit anfänglich positiver Restzeit, welche stetig abnahm und etwa zu Zeitpunkt 58s einen Nulldurchgang (Abriss) erfuhr.

Im untere Diagramm von Abbildung D.31 wurde die anfängliche Vermutung durch eine ideale Nutzung des Ersatzlinks bestätigt. Bereits beim ersten „schlechten“ Signalstärkewert (die nach unten gerichtete Spitze zum Zeitpunkt 58s) stand der aufgebaute Ersatzlink zur Verfügung (siehe Tabelle D.5).

Die im Bereich 40s bis 60s näherungsweise linear abfallende Restzeit im unteren Diagramm von Abbildung D.31 zeigt, dass die Prognose auch für andere Ersatzlinktechnologien mit geringeren Assoziationszeiten (hier: 22 Sekunden) brauchbare Ergebnisse liefern würde.

D.4 Messreihe 4

Szenario

Diese Messreihe (Abbildung D.32) begann am Newtonbau auf Höhe des gegenüber liegenden Eingangs zum „großen Hörsaal“. Der Nutzer bewegte sich in Richtung Humboldtbau auf den Kirchhoffbau zu, nahm dabei die kleinere Treppe am Newtonbau (30s) und passierte anschließend die Fahrradständer unterhalb des Kirchhoffbaus (75s). Hinter diesen machte er kehrt (105s), nahm aber den Rückweg in Richtung Helmholtzbau oberhalb der Fahrradständer. Er ging die Treppe herauf (150s), und bog links herum in Richtung des Brunnens vor dem Kirchhoffbau ab (190s). Beim Passieren der Ecke des Helmholtzbaus war der Abstand zur Basisstation am geringsten (170s). Die Fortbewegung erfolgte betont langsam.

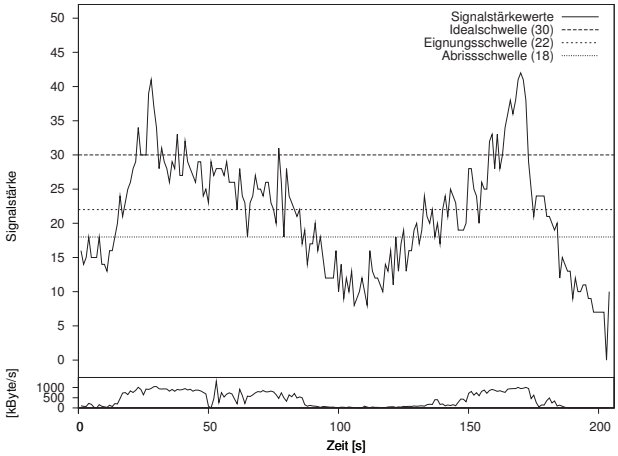

Abbildung D.32: Signalstärkewerte und TCP-Datenrate

Es wurde immer ein Signalstärkewert pro Sekunde ermittelt. Die Fenstergröße betrug 30 Sekunden (30 Messwerte) bei einem minimalen Füllstand von 10 Sekunden (10 Messwerte). Die gezeigten Diagramme beruhen durchgängig auf linearer Regressionsrechnung.

Messreihe 4, Signalstärkewerte, lineare Regression

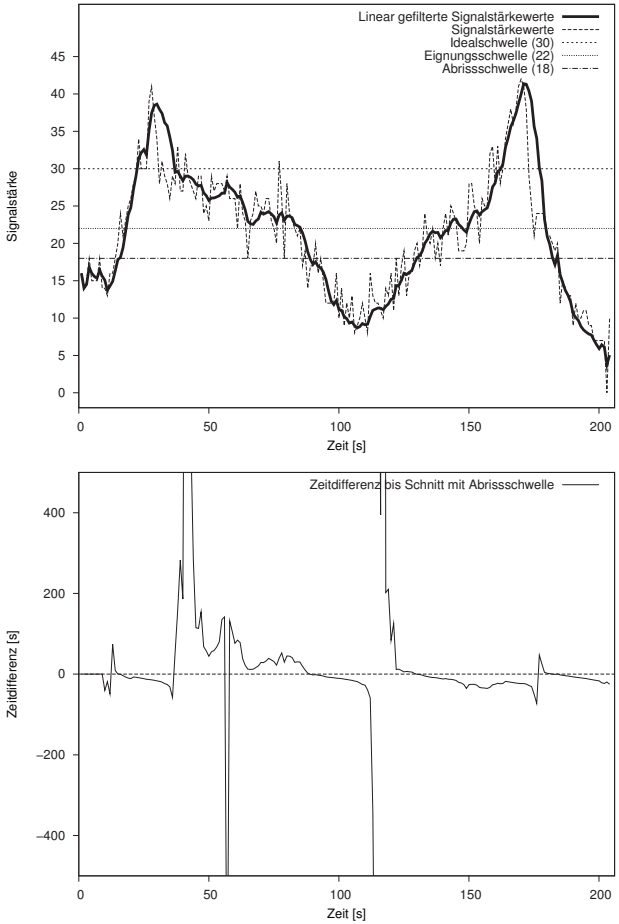

Abbildung D.33: Das obere Diagramm zeigt die linear gefilterten Signalstärkewerte im Vergleich mit den Rohdaten. Das untere Diagramm führt zu jedem Zeitpunkt den zeitlichen Abstand zum errechneten Schnittpunkt zwischen der momentan gültigen Regressionsgeraden und der Abrissschwelle auf. Innerhalb der ersten 10 Sekunden war die Mindestwertemenge für die Regressionsrechnung noch nicht verfügbar, sodass hier noch auf eine Berechnung der Zeitdifferenz verzichtet worden war.

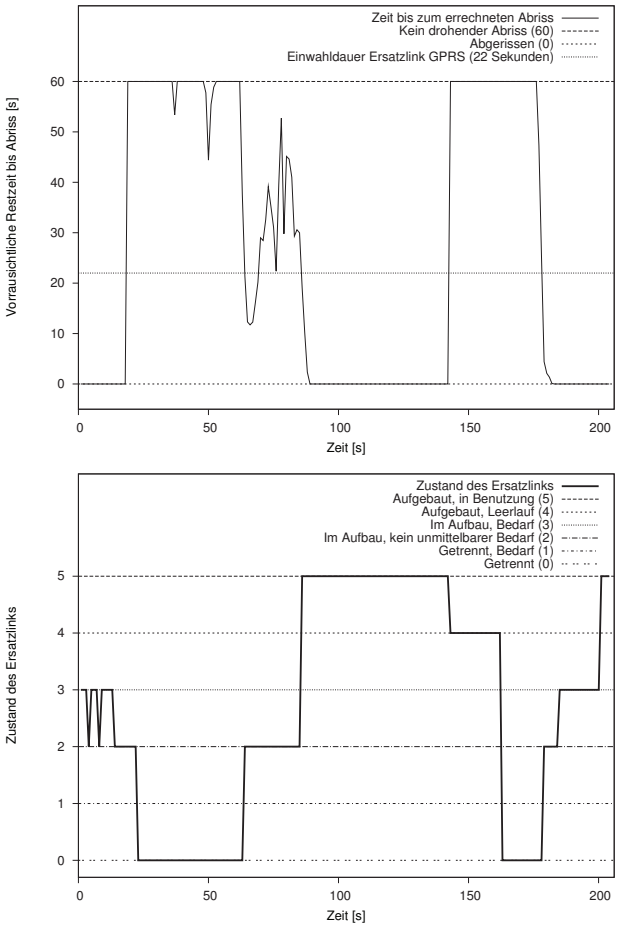

Abbildung D.34: Voraussichtliche Restzeit bis zum Abriss des Hauptlinks (oberes Diagramm) und Ersatzlinknutzung durch den Handoverentscheider (unteres Diagramm)

Verstrichene Zeit [s]	Ereignis Ersatzlink GPRS
1	Assoziiere Ersatzlink
23	Assoziation abgebaut
64	Assoziiere Ersatzlink
86	Assoziation aufgebaut
86	Beginne mit Datenübertragung
143	Stoppe Datenübertragung
163	Assoziation abgebaut
179	Assoziiere Ersatzlink
201	Assoziation aufgebaut
201	Beginne mit Datenübertragung

Tabelle D.6: Diese Kommandos und Statusänderungen wurden bezüglich des hier betrachteten Ersatzlinks (GPRS mit einer Einwahlzeit von 22 Sekunden) vom Handoverentscheider abgesetzt bzw. empfangen.

Diskussion

Das vorliegende Szenario zeichnet sich durch eine Vielzahl an unterschiedlichen Zeitabschnitten aus. Anfangs, als der Entscheider noch auf Grund der nicht ausreichend verfügbaren Messwerte reaktiv arbeitete, wurde der Ersatzlink angefordert (gut erkennbar im unteren Diagramm von Abbildung D.34). Zum Zeitpunkt 23s (aus Tabelle D.6 entnommen) stand der Ersatzlink bereit; er wurde allerdings nicht mehr benötigt und gleich wieder getrennt. Sein Aufbau entpuppte sich rückwirkend als Fehlentscheidung. Diese war allerdings unvermeidbar und begründet sich aus den anfänglichen „schlechten“ Signalstärkewerten.

Zum Zeitpunkt 64s unterschritt die prognostizierte Restzeit erstmals die Marke von 22 Sekunden (oberes Diagramm aus Abbildung D.34), was allerdings nur knapp erfolgte, da sich die Prognose im Folgenden wieder in Richtung eines späteren Abrisszeitpunkts korrigierte. Dies reichte jedoch aus, um die Assoziation des Ersatzlinks anzustoßen, und genau zum Zeitpunkt 86s, als der Ersatzlink verfügbar war, wurde der Hauptlink schließlich unbrauchbar. Der Anforderungszeitpunkt erwies sich als ideal.

Der zweite „Höcker“ zum Zeitpunkt 170s lag oberhalb der Idealschwelle, sodass der Ersatzlink wieder abgebaut wurde. Das führte beim anschließenden raschen Abfall zu einer baldigen Neuassoziation, welche allerdings zu lange benötigte, um rechtzeitig vor dem folgenden Abrissereignis abgeschlossen zu sein. Dies wird durch die sehr rasch fallende Abrissprognose bei 180s im unteren Diagramm aus Abbildung D.34 deutlich.

Es wurde zum Zeitpunkt 64s ein idealer Assoziationszeitpunkt gefunden, allerdings standen ab ca. 180s weder Haupt-, noch Ersatzlink zur Verfügung.

D.5 Messreihe 5

Szenario

Diese Messreihe (siehe Abbildung D.35) begann neben dem Newtonbau, auf Seite den Helmholtzbaus in Höhe des „großen Hörsaals“. Die Marschroute ging in Richtung des Humboldtbaus, sodass etwa zum Zeitpunkt 20s die Basisstation in Sichtweite geriet. Der Laptop wurde weiter in Richtung der Fahrradständer unterhalb des Kirchhoffbaus getragen, welche unterhalb in Richtung Humboldtbau passiert wurden (etwa zum Zeitpunkt 60s). Nachdem die Fahrradständer vollständig passiert waren (90s), ging der Träger auf die dortige Eingangstür des Kirchhoffbaus zu. Diese wurde zum Zeitpunkt 105s geöffnet und durchschritten, wonach die Funkverbindung komplett abriss.

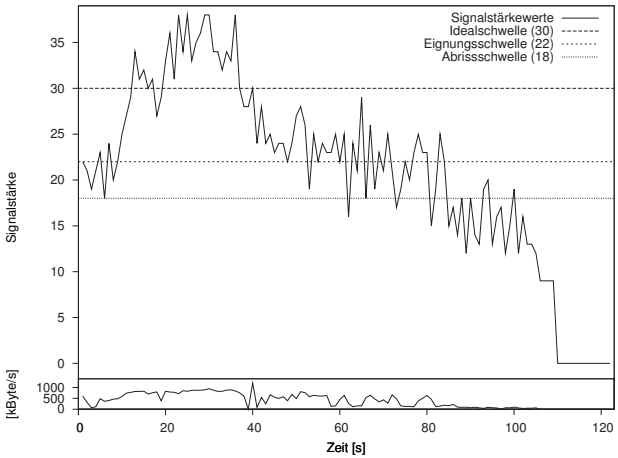

Abbildung D.35: Signalstärkewerte und TCP-Datenrate

Es wurde immer ein Signalstärkewert pro Sekunde ermittelt. Die Fenstergröße betrug 30 Sekunden (30 Messwerte) bei einem minimalen Füllstand von 10 Sekunden (10 Messwerte). Die gezeigten Diagramme beruhen durchgängig auf linearer Regressionsrechnung.

Messreihe 5, Signalstärkewerte, lineare Regression

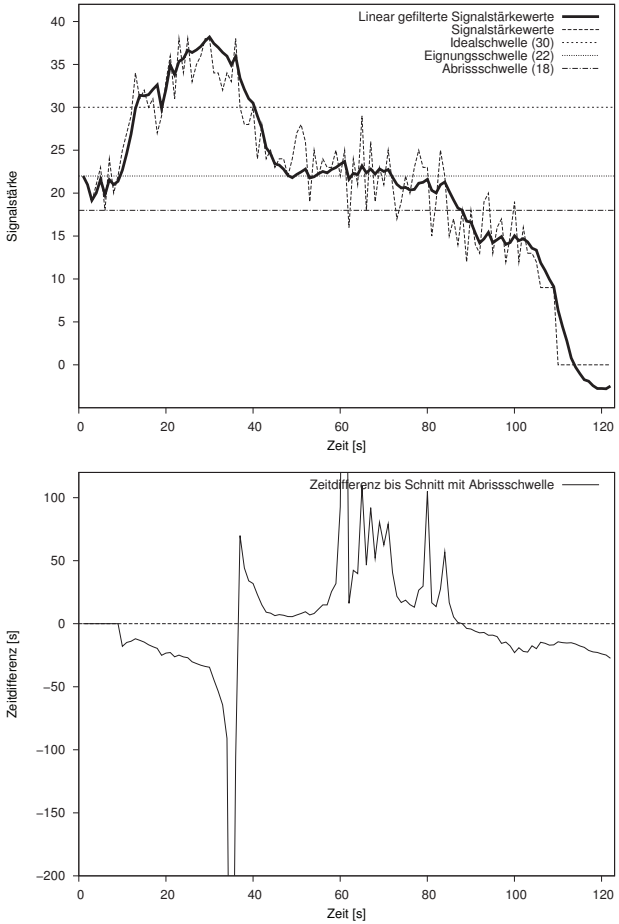

Abbildung D.36: Das obere Diagramm zeigt die linear gefilterten Signalstärkewerte im Vergleich mit den Rohdaten. Das untere Diagramm führt zu jedem Zeitpunkt den zeitlichen Abstand zum errechneten Schnittpunkt zwischen der momentan gültigen Regressionsgeraden und der Abrissschwelle auf. Innerhalb der ersten 10 Sekunden war die Mindestwertemenge für die Regressionsrechnung noch nicht verfügbar, sodass hier noch auf eine Berechnung der Zeitdifferenz verzichtet worden war.

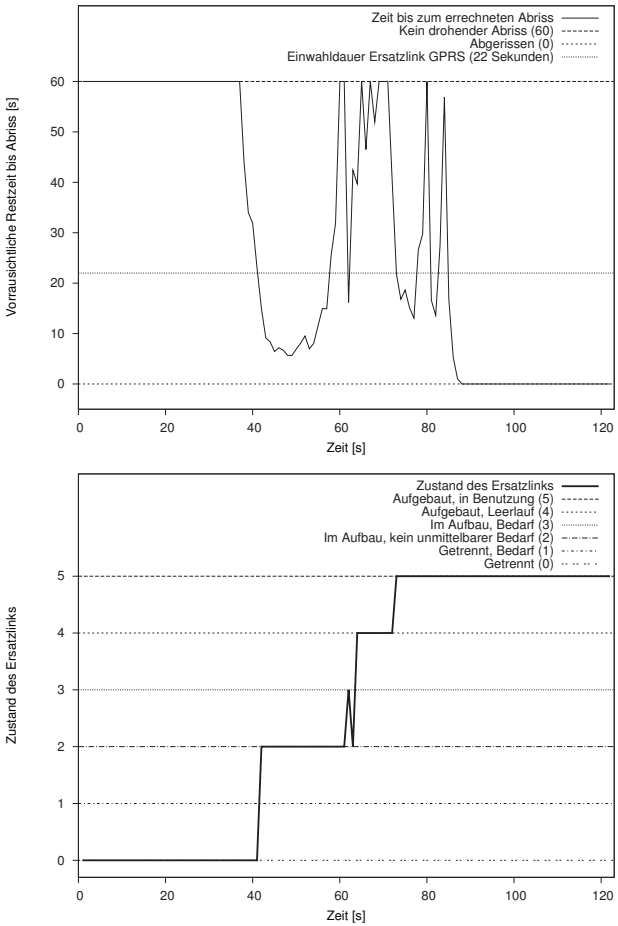

Abbildung D.37: Voraussichtliche Restzeit bis zum Abriss des Hauptlinks (oberes Diagramm) und Ersatzlinknutzung durch den Handoverentscheider (unteres Diagramm)

Verstrichene Zeit [s]	Ereignis Ersatzlink GPRS
42	Assoziiere Ersatzlink
64	Assoziation aufgebaut
73	Beginne mit Datenübertragung

Tabelle D.7: Diese Kommandos und Statusänderungen wurden bezüglich des hier betrachteten Ersatzlinks (GPRS mit einer Einwahlzeit von 22 Sekunden) vom Handoverentscheider abgesetzt bzw. empfangen.

Diskussion

Der aufgenommene Signalstärkeverlauf ist durch einen anfänglichen Anstieg und ab Zeitpunkt 30s von einem längerfristigen Abfall geprägt. Zum Zeitpunkt 35s trat im Zeitdifferenzdiagramm eine Polstelle auf, da hier die Regressionsgerade „umklappte“ und kurzzeitig horizontal verlief. Im Zeitraum von 50s bis 80s formten die Signalstärkewerte eine Art Sattelpunkt, welcher im Restzeitdiagramm ab Zeitpunkt 60s zu einem starken Anstieg führte. Die Abrissprognose korrigierte sich damit auf einen späteren Zeitpunkt, was allerdings keine Auswirkung auf die Nutzung des Ersatzlinks hatte, da dessen Assoziation bereits zum Zeitpunkt 42s abgestoßen worden war. Nachdem der Ersatzlink verfügbar wurde, bleibt er noch für ca. 9 Sekunden ungenutzt, bis der Handoverentscheider schließlich seine Nutzung anordnete. Er wurde bis zum Ende der Messung involviert, da sich der Hauptlink bis hin zum vollständigen Abriss weiter verschlechterte.

Zum Zeitpunkt 110s gab es einen interessanten Effekt zu beobachten, welcher auf die verwendete lineare Regressionsrechnung zurückzuführen ist. Obwohl nur positive Signalstärkewerte „erlaubt“ sind, rutschte der linear interpolierte Signalstärkewerte in den negativen Bereich ab (siehe Abbildung D.36). Bei der linearen Regressionsrechnung ist das jedoch ein normales Verhalten, sodass hier noch eine anschließende Begrenzung des Wertebereiches hätte stattfinden müssen. Sowohl die exponentielle als auch die inversquadratische Regressionsrechnung weisen diesen Effekt jedoch nicht auf, da die hierbei verwendeten Rücktransformationen nur positive Werte liefern können.

D.6 Messreihe 6

Szenario

Diese Messreihe (Abbildung D.38) begann am hinteren Ende des Parkplatzes hinter dem Humboldtbau, neben den Fenstern der Seminarräume und damit ohne Sichtverbindung zur Basisstation. Der Laptop wurde mit sehr langsamer Schrittgeschwindigkeit in Richtung der Basisstation getragen, also quer über den Platz in Richtung des Brunnens vor dem Kirchhoffbau. Bei ca. 120s wurde die Hausecke mit der Basisstation passiert, wobei es zu einer sofortigen Abschattung kam. Am Eingang des Helmholtzbaus wurde die Messung unterbrochen.

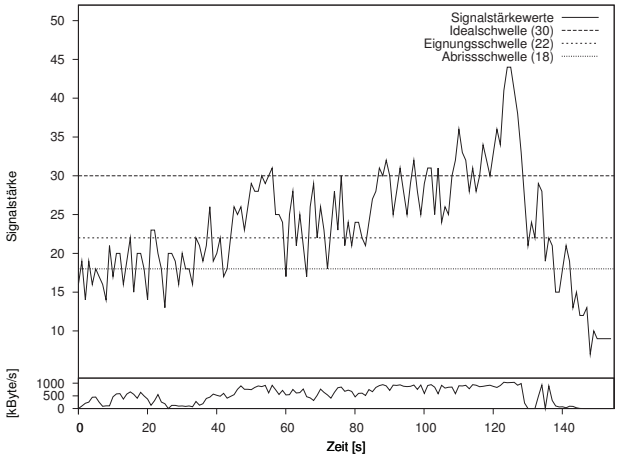

Abbildung D.38: Signalstärkewerte und TCP-Datenrate

Es wurde immer ein Signalstärkewert pro Sekunde ermittelt. Die Fenstergröße betrug 30 Sekunden (30 Messwerte) bei einem minimalen Füllstand von 10 Sekunden (10 Messwerte). Die gezeigten Diagramme beruhen durchgängig auf linearer Regressionsrechnung.

Messreihe 6, Signalstärkewerte, lineare Regression

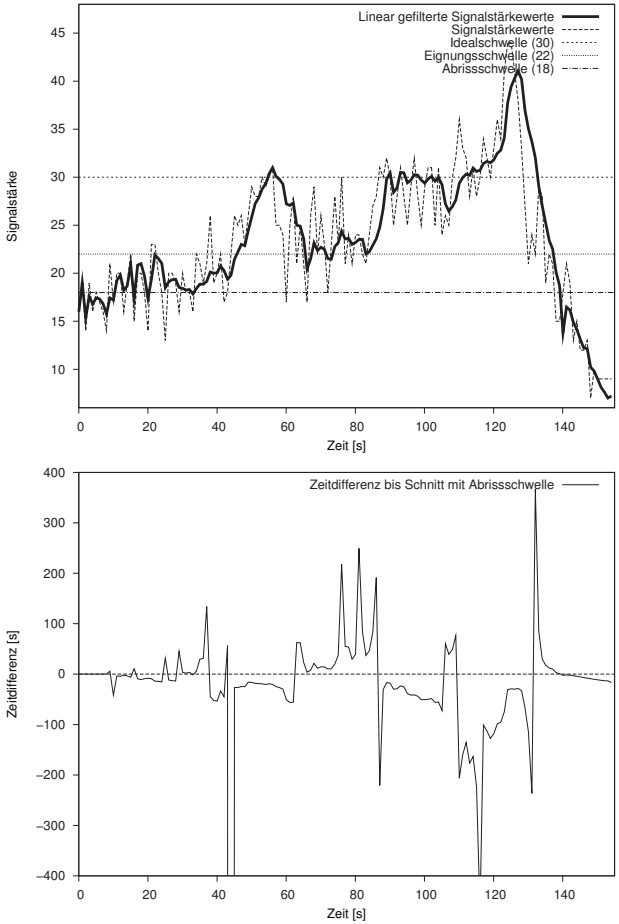

Abbildung D.39: Das obere Diagramm zeigt die linear gefilterten Signalstärkewerte im Vergleich mit den Rohdaten. Das untere Diagramm führt zu jedem Zeitpunkt den zeitlichen Abstand zum errechneten Schnittpunkt zwischen der momentan gültigen Regressionsgeraden und der Abrissschwelle auf. Innerhalb der ersten 10 Sekunden war die Mindestwertemenge für die Regressionsrechnung noch nicht verfügbar, sodass hier noch auf eine Berechnung der Zeitdifferenz verzichtet worden war.

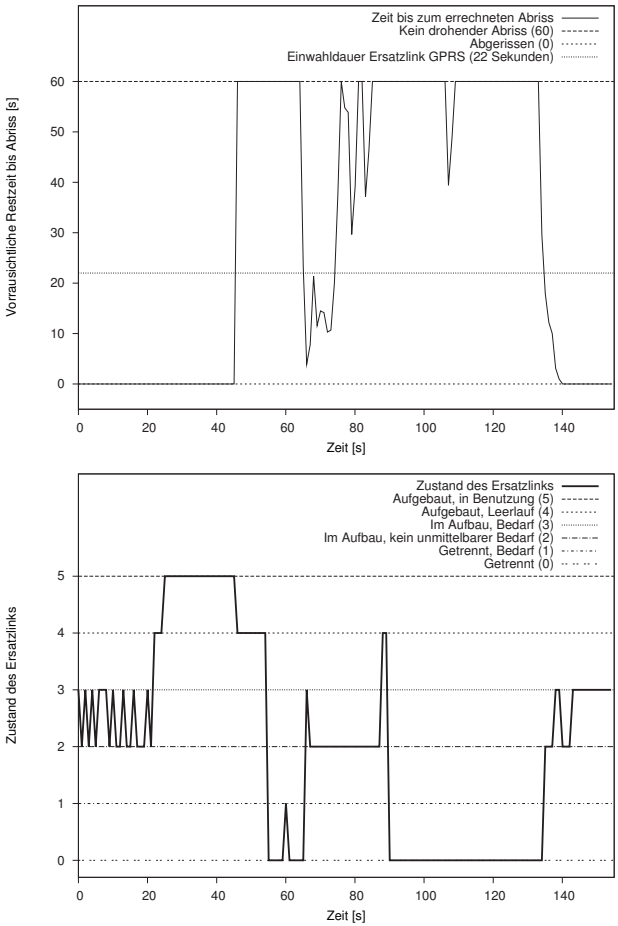

Abbildung D.40: Voraussichtliche Restzeit bis zum Abriss des Hauptlinks (oberes Diagramm) und Ersatzlinknutzung durch den Handoverentscheider (unteres Diagramm)

Verstrichene Zeit [s]	Ereignis Ersatzlink GPRS
0	Assoziiere Ersatzlink
22	Assoziation aufgebaut
25	Beginne mit Datenübertragung
46	Stoppe Datenübertragung
55	Assoziation abgebaut
66	Assoziiere Ersatzlink
88	Assoziation aufgebaut
90	Assoziation abgebaut
135	Assoziiere Ersatzlink

Tabelle D.8: Diese Kommandos und Statusänderungen wurden bezüglich des hier betrachteten Ersatzlinks (GPRS mit einer Einwahlzeit von 22 Sekunden) vom Handoverentscheider abgesetzt bzw. empfangen.

Diskussion

Dieser Signalstärkeverlauf ist von einer stetigen Verbesserung geprägt, welche sich jedoch über einen längeren Zeitraum erstreckt und zudem von diversen mehrsekündlich anhaltenden Tendenzen durchsetzt ist. Die Signalstärke befand sich anfangs unterhalb der Abrissschwelle, sodass eine sofortige Assoziation des Ersatzlinks angeordnet worden war. Diese Entscheidung erwies sich trotz des tendenziell ansteigenden Verlaufs als korrekt, sodass der aufgebaute Ersatzlink ab Zeitpunkt 25s noch für ganze 21 Sekunden benutzt werden konnte. Bei Zeitpunkt 55s durchschritt der linear gefilterte Signalstärkewert die Idealschwelle, was zum Abbau des Ersatzlinks führte. Der anschließende steile Abfall sorgte jedoch wieder für eine Abrissprognose in der nahen Zukunft, sodass der Ersatzlink zu Zeitpunkt 66s wieder angefordert worden war. Diese Entscheidung entpuppte sich jedoch als Fehlentscheidung, da sich der abfallende Trend zeitnah wieder umkehrte. Der Ersatzlink wurde lediglich für zwei Sekunden benutzt.

Als bei Zeitpunkt 125s die Häuserecke passiert wurde und sich die Signalstärkewerte rapide verschlechterten, kam die Regressionsrechnung nicht schnell genug hinterher. Erst ca. 10 Sekunden später „zog“ die Prognose entsprechend „nach“, und der Ersatzlink wurde zu Zeitpunkt 135s angefordert. Nur ein paar Sekunden später riss der Hauptlink jedoch bereits ab, sodass an dieser Stelle kein signifikanter Zeitvorteil gegenüber einem reaktiven Verhalten feststellbar war. Ein solcher Verlauf kann mit einem 30 Sekunden langen Mittelungsfenster sowie einem Ersatzlink mit vergleichsweise langer Assoziationsdauer nicht vernünftig behandelt werden.

D.7 Messreihe 7

Szenario

Bei diesem „Indoor“-Szenario (Abbildung D.41) stand der Laptop anfangs auf dem Schreibtisch seines Besitzers, gleich neben der Basisstation. Nach 20 Sekunden wurde der Laptop aus seiner „Dockingstation“ entnommen und vom Benutzer in Richtung Bürotür getragen. Im Zeitraum zwischen 35s und 40s passierte der Benutzer die Bürotür; er blieb kurz stehen, drehte sich zur Seite und öffnete die Tür, und bog dann links herum in den Flur ab. Die Bürotür blieb offen. Es ging links herum mit langsamer Schrittgeschwindigkeit den Flur entlang, bis zum zentralen Treppenhaus des Helmholtzgebäudes. Die Messung wurde unterbrochen, als kein Datenverkehr mehr festgestellt werden konnte.

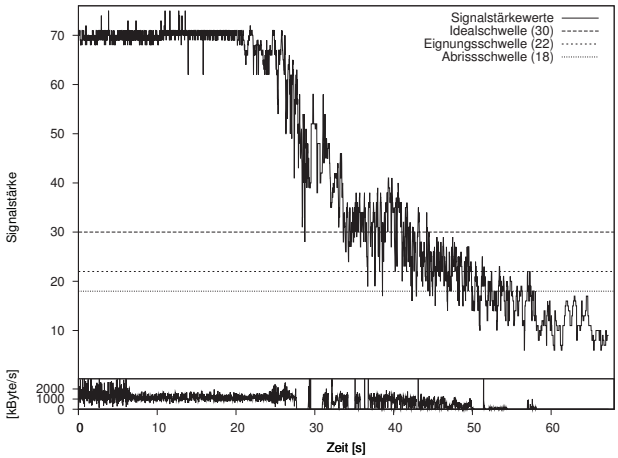

Abbildung D.41: Signalstärkewerte und TCP-Datenrate beim ersten Indoor-Szenario

Es wurden 50 Signalstärkemesswerte pro Sekunde („Werte pro Sekunde“ (WpS)) ermittelt. Anschließend wurden drei unterschiedliche Fenstergrößen betrachtet: 12 Sekunden ($12 * 50 = 600$ Messwerte), 20 Sekunden ($20 * 50 = 1000$ Messwerte) und 30 Sekunden ($30 * 50 = 1500$ Messwerte). Der minimale Füllstand betrug in allen Fällen 10 Sekunden. Die gezeigten Diagramme beruhen durchgängig auf linearer Regressionsrechnung.

D.7.1 Messreihe 7, 12s/10s (600/500 Messwerte, 50 WpS)

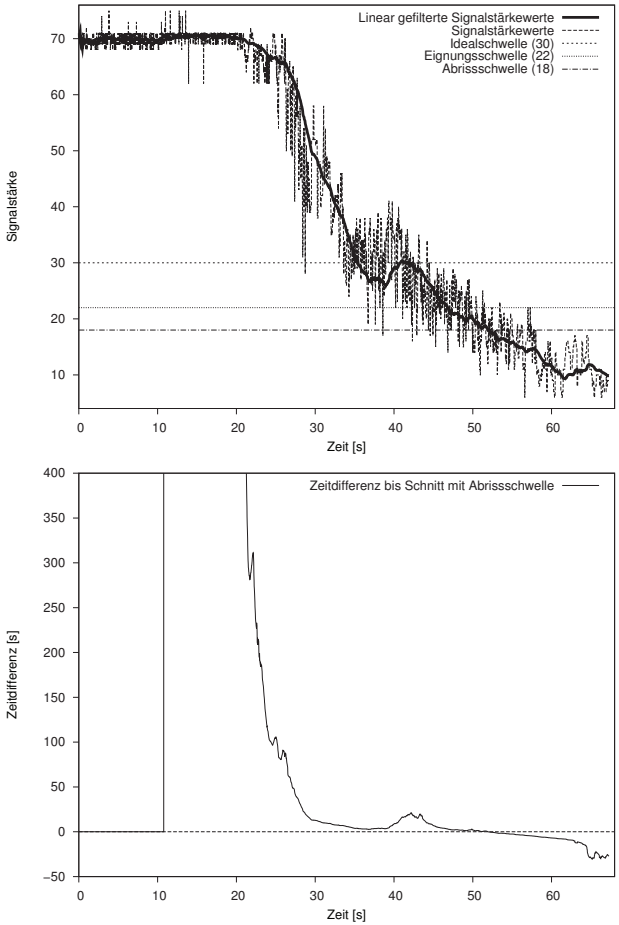

Abbildung D.42: Das obere Diagramm zeigt die linear gefilterten Signalstärkewerte im Vergleich mit den Rohdaten. Das untere Diagramm führt zu jedem Zeitpunkt den zeitlichen Abstand zum errechneten Schnittpunkt zwischen der momentan gültigen Regressionsgeraden und der Abrissschwelle auf. Innerhalb der ersten 10 Sekunden war die Mindestwertemenge für die Regressionsrechnung noch nicht verfügbar, sodass hier noch auf eine Berechnung der Zeitdifferenz verzichtet worden war.

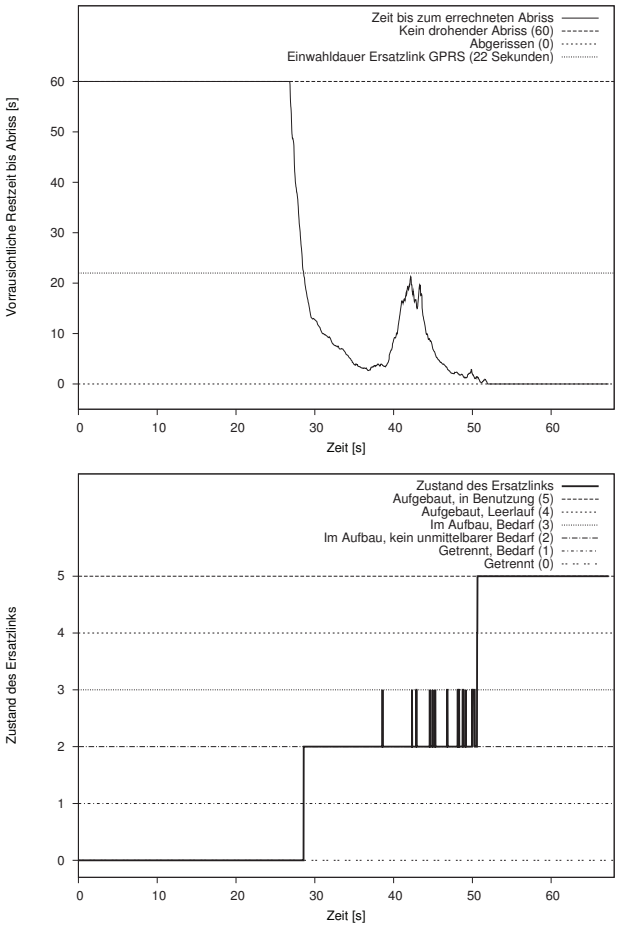

Abbildung D.43: Voraussichtliche Restzeit bis zum Abriss des Hauptlinks (oberes Diagramm) und Ersatzlinknutzung durch den Handoverentscheider (unteres Diagramm)

D.7.2 Messreihe 7, 20s/10s (1000/500 Messwerte, 50 WpS)

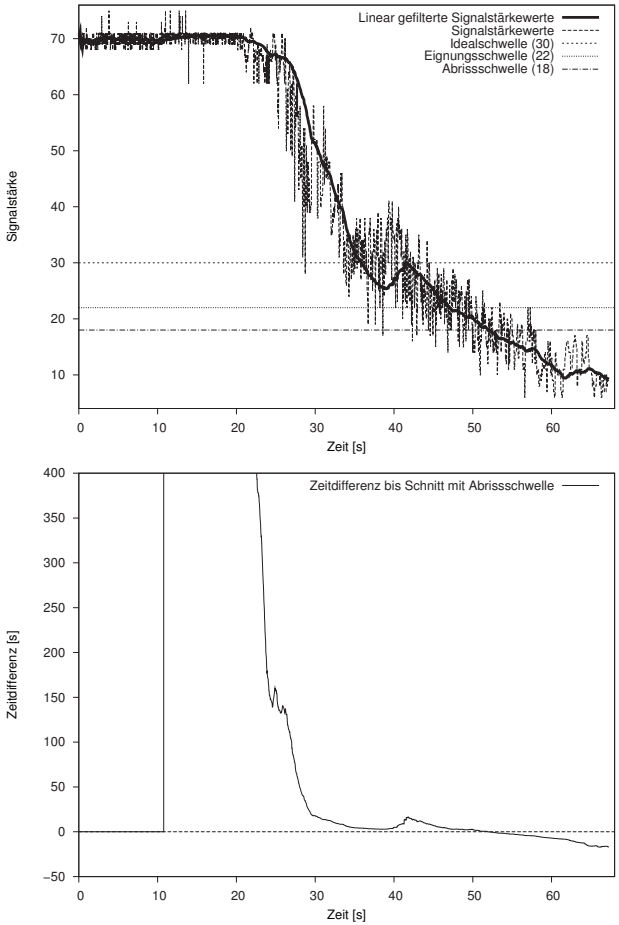

Abbildung D.44: Das obere Diagramm zeigt die linear gefilterten Signalstärkewerte im Vergleich mit den Rohdaten. Das untere Diagramm führt zu jedem Zeitpunkt den zeitlichen Abstand zum errechneten Schnittpunkt zwischen der momentan gültigen Regressionsgeraden und der Abrissschwelle auf. Innerhalb der ersten 10 Sekunden war die Mindestwertemenge für die Regressionsrechnung noch nicht verfügbar, sodass hier noch auf eine Berechnung der Zeitdifferenz verzichtet worden war.

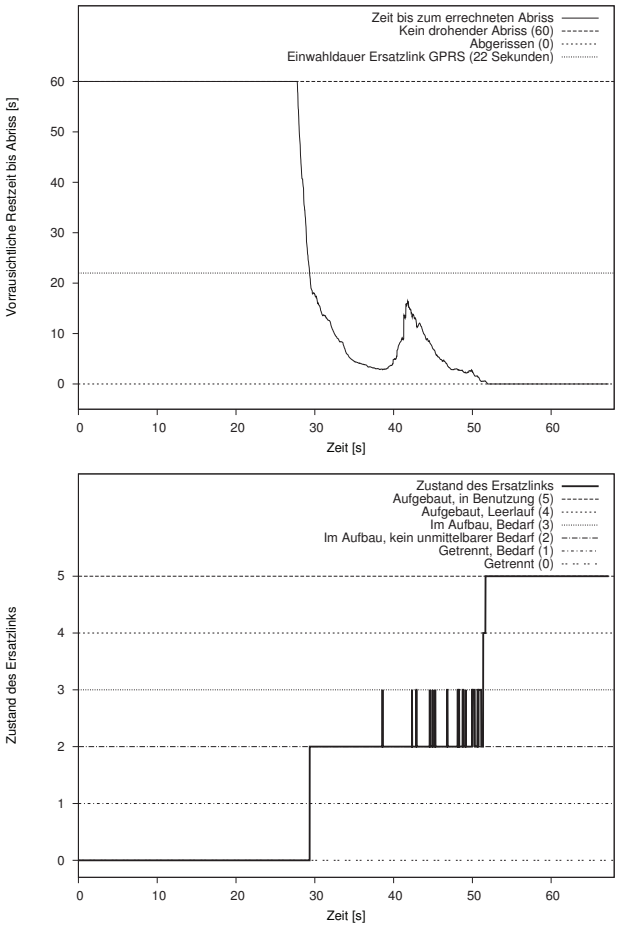

Abbildung D.45: Voraussichtliche Restzeit bis zum Abriss des Hauptlinks (oberes Diagramm) und Ersatzlinknutzung durch den Handoverentscheider (unteres Diagramm)

D.7.3 Messreihe 7, 30s/10s (1500/500 Messwerte, 50 WpS)

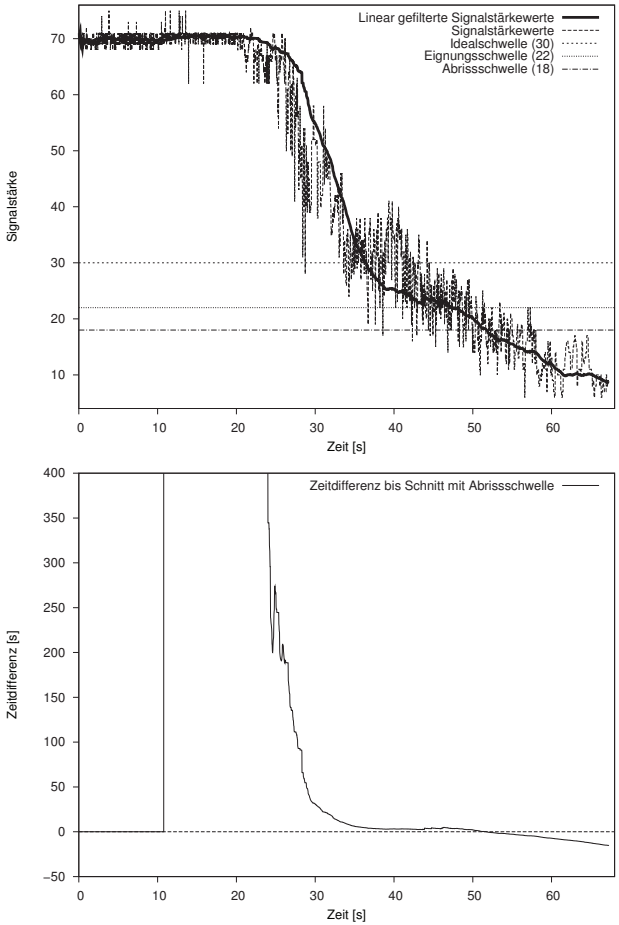

Abbildung D.46: Das obere Diagramm zeigt die linear gefilterten Signalstärkewerte im Vergleich mit den Rohdaten. Das untere Diagramm führt zu jedem Zeitpunkt den zeitlichen Abstand zum errechneten Schnittpunkt zwischen der momentan gültigen Regressionsgeraden und der Abrissschwelle auf. Innerhalb der ersten 10 Sekunden war die Mindestwertemenge für die Regressionsrechnung noch nicht verfügbar, sodass hier noch auf eine Berechnung der Zeitdifferenz verzichtet worden war.

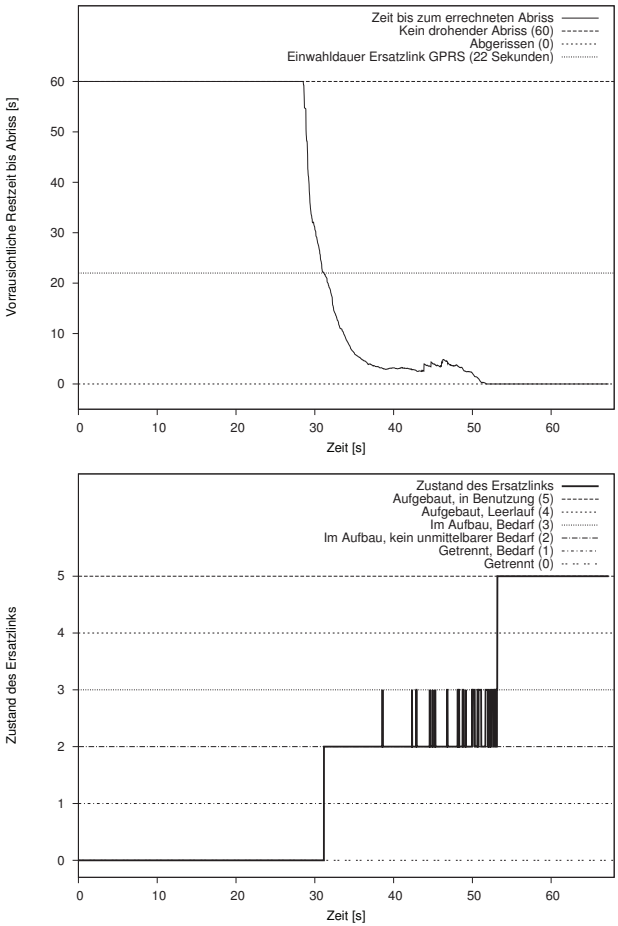

Abbildung D.47: Voraussichtliche Restzeit bis zum Abriss des Hauptlinks (oberes Diagramm) und Ersatzlinknutzung durch den Handoverentscheider (unteres Diagramm)

Verstrichene Zeit [s]	Ereignis Ersatzlink GPRS
29	Assoziiere Ersatzlink
51	Assoziation aufgebaut
52	Beginne mit Datenübertragung

Tabelle D.9: Diese Kommandos und Statusänderungen wurden bezüglich des hier betrachteten Ersatzlinks (GPRS mit einer Einwahlzeit von 22 Sekunden) vom Handoverentscheider abgesetzt bzw. empfangen. Das verwendete Fenster umfasste 20 Sekunden (1000 Messwerte).

Diskussion

Dieses Szenario zeigt ein typisches „sich von einer Basisstation entfernen“-Schema. Im unteren Diagramm von Abbildung D.46 kann man deutlich eine abfallende Restzeit zum prognostizierten Abrissereignis hin erkennen. Zum Zeitpunkt 29s betrug die erwartete Restzeit bis zum Linkabriss weniger als 22 Sekunden (oberes Diagramm in Abbildung D.47), weshalb der Ersatzlink angefordert worden war (selbe Abbildung, unteres Diagramm). Zum Zeitpunkt 52s durchstieß der linear gefilterte Signalstärkewert schließlich die Abrissschwelle, und bereits eine Sekunde später war der Ersatzlink vollständig aufgebaut und verfügbar.

Betrachtenswert ist, dass die gefilterten Signalstärkewerte in zwei Intervallen beinahe linear abfielen. Zwischen 25s und 35s bewegte sich der Nutzer innerhalb seines Büros von der Basisstation weg, und im Zeitraum von 40s bis 60s ging er den Flur entlang. Das ist der Grund dafür, dass die lineare Regressionsrechnung in diesem Szenario so gut funktionierte: Zum Zeitpunkt 29s wurde der Ersatzlink angefordert (erwartete Komplettierung bei 51s), und nur eine Sekunde später wurde die Abrissschwelle unterschritten.

An dieser Stelle muss jedoch darauf hingewiesen werden, dass anhand der beiden linear anmutenden Bereiche der gefilterten Messwerte nur sehr grob auf den Verlauf der Regressionsgeraden zu diesen Zeitpunkten geschlossen werden könnte. Diese fiel nämlich zum Zeitpunkt 29s noch deutlich flacher ab als der anschließende „lineare“ Abschnitt, was an der Krümmung des Werteverlaufs und der Betrachtung vergangener Werte begründet liegt.

Nichtsdestotrotz ist diese Messreihe ein sehr anschauliches Beispiel dafür, dass die Regressionsrechnung auch „Indoor“ eingesetzt werden kann und dabei brauchbare Ergebnisse liefert. Dazu müssen allerdings Messwerte mit einer hohen Rate zur Verfügung stehen (hier: willkürlich 50 Signalstärkewerte pro Sekunde).

D.8 Messreihe 8

Szenario

Dieses Szenario (Abbildung D.48) startete im Obergeschoss nahe des zentralen Treppenhauses des Helmholtzbaus. Nachdem die Messung begonnen wurde, verweilte der Nutzer noch ca. 10 Sekunden auf der Stelle, damit genügend Messwerte für die Regressionsrechnung bereit standen. Zu Zeitpunkt 20s wurde das offene Büro des Nutzers passiert, in welchem sich die Basisstation befand. Anschließend ging es weiter in Richtung Treppe, eine Etage herunter (40s) und unterhalb des Büros mit der Basisstation den Flur entlang. Zu Zeitpunkt 50s befand sich der Laptop knapp unterhalb des Büros, weshalb die Signalstärke im Folgenden wieder abfiel, bis der Datentransfer schließlich komplett zusammenbrach.

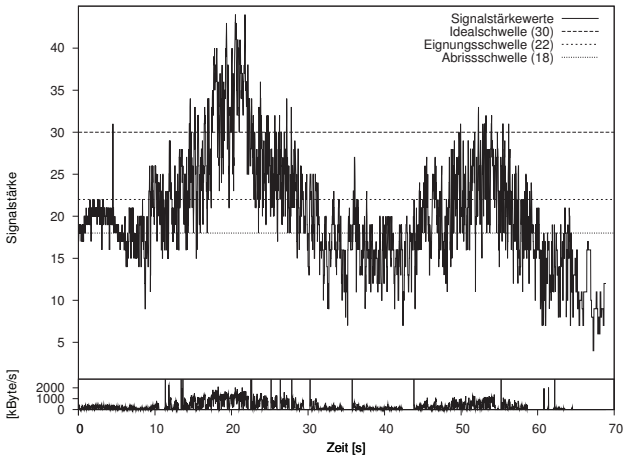

Abbildung D.48: Signalstärkewerte und TCP-Datenrate beim zweiten Indoor-Szenario

Es wurden 50 Signalstärkemesswerte pro Sekunde ermittelt, wobei anschließend vier von fünf Messwerten ignoriert worden sind, was einer effektiven Erfassungsrate von 10 Signalstärkemesswerten pro Sekunde entspricht. Es wurden wiederum drei unterschiedliche Fenstergrößen betrachtet: 12 Sekunden ($12 * 10 = 120$ Messwerte), 20 Sekunden ($20 * 10 = 200$ Messwerte) und 30 Sekunden ($30 * 10 = 300$ Messwerte). Der minimale Füllstand betrug in allen Fällen 10 Sekunden. Die gezeigten Diagramme beruhen durchgängig auf linearer Regressionsrechnung.

D.8.1 Messreihe 8, 12s/10s (120/100 Messwerte, 10 WpS)

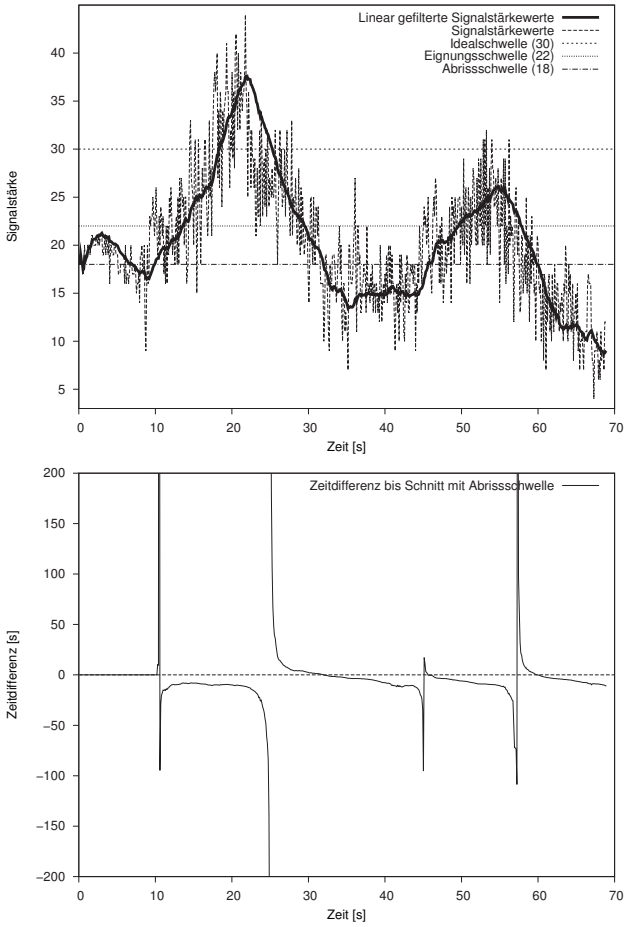

Abbildung D.49: Das obere Diagramm zeigt die linear gefilterten Signalstärkewerte im Vergleich mit den Rohdaten. Das untere Diagramm führt zu jedem Zeitpunkt den zeitlichen Abstand zum errechneten Schnittpunkt zwischen der momentan gültigen Regressionsgeraden und der Abrissschwelle auf. Innerhalb der ersten 10 Sekunden war die Mindestwertemenge für die Regressionsrechnung noch nicht verfügbar, sodass hier noch auf eine Berechnung der Zeitdifferenz verzichtet worden war.

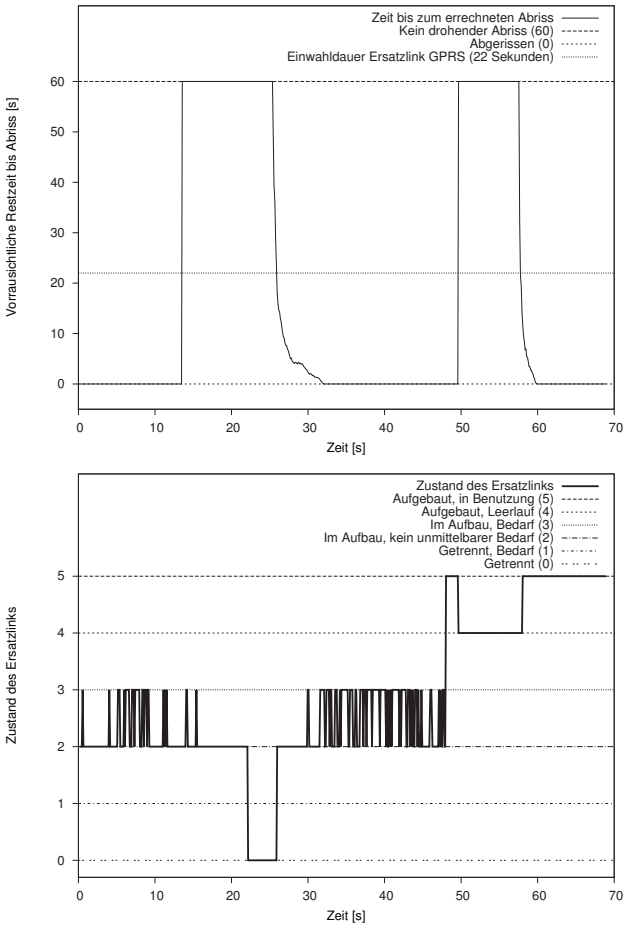

Abbildung D.50: Voraussichtliche Restzeit bis zum Abriss des Hauptlinks (oberes Diagramm) und Ersatzlinknutzung durch den Handoverentscheider (unteres Diagramm)

D.8.2 Messreihe 8, 20s/10s (200/100 Messwerte, 10 WpS)

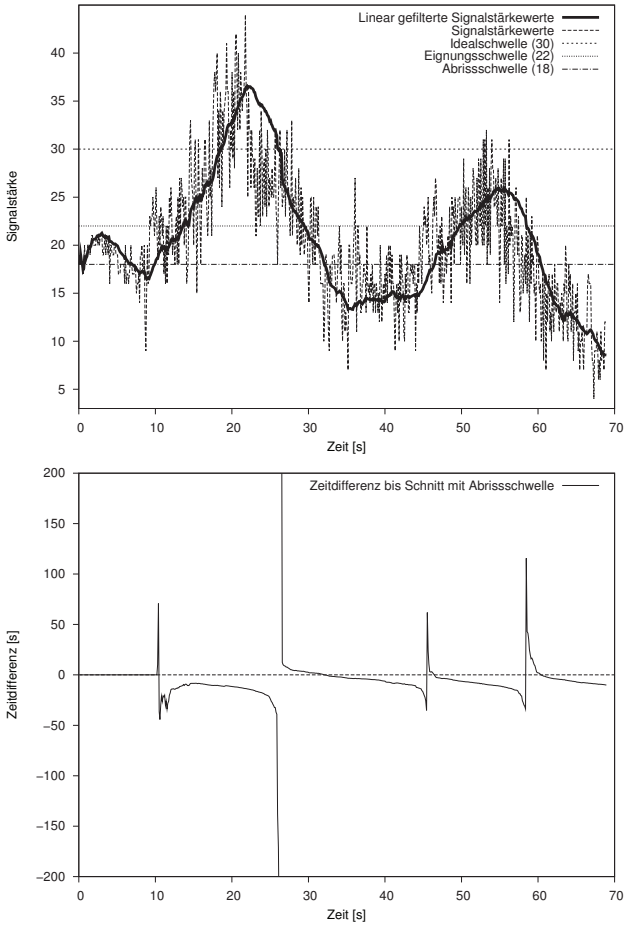

Abbildung D.51: Das obere Diagramm zeigt die linear gefilterten Signalstärkewerte im Vergleich mit den Rohdaten. Das untere Diagramm führt zu jedem Zeitpunkt den zeitlichen Abstand zum errechneten Schnittpunkt zwischen der momentan gültigen Regressionsgeraden und der Abrissschwelle auf. Innerhalb der ersten 10 Sekunden war die Mindestwertemenge für die Regressionsrechnung noch nicht verfügbar, sodass hier noch auf eine Berechnung der Zeitdifferenz verzichtet worden war.

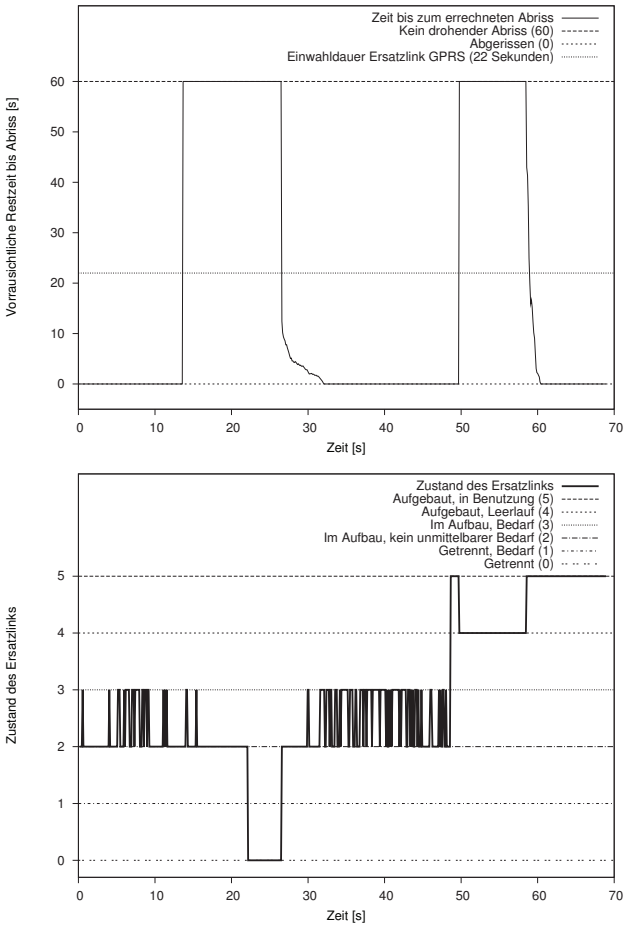

Abbildung D.52: Voraussichtliche Restzeit bis zum Abriss des Hauptlinks (oberes Diagramm) und Ersatzlinknutzung durch den Handoverentscheider (unteres Diagramm)

D.8.3 Messreihe 8, 30s/10s (300/100 Messwerte, 10 WpS)

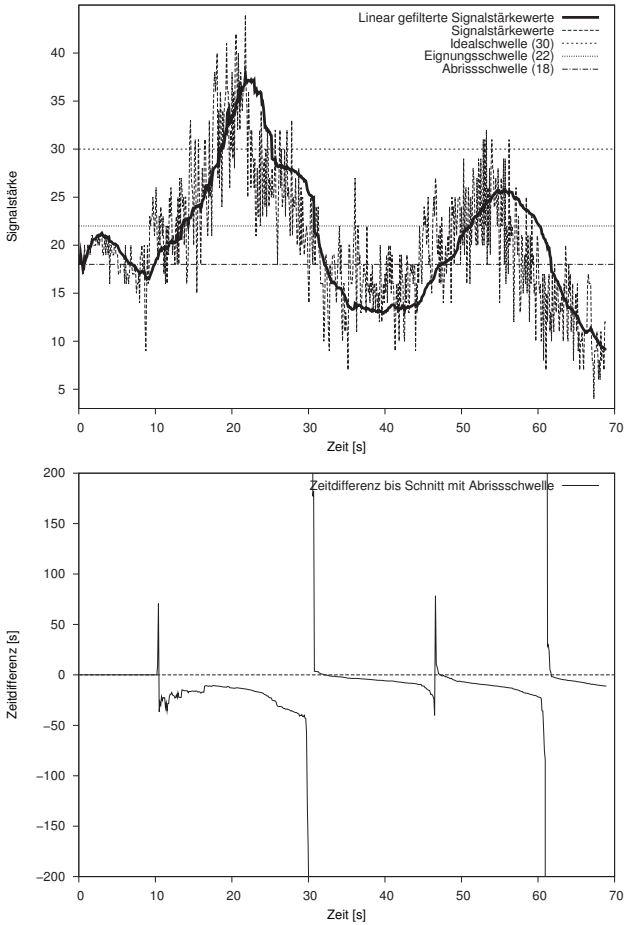

Abbildung D.53: Das obere Diagramm zeigt die linear gefilterten Signalstärkewerte im Vergleich mit den Rohdaten. Das untere Diagramm führt zu jedem Zeitpunkt den zeitlichen Abstand zum errechneten Schnittpunkt zwischen der momentan gültigen Regressionsgeraden und der Abrissschwelle auf. Innerhalb der ersten 10 Sekunden war die Mindestwertemenge für die Regressionsrechnung noch nicht verfügbar, sodass hier noch auf eine Berechnung der Zeitdifferenz verzichtet worden war.

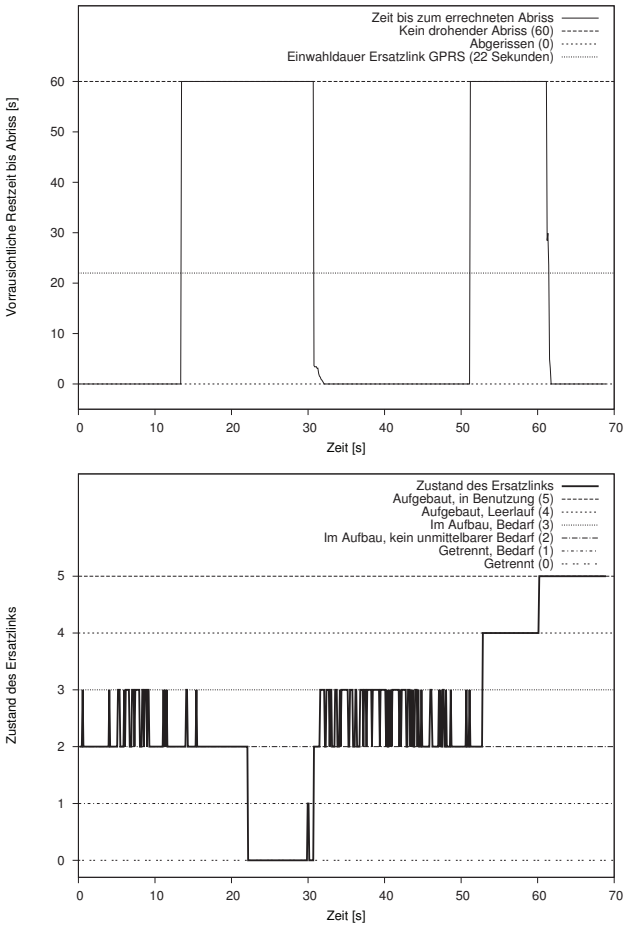

Abbildung D.54: Voraussichtliche Restzeit bis zum Abriss des Hauptlinks (oberes Diagramm) und Ersatzlinknutzung durch den Handoverentscheider (unteres Diagramm)

Verstrichene Zeit [s]	Ereignis Ersatzlink GPRS
0	Assoziiere Ersatzlink
22	Assoziation abgebaut
27	Assoziiere Ersatzlink
49	Assoziation aufgebaut
49	Beginne mit Datenübertragung
50	Stoppe Datenübertragung
59	Beginne mit Datenübertragung

Tabelle D.10: Diese Kommandos und Statusänderungen wurden bezüglich des hier betrachteten Ersatzlinks (GPRS mit einer Einwahlzeit von 22 Sekunden) vom Handoverentscheider abgesetzt bzw. empfangen. Das verwendete Fenster umfasste 20 Sekunden (200 Messwerte).

Diskussion

In den ersten 10 Sekunden lagen noch nicht genügend Messwerte für die Regressionsrechnung vor, daher wurde reaktiv entschieden und in allen Fällen eine sofortige Assoziation des Ersatzlinks angeordnet (siehe Tabelle D.10). In den ersten 15 Sekunden hätte der Ersatzlink auch gebraucht werden können, was an dem häufigen Wechsel zu Zustand 3 in allen drei Ersatzlinknutzungsdiagrammen erkennbar ist.

Zu Zeitpunkt 27s wurde ein Abriss für 22 Sekunden in der Zukunft prognostiziert, daher kam es zu einem Aufbau des Ersatzlinks. Auf Grund des raschen Abfalls der Signalstärke innerhalb von 10 Sekunden stand zum Abrisszeitpunkt (34s) noch kein aufgebauter Ersatzlink bereit. Die verbleibende Zeit bis zum erfolgreichen Aufbau (bei 49s) betrug allerdings nur noch 15 Sekunden, was einem Zeitvorteil von 7 Sekunden gegenüber einem reaktiven Entscheider entsprach.

Beim zweiten „Höcker“ wurde der Ersatzlink nicht länger benutzt, jedoch auch nicht abgebaut (Idealschwelle nicht überschritten). Dadurch stand beim anschließenden Abfall (Abriss bei 59s) sofort der Ersatzlink zur Verfügung.

Anhand dieser Messreihe lässt sich der Einfluss der verschiedenen Fenstergrößen erkennen. Je größer das Fenster gewählt wird, desto weiter rücken die prognostizierten Abrissereignisse und damit die Assoziationskommandos in die Zukunft. Der Effekt umfasst allerdings nur wenige Sekunden, welche in der Praxis aber einen spürbaren Unterschied ausmachen können.

D.9 Messreihe 9

Szenario

Die vorliegende Messreihe (Abbildung D.55) wurde draußen aufgenommen, und entspricht von Ihrer Konzeption her der Messreihe 3 (siehe Seite 371). Im Unterschied zu Messreihe 3 wurden die Signalstärkewerte hier jedoch nicht sekündlich, sondern mit einer Rate von 50 Messwerten pro Sekunde ermittelt. Damit sollten ein Vergleich zwischen einer Regressionsrechnung mit niedriger Rate und einer mit einer hohen Rate ermöglicht werden. Die Verwendung von Messwerten mit einer hohen Rate hatte bei den „Indoor"-Messungen (Messreihen 7 und 8) gute Ergebnisse erzielt. Im Unterschied zu Messreihe 3 wurde zudem anfangs mehr als 10 Sekunden auf der Stelle verweilt, damit genügend Werte für die Regressionsrechnung bereit standen.

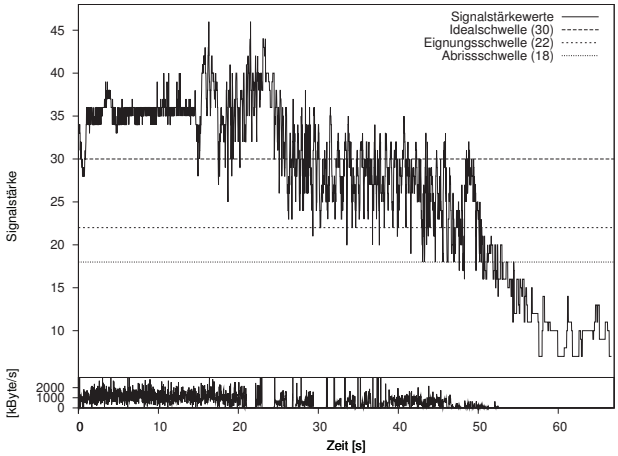

Abbildung D.55: Signalstärkewerte und TCP-Datenrate

Es wurden 50 Signalstärkemesswerte pro Sekunde ermittelt. Anschließend wurden drei unterschiedliche Fenstergrößen betrachtet: 12 Sekunden ($12 * 50 = 600$ Messwerte), 20 Sekunden ($20 * 50 = 1000$ Messwerte) und 30 Sekunden ($30 * 50 = 1500$ Messwerte). Der minimale Füllstand betrug in allen Fällen 10 Sekunden. Die gezeigten Diagramme beruhen durchgängig auf linearer Regressionsrechnung.

D.9.1 Messreihe 9, 12s/10s (600/500 Messwerte, 50 WpS)

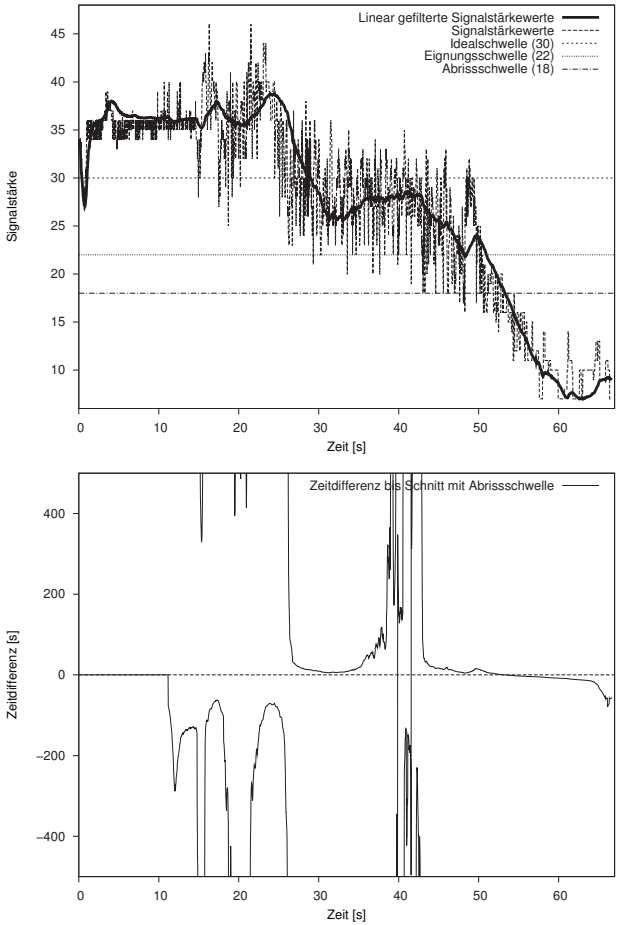

Abbildung D.56: Das obere Diagramm zeigt die linear gefilterten Signalstärkewerte im Vergleich mit den Rohdaten. Das untere Diagramm führt zu jedem Zeitpunkt den zeitlichen Abstand zum errechneten Schnittpunkt zwischen der momentan gültigen Regressionsgeraden und der Abrissschwelle auf. Innerhalb der ersten 10 Sekunden war die Mindestwertemenge für die Regressionsrechnung noch nicht verfügbar, sodass hier noch auf eine Berechnung der Zeitdifferenz verzichtet worden war.

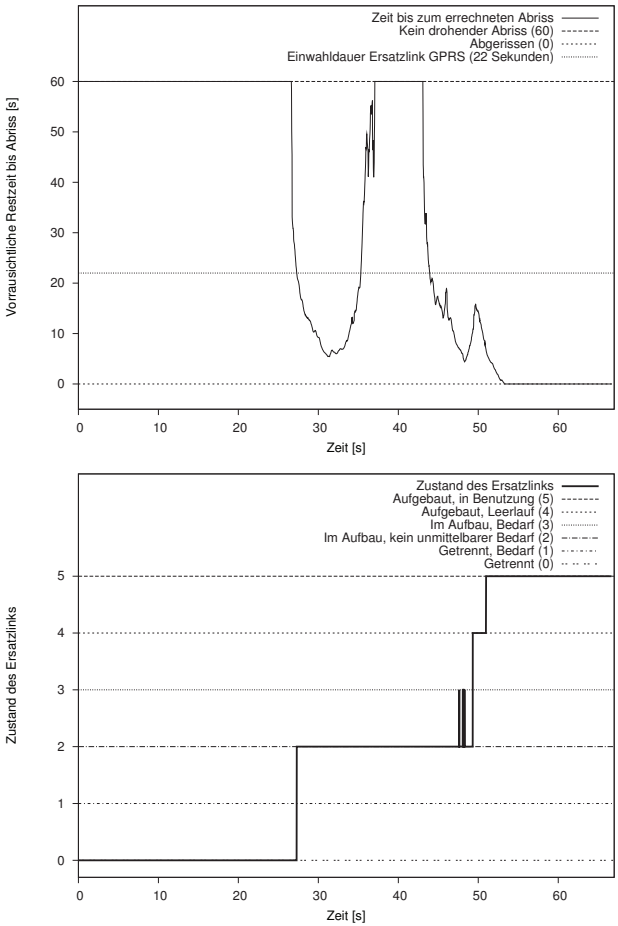

Abbildung D.57: Voraussichtliche Restzeit bis zum Abriss des Hauptlinks (oberes Diagramm) und Ersatzlinknutzung durch den Handoverentscheider (unteres Diagramm)

D.9.2 Messreihe 9, 20s/10s (1000/500 Messwerte, 50 WpS)

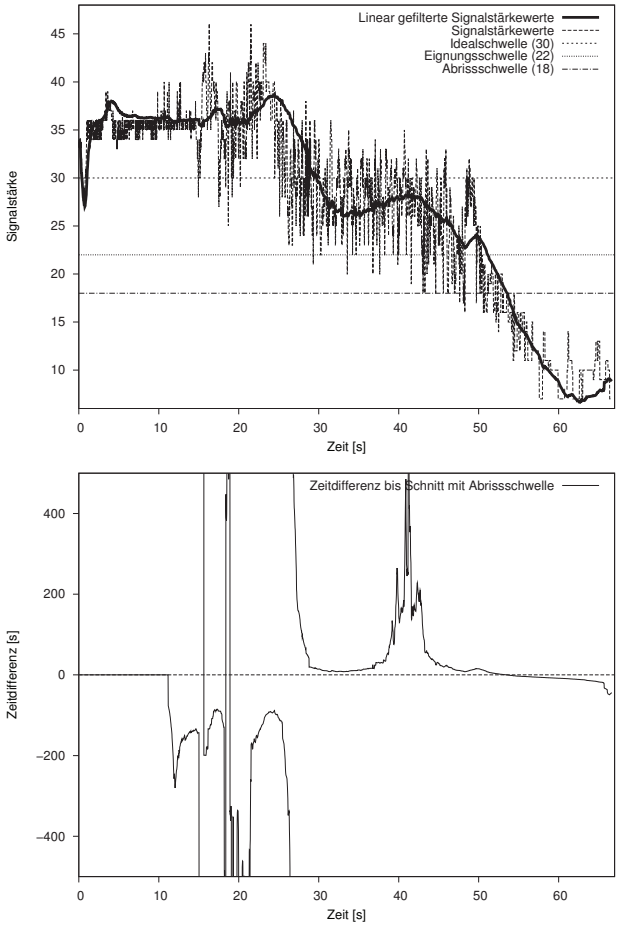

Abbildung D.58: Das obere Diagramm zeigt die linear gefilterten Signalstärkewerte im Vergleich mit den Rohdaten. Das untere Diagramm führt zu jedem Zeitpunkt den zeitlichen Abstand zum errechneten Schnittpunkt zwischen der momentan gültigen Regressionsgeraden und der Abrissschwelle auf. Innerhalb der ersten 10 Sekunden war die Mindestwertemenge für die Regressionsrechnung noch nicht verfügbar, sodass hier noch auf eine Berechnung der Zeitdifferenz verzichtet worden war.

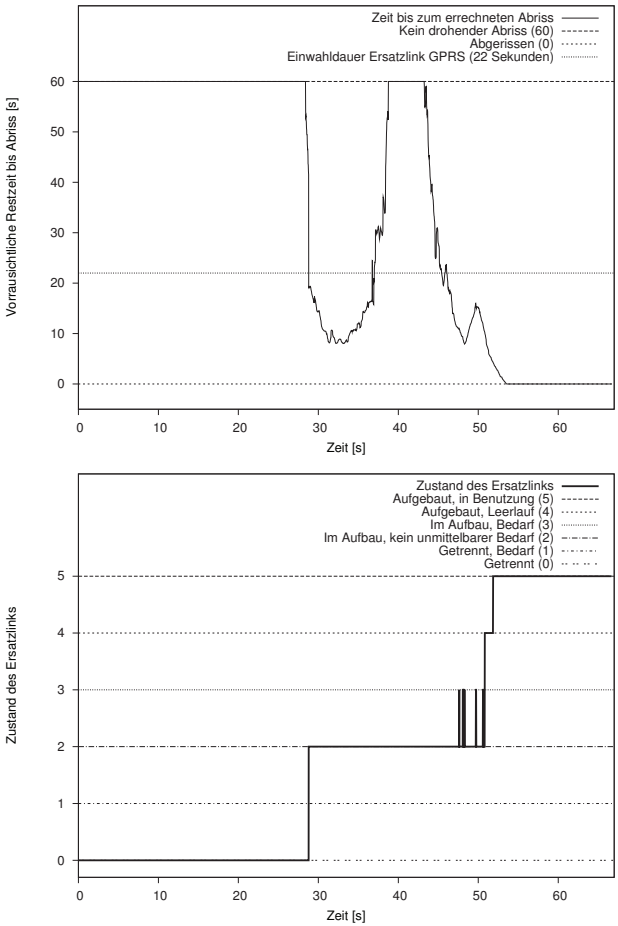

Abbildung D.59: Voraussichtliche Restzeit bis zum Abriss des Hauptlinks (oberes Diagramm) und Ersatzlinknutzung durch den Handoverentscheider (unteres Diagramm)

D.9.3 Messreihe 9, 30s/10s (1500/500 Messwerte, 50 WpS)

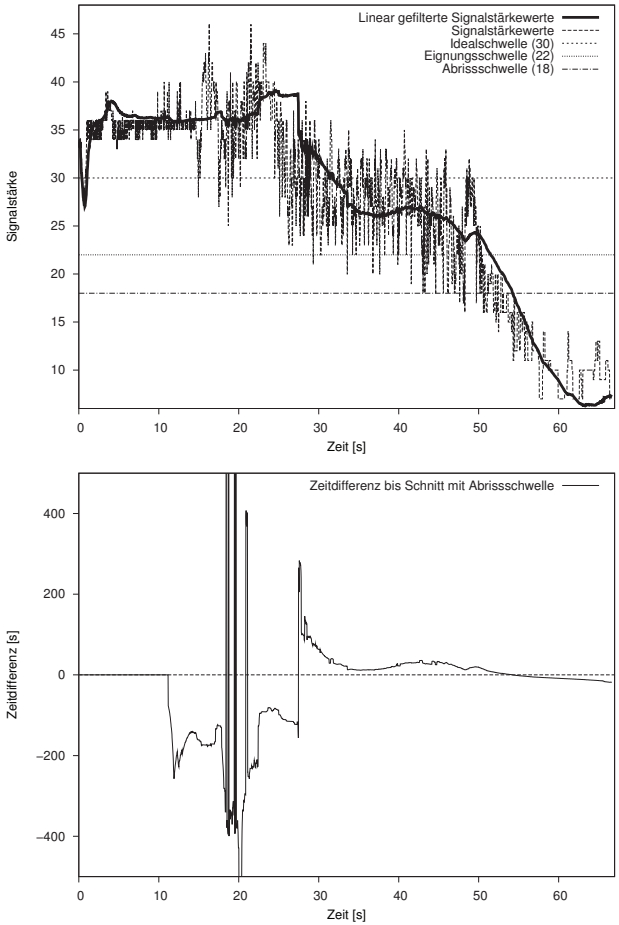

Abbildung D.60: Das obere Diagramm zeigt die linear gefilterten Signalstärkewerte im Vergleich mit den Rohdaten. Das untere Diagramm führt zu jedem Zeitpunkt den zeitlichen Abstand zum errechneten Schnittpunkt zwischen der momentan gültigen Regressionsgeraden und der Abrissschwelle auf. Innerhalb der ersten 10 Sekunden war die Mindestwertemenge für die Regressionsrechnung noch nicht verfügbar, sodass hier noch auf eine Berechnung der Zeitdifferenz verzichtet worden war.

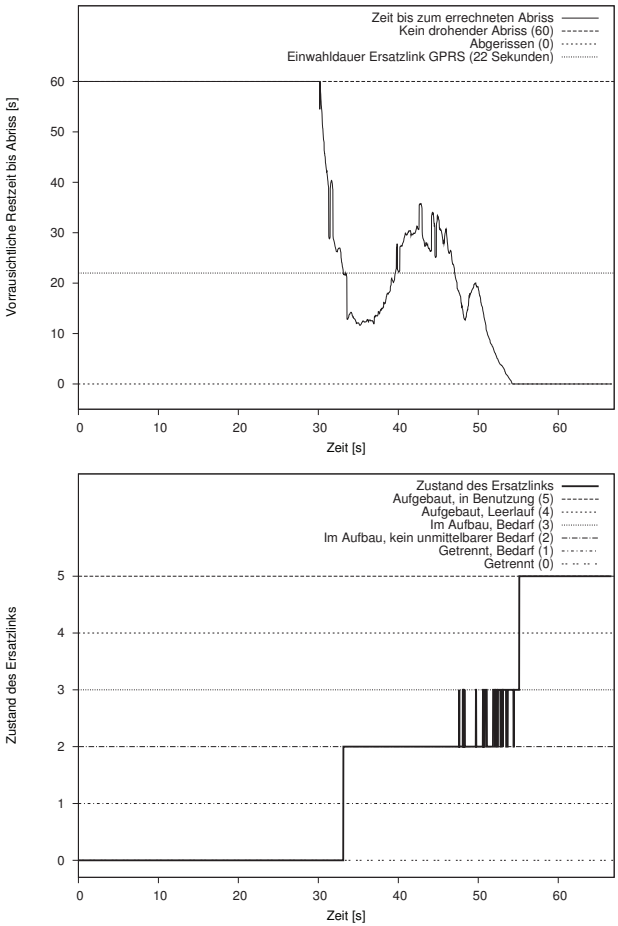

Abbildung D.61: Voraussichtliche Restzeit bis zum Abriss des Hauptlinks (oberes Diagramm) und Ersatzlinknutzung durch den Handoverentscheider (unteres Diagramm)

Verstrichene Zeit [s]	Ereignis Ersatzlink GPRS
29	Assoziiere Ersatzlink
51	Assoziation aufgebaut
52	Beginne mit Datenübertragung

Tabelle D.11: Diese Kommandos und Statusänderungen wurden bezüglich des hier betrachteten Ersatzlinks (GPRS mit einer Einwahlzeit von 22 Sekunden) vom Handoverentscheider abgesetzt bzw. empfangen. Das verwendete Fenster umfasste 20 Sekunden (1000 Messwerte).

Diskussion

Die vorliegende Messreihe eignet sich gut, um die Auswirkungen unterschiedlicher Fenstergrößen zu untersuchen. Bei der Auswertung mit der geringsten Fenstergröße (12 Sekunden, Abbildungen auf Seite 404) zeigt sich ein gutes Prognoseergebnis. Beim nächstgrößeren Berechnungsfenster von 20 Sekunden (siehe Seite 406) machte sich aber bereits eine „Trägheit“ bemerkbar, da die Regressionsgerade dem fallenden Verlauf nicht schnell genug folgen konnte. So kam es bereits zu einem verspäteten Aufbau des Ersatzlinks. Beim dritten Regressionsfenster mit 30 Sekunden Dauer (siehe Seite 408) machte sich diese Trägheit am deutlichsten bemerkbar. Der prognostizierte Abrisszeitpunkt wurde auf Grund der langfristigen Betrachtung und der damit verbundenen zu flach abfallenden Regressionsgeraden in Richtung Zukunft verschoben, sodass der Ersatzlink schlussendlich nicht mehr schnell genug aufgebaut werden konnte.

D.10 Messreihe 10

Szenario

Messreihe 10 (Abbildung D.62) wurde ebenfalls draußen aufgenommen, und entspricht von Ihrer Konzeption her der Messreihe 4 (siehe Seite 375). Auch hier wurden die Signalstärkewerte, ähnlich wie in den Messreihen 7,8 und 9, mit 50 Messungen pro Sekunde aufgenommen. Die Motivation zur Wiederholung der Messreihe war identisch mit der Erklärung aus Messreihe 9.

Beim Vergleich des Signalstärkeverlaufs mit dem aus Messreihe 4 ist deutlich die Ähnlichkeit erkennbar, da hier derselbe Weg abgeschritten worden ist. Allerdings hat der Autor in dieser Messreihe sich mir normaler Schrittgeschwindigkeit fortbewegt, wodurch die Zeitachse hier im Vergleich gestaucht erscheint.

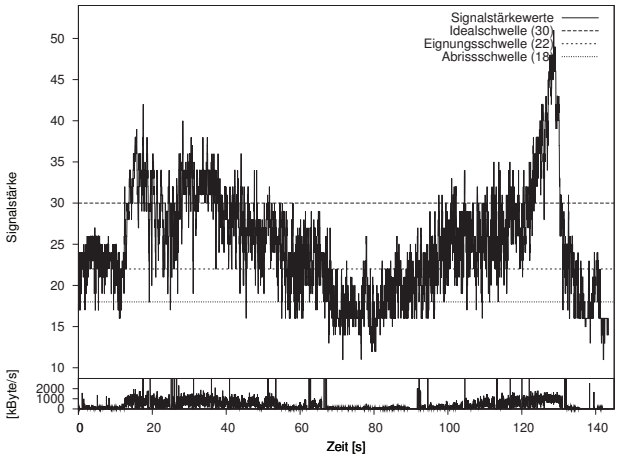

Abbildung D.62: Signalstärkewerte und TCP-Datenrate

Es wurden 50 Signalstärkemesswerte pro Sekunde ermittelt, wobei anschließend vier von fünf Messwerten ignoriert worden sind, was einer effektiven Erfassungsrate von 10 Signalstärkemesswerten pro Sekunde entspricht. Es wurden wiederum drei unterschiedliche Fenstergrößen betrachtet: 12 Sekunden ($12 * 10 = 120$ Messwerte), 20 Sekunden ($20 * 10 = 200$ Messwerte) und 30 Sekunden ($30 * 10 = 300$ Messwerte). Der minimale Füllstand betrug in allen Fällen 10 Sekunden. Die gezeigten Diagramme beruhen durchgängig auf linearer Regressionsrechnung.

D.10.1 Messreihe 10, 12s/10s (120/100 Messwerte, 10 WpS)

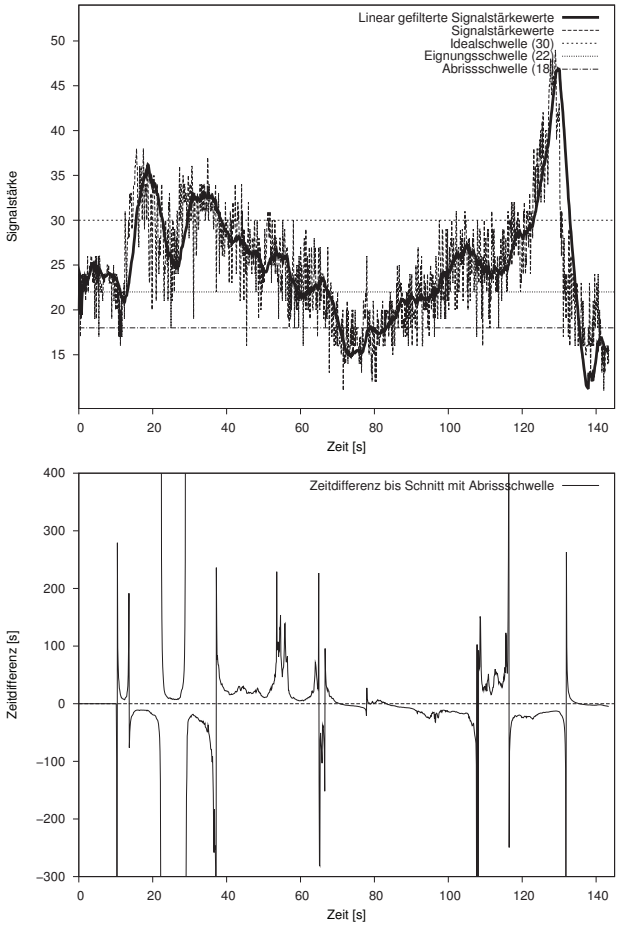

Abbildung D.63: Das obere Diagramm zeigt die linear gefilterten Signalstärkewerte im Vergleich mit den Rohdaten. Das untere Diagramm führt zu jedem Zeitpunkt den zeitlichen Abstand zum errechneten Schnittpunkt zwischen der momentan gültigen Regressionsgeraden und der Abrissschwelle auf. Innerhalb der ersten 10 Sekunden war die Mindestwertemenge für die Regressionsrechnung noch nicht verfügbar, sodass hier noch auf eine Berechnung der Zeitdifferenz verzichtet worden war.

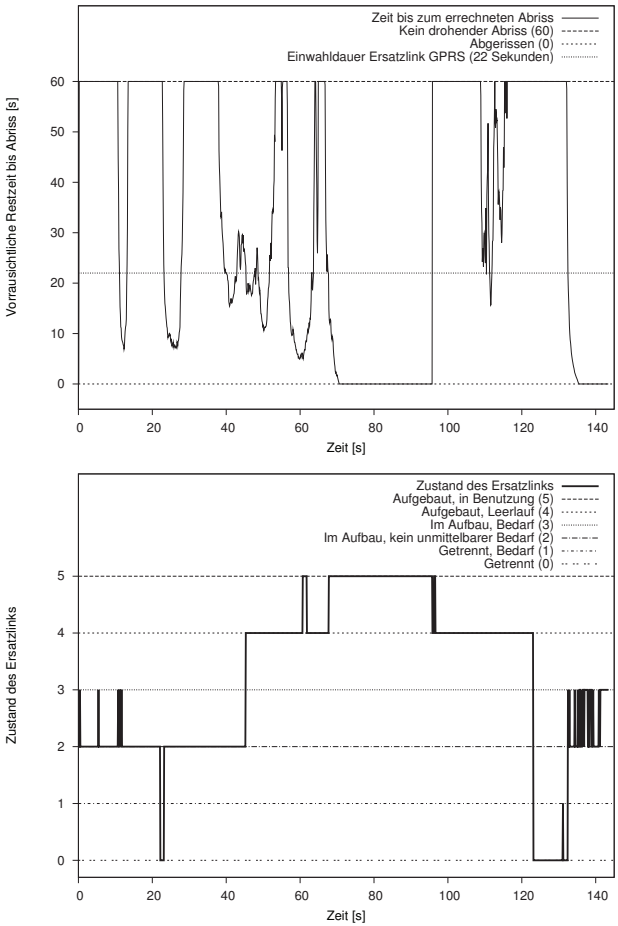

Abbildung D.64: Voraussichtliche Restzeit bis zum Abriss des Hauptlinks (oberes Diagramm) und Ersatzlinknutzung durch den Handoverentscheider (unteres Diagramm)

D.10.2 Messreihe 10, 20s/10s (200/100 Messwerte, 10 WpS)

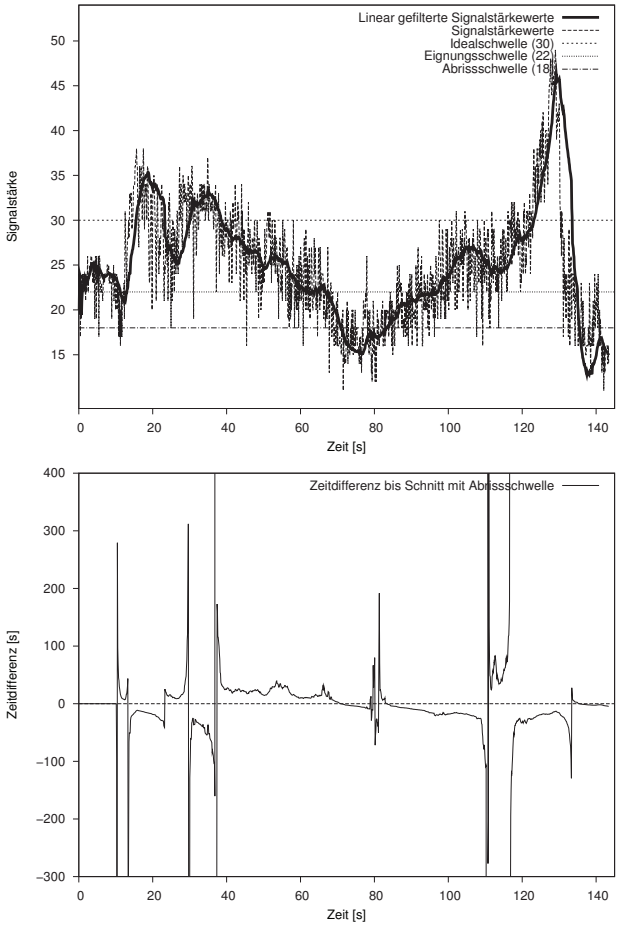

Abbildung D.65: Das obere Diagramm zeigt die linear gefilterten Signalstärkewerte im Vergleich mit den Rohdaten. Das untere Diagramm führt zu jedem Zeitpunkt den zeitlichen Abstand zum errechneten Schnittpunkt zwischen der momentan gültigen Regressionsgeraden und der Abrissschwelle auf. Innerhalb der ersten 10 Sekunden war die Mindestwertemenge für die Regressionsrechnung noch nicht verfügbar, sodass hier noch auf eine Berechnung der Zeitdifferenz verzichtet worden war.

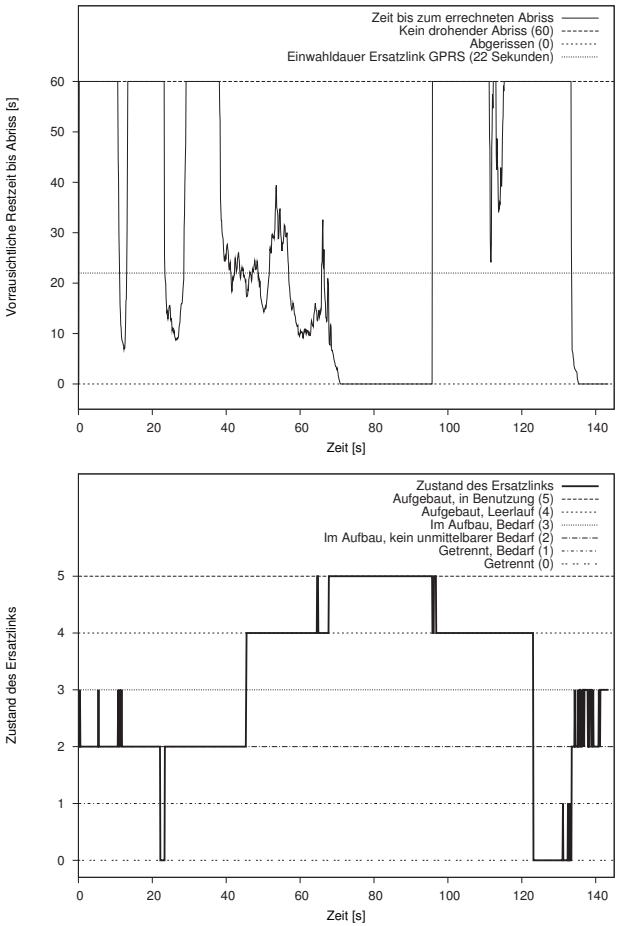

Abbildung D.66: Voraussichtliche Restzeit bis zum Abriss des Hauptlinks (oberes Diagramm) und Ersatzlinknutzung durch den Handoverentscheider (unteres Diagramm)

D.10.3 Messreihe 10, 30s/10s (300/100 Messwerte, 10 WpS)

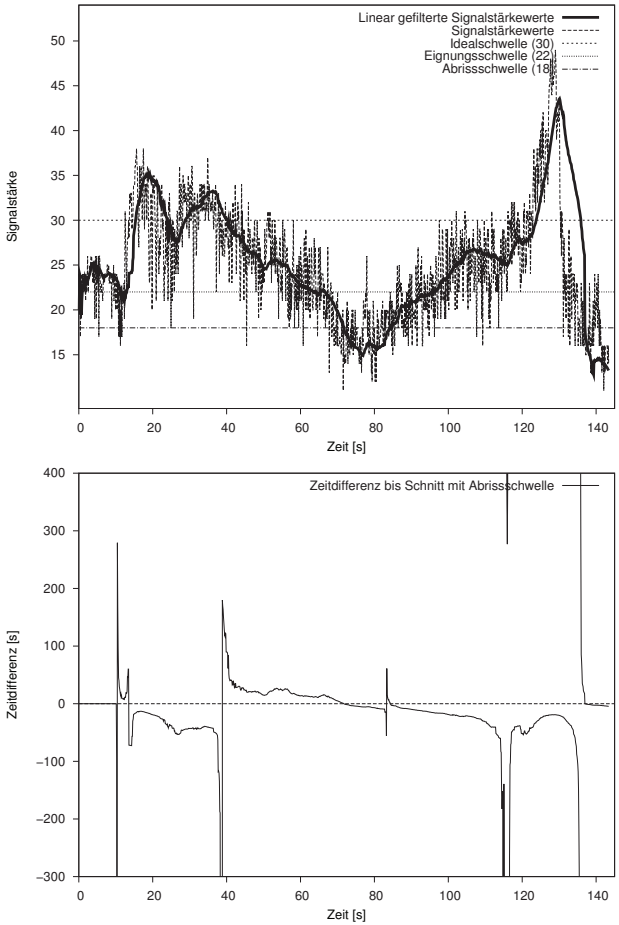

Abbildung D.67: Das obere Diagramm zeigt die linear gefilterten Signalstärkewerte im Vergleich mit den Rohdaten. Das untere Diagramm führt zu jedem Zeitpunkt den zeitlichen Abstand zum errechneten Schnittpunkt zwischen der momentan gültigen Regressionsgeraden und der Abrissschwelle auf. Innerhalb der ersten 10 Sekunden war die Mindestwertemenge für die Regressionsrechnung noch nicht verfügbar, sodass hier noch auf eine Berechnung der Zeitdifferenz verzichtet worden war.

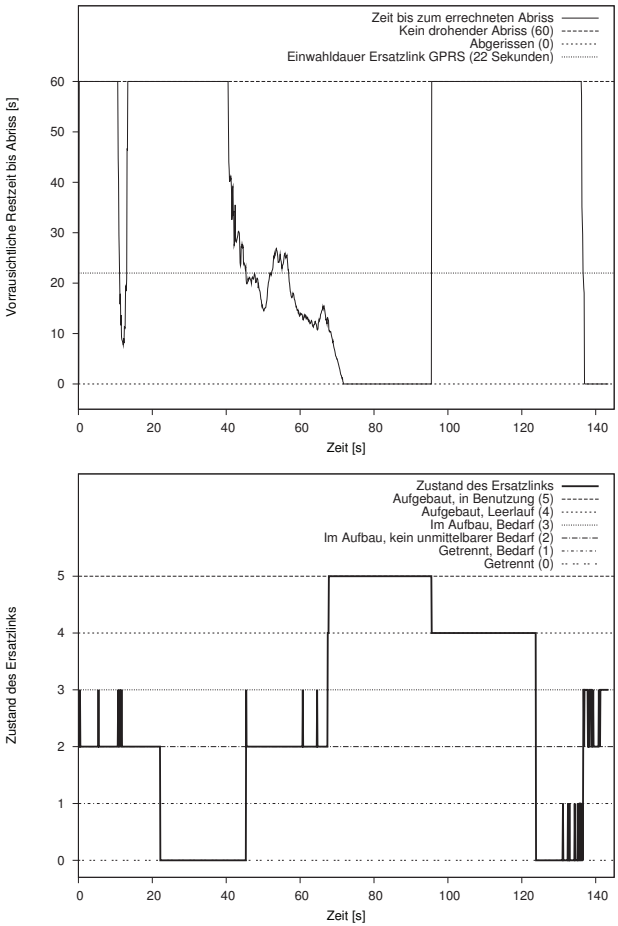

Abbildung D.68: Voraussichtliche Restzeit bis zum Abriss des Hauptlinks (oberes Diagramm) und Ersatzlinknutzung durch den Handoverentscheider (unteres Diagramm)

Verstrichene Zeit [s]	Ereignis Ersatzlink GPRS
0	Assoziiere Ersatzlink
22	Assoziation abgebaut
23	Assoziiere Ersatzlink
45	Assoziation aufgebaut
65	Beginne mit Datenübertragung
65	Stoppe Datenübertragung
68	Beginne mit Datenübertragung
96	Stoppe Datenübertragung
96	Beginne mit Datenübertragung
97	Stoppe Datenübertragung
123	Assoziation abgebaut
134	Assoziiere Ersatzlink

Tabelle D.12: Diese Kommandos und Statusänderungen wurden bezüglich des hier betrachteten Ersatzlinks (GPRS mit einer Einwahlzeit von 22 Sekunden) vom Handoverentscheider abgesetzt bzw. empfangen. Das verwendete Fenster umfasste 20 Sekunden (200 Messwerte).

Diskussion

Auch diese Messreihe eignete sich gut, um die Auswirkungen unterschiedlicher Fenstergrößen zu untersuchen. Deutlich machte sich hier wieder die mit wachsender Ausdehnung des zeitlichen Fensters ansteigende Trägheit bemerkbar. Bei dem Fenster mit 12 Sekunden Dauer erfolgte die Anforderung des Ersatzlinks zu früh, aber bei dem 20 Sekunden umfassenden Fenster ergab sich eine „gute“ Prognose.

Zudem ist deutlich erkennbar, wie sich die Diagramme mit steigender Größe des Berechnungsfensters „beruhigen“, also einen glatteren Verlauf zeigen. Dieser Vorteil ist jedoch vernachlässigbar, da durch die steigende Trägheit teilweise schlechtere Abrissprognosen errechnet worden sind. Deutlich wird dies beim raschen Abfall der Signalstärkewerte ab 130s, welcher in keinem der drei Rechnungen abgebildet werden konnte. Mit steigender Fenstergröße ist also wieder eine nachteilig anwachsende Trägheit erkennbar.

D.11 Fazit

Anhand der zehn Messreihen konnte verdeutlicht werden, dass die Durchführung einer Abrissprognose mit Hilfe der Regressionsrechnung brauchbare Ergebnisse liefert. Dieses positive Ergebnis hängt jedoch in hohem Maße von der Technologie des Hauptlinks und von der Assoziationsdauer des Ersatzlinks ab. Bei langen Assoziationszeiten wie den kalkulierten 22 Sekunden kam die Prognose gerade „Indoor“ an ihre Grenzen, da sich die Signalstärkewerte bzw. Linkgütewerte vergleichsweise kurzfristig verschlechtern können. Es ist angebracht, die Regressionsparameter sorfältig auszuwählen. Dazu bietet sich eine empirische Wahl an, was durch die Aufnahme von Messreihen und sukzessives Ausprobieren verschiedener Parametersätze zum Erfolg führt. Für den in REACH implementierten Handoverentscheider können die vorgestellten Parameter beliebig durch den Nutzer vorgegeben werden.

Inversquadratische vs. exponentielle vs. lineare Regression

In den Messreihen 1 und 2 hat sich gezeigt, dass sich die Ergebnisse der drei Regressionstypen nicht nennenswert voneinander unterscheiden. Dies gilt jedenfalls für die vorliegende Kombination aus WLAN-Adapter, dessen Treiber und der Version des verwendeten Linux-Kernels. In der vorliegenden Konfiguration erwies sich die lineare Regressionsrechnung als praktikabel. Es steht dem Nutzer allerdings frei, seine jeweils vorliegende Konfiguration gegen alle Regressionsvarianten zu testen und das als am geeignetsten erscheinende Schema empirisch auszuwählen. Daher wurde in dieser Ausarbeitung entschieden, ab Messreihe 3 nur noch die lineare Regressionsrechnung einzusetzen.

Bei den drei untersuchten Verfahren wird bei der linearen Regressionsrechnung der Ersatzlink tendenziell am frühesten aufgebaut, da im Vergleich zur exponentiellen und zur inversquadratischen Regressionsrechnung bei einem Abfall kein konkav gekrümmter Verlauf angenommen wird.

Signalstärkewerte vs. Linkgütewerte

In den Messreihen 1 und 2, welche mit Hilfe eines Linux-Kernels der Version 2.6.27 aufgenommen worden sind, waren für die Signalstärke und die Linkgüte jeweils noch unterschiedliche Werte verfügbar. Anhand der abgebildeten Beispiele lässt sich jedoch kein *klarer Favorit* ausmachen; dafür hätten zwecks Vergleichbarkeit noch Messungen mit höheren Erfassungsraten durchgeführt werden müssen. Die Signalstärkeverläufe wiesen jedoch einen „ruhigeren Verlauf“ auf, waren also weniger von Ausreißern durchsetzt. Als dies erkannt wurde, lieferte der zu dieser Zeit aktuelle Linux-Kernel in der Version

2.6.31 bezüglich des verwendeten WLAN-Adapters nur noch Signalstärkewerte, wobei die Linkgütewerte von diesen nicht länger abwichen. Dieses Verhalten ist nicht nur abhängig von der Kernelversion, sondern hängt auch vom verwendeten WLAN-Treiber ab. Der Autor empfiehlt daher, nach Erhalt der ersten Messreihen zur Bestimmung der Entscheidungsschwellen *beide* Varianten zu testen. Im vorliegenden Fall fiel die Wahl zwangsweise auf die Auswertung von Signalstärkewerten.

Proaktiver vs. reaktiver vs. gieriger Entscheider

Eine Prognose in die Zukunft kann keine idealen Ergebnisse liefern. Mit Fehlentscheidungen, wie unnötigerweise aufgebaute Ersatzlinks oder zu spät angestoßene Assoziationen, muss gerechnet werden.

Einerseits konnte allerdings verdeutlicht werden, dass durch den vorgestellten proaktiven Handoverentscheider weniger Ressourcen belegt werden als durch den „gierigen“ Entscheider, bei welchem ein Ersatzlink vorsorglich dauerhaft vorgehalten wird.

Andererseits sind beim proaktiven Entscheider die Zeitspannen ohne Ersatzlink bei unbrauchbar gewordenem Hauptlink geringer als bei einem rein reaktiven Entscheider. Es ist erkennbar, dass in den meisten Fällen wenigstens ein kleiner Zeitvorteil besteht, also die Assoziation des Ersatzlink wenigstens ein paar Sekunden noch vor Abriss des Hauptlinks angestoßen wird. Dieser „Vorlauf“ stellt den eigentlichen Vorteil des vorgestellten proaktiven Entscheiders gegenüber einem reaktiven Entscheider dar.

Welcher Entscheider jedoch der „ideale Entscheider“ ist, kann nur in Verbindung mit einer weiteren nutzerseitigen Zielstellung entschieden werden. Eine möglichst ununterbrochene Verbindung zum Internet kann nur durch eine größtmögliche Belegung von Ressourcen anvisiert werden. Eine minimale Nutzung des Ersatzlinks (beispielsweise bei einem „teuren“ Ersatzlink) erreicht auch ein rein reaktiv arbeitender Entscheider. Der vorgestellte proaktiv arbeitende Entscheider ist damit zwischen den beiden Varianten angesiedelt. Er ist in Bezug auf die Ersatzlinkverfügbarkeit niemals schlechter als der reaktive Entscheider, und in Bezug auf die Ressourcenbelegung niemals schlechter als der „gierige“ Entscheider.

Hohe vs. niedrige Rate der Messdatenerfassung

Die Erfassung der aktuellen Signalstärke oder Linkgüte können bereits mit einer Rate von einem Wert pro Sekunde brauchbare Ergebnisse liefern, was durch die Auswertung der Messreihen 1 bis 6 gezeigt worden ist. Zudem war der dabei benötigte Rechenaufwand so gering, dass es zu keiner spürbaren Belastung kam. Für einen Vergleich ist es jedoch

nachteilig, dass alle sechs Messreihen außerhalb von Gebäuden aufgenommen worden sind. Bei einer „Indoor“-Messung können sich die Eigenschaften der Funkverbindung nämlich schneller ändern als draußen. Dabei besteht die Gefahr, dass der Entscheider nicht schnell genug reagiert, da die Dichte der Messwerte zu gering ist und zudem eine rapide Verschlechterung durch die Ausreißerbehandlung maskiert werden kann.

Die Empfindlichkeit der Regressionsrechnung gegenüber Rauschen und Fading wird mit wachsender Dichte der Messwerte geringer, und es können dann auch zeitlich kurze Ereignisse durch mehrere Messwerte abgebildet werden. Der resultierende Vorteil ist beim Vergleich der Messreihen 7 bis 10 gegenüber den Messreihen 1 bis 6 erkennbar. Mit steigender Erfassungsrate wächst allerdings auch die Anzahl der Messwerte im Fenster, was zu einem höheren Berechnungsaufwand führt.

Die anfangs gewählte Rate von einem Wert pro Sekunde wird vom Autor als zu niedrig bewertet. Die später gewählte Rate von 50 Werten pro Sekunde führte dann zwar zu einer weitaus besseren Filterung, allerdings war der Rechenaufwand enorm: Die Berechnung der Diagramme der 67s langen Messreihe 9 benötigte für das 1500 Messwerte lange Fenster einen Zeitaufwand von knapp 25 Minuten, durchgeführt auf dem leistungsfähigen Arbeitsplatz-PC des Autors. Ein solcher Rechenaufwand einzig für Prognosezwecke ist nicht praxistauglich, und wie sich zeigte, auch unnötig. Nachdem in den Messreihen 8 und 10 jeweils vier von fünf Messwerten ignoriert wurden, also die Erfassungsrate nachträglich auf 10 Messwerte pro Sekunde reduziert worden ist, waren sowohl die Ergebnisse noch brauchbar als auch der Rechenaufwand unkritisch. Daher empfiehlt der Autor eine Erfassungsrate von 10 Messwerten pro Sekunde.

Große vs. kleine zeitliche Ausdehnung des Fensters

Um die Auswirkung der Fenstergröße auf das Ergebnis der Regressionsrechnung zu beurteilen, lohnt sich ein Blick auf die Messreihen 7 bis 10. Diese wurden jeweils mit einer Fenstergröße von 12, 20 und 30 Sekunden ausgewertet, durchgängig mit einem minimalen Füllstand von 10 Sekunden.

Bei allen vier Messreihen zeigte sich der folgende Effekt: Je geringer die zeitliche Ausdehnung des Fensters war, desto eher wurde der Ersatzlink angefordert. Man kann dies jeweils an den Ersatzlinknutzungsdiagrammen erkennen, wenn man die Zeitpunkte des Assoziationskommandos miteinander vergleicht. Es werden zwei Ursachen für diesen Effekt verantwortlich gemacht: „Rauschen“ und „Trägheit“.

Zur Verdeutlichung des „Rauscheffektes“ bietet sich Messreihe 10 an. Beim „kürzesten“ Fenster von 12 Sekunden ist im Restzeitdiagramm (oberes Diagramm in Abbildung D.64 auf Seite 413) ein derart unruhiger Verlauf erkennbar, dass die Entscheidungsschwelle

von 22s mehrfach unterlaufen wurde. Der Ersatzlink wurde daher sehr früh aufgebaut und die Ressourcenbelegung war vergleichsweise hoch. Positiverweise war der Ersatzlink aber immer dann verfügbar, wenn er benötigt wurde. Mit steigender Fenstergröße (Diagramme auf den Seiten 415 und 417) „beruhigte“ sich der gezeigte Verlauf zwar, aber die Zeitpunkte der Anforderung rückten dabei in die Zukunft.

Mit „Trägheit“ ist die Eigenschaft eines Handoverentscheiders gemeint, sich weniger gut auf wechselnde Rahmenbedingungen einzustellen zu können. Die Ursache für Trägheit liegt in der Berücksichtigung vergangener Messwerte begründet. Je größer das zeitliche Fenster ist, desto älter sind die in die Regressionsrechung einfließenden Messwerte. Bei gekrümmten Verläufen sorgen die alten Werte damit tendenziell dafür, dass die Regressionsgerade nicht schnell genug mitschwenkt. Damit wird ein Schnittpunkt mit der Abrissschwelle berechnet, welcher zu weit in der Zukunft liegt. Mit weiter voranschreitender Zeit schmiegt sich die Regressionsgerade immer weiter an den realen Verlauf an, sodass der errechnete Abrisszeitpunkt zusätzlich näher heranrückt. Dieser Effekt ist jeweils an den Restzeitdiagrammen erkennbar, welche mit zunehmender Fenstergröße immer steiler verlaufen. Die Assoziationsentscheidung wird damit zu spät gefällt, und der Entscheider liefert keine guten Ergebnisse mehr.

Eine Fenstergröße von 30 Sekunden ist damit für WLAN zu lang. Zwischen 12 Sekunden und 20 Sekunden ist in den gezeigten Beispielen jedoch kaum ein Unterschied feststellbar, beide reagieren hinreichend „flink“. Bei Messreihe 10 ist auf Seite 412 in Abbildung D.63 jedoch erkennbar, dass die Diagramme bei einer Fenstergröße von 12s „unruhig“ verlaufen. Daraus wird gefolgert, dass ein Beobachtungshorizont von 12 Sekunden für die bei „Outdoor“-Verhältnissen typisch langsamen Änderungen nicht ausreichend ist. Statt des gewünschten langfristigen Verlaufs fließen hier verstärkt auch kurzfristige Änderungen mit in die Abrissprognose ein, was zu einer „unruhigen“ Prognose führt. Der Autor empfiehlt daher eine Fenstergröße von 20 Sekunden. In Verbindung mit der bereits vorgeschlagenen Erfassungsrate von 10 Messwerten pro Sekunde ist der Rechenaufwand vernachlässigbar, und es sind sowohl innerhalb als auch außerhalb von Gebäuden brauchbare Ergebnisse zu erwarten.

E Betrachtung des Overheads

In Kapitel 8 wurden die Betriebsmodi „harter Handover“, Redundanzerhöhung und Kanalbündelung vorgestellt und miteinander verglichen. Aus Platzgründen wurde allerdings bislang keine Betrachtung des Overheads, also des durch REACH *zusätzlich* verursachten Datenübertragungsaufwands, durchgeführt. Eine diesbezügliche Betrachtung wird in diesem Kapitel nachgeholt.

Es handelt sich im Folgenden um ergänzende Diagramme zu den Messreihen der bereits in Abschnitt 8.2 vorgestellten Szenarien. Eine ausführliche Vorstellung des Testaufbaus und des Testablaufs kann dort nachgelesen werden.

Problematik der Berechnung des Overheads

Vorsicht ist bei Betrachtung der nachfolgenden Abbildungen E.5, E.10 und E.15 geboten. Diese sollen den Umfang des Overheads (Verwaltungsinformationen und redundante Nutzdaten) bezüglich des Empfangsdatenstroms am mobilen Endgerät verdeutlichen. Betrachtet man sich insbesondere die Diagramme E.5 und E.15, fällt eine zeitweise negative Datenübertragungsrate auf. Der Grund für diesen sonderbaren Umstand liegt in der verwendeten Berechnungsvorschrift begründet. Der REACH-Client erfasst nämlich den anfallenden Overhead nicht direkt, sondern berechnet ihn aus der empfangenen Gesamtdatenmenge (hier: der Umfang aller empfangenen SPPR-PDUs), von welcher die Menge der schlussendlich abgerufenen Nutzdaten abgezogen wird.

Wird eine SPPR-PDU ausgewertet, erhöht sich augenblicklich das empfangene Gesamtdatenmenge. Einer SPPR-PDU kann allerdings noch nicht angesehen werden, wie groß ihr Anteil am Overhead ist. Während der Paketkopf bereits eindeutig dem Overhead zugerechnet werden dürfte, kann das Nutzdatenfeld noch nicht klassifiziert werden. Schließlich könnte das enthaltene Datenfragment bereits empfangen worden sein, dann wäre auch das Nutzdatenfeld als Overhead zu werten. Aber selbst wenn das Datenfragment „neu“ wäre, könnte auf dieser Ebene nicht festgestellt werden, wie groß der Anteil am Overhead sein würde. Schließlich beschreiben die Datenfragmente ihrerseits einen Strom aus SCPR-PDUs, welcher wiederum eine Mischung aus Paketköpfen, Nutzdatenfeldern und Signalisierungsblöcken darstellt. Der Overhead kann erst dann bestimmt

werden, wenn die Daten durch alle Schichten von REACH „gewandert“ sind. Schlussendlich muss sämtliches Datenvolumen als Verwaltungsaufwand angesehen werden, welches durch REACH selbst ausgewertet, also nicht an die Anwendungen weitergereicht worden ist.

Die Idee hinter dieser Berechnungsvorschrift ist zwar korrekt, jedoch manifestiert sich hierbei ein Problem, welches zu den dargestellten negativen Datenraten führte. Zwischen Empfang einer SPPR-PDU und dem Abruf der enthaltenen Nutzdaten durch die Anwendungen liegt ein zeitlicher Versatz. Dieser tritt besonders dann zu Tage, wenn der SPPR-Empfangsdatenstrom Lücken aufweist, und ein erhaltenes Datenfragment zwar vermerkt, jedoch noch nicht „hochgereicht“ werden kann.

Wird in diesem Fall eine SPPR-PDU empfangen (das enthaltene Fragment sei „neu“, liegt jedoch nicht lückenlos an), trägt es unmittelbar zum Anwachsen der Gesamtdatenmenge bei, jedoch ist noch kein Abruf weiterer Anwendungsdaten möglich. Gemäß der verwendeten Berechnungsvorschrift würde die PDU für den Moment zu 100% dem Overhead zugerechnet werden, und die momentane Datenrate des Overheads würde fälschlicherweise als zu hoch angenommen werden. Wird zu einem späteren Zeitpunkt die Lücke im SPPR-Datenstrom geschlossen, stehen plötzlich SCPR-PDUs zur Auswertung bereit, welche einen signifikanten Anteil an Nutzdaten transportieren können. Es kann passieren, dass *eine* SPPR-PDU empfangen wird, welche eine Lücke schließt, sodass plötzlich *viele* wartende SCPR-PDUs ausgewertet werden können. Die Menge der ausgelieferten Anwendungsdaten erhöht sich, und kann dabei die Größe der lückenschließenden SPPR-PDU überschreiten. Der REACH-Client würde in diesem Fall eine negative Datenrate bezüglich des Overheads errechnen, da im vorliegenden Betrachtungsintervall mehr Anwendungsdaten zugestellt werden konnten, als seitens der Partnerinstanz empfangen worden sind. In Wahrheit wurde jedoch nur die ursprünglich zu hoch errechnete Menge an Overhead nach unten korrigiert.

Nichtsdestotrotz sind die Diagramme geeignet, die anfallende Menge an Overhead qualitativ zu verdeutlichen. Abbildung E.10, welche den empfangenen Overhead im Szenario einer Redundanzerhöhung darstellt, unterstreicht diese Aussage.

Bei Betrachtung des senderseitigen Overheads tritt das beschriebene Problem ebenfalls auf. Da im dargestellten Szenario jedoch lediglich ein Download stattfand, also keinerlei Nutzdaten versendet worden sind, trat der Effekt nicht zu Tage. Jede versendete SPPR-PDU wurde somit zu 100% dem Overhead zugerechnet.

E.1 „Harter Handover“

In diesem Abschnitt wird das Szenario des „harten Handovers“, welches in Abschnitt 8.2.1 des Hauptteils vorgestellt worden ist, in Bezug auf den angefallenen Overhead hin untersucht. Abbildung E.1 ist identisch mit der bereits gezeigten Abbildung 8.6. Sie wurde aus Gründen der Übersichtlichkeit erneut aufgeführt.

Auffällig war der Einbruch der Datenrate zum Zeitpunkt 30s, was typisch für den Betriebsmodus des „harten Handovers“ ist. In den Abbildungen E.2 und E.3 ist erkennbar, dass immer nur ein Netzzugangsgerät aktiv war. Die kurzzeitige Überschneidung der Linien in den linken Diagrammen zum Zeitpunkt 32s ist hingegen auf Laufzeiteffekte innerhalb des REACH-Clients zurückzuführen.

Beim Vergleich des rechten Diagramms aus Abbildung E.1 mit den rechten Diagrammen aus den Abbildungen E.4 und E.5 wird deutlich, dass zusätzlich zu den 950 MB empfangenen Nutzdaten circa 550 kB durch REACH verursachten Overhead empfangen und 800 kB versendet worden sind. Der Overhead belief sich in der Größenordnung von ein Promille, bezogen auf den TCP-Nutzdatenstrom. Eine umfassende Betrachtung des Übertragungsaufwands müsste aber eigentlich auf einer tieferen Ebene ansetzen, da bislang nicht berücksichtigt wurde, dass auf Ebene von TCP zusätzliche Segmente und Quittungen anfallen können.

Datenstrom ohne Overhead

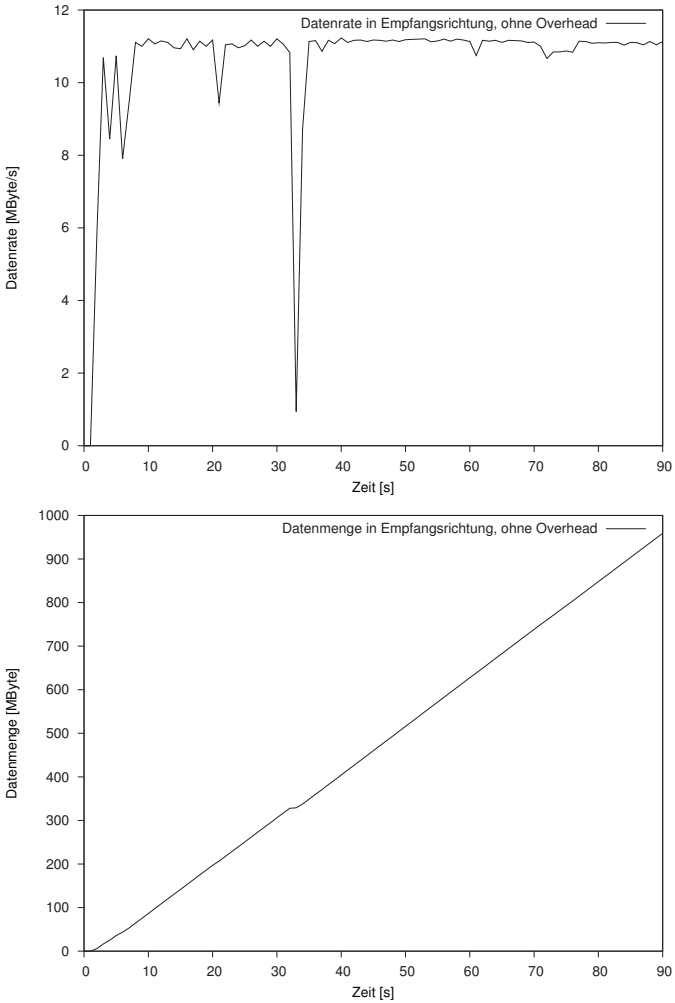

Abbildung E.1: Empfängerseitige Nutzdatenrate und Nutzdatenmenge, jeweils ohne Overhead

Datenstrom mit Overhead, senderseitig

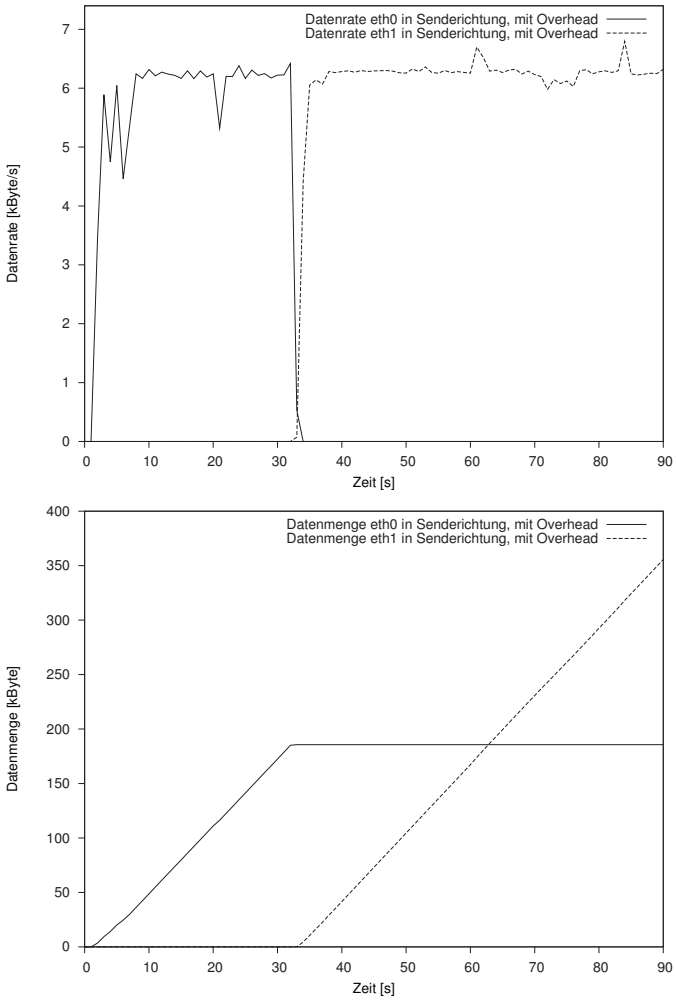

Abbildung E.2: Senderseitige Gesamtdatenrate und Gesamtdatenmenge, jeweils mit Overhead. Beide Interfaces (`eth0` und `eth1`) werden einzeln aufgeführt. Da keinerlei Nutzdaten *versendet* worden sind, war das gesamte Datenaufkommen als Overhead zu klassifizieren. Es genügte, die angefallenen Datenmengen in der Größenordnung Kilobyte (kB) anzugeben.

Datenstrom mit Overhead, empfängerseitig

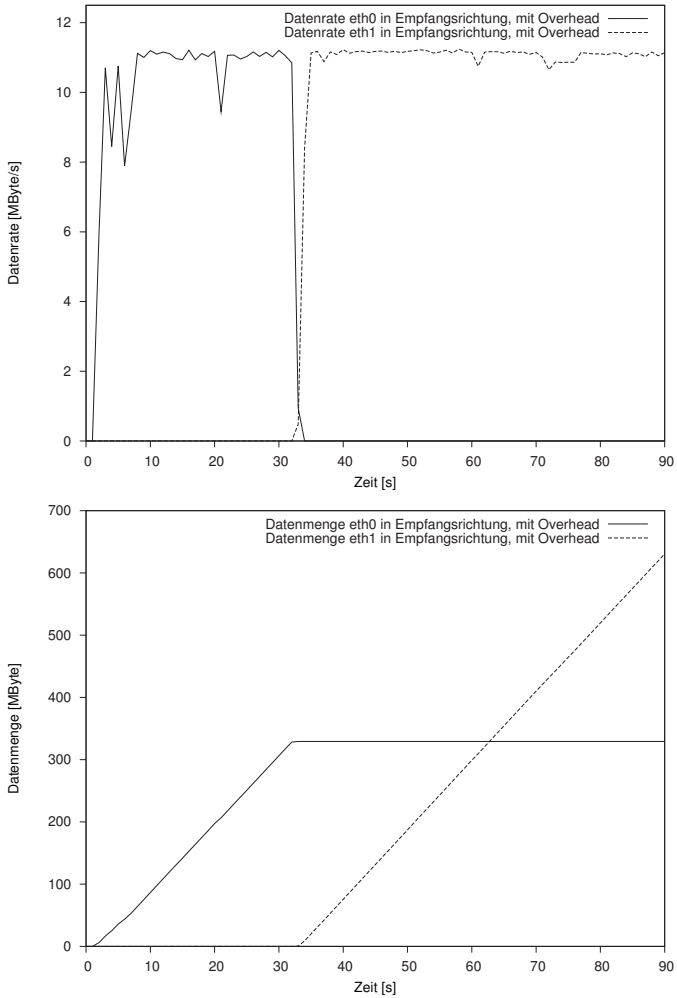

Abbildung E.3: Empfängerseitige Gesamtdatenrate und Gesamtdatenmenge, jeweils mit Overhead. Beide Interfaces (`eth0` und `eth1`) werden einzeln aufgeführt. Hierbei handelt es sich tatsächlich um eine Mischung aus Nutzdaten („der Download“) und dem durch REACH verursachten Overhead.

Nur Overhead, senderseitig

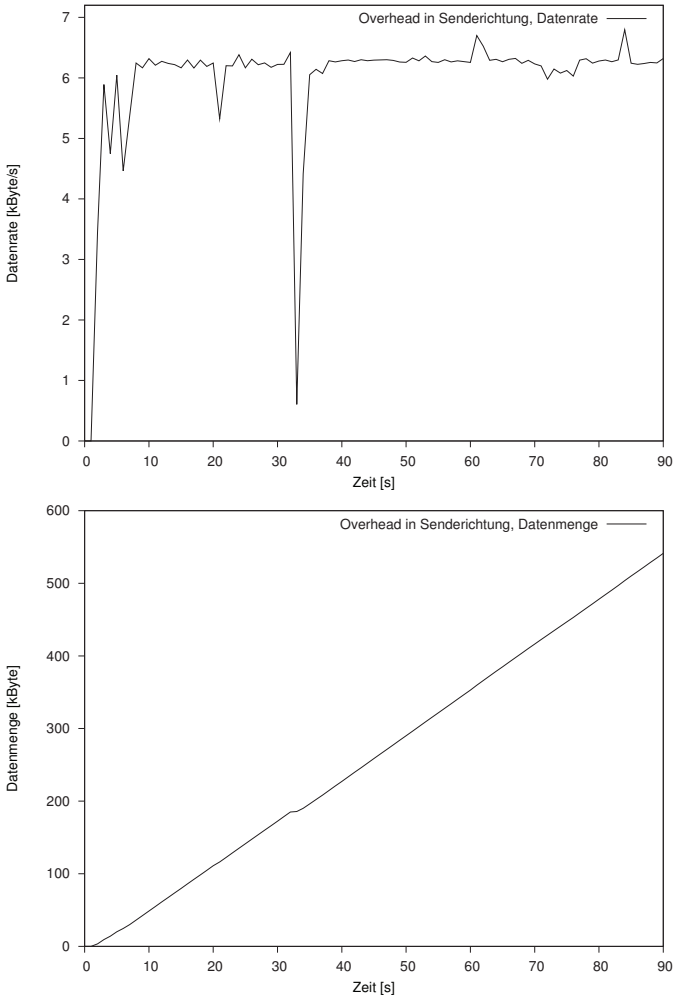

Abbildung E.4: Senderseitiger Overhead, Datenrate und Datenmenge. Das angefallene Datenvolumen wurde in der Größenordnung Kilobyte angegeben.

Nur Overhead, empfängerseitig

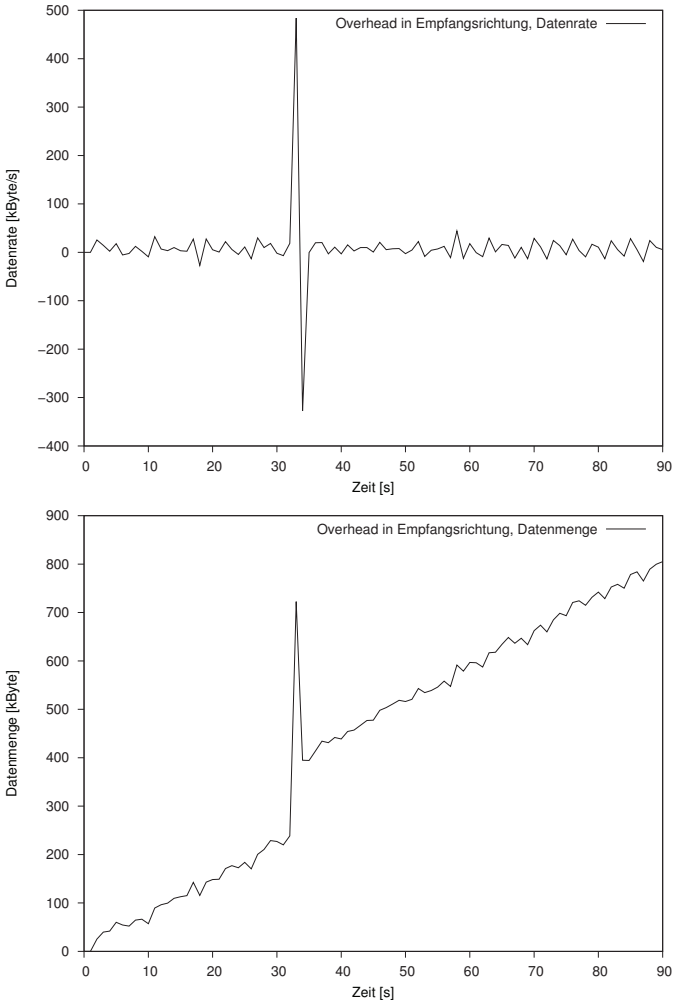

Abbildung E.5: Empfängerseitiger Overhead, Datenrate und Datenmenge. Das angefallene Datenvolumen wurde in der Größenordnung Kilobyte angegeben. (Achtung: Bitte Anmerkungen im Text beachten)

E.2 Redundanzerhöhung

In Abschnitt 8.2.2 wurde der redundante Übertragungsmodus von REACH untersucht. Abbildung E.6, welche den Inhalt aus Abbildung 8.7 wiederholt, zeigt die gemessene Nutzdatenrate sowie den Anstieg der empfangenen Nutzdatenmenge.

Zum Zeitpunkt 30s wurde ein zusätzliches Netzzugangsgerät aktiviert und mit in die Datenübertragung einbezogen. Dabei wurden identische Nutzdaten über unterschiedliche Wege versendet. Die Diagramme aus Abbildung E.6 zeigen, dass dadurch ein Absinken der Datenrate bei einem spontanen Verlust eines Netzzugangs (zum Zeitpunkt 60s) verhindert werden konnte.

Die Diagramme aus Abbildung E.7 untermauern, das hierbei eine redundante Versendung über beide Wege stattfand. Jeder Weg war ca. 60 Sekunden lang aktiv, und schlussendlich wurde über beide Wege annähernd die gleiche Datenmenge versendet bzw. empfangen (dargestellt in Abbildung E.8). Laut Abbildung E.9 hatte sich die Senderate während der Phase der redundanten Datenübertragung näherungsweise verdoppelt, was daran lag, dass jede Bestätigung immer über beide Wege versendet worden war. Betrachtet man die Menge des empfängerseitigen Overheads in Abbildung E.10, wird deutlich, dass das Datenaufkommen eines Weges (mit ca. 11 MB/s) redundant war. Im Zeitintervall der mehrfachen Datenübertragung ist ein redundantes Datenvolumen von ca. 310 MB angefallen.

Datenstrom ohne Overhead

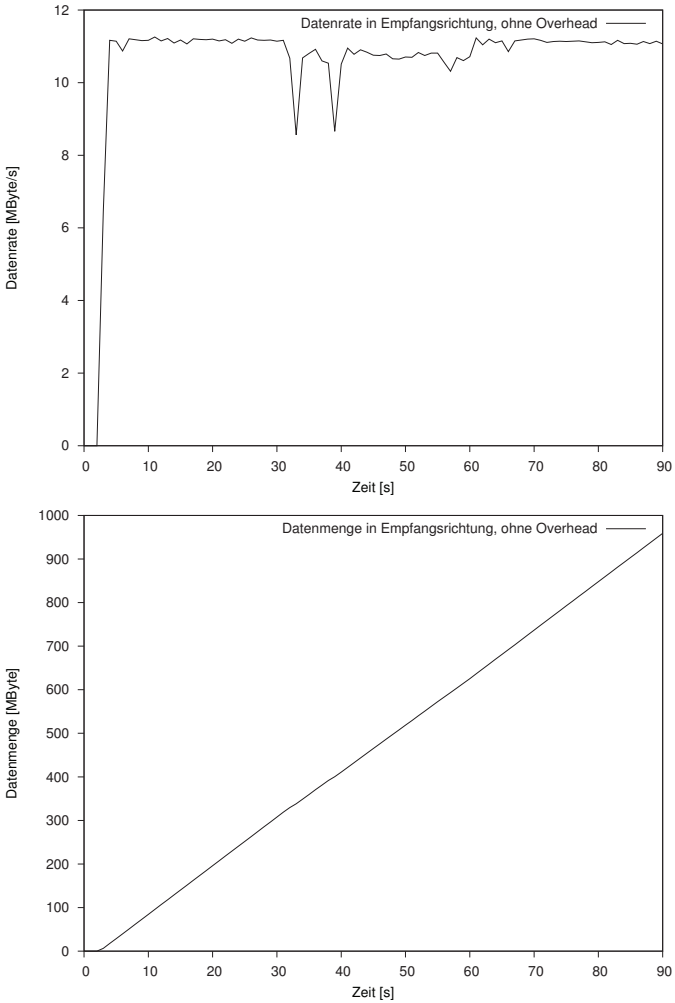

Abbildung E.6: Empfängerseitige Nutzdatenrate und Nutzdatenmenge, jeweils ohne Overhead

Datenstrom mit Overhead, senderseitig

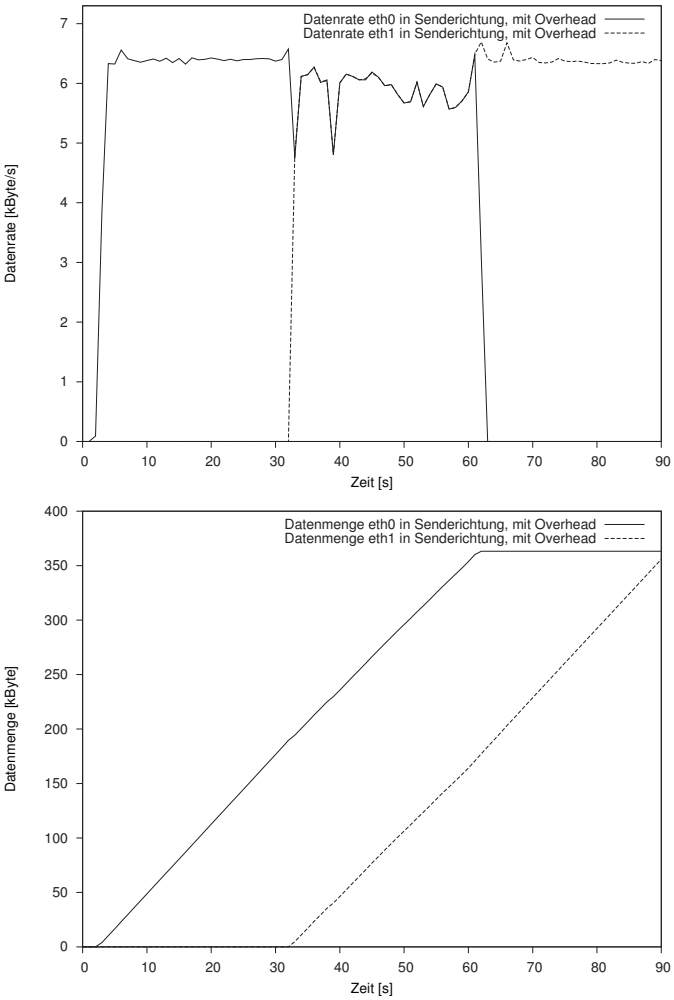

Abbildung E.7: Senderseitige Gesamtdatenrate und Gesamtdatenmenge, jeweils mit Overhead. Beide Interfaces (`eth0` und `eth1`) werden einzeln aufgeführt. Da keinerlei Nutzdaten *versendet* worden sind, war das gesamte Datenaufkommen als Overhead zu klassifizieren. Es genügte, die angefallenen Datenmengen in der Größenordnung Kilobyte anzugeben.

Datenstrom mit Overhead, empfängerseitig

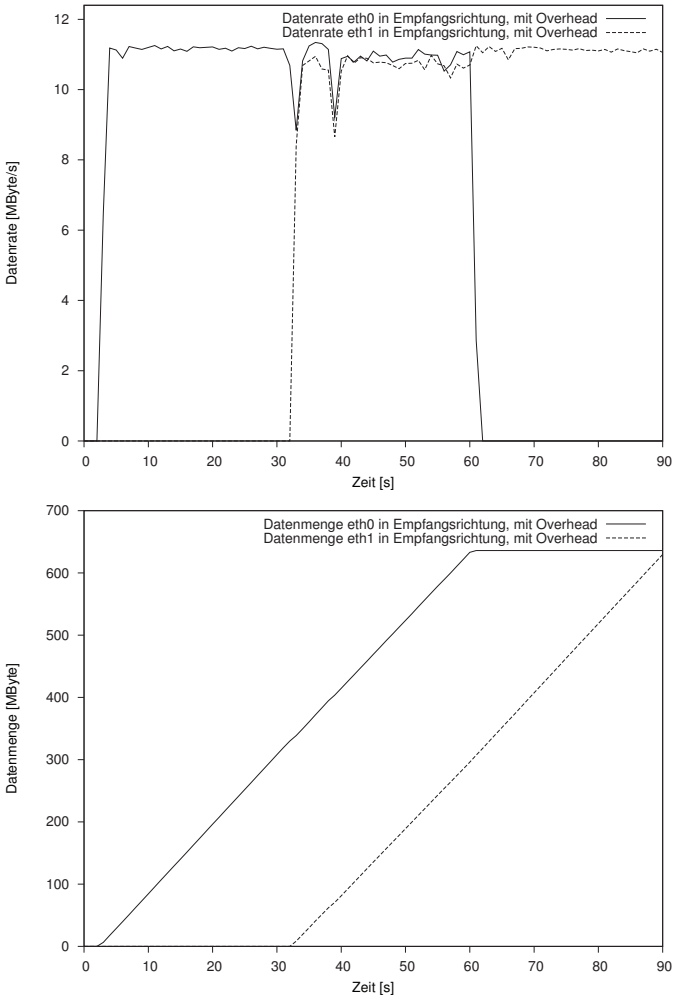

Abbildung E.8: Empfängerseitige Gesamtdatenrate und Gesamtdatenmenge, jeweils mit Overhead. Beide Interfaces (`eth0` und `eth1`) werden einzeln aufgeführt. Hierbei handelt es sich tatsächlich um eine Mischung aus Nutzdaten („der Download“) und dem durch REACH verursachten Overhead.

Nur Overhead, senderseitig

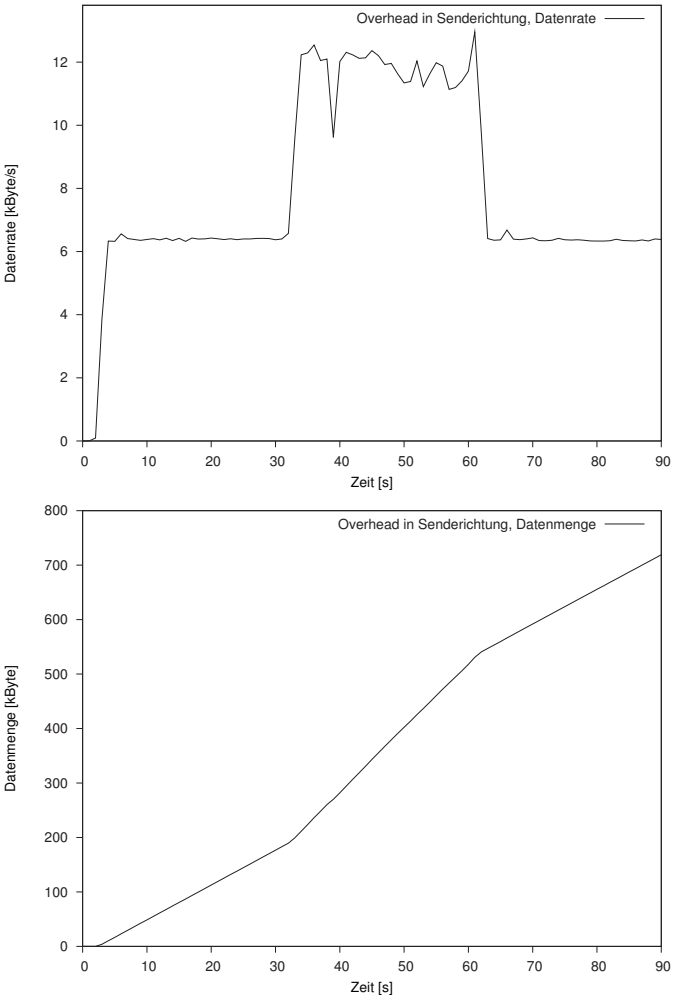

Abbildung E.9: Senderseitiger Overhead, Datenrate und Datenmenge. Das angefallene Datenvolumen wurde in der Größenordnung Kilobyte angegeben.

Nur Overhead, empfängerseitig

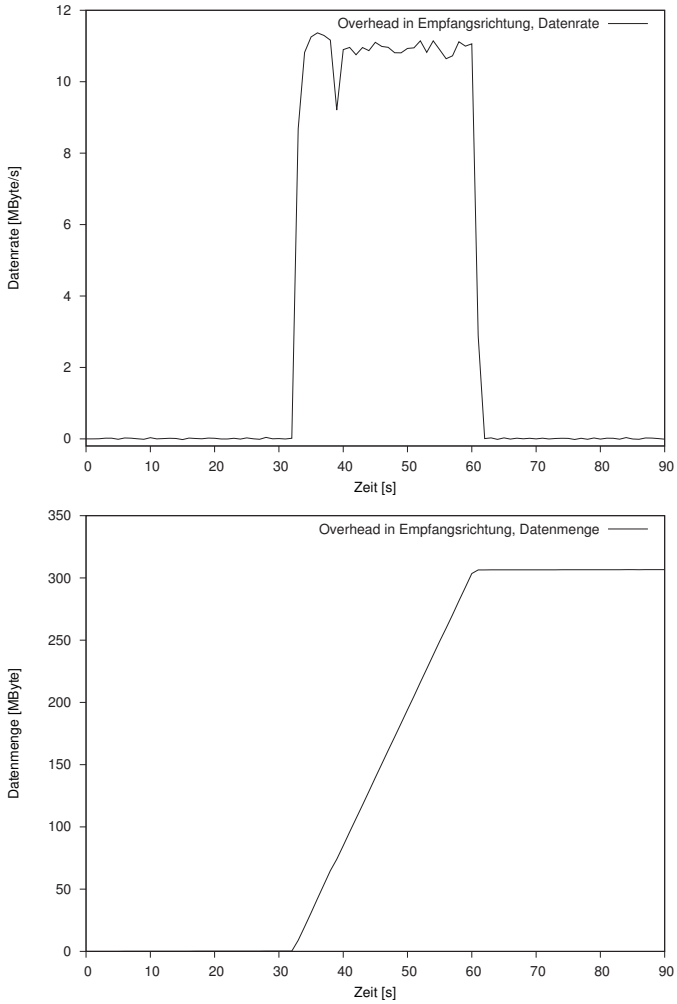

Abbildung E.10: Empfängerseitiger Overhead, Datenrate und Datenmenge. Auf Grund der redundaten Übertragung großer Datenmengen war es notwendig, den Overhead in der Größenordnung MByte anzugeben.
(Achtung: Bitte Anmerkungen im Text beachten)

E.3 Kanalbündelung

Die folgenden Diagramme dienen als Ergänzung zu Abschnitt 8.2.3, in welchem ein Kanalbündelungsszenario vorstellt worden ist. Abbildung E.11, welche bereits im Hauptteil als Abbildung 8.8 vorgestellt worden ist, zeigt die hierbei erreichte Nutzdatenrate und die übertragene Nutzdatenmenge.

Im Zeitraum zwischen 30s und 60s konnte eine höhere Datenübertragungsrate erreicht werden, als es über eine der auf Ethernet basierenden Übertragungsstrecken alleine möglich gewesen wäre. Bei Betrachtung der Abbildungen E.12 und E.13 wird jedoch deutlich, dass keine Verdoppelung der Nutzdatenrate erreicht worden ist. Als Ursache hierfür vermutet der Autor Laufzeiteffekte in den Flusssteuerungsmechanismen von REACH, was allerdings bislang nicht weiter untersucht worden ist.

Aus den Abbildungen E.14 und E.15 geht hervor, dass während der Kanalbündelung die Rate, mit welcher der Overhead anfiel, geringer geworden war. Einerseits liegt dies darin begründet, dass die Gesamtdatenrate nicht das anvisierte Maximum erreicht hatte. Zudem wird vermutet, dass die bislang ungeklärte Effekte der Flusssteuerungsmechanismen dafür sorgten, dass seltener Bestätigungsnachrichten verschickt worden waren und sich die resultierenden Sammelquittungen als „effizienter“ erwiesen.

Schlussendlich muss festgestellt werden, dass die Kanalbündelung zwar funktionierte, jedoch bislang kein „ideales Verhalten“ beobachtet werden konnte. Hierfür ist eine tiefgreifende Analyse des Laufzeitverhaltens von REACH notwendig, um die Ursache für die gedrosselte Datenrate ausfindig zu machen und zu beheben.

Datenstrom ohne Overhead

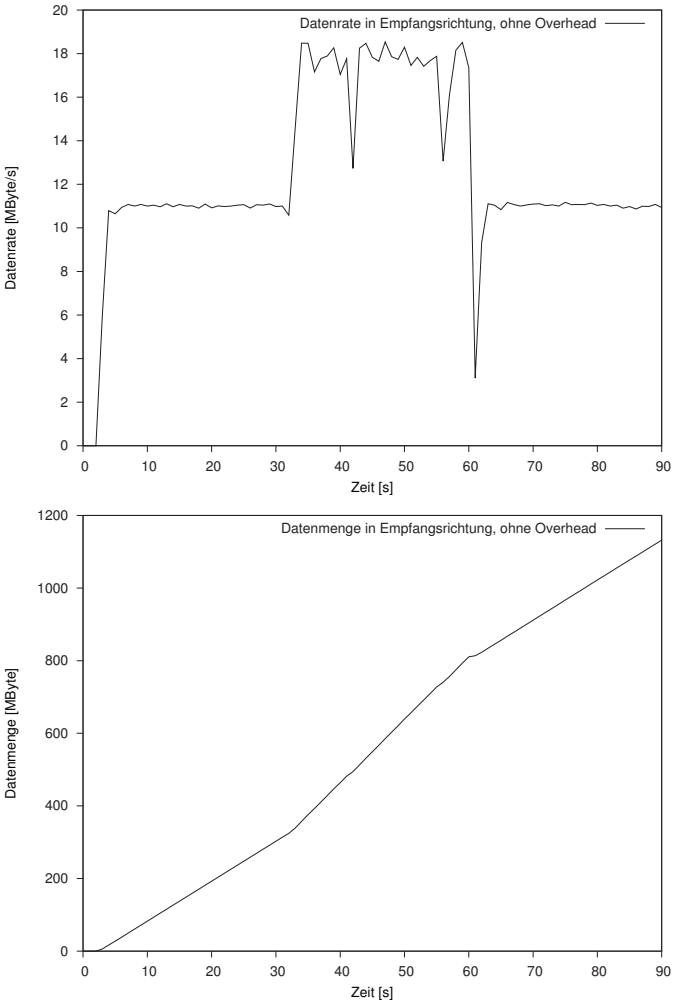

Abbildung E.11: Empfängerseitige Nutzdatenrate und Nutzdatenmenge, jeweils ohne Overhead

Datenstrom mit Overhead, senderseitig

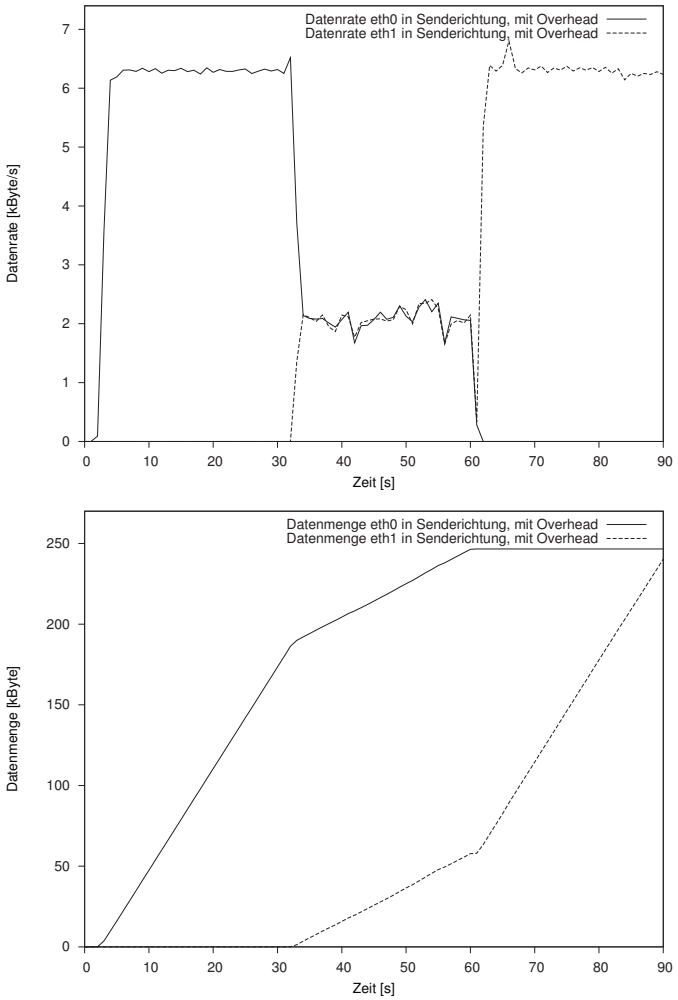

Abbildung E.12: Senderseitige Gesamtdatenrate und Gesamtdatenmenge, jeweils mit Overhead. Beide Interfaces (`eth0` und `eth1`) werden einzeln aufgeführt. Da keinerlei Nutzdaten *versendet* worden sind, war das gesamte Datenaufkommen als Overhead zu klassifizieren. Es genügte, die angefallenen Datenmengen in der Größenordnung Kilobyte anzugeben.

Datenstrom mit Overhead, empfängerseitig

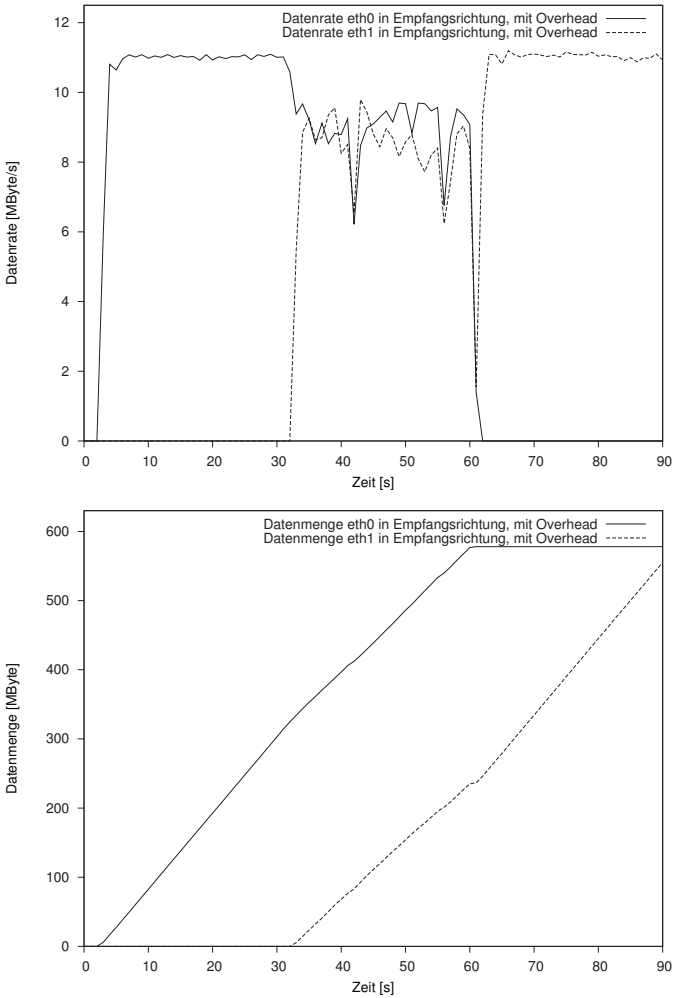

Abbildung E.13: Empfängerseitige Gesamtdatenrate und Gesamtdatenmenge, jeweils mit Overhead. Beide Interfaces (`eth0` und `eth1`) werden einzeln aufgeführt. Hierbei handelt es sich tatsächlich um eine Mischung aus Nutzdaten („der Download“) und dem durch REACH verursachten Overhead.

Nur Overhead, senderseitig

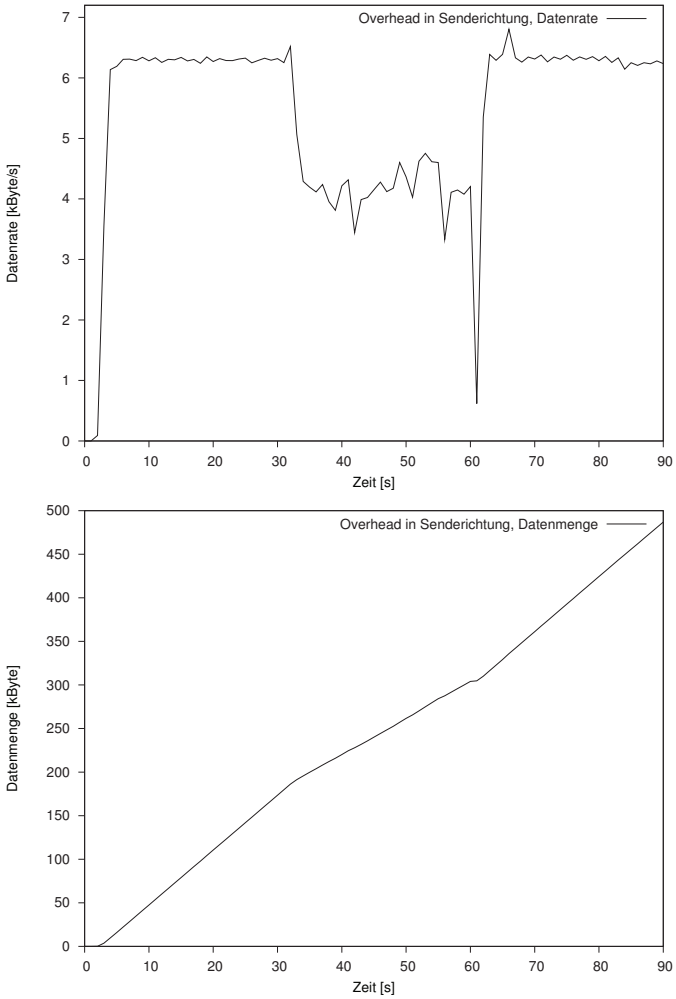

Abbildung E.14: Senderseitiger Overhead, Datenrate und Datenmenge. Das angefallene Datenvolumen wurde in der Größenordnung Kilobyte angegeben.

Nur Overhead, empfängerseitig

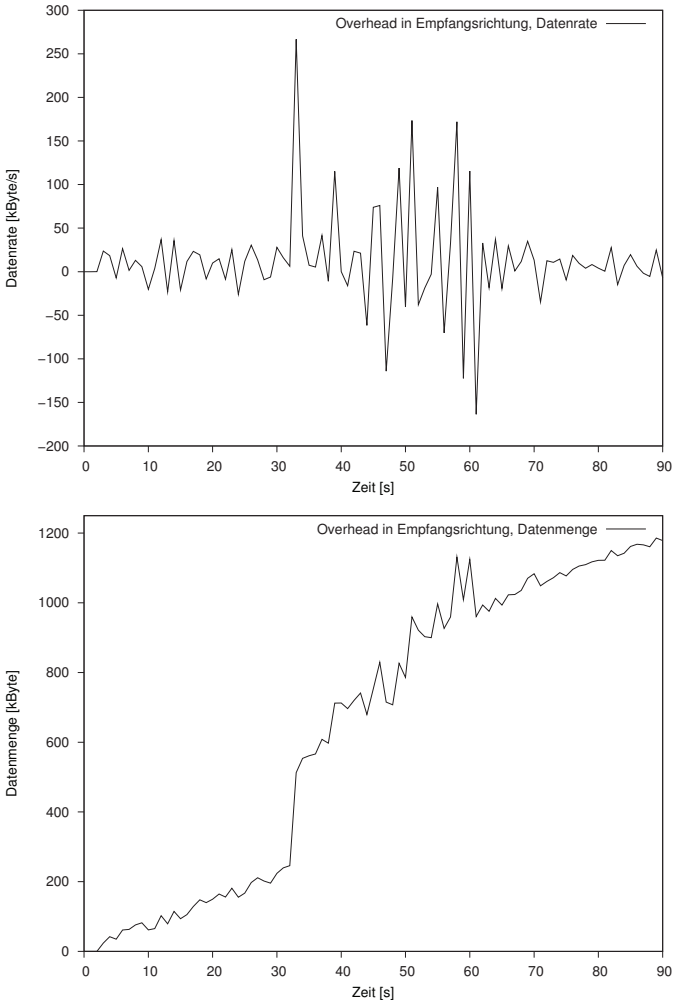

Abbildung E.15: Empfängerseitiger Overhead, Datenrate und Datenmenge. Das angefallene Datenvolumen wurde in der Größenordnung Kilobyte angegeben. (Achtung: Bitte Anmerkungen im Text beachten)

F Systeme der Testumgebung

In diesem Kapitel werden die Systeme der Testumgebung von REACH mit ihren technischen Daten vorgestellt. Die Testumgebung umfasst ein mobiles Endgerät, eine REACH-Box, mehrere REACH-Proxyserver sowie zwei WLAN-Basisstationen. Der bei einer Demonstration für das Publikum „sichtbare“ Aufbau wird in Abbildung F.1 dargestellt. Alle involvierten Systeme, einschließlich der REACH-Proxyserver, werden im Folgenden vorgestellt. Die dabei verwendeten Abkürzungen können im Abkürzungsverzeichnis nachgeschlagen werden.

Abbildung F.1: Der Testaufbau im Büro des Autors, mit einem mobilen Endgerät, der REACH-Box und zwei WLAN-Basisstationen (teilweise verdeckt)

F.1 Der Laptop als mobiles Endgerät

Abbildung F.2: Als mobiles Endgerät kam dieser Laptop zum Einsatz. Er konnte mit zwei Ethernetschnittstellen, zwei WLAN-Adaptern und einem Bluetooth-Adapter zwecks Kopplung an ein Handy ausgestattet werden.

Rechnername:	`neelix.e-technik.tu-ilmenau.de`
Einsatzzweck:	Mobiles Endgerät
Betriebssystem:	Gentoo GNU/Linux
Prozessor:	2,2 GHz Intel Pentium 4 Mobile
Arbeitsspeicher:	768 MiB
Ethernet:	1x Realtek RTL-8139 10/100 (PCI)
	1x Realtek RTL-8139 10/100 (PCMCIA)
USB-Anschlüsse:	4x USB 1.1
	2x USB 2.0 mittels PCMCIA-Adapter, NEC
WLAN:	Intersil Prism 2.5; 2,4 GHz IEEE 802.11b
	2x Zyxel G-220 v2 USB (ZD1211); 2,4 GHz IEEE 802.11b/g
Bluetooth:	USB-Adapter Celink BTA-3100
GPRS/GSM:	Mobiltelefon Siemens S55, Zugriff über Bluetooth
Seriell:	9-poliger Anschluss

Tabelle F.1: Die technischen Daten des mobilen Endgeräts

F.2 Die REACH-Box

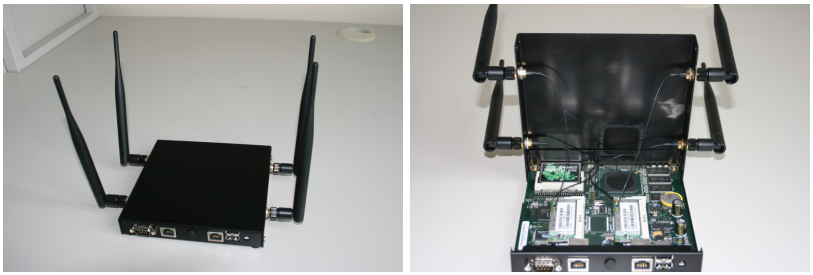

Abbildung F.3: Als REACH-Box kam ein Industrie-PC zum Einsatz. Dieser wurde mit zwei Ethernetschnittstellen und zwei WLAN-Adaptern ausgestattet. Zudem besaß er zwei USB-Schnittstellen zum Anschluss weiterer Netzzugangsgeräte.

Rechnername:	`reachbox`, ohne Eintrag im DNS
Einsatzzweck:	REACH-Box als „Vorschaltgerät“
Betriebssystem:	Gentoo GNU/Linux
Bezeichnung:	PC Engines ALIX.2D2
Prozessor:	500 MHz AMD Geode LX800
Arbeitsspeicher:	256 MiB DDR DRAM
Massenspeicher:	CompactFlash-Anschluss mit 44 Pins, IDE
bestückt mit:	8 GB CF Kingston Elite Pro 133x
Interne Anschlüsse:	2x MiniPCI
bestückt mit:	2x Wistron DCMA-81 MiniPCI; 2,4 / 5 GHz IEEE 802.11a/b/g
Ethernet:	2x VIA VT6105M 10/100
Grafik:	Nicht verfügbar, aber mit 9-poligem Konsolenanschluss
USB-Anschlüsse:	2x USB 2.0
Abmessungen:	152,4 x 152,4 mm
Leistungsaufnahme:	ca. 4-5 Watt (ohne Erweiterungskarten) bei 12 Volt

Tabelle F.2: Die technischen Daten der REACH-Box

F.3 Der REACH-Proxyserver im Büro

Abbildung F.4: Der Arbeitsplatz-PC des Autors konnte zu Testzwecken auch die Aufgaben eines REACH-Proxyservers mit übernehmen.

Rechnername:	`wesley.e-technik.tu-ilmenau.de`
Einsatzzweck:	Entwicklungsumgebung, REACH-Proxyserver
Betriebssystem:	Gentoo GNU/Linux
Prozessor:	3,0 GHz Intel Pentium 4, mit Hyperthreading
Arbeitsspeicher:	2 GiB
Ethernet:	1x ADMtek NC100 10/100
	2x AMD 79c970 10/100

Tabelle F.3: Die technischen Daten des Arbeitsplatz-PCs des Autors

F.4 Der REACH-Proxyserver im Rechnerraum

Abbildung F.5: Die einzige Aufgabe dieses Servers bestand darin, einen REACH-Proxyserver im Campusnetz der TU Ilmenau darzustellen.

Rechnernamen:	`equinox.e-technik.tu-ilmenau.de`
	`defiant.e-technik.tu-ilmenau.de`
	`phoenix.e-technik.tu-ilmenau.de`
	`excelsior.e-technik.tu-ilmenau.de`
Einsatzzweck:	REACH-Proxyserver, VPN-Server
Betriebssystem:	Gentoo GNU/Linux
Prozessor:	2,4 GHz Intel Pentium 4
Arbeitsspeicher:	1 GiB
Ethernet:	1x AMD 79c970 10/100

Tabelle F.4: Die technischen Daten des dedizierten REACH-Proxyservers zeigen, dass dieser mit insgesamt vier öffentlichen IP-Adressen (hier nur mit den zugehörigen DNS-Namen aufgeführt) ausgestattet worden war.

F.5 Der REACH-Proxyserver Daheim

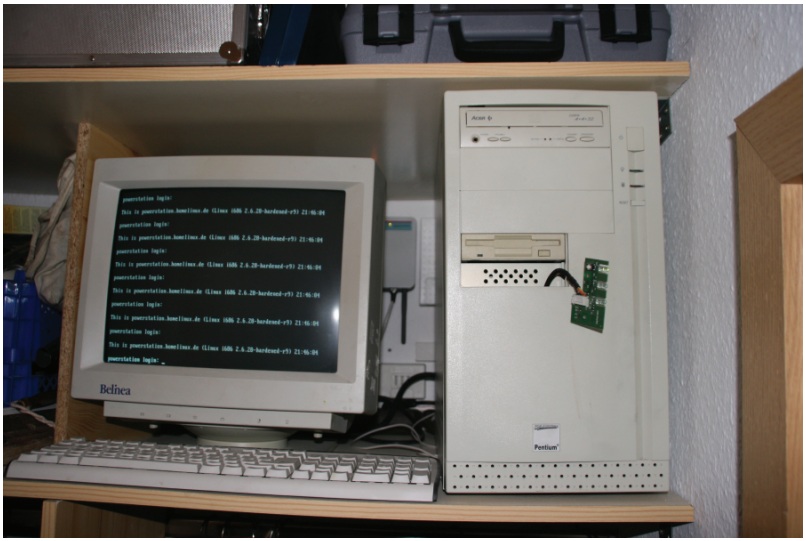

Abbildung F.6: Dieser Server stand in der Wohnung des Autors und stellte ebenfalls einen REACH-Proxyserver dar. Zudem verfügte er über eine Anbindung an das digitale Telefonnetz (ISDN).

Rechnername:	`powerstation.homelinux.org`, hinter einer NAT-Grenze
Einsatzzweck:	REACH-Proxyserver, VoIP-PBX, ISDN-Gateway
Betriebssystem:	Gentoo GNU/Linux
Prozessor:	1 GHz Intel Pentium 3
Arbeitsspeicher:	512 MiB
Ethernet:	3x Intel Ethernet Pro 100
Telefonie:	2x ISDN-PCI-Karte, Chipsatz HFC-S
ADSL:	T-Com Speedport W 500V, 3000/512 kBit/s, Flatrate

Tabelle F.5: Die technischen Daten des privaten REACH-Proxyservers des Autors

F.6 Die WLAN-Basisstationen im Büro

Abbildung F.7: Der Versuchsaufbau schloss diese beiden WLAN-Basisstationen, welche formschön auf einem Brett montiert worden waren, mit ein. Sie boten jeweils einen Zugang zum Internet mittels Ethernet und WLAN an.

Rechnernamen:	`midas1.e-technik.tu-ilmenau.de`
	`midas2.e-technik.tu-ilmenau.de`
Einsatzzweck:	WLAN-Basisstation und Ethernet-Switch
Betriebssystem:	Nicht modifizierte Firmware
Bezeichnung:	ASUS WL-500g Premium
Ethernet:	1x WAN 100 MBit/s
	4x LAN 100 MBit/s
WLAN:	1x 2,4 GHz IEEE 802.11b/g

Tabelle F.6: Die technischen Daten der beiden WLAN-Basisstationen

Literaturverzeichnis

[Ahma07] Ahmad Ahmad. Entwurf und Implementierung eines Handoverentscheiders. Diplomarbeit, Fachgebiet Kommunikationsnetze, Technische Universität Ilmenau, Juni 2007.

[Aste] Digium, Inc., `http://www.asterisk.org`. *Homepage der freien IP-PBX Asterisk*. [Online; Stand 3. Februar 2010].

[AuJe07] F. Audet und C. Jennings. *Network Address Translation (NAT) Behavioral Requirements for Unicast UDP*. Internet Engineering Task Force, `http://www.ietf.org/rfc/rfc4787.txt`, Januar 2007. RFC 4787 (Best Current Practice).

[BaAD06] N. Banerjee, A. Acharya und S.K. Das. Seamless SIP-based mobility for multimedia applications. *IEEE Network* 20(2), März 2006, S. 6–13.

[BaBa95] Ajay Bakre und B. R. Badrinath. I-TCP: Indirect TCP for Mobile Hosts. In *International Conference on Distributed Computing Systems (ICDCS)*. Rutgers University, Piscataway, NJ, Juni 1995.

[BBCG+03] R. Bagrodia, S. Bhattacharyya, F. Cheng, S. Gerding, G. Glazer, R. Guy, Z. Ji, J. Lin, T. Phan, E. Skow, M. Varshney und G. Zorpas. iMASH: Interactive Mobile Application Session Handoff. In *Proceedings of MobiSys*, 2003, S. 259–272.

[BeCF07] Paolo Bellavista, Antonio Corradi und Luca Foschini. Context-aware handoff middleware for transparent service continuity in wireless networks. *Pervasive Mobile Computing* 3(4), 2007, S. 439–466.

[Blue99] The Bluetooth SIG, `http://ece.wpi.edu/analog/resources/bluetooth_a.pdf`. *BLUETOOTH SPECIFICATION Version 1.0 A*, Juli 1999. [Online; Stand 3. Juni 2010].

[BrHW08] Shenja Brückner, Philip Herold und Stefan Wiebel. Anbindung eines Sensornetzwerkes an das Internet mit Hilfe transparenter Proxies. Medienprojekt, Fachgebiet Kommunikationsnetze, Technische Universität Ilmenau, Mai 2008.

[BrSi96] Kevin Brown und Suresh Singh. M-UDP: UDP for Mobile Cellular Networks. *ACM SIGCOMM Computer Communication Review* 26(5), Oktober 1996, S. 60–78.

[BrSi97] Kevin Brown und Suresh Singh. M-TCP: TCP for Mobile Cellular Networks. *ACM SIGCOMM Computer Communication Review* 27(5), Oktober 1997, S. 19–43.

[BTGR] Palo Wireless, `http://www.palowireless.com/INFOTOOTH/tutorial/radio.asp`. *Bluetooth Radio Layer Tutorial*. [Online; Stand 17. November 2009].

[BTPS07] Novell, Inc., `http://www.novell.com/coolsolutions/feature/18684.html`. *How to Set Up a Bluetooth Proximity System*, Februar 2007. [Online; Stand 17. November 2009].

[CDKF+02] B. Cain, S. Deering, I. Kouvelas, B. Fenner und A. Thyagarajan. *Internet Group Management Protocol, Version 3*. Internet Engineering Task Force, `http://www.ietf.org/rfc/rfc3376.txt`, Oktober 2002. RFC 3376 (Proposed Standard), updated by RFC 4604.

[ChHa09] J. Chesterfield und B. Haberman. *Multicast Group Membership Discovery MIB*. Internet Engineering Task Force, `http://www.ietf.org/rfc/rfc5519.txt`, April 2009. RFC 5519 (Proposed Standard).

[ChSG05] Ling-Jyh Chen, Tony Sun und Mario Gerla. USHA: A Practical Vertical Handoff Solution. In *The First International Conference on Multimedia Services Access Networks (MSAN'05)*, Orlando, USA, 2005.

[Clar82] D.D. Clark. *Window and Acknowledgement Strategy in TCP*. Internet Engineering Task Force, `http://www.ietf.org/rfc/rfc813.txt`, Juli 1982. RFC 813.

[CSYG06] Ling-Jyh Chen, Tony Sun, Guang Yang und Mario Gerla. USHA: a simple and practical seamless vertical handoff solution. In *IEEE Consumer Communications and Networking Conference 2006*, Band 2, Las Vegas, Nevada, USA, Februar 2006. S. 1284–1285. Demo paper.

[Dant] Inferno Nettverk A/S, `http://www.inet.no/dante`. *Dante – A Free Socks Implementation*. [Online; Stand 3. Juni 2010].

[dBoe] Michel de Boer. *Homepage des Softphones Twinkle*. `http://www.xs4all.nl/~mfnboer/twinkle`. [Online; Stand 18. Februar 2010].

[DBus] Freedesktop.org, `http://www.freedesktop.org/wiki/Software/dbus`. *What is D-Bus?* [Online; Stand 2. September 2009].

[Deep] *What is „Deep Inspection“?* `http://www.ranum.com/security/computer_security/editorials/deepinspect`. [Online; Stand 3. Juni 2009].

[Deer89] S.E. Deering. *Host extensions for IP multicasting*. Internet Engineering Task Force, `http://www.ietf.org/rfc/rfc1112.txt`, August 1989. RFC 1112 (Standard), updated by RFC 2236.

[Dell08] Sebastian Delling. Service Discovery in globalen Netzen. Hauptseminar, Fachgebiet Kommunikationsnetze, Technische Universität Ilmenau, Juli 2008.

[Diab10] Ali Diab. *Mobility Management in IP-Based Networks – Analysis, Design, Programming and Computer-Based Learning Modules*. Dissertation, Technische Universität Ilmenau, März 2010.

[Dour07] François Dourthe. Extending a Graphical User Interface for a Demonstrator for Vertical Handovers. Students Project, Fachgebiet Kommunikationsnetze, Technische Universität Ilmenau, Juli 2007.

[East] Thomas M. Eastep. *Shoreline Firewall - iptables made easy*. `http://www.shorewall.net`. [Online; Stand 3. Juni 2009].

[Ever04] Florian Evers. Unterstützung eines vertikalen Handovers zwischen GPRS und WLAN durch einen Proxy. Diplomarbeit, Fachgebiet Kommunikationsnetze, Technische Universität Ilmenau, November 2004.

[Ever08] Florian Evers. Mobilität im Internet – Die „Roaming-Enabled Architecture“ (REACH) als Middleware für vertikale Handover. In *10. Ilmenauer TK-Manager Workshop*. Fachgebiet Kommunikationsnetze, Technische Universität Ilmenau, Universitätsverlag Ilmenau, September 2008, S. 65–69. 978–3–939473–33–6.

[EvSe06] Florian Evers und Jochen Seitz. A VPN-driven Infrastructure for Vertical Handovers. In *IEEE Sarnoff Symposium 2006*, Princeton, New Jersey, USA, März 2006. Fachgebiet Kommunikationsnetze, Technische Universität Ilmenau.

[EvSe08] Florian Evers und Jochen Seitz. REACH: A Roaming-Enabled Architecture for Multi-Layer Capturing. In *IEEE Wireless Communications and Networking Conference 2008 (WCNC 2008)*, Las Vegas, Nevada, USA, 2008. Fachgebiet Kommunikationsnetze, Technische Universität Ilmenau. 978–1–4244–1997–5.

[EvYS08] Florian Evers, Yevgeniy Yeryomin und Jochen Seitz. Handover-aware SIP-based VoIP provided by a Roaming-Enabled Architecture (REACH). In *IEEE Sarnoff Symposium 2008*, Princeton, New Jersey, USA, 2008. Fachgebiet Kommunikationsnetze, Technische Universität Ilmenau.

[Fire] Mozilla Foundation, `http://www.mozilla-europe.org/de/firefox`. *Der Webbrowser Firefox*. [Online; Stand 3. Juni 2009].

[GLDC+08] S. Gundavelli, K. Leung, V. Devarapalli, K. Chowdhury und B. Patil. *Proxy Mobile IPv6*. Internet Engineering Task Force, `http://www.ietf.org/rfc/rfc5213.txt`, August 2008. RFC 5213 (Proposed Standard).

[GMH0] LinuxWiki.org, `http://linuxwiki.de/GPRS`. *GPRS Mini-Howto*. [Online; Stand 17. November 2009].

[GuJP02] Eva Gustafsson, Annika Jonsson und Charles E. Perkins. Mobile IPv4 Regional Registration, IETF Internet draft, draft-ietf-mobileip-reg-tunnel-06.txt. *ACM Computer Communication Review*, März 2002.

[HaJP06] M. Handley, V. Jacobson und C. Perkins. *SDP: Session Description Protocol*. Internet Engineering Task Force, `http://www.ietf.org/rfc/rfc4566.txt`, Juli 2006. RFC 4566 (Proposed Standard).

[Hal0] Freedesktop.org, `http://www.freedesktop.org/wiki/Software/hal`. *HAL - Hardware Abstraction Layer*. [Online; Stand 2. September 2009].

[Hess06] André Hesse. Entwurf und Implementierung einer Testumgebung zur Anbindung von Sensornetzwerken an IP-basierte Kommunikationssysteme. Diplomarbeit, Fachgebiet Kommunikationsnetze, Technische Universität Ilmenau, März 2006.

[Hoff08] Danilo Hoffmann. GPRS-Anbindung über Bluetooth-Handys unter Linux. Hauptseminar, Fachgebiet Kommunikationsnetze, Technische Universität Ilmenau, Oktober 2008.

[Hoff10] Danilo Hoffmann. Vollautomatische Ansteuerung von GPRS-fähigen Handys mittels Bluetooth unter Linux. Projektseminar, Fachgebiet Kommunikationsnetze, Technische Universität Ilmenau, Januar 2010.

[HoSr01] M. Holdrege und P. Srisuresh. *Protocol Complications with the IP Network Address Translator*. Internet Engineering Task Force, `http://www.ietf.org/rfc/rfc3027.txt`, Januar 2001. RFC 3027 (Informational).

[HSHS+06] Huaizhong Han, Srinivas Shakkottai, C. V. Hollot, R. Srikant und Don Towsley. Multi-path TCP: A joint congestion control and routing scheme to exploit path diversity in the internet. *IEEE/ACM Transactions on Networking* 14(6), 2006, S. 1260–1271.

[Huit06] C. Huitema. *Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)*. Internet Engineering Task Force, `http://www.ietf.org/rfc/rfc4380.txt`, Februar 2006. RFC 4380 (Proposed Standard).

[Hörn08] Jens Hörnlein. SCTP und DCCP. Hauptseminar, Fachgebiet Kommunikationsnetze, Technische Universität Ilmenau, Juli 2008.

[Hütt] Norbert Hüttisch. *AT-Kommandos für GSM-Geräte*. `http://www.nobbi.com/atgsm.html`. [Online; Stand 17. November 2009].

[I2P0] *I2P Anonymous Network*. `http://www.i2p2.de`. [Online; Stand 3. Juni 2009].

[IANA02] IANA. *Special-Use IPv4 Addresses*. Internet Engineering Task Force, `http://www.ietf.org/rfc/rfc3330.txt`, September 2002. RFC 3330 (Informational).

[IEEE08] *Draft Standard for Local and Metropolitan Area Networks: Media Independent Handover Services*. `http://tinyurl.com/34koq74`, September 2008. [Online; Stand 3. Juni 2010].

[JoPA04] D. Johnson, C. Perkins und J. Arkko. *Mobility Support in IPv6*. Internet Engineering Task Force, `http://www.ietf.org/rfc/rfc3775.txt`, Juni 2004. RFC 3775 (Proposed Standard).

[KDE0] KDE e. V., http://www.kde.org. *KDE Homepage: Experience Freedom*. [Online; Stand 3. Juni 2009].

[KeSe05] S. Kent und K. Seo. *Security Architecture for the Internet Protocol*. Internet Engineering Task Force, http://www.ietf.org/rfc/rfc4301.txt, Dezember 2005. RFC 4301 (Proposed Standard).

[Kita01] H. Kitamura. *A SOCKS-based IPv6/IPv4 Gateway Mechanism*. Internet Engineering Task Force, http://www.ietf.org/rfc/rfc3089.txt, April 2001. RFC 3089 (Informational).

[KoHF06] E. Kohler, M. Handley und S. Floyd. *Datagram Congestion Control Protocol (DCCP)*. Internet Engineering Task Force, http://www.ietf.org/rfc/rfc4340.txt, März 2006. RFC 4340 (Proposed Standard).

[Kphn] Geeknet, Inc., http://sourceforge.net/projects/kphone. *Homepage des Softphones KPhone*. [Online; Stand 3. Februar 2010].

[Kras] Maxim Krasnyansky. *Universal TUN/TAP device driver*. http://www.kernel.org/pub/linux/kernel/people/marcelo/linux-2.4/Documentation/networking/tuntap.txt. [Online; Stand 3. Juni 2010].

[lapt] laptopkarten.de, http://www.laptopkarten.de/Lexikon/Handover.html. *Laptopkarten.de – Lexikon*. [Online; Stand 2. September 2009].

[Leec96] M. Leech. *Username/Password Authentication for SOCKS V5*. Internet Engineering Task Force, http://www.ietf.org/rfc/rfc1929.txt, März 1996. RFC 1929 (Proposed Standard).

[LGLK+96] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas und L. Jones. *SOCKS Protocol Version 5*. Internet Engineering Task Force, http://www.ietf.org/rfc/rfc1928.txt, März 1996. RFC 1928 (Proposed Standard).

[Linu] Linux kernel documentation, Documentation/networking/tproxy.txt. *Transparent proxy support*. Diese Dokumentation ist Bestandteil der Linux Kernelquellen seit Version 2.6.28.

[MaBh98] David A. Maltz und Pravin Bhagwat. MSOCKS: An Architecture for Transport Layer Mobility. In *IEEE INFOCOM'98*, Band 3, San Francisco, CA, USA, 1998. S. 1037–1045.

[MMFR96] M. Mathis, J. Mahdavi, S. Floyd und A. Romanow. *TCP Selective Acknowledgment Options*. Internet Engineering Task Force, `http://www.ietf.org/rfc/rfc2018.txt`, Oktober 1996. RFC 2018 (Proposed Standard).

[MNJH08] R. Moskowitz, P. Nikander, P. Jokela und T. Henderson. *Host Identity Protocol*. Internet Engineering Task Force, `http://www.ietf.org/rfc/rfc5201.txt`, April 2008. RFC 5201 (Experimental).

[NaHH06] Nidal Nasser, Ahmed Hasswa und Hossam Hassanein. Handoffs in Fourth Generation Heterogenous Networks. *Topics in Internet Technology* 44(10), Oktober 2006, S. 96–103.

[NATT] *NAT Traversal for VoIP and Internet Communications using STUN, TURN and ICE*. `http://www.voiptraversal.com`. [Online; Stand 3. Juni 2010].

[Ncat] OpenVPN Technologies, `http://netcat.sourceforge.net`. *The GNU Netcat project*. [Online; Stand 3. Juni 2009].

[NMan] Red Hat, Inc, `http://projects.gnome.org/NetworkManager`. *NetworkManager – Great Networking powered by DBUS and HAL*. [Online; Stand 2. September 2009].

[OtKu04] Jörg Ott und Dirk Kutscher. The Drive-Thru Architecture: WLAN-based Internet Access on the Road. In *IEEE Semiannual Vehicular Technology Conference 2004*, Milan, Italien, Mai 2004.

[OVHT] OpenVPN Technologies, `http://openvpn.net/index.php/open-source/documentation/howto.html`. *OpenVPN HOWTO*. [Online; Stand 3. Juni 2009].

[OVPN] OpenVPN Technologies, Inc., `http://www.openvpn.net`. *OpenVPN – Open Source VPN*. [Online; Stand 3. Juni 2009].

[Papu99] Lothar Papula. *Mathematik für Ingenieure und Naturwissenschaftler*, Band 3 der *Viewegs Fachbücher der Technik*. Vieweg Verlag. 3–528–24937–4, 3. Auflage, 1999.

[Perk02] C. Perkins. *IP Mobility Support for IPv4*. Internet Engineering Task Force, `http://www.ietf.org/rfc/rfc3344.txt`, August 2002. RFC 3344 (Proposed Standard), updated by RFC 4721.

[PoRe85] J. Postel und J. Reynolds. *File Transfer Protocol*. Internet Engineering Task Force, `http://www.ietf.org/rfc/rfc959.txt`, Oktober 1985. RFC 959 (Standard), updated by RFCs 2228, 2640, 2773, 3659.

[Post81] J. Postel. *Internet Control Message Protocol*. Internet Engineering Task Force, `http://www.ietf.org/rfc/rfc792.txt`, September 1981. RFC 792 (Standard), updated by RFCs 950, 4884.

[Post09] Wesley Post. *The Linux GPRS HOWTO*. `http://www.xs4all.nl/~ernstagn/GPRS-HOWTO`, Dezember 2009. [Online; Stand 3. Juni 2010].

[Reyn03] Patrick Reynolds. *[Dulug] info on /proc/net/wireless and iwconfig*. `https://lists.dulug.duke.edu/pipermail/dulug/2003-September/007550.html`, September 2003. [Online; Stand 3. Juni 2010].

[RFIG07] Letian Rong, Manel Fredj, Valérie Issarny und Nikolaos Georgantas. Mobility management in B3G networks: a middleware-based approach. In *ESSPE '07: International workshop on Engineering of software services for pervasive environments*, New York, NY, USA, 2007. ACM, S. 41–45.

[RiTu05] M. Riegel und M. Tuexen. *Mobile SCTP (draft-riegel-tuexen-mobile-sctp-05.txt)*. `http://tools.ietf.org/html/draft-riegel-tuexen-mobile-sctp-05`, Juli 2005. [Online; Stand 3. Juni 2010].

[RMKG+96] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot und E. Lear. *Address Allocation for Private Internets*. Internet Engineering Task Force, `http://www.ietf.org/rfc/rfc1918.txt`, Februar 1996. RFC 1918 (Best Current Practice).

[RMMW08] J. Rosenberg, R. Mahy, P. Matthews und D. Wing. *Session Traversal Utilities for NAT (STUN)*. Internet Engineering Task Force, `http://www.ietf.org/rfc/rfc5389.txt`, Oktober 2008. RFC 5389 (Proposed Standard).

[RSCJ+02] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley und E. Schooler. *SIP: Session Initiation Protocol*. Internet Engineering Task Force, `http://www.ietf.org/rfc/rfc3261.txt`, Juni 2002. RFC 3261 (Proposed Standard), updated by RFCs 3265, 3853, 4320, 4916, 5393.

[SCFJ03] H. Schulzrinne, S. Casner, R. Frederick und V. Jacobson. *RTP: A Transport Protocol for Real-Time Applications*. Internet Engineering Task Force, `http://www.ietf.org/rfc/rfc3550.txt`, Juli 2003. RFC 3550 (Standard), updated by RFC 5506.

[Schm08] Val Schmidt. *[SIO-SWAP] Meaning of "link quality" info in /proc/net/wireless?* `http://siomail.ucsd.edu/pipermail/swap/2008-September/000662.html`, September 2008. [Online; Stand 3. Juni 2010].

[Schu08] Dominik Schulz. Erweiterung eines Entscheiders für vertikale Handover. Studienarbeit, Fachgebiet Kommunikationsnetze, Technische Universität Ilmenau, Juni 2008.

[ScWe00] Henning Schulzrinne und Elin Wedlund. Application-Layer Mobility Using SIP. *Mobile Computing and Communications Review* Band 4, 2000, S. 47–57.

[Sieg99] Gerd Siegmund. *Technik der Netze*. Hüthig Verlag, Heidelberg. 3–7785–2637–5, 4. Auflage, 1999.

[Skyp] Skype Limited, `http://www.skype.com`. *Homepage der VoIP-Software Skype*. [Online; Stand 3. Februar 2010].

[SnBK01] Alex C. Snoeren, Hari Balakrishnan und M. Frans Kaashoek. The Migrate Approach to Internet Mobility. In *Proceedings of the Oxygen Student Workshop*, Gloucester, MA, Juli 2001. M. I. T. Computer Science and Artificial Intelligence Laboratory.

[SNLW08] Enrique Stevens-Navarro, Yuxia Lin und Vincent W.S. Wong. An MDP-Based Vertical Handoff Decision Algorithm for Heterogeneous Wireless Networks. *IEEE Transactions on Vehicular Technology* 57(2), März 2008, S. 1243–1254.

[Sock] Permeo Technologies, Inc., `http://www.socks.permeo.com/AboutSOCKS/SOCKSOverview.asp`. *SOCKS Overview*. [Seit Juli 2008 ersatzlos offline].

[Squi] *Squid: Optimising Web Delivery*. `http://www.squid-cache.org`. [Online; Stand 3. Juni 2009].

[SrEg01] P. Srisuresh und K. Egevang. *Traditional IP Network Address Translator (Traditional NAT)*. Internet Engineering Task Force, `http://www.ietf.org/rfc/rfc3022.txt`, Januar 2001. RFC 3022 (Informational).

[Stau08] Oliver Stauch. *Navigieren ohne GPS. Ortung: So funktionierts*. `http://www.connect.de/themen_spezial/Ortung-So-funktioniert_s_1565419.html`, April 2008. [Online; Stand 3. Juni 2010].

[Stew07] R. Stewart. *Stream Control Transmission Protocol*. Internet Engineering Task Force, `http://www.ietf.org/rfc/rfc4960.txt`, September 2007. RFC 4960 (Proposed Standard).

[Tele09] *Datenschutz ist ein Menschenrecht: Telefonüberwachung*. `http://www.ammering.org/pageID_5870581.html`, Juli 2009. [Online; Stand 3. Juni 2010].

[Tor0] The Tor Project, Inc., `https://www.torproject.org/overview`. *Tor: Overview*. [Online; Stand 3. Juni 2009].

[Toss09] Matthias Tosse. Entwicklung von Kommunikations-Backends für einen Handover Demonstrator. Studienarbeit, Fachgebiet Kommunikationsnetze, Technische Universität Ilmenau, Juli 2009.

[Tui8] Universitätsrechenzentrum der Technischen Universität Ilmenau, `http://tu-ilmenau.de/unirz/X-WPA_Supplicant.2311.0.html`. *DFNRoaming/802.1X Konfiguration des X/WPA_Supplicanten*. [Online; Stand 04. Juni 2010].

[Viet08] Hoang Nguyen Viet. Entwicklung von Mechanismen für einen kostenoptimierten Verbindungsaufbau für FMC-Privatnetze. Diplomarbeit, Fachgebiet Kommunikationsnetze, Technische Universität Ilmenau, Dezember 2008.

[VTRB97] P. Vixie, S. Thomson, Y. Rekhter und J. Bound. *Dynamic Updates in the Domain Name System (DNS UPDATE)*. Internet Engineering Task Force, `http://www.ietf.org/rfc/rfc2136.txt`, April 1997. RFC 2136 (Proposed Standard), updated by RFCs 3007, 4035, 4033, 4034.

[Walk01] Bernhard Walke. *Mobilfunknetze und ihre Protokolle*, Band 1 der *Informationstechnik*. Teubner Verlag. 3–519–26430–7, 3. Auflage, 2001.

[Weis98] Christian Weisgerber. *Die Unix-Philosophie*. `http://sites.inka.de/mips/unix/unixphil.html`, Juni 1998. [Online; Stand 3. Juni 2009].

[Wicd] Sourceforge.net, `http://wicd.sourceforge.net`. *Home of wicd*. [Online; Stand 2. September 2009].

[Wiki09a] Wikipedia. *Galileo (Satellitennavigation) — Wikipedia, Die freie Enzyklopädie*. `http://de.wikipedia.org/w/index.php?title=Galileo_(Satellitennavigation)&oldid=63100879`, 2009. [Online; Stand 2. September 2009].

[Wiki09b] Wikipedia. *Global Positioning System — Wikipedia, Die freie Enzyklopädie*. `http://de.wikipedia.org/w/index.php?title=Global_Positioning_System&oldid=64022962`, 2009. [Online; Stand 2. September 2009].

[Wiki09c] Wikipedia. *Global System for Mobile Communications — Wikipedia, Die freie Enzyklopädie*. `http://de.wikipedia.org/w/index.php?title=Global_System_for_Mobile_Communications&oldid=64037604`, 2009. [Online; Stand 2. September 2009].

[Wiki09d] Wikipedia. *Heimbereich — Wikipedia, Die freie Enzyklopädie*. `http://de.wikipedia.org/w/index.php?title=Heimbereich&oldid=61634456`, 2009. [Online; Stand 21. Juli 2009].

[Wiki09e] Wikipedia. *Kppp — Wikipedia, The Free Encyclopedia*. `http://en.wikipedia.org/w/index.php?title=Kppp&oldid=333432453`, 2009. [Online; Stand 19 Januar 2010].

[Wiki09f] Wikipedia. *Sudo — Wikipedia, Die freie Enzyklopädie*. `http://de.wikipedia.org/w/index.php?title=Sudo&oldid=64732312`, 2009. [Online; Stand 25. Januar 2010].

[Wiki09g] Wikipedia. *Udev — Wikipedia, Die freie Enzyklopädie*. `http://de.wikipedia.org/w/index.php?title=Udev&oldid=64520373`, 2009. [Online; Stand 24. November 2009].

[Wiki10] Wikipedia. *Universal Plug and Play — Wikipedia, Die freie Enzyklopädie*. `http://de.wikipedia.org/w/index.php?title=Universal_Plug_and_Play&oldid=69694684`, 2010. [Online; Stand 3. Februar 2010].

[Wild02] WildPackets, Inc., `http://www.wildpackets.com/elements/whitepapers/Converting_Signal_Strength.pdf`. *Converting Signal Strength Percentage to dBm Values*, November 2002. [Online; Stand 3. Juni 2010].

[Will08] Michael Will. *Linux GPRS HOWTO*. LinuxWiki.org, `http://linuxwiki.de/GPRS`, März 2008. [Online; Stand 17. November 2009].

[Will09] Dan Williams. *Dan Williams blog. Whipping out the pipe snake. Background Scanning.* `http://blogs.gnome.org/dcbw/2009/10/06/whipping-out-the-pipe-snake`, Oktober 2009. [Online; Stand 3. Juni 2010].

[Wint03] Sven Winter. Barrierefreier Campus: Lokalisierung in Gebäuden. Medienprojekt, Fachgebiet Kommunikationsnetze, Technische Universität Ilmenau, `http://www.ammering.org/pageID_5870581.html`, Dezember 2003.

[Wshk] The Wireshark Foundation, `http://www.wireshark.org`. *Homepage des Protokollanalysators Wireshark*. [Online; Stand 3. Februar 2010].

[XLte] CounterPath Corporation, `http://www.counterpath.com/x-lite.html`. *Homepage des Softphones X-Lite*. [Online; Stand 3. Februar 2010].

[YeES08] Yevgeniy Yeryomin, Florian Evers und Jochen Seitz. Solving the Firewall and NAT Traversal Issues for SIP-based VoIP. In *15th International Conference on Telecommunications (ICT'08)*, St. Petersburg, Russland, 2008. Fachgebiet Kommunikationsnetze, Technische Universität Ilmenau.

[Zech06] Stefan Zech. Entwicklung einer grafischen Benutzeroberfäche zur Steuerung eines Handoverdemonstrators. Studienarbeit, Fachgebiet Kommunikationsnetze, Technische Universität Ilmenau, September 2006.

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis und Formelzeichen

3GPP ..... **3**rd **G**eneration **P**artnership **P**roject
3GPP2 ..... **3**rd **G**eneration **P**artnership **P**roject **2**
$\bar{x}$ ..... Empirischer Mittelwert der zeitlichen Komponenten
$\bar{y}$ ..... Empirischer Mittelwert der gütebezogenen Komponenten
ω_c ..... Gewicht für Geldkosten (‘costs’)
ω_d ..... Gewicht für Zustand des Netzes
ω_f ..... Gewicht für Linkgüte (‘performance’)
ω_p ..... Gewicht für Energieverbrauch (‘power’)
ω_s ..... Gewicht für Sicherheit (‘security’)
$\overline{x^2}$ ..... Empirischer Mittelwert der quadrierten zeitlichen Komponenten
$\overline{xy}$ ..... Empirischer Mittelwert der Produkte aus Zeit- und Gütewerten
σ_f ..... Empirische Standardabweichung bezüglich f
C_i ..... Geldkosten (‘costs’) bezüglich Netzzugang i
D_i ..... Zustand des Netzes bezüglich Netzzugang i
F_i ..... Linkgüte (‘performance’) bezüglich Netzzugang i
n ..... Größe des Mittelungsfensters
P_i ..... Energieverbrauch (‘power’) bezüglich Netzzugang i
S_i ..... Sicherheit (‘security’) bezüglich Netzzugang i
t ..... Zeit (im Englischen: ‘**t**ime’)
`A` ..... Feld für **A**nzahl SACK-Blöcke
`F` ..... Feld für **F**lusssteuerungszwecke
`L` ..... **L**ängenfeld Nutzdaten
`P` ..... Feld für **P**DU-Typ
A ..... **a**usgehend
Abk. ..... **Abk**ürzung
AEK ..... **A**briss**erk**ennung
AES ..... **A**dvanced **E**ncryption **S**tandard
ALG ..... **A**pplication **L**ayer **G**ateway

AMD ..... **A**dvanced **M**icro **D**evices
API ..... **A**pplication **P**rogramming **I**nterface
ARP ..... **A**ddress **R**esolution **P**rotocol
AT ..... **At**tention, Hayes-Befehlssatz
B ..... **B**earer
B2BUA ..... **B**ack-**to**-**b**ack **U**ser **A**gent
Bit ..... **B**inary Dig**it**
bzw. ..... **be**zie**h**ungs**w**eise
ca. ..... **c**irc**a**
CF ..... **C**ompact**F**lash
CMS ..... **C**ontent **M**anagement **S**ystem
Codec ..... **Co**der and **Dec**oder
dBm ..... **D**ezi**b**el, bezogen auf 1 **M**illiwatt
DBUS ..... **D**esktop **Bus**
DCCP ..... **D**atagram **C**ongestion **C**ontrol **P**rotocol
DDR ..... **D**ouble **D**ata **R**ate
DFN ..... **D**eutsches **F**orschungs**n**etz
DHCP ..... **D**ynamic **H**ost **C**onfiguration **P**rotocol
div ..... **div**idiert durch
DNS ..... **D**omain **N**ame **S**ervice
DPI ..... **D**eep **P**acket **I**nspection
DRAM ..... **D**ynamic **R**andom **A**ccess **M**emory
DSL ..... **D**igital **S**ubscriber **L**ine
DSO ..... **D**ynamically loadable **S**hared **O**bject
E ..... **e**ingehend
EAP ..... **E**xtensible **A**uthentication **P**rotocol
EDGE ..... **E**nhanced **D**ata Rates for **G**SM **E**volution
Engl. ..... **Engl**isch
ESSID ..... **E**xtended **S**ervice **S**et **Id**entifier
etc. ..... **et c**etera
evtl. ..... **e**v**e**n**t**u**e**ll
exp. ..... **exp**onentiell
FTP ..... **F**ile **T**ransfer **P**rotocol
FW ..... **F**ire**w**all
GB ..... **G**iga**b**yte = 10^9 Byte
ggf. ..... **g**e**g**ebenen**f**alls
GGSN ..... **G**ateway **G**PRS **S**upport **N**ode

GHz ..... **G**iga**h**ert**z**
GiB ..... **G**igab**i**nary**b**yte = 2^{30} Byte
GNU ..... **G**NU is **n**ot **U**nix
GPRS ..... **G**eneral **P**acket **R**adio **S**ervice
GSM ..... **G**lobal **S**ystem for **M**obile Communications
GUI ..... **G**raphical **U**ser **I**nterface
HAL ..... **H**ardware **A**bstraction **L**ayer
HCI ..... **H**ost **C**ontroller **I**nterface
HIP ..... **H**ost **I**dentity **P**rotocol
HMIP ..... **H**ierarchical **M**obile **IP**
HMIPv6 ..... **H**ierarchical **M**obile **IPv6**
HSDPA ..... **H**igh **S**peed **D**ownlink **P**acket **A**ccess
HTTP ..... **H**yper**t**ext **T**ransfer **P**rotocol
HTTPS ..... **H**yper**t**ext **T**ransfer **P**rotocol **S**ecure
I ..... **I**nitiator
I-TCP ..... **I**ndirect **T**ransmission **C**ontrol **P**rotocol
I2P ..... **I**nvisible **I**nternet **P**roject
ICMP ..... **I**nternet **C**ontrol **M**essage **P**rotocol
ID ..... **Id**entifikator
IDE ..... **I**ntegrated **D**evice **E**lectronics
IEEE ..... **I**nstitute of **E**lectrical and **E**lectronics **E**ngineers
IEK ..... **I**solations**erk**ennung
IGMP ..... **I**nternet **G**roup **M**anagement **P**rotocol
iMASH ..... **I**nteractive **M**obile **A**pplication **S**ession **H**andoff
invsqr. ..... **inversq**uad**r**atisch (**inv**erse **squ**a**r**ed)
IP ..... **I**nternet **P**rotocol
IPv4 ..... **I**nternet **P**rotocol, **V**ersion **4**
IPv6 ..... **I**nternet **P**rotocol, **V**ersion **6**
IrDA ..... **I**nfra**r**ed **D**ata **A**ssociation
ISDN ..... **I**ntegrated **S**ervices **D**igital **N**etwork
kB ..... **K**ilo**b**yte = 10^3 Byte
kbit ..... **K**ilo**bit** = 10^3 Bit
kByte ..... **K**ilo**byte** = 10^3 Byte
KPPP ..... **K**DE **P**oint-to-**P**oint **P**rotocol, ein Einwahlprogramm
LAN ..... **L**ocal **A**rea **N**etwork
libltdl ..... **lib**tool **d**ynamic **l**oader **lib**rary
LL ..... **L**ogical **L**ink

LSB ..... **L**east **S**ignificant **B**it
LTE ..... **L**ong **T**erm **E**volution
M-TCP ..... **M**obile **T**ransmission **C**ontrol **P**rotocol
M-UDP ..... **M**obile **U**ser **D**atagram **P**rotocol
MAC ..... **M**edium **A**ccess **C**ontrol
MB ..... **M**ega**b**yte = 10^6 Byte
Mbit ..... **M**ega**bit** = 10^6 Bit
MByte ..... **M**ega**byte** = 10^6 Byte
ME ..... **M**obiles **E**ndgerät
MHz ..... **M**ega**h**ert**z**
MiB ..... **M**egab**i**nary**b**yte = 2^{20} Byte
MIHF ..... **M**edia **I**ndependent **H**andover **F**unction
MIPv6 ..... **M**obile **IPv6**
mod. ..... **mod**ifiziert
Modem ..... **Mo**dulator and **Dem**odulator
MPTCP ..... **M**ulti**p**ath **T**ransmission **C**ontrol **P**rotocol
ms ..... **M**illi**s**ekunden
MSB ..... **M**ost **S**ignificant **B**it
mSCTP ..... **M**obile **S**tream **C**ontrol **T**ransmission **P**rotocol
MTU ..... **M**aximum **T**ransmission **U**nit
MUM ..... **M**obile agent-based **U**biquitous multimedia **M**iddleware
MUX ..... **Mu**ltiple**x**er
NAT ..... **N**etwork **A**ddress **T**ranslation
NEC ..... **NEC** Corporation (ehemals: Nippon Electric Company)
NZG ..... **N**etz**zug**ang
OBU ..... **O**n-**B**oard-**U**nit
PBX ..... **P**rivate **B**ranch **E**xchange
PC ..... **P**ersonal **C**omputer
PCI ..... **P**eripheral **C**omponent **I**nterconnect
PCMCIA ..... **P**ersonal **C**omputer **M**emory **C**ard **I**nternational **A**ssociation
PCPR ..... **P**acket **C**ontrol **P**rotocol for **R**EACH
PDA ..... **P**ersonal **D**igital **A**ssistant
PDP ..... **P**acket **D**ata **P**rotocol
PDU ..... **P**rotocol **D**ata **U**nit
PF ..... **P**ort **F**orwarding
PI ..... **P**rotection **I**ndication
POP3 ..... **P**ost **O**ffice **P**rotocol, Version **3**

PPP ..... **P**oint-to-**P**oint **P**rotocol
PPPR ..... **P**acket **P**rotection **P**rotocol for **R**EACH
PSK ..... **P**re-**s**hared **K**ey
PSTN ..... **P**ublic **S**witched **T**elephone **N**etwork
QoS ..... **Q**uality-**o**f-**S**ervice
RDS ..... **R**eliable **D**atagram **S**ockets
REACH ..... **R**oaming-**E**nabled **Arch**itecture
RFC ..... **R**equest **f**or **C**omments
RP ..... **R**elay **P**lugin
RPS ..... **R**EACH-**P**roxy**s**erver
RSSI ..... **R**eceived **S**ignal **S**trength **I**ndicator
RTCP ..... **R**eal**T**ime **C**ontrol **P**rotocol
RTP ..... **R**eal-Time **T**ransport **P**rotocol
RTT ..... **R**ound **T**rip **T**ime
s ..... Sekunden
SACK ..... **S**elective **Ack**nowledgement
SBC ..... **S**ession **B**order **C**ontroller
SCPR ..... **S**tream **C**ontrol **P**rotocol for **R**EACH
SCTP ..... **S**tream **C**ontrol **T**ransmission **P**rotocol
SDP ..... **S**ession **D**escription **P**rotocol
SDU ..... **S**ervice **D**ata **U**nit
Signalst. ..... **Signalst**ärke
SIM ..... **S**ubscriber **I**dentity **M**odule
SIP ..... **S**ession **I**nitiation **P**rotocol
SMTP ..... **S**imple **M**ail **T**ransfer **P**rotocol
SN ..... **S**equenz**n**ummer
SOCKSv5 ..... **SOCKS**, **V**ersion **5**
SPPR ..... **S**tream **P**rotection **P**rotocol for **R**EACH
SRP ..... SIP-RTP-Proxy
SSH ..... **S**ecure **Sh**ell
SSHD ..... **S**ecure **Sh**ell **D**aemon (`sshd`)
SSID ..... **S**ervice **S**et **Id**entifier
STUN ..... **S**imple **T**raversal of **U**DP through **N**AT
SYN ..... **Syn**chronisierung
TCP ..... **T**ransmission **C**ontrol **P**rotocol
TOR ..... **T**he **O**nion **R**outer
TTL ..... **T**ime **t**o **l**ive

TU ..... **T**echnische **U**niversität
TURN ..... **T**raversal **U**sing **R**elay **N**AT
UDP ..... **U**ser **D**atagram **P**rotocol
UMTS ..... **U**niversal **M**obile **T**elecommunications **S**ystem
UPnP ..... **U**niversal **P**lug a**n**d **P**lay
USB ..... **U**niversal **S**erial **B**us
USHA ..... **U**niversal **S**eamless **H**andoff **A**rchitecture
VAD ..... **V**oice **A**ctivity **D**etection
VC ..... **V**irtual **C**hannel
VCI ..... **V**irtual **C**hannel **I**dentifier
VHDF ..... **V**ertical **H**andoff **D**ecision **F**unction
VLAN ..... **V**irtual **L**ocal **A**rea **N**etwork
VoIP ..... **V**oice **o**ver **IP**
VPN ..... **V**irtual **P**rivate **N**etwork
vs. ..... **v**er**s**us, lateinisch für gegenüber gestellt
WAN ..... **W**ide **A**rea **N**etwork
WEP ..... **W**ired **E**quivalent **P**rivacy
Wicd ..... **W**ireless **I**nterface **C**onnection **D**aemon
WiMAX ..... **W**orldwide **I**nteroperability for **M**icrowave **A**ccess
WLAN ..... **W**ireless **L**ocal **A**rea **N**etwork
WPA ..... **W**i-Fi **P**rotected **A**ccess
WPA-EAP ..... **W**i-Fi **P**rotected **A**ccess - **E**xtensible **A**uthentication **P**rotocol
WPA-PSK ..... **W**i-Fi **P**rotected **A**ccess - **P**re-**s**hared **K**ey
WPA2 ..... **W**i-Fi **P**rotected **A**ccess **2**
WpS ..... **W**erte **p**ro **S**ekunde
X.25 ..... Ein Netzzugangsprotokoll
XML ..... **E**xtended **M**arkup **L**anguage
zzgl. ..... **z**u**z**ü**gl**ich

Thesen zur Dissertation

1. Die „Roaming-Enabled Architecture“ (REACH) stellt eine Mobilitätsunterstützung dar, welche mit beliebigen Anwendungen funktioniert. Es werden keine Modifikationen der Anwendungen, der Betriebssysteme oder der Komponenten des Netzwerks benötigt.

2. Nur die Gruppe der Middleware-basierten Mobilitätsunterstützungen, zu der auch REACH gehört, eignet sich zur Erstellung einer praktikablen Lösung zum Einsatz im heutigen Internet.

3. REACH implementiert einen „Multi-Layer“-Ansatz. Die Datenströme der nicht modifizierten Anwendungen lassen sich auf unterschiedlichen Schichten abfangen, wobei REACH auch eine Mobilitätsunterstützung auf Benutzerebene anbietet.

4. Nur ein kombiniertes Angebot unterschiedlicher Mechanismen erlaubt die bestmögliche Unterstützung einer Vielzahl an Anwendungen und Nutzungsszenarien.

5. Eine praxisrelevante Architektur zur Mobilitätsunterstützung muss Komponenten zur Verwaltung von Netzzugangsgeräten, zum Treffen von Handoverentscheidungen sowie eine optionale Bedienschnittstelle aufweisen.

6. Durch Auswertung von Signalstärke- oder Linkgütewerten kann REACH zukünftige Abrissereignisse vorhersagen, und somit auch „prädiktive weichere Handover“ durchführen.

7. Mit der REACH-Box kann ein lokales Netzwerk „mobil gemacht“ werden, wobei keinerlei Modifikationen der beteiligten Endgeräte mehr erforderlich sind.

Ilmenau, den 30. September 2010 Dipl.-Ing. Florian Evers

Erklärung

Ich versichere, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Weitere Personen waren an der inhaltlich-materiellen Erstellung der vorliegenden Arbeit nicht beteiligt. Insbesondere habe ich hierfür nicht die entgeltliche Hilfe von Vermittlungs- bzw. Beratungsdiensten (Promotionsberater oder anderer Personen) in Anspruch genommen. Niemand hat von mir unmittelbar oder mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalte der vorgelegten Dissertation stehen.

Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form einer Prüfungsbehörde vorgelegt.

Ich bin darauf hingewiesen worden, dass die Unrichtigkeit der vorstehenden Erklärung als Täuschungsversuch angesehen wird und den erfolglosen Abbruch des Promotionsverfahrens zu Folge hat.

Ilmenau, den 30. September 2010 Dipl.-Ing. Florian Evers