Das Transportprotokoll SCTP

Leistungsbewertung und Optimierung eines neuen Transportprotokolls

D I S S E R T A T I O N

zur Erlangung des akademischen Grades
Doctor rerum naturalium
(Dr. rer. nat.)
im Fach Informatik

vorgelegt dem
Fachbereich Wirtschaftswissenschaften
Institut für Informatik und Wirtschaftsinformatik (ICB)
Universität Duisburg-Essen

von
Dipl.-Ing. Andreas Jungmaier
geboren am 12.06.1972 in Oberhausen

Gutachter:

1. Prof. Dr.-Ing. Erwin P. Rathgeb
2. Prof. Dr. Bruno Müller-Clostermann

Tag der mündlichen Prüfung: 16. August 2005

Abstract

The new Stream Control Transmission Protocol (SCTP) was originally standardised for the transport of signaling messages – originating from the Common Channel Signaling System No.7 (SS7) – over IP based networks. Nonetheless, SCTP is a general purpose IP-based reliable transport protocol which is connection oriented and offers message based data transfer. It supports multiple independent message streams and flexible data delivery mechanisms. In contrast to TCP, SCTP protocol endpoints support multiple addresses (multihoming), and therefore an endpoint may be reached via different and possibly redundant network paths. Several SCTP enhancements, e.g. for partially reliable message transfer [82], are currently being proposed, and it is conceivable that SCTP is a suitable transport protocol for many established and future applications [5, 20].

In this thesis, the behaviour and performance of SCTP was investigated within different scenarios. A suite of tools were developed to this end: in a testbed environment, a Unix-based protocol implementation named *sctplib* was created for demonstrating SCTP fairness towards TCP, and the applicability of SCTP for signaling transport was investigated.

Discrete event-based simulation models of the SCTP data path and some control path elements were created and validated against the results from the testbed experiments. These models were later enhanced to investigate the simultaneous use of several available paths (also named *load sharing*). From an operator perspective, load sharing is desirable for signaling transport at high network loads, even though it requires substantial protocol adaptations. Load sharing mechanisms from the literature were evaluated, and a number of modifications of these algorithms were suggested and evaluated, as well.

It could be shown that the modifications of the existing load sharing algorithms that were developed within the scope of this dissertation indeed lead to an optimisation of the load sharing algorithms so far proposed, both in terms of overall throughput and maximum message delay that can be expected.

Keywords:
SCTP, Transport Protocol, Performance Evaluation, Load Sharing

Zusammenfassung

Das Stream Control Transmission Protocol (SCTP) wurde als Basis für den effizienten Transport von Signalisierungsnachrichten aus dem Zentralen Zeichengabesystem No.7 (SS7) über IP-basierte Netze entwickelt. SCTP ist ein generisches Vielzweck-Transportprotokoll, welches verbindungsorientiert operiert und eine zuverlässige, nachrichtenorientierte Datenübertragung bietet. So unterstützt es mehrere unabhängige Nachrichtenströme in einer Verbindung sowie flexible Zustellmechanismen. Im Gegensatz zu TCP unterstützt SCTP mehrere Netzadressen pro Verbindung (Multihoming), und aus diesem Grund sind SCTP-Endpunkte über mehrere – möglicherweise redundante – Netzwerkpfade erreichbar.
In der vorliegenden Dissertation wurde das Verhalten und die Leistungsfähigkeit des SCTP in verschiedenen Szenarien untersucht. Geeignete Werkzeuge wurden zu diesem Zweck entwickelt: in einer Testbett-Umgebung wurde eine Unix-basierte Protokollimplementation namens *sctplib* dazu benutzt, die Fairness von SCTP im Zusammenspiel mit TCP sowie die Eignung des SCTP für den SS7-basierten Signalisierungstransport zu untersuchen.
Ein ereignisorientiertes Simulationsmodell des SCTP-Datenpfades und einiger Elemente des Kontrollpfades wurde erstellt und mit Hilfe der Ergebnisse aus den Untersuchungen im Testbett validiert. Mit einer Erweiterung dieses Modells wurden Lastverteilungsalgorithmen untersucht. Lastverteilung ist aus der Perspektive eines Netzbetreibers wünschenswert, um eine gleichmäßige Verteilung der Verkehrslast zu erreichen, und Spitzenlasten im Netz abzufangen. Ihre effiziente Unterstützung erfordert jedoch beträchtliche Protokollmodifikationen beim SCTP.
Neben einer Bewertung der in der Literatur beschriebenen Lastverteilungsalgorithmen wurden eigene Modifikationen dieser Algorithmen vorgeschlagen und gleichfalls bewertet. Dabei konnte gezeigt werden, dass die beschriebenen Modifikationen zu einer Optimierung der existierenden Lastverteilungsalgorithmen führen, sowohl im Hinblick auf den Gesamtdurchsatz als auch auf die zu erwartende Nachrichtenverzögerung.

Schlagwörter:
SCTP, Transportprotokoll, Leistungsbewertung, Simulation

Für Helena.

Danksagung

Die vorliegende Dissertation ist im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl “Technik der Rechnernetze” der Universität Duisburg-Essen entstanden. An dieser Stelle möchte ich all denjenigen danken, die dazu beigetragen haben, dass ich diese Arbeit fertigstellen konnte.
Mein ganz persönlicher Dank gilt Herrn Professor Dr.-Ing. Rathgeb, der mir mit Rat, Hilfe und konstruktiver Kritik stets vertrauensvoll und geduldig zur Seite gestanden hat.
Besonders danken möchte ich außerdem Herrn Professor Dr. Müller-Clostermann, der als Zweitgutachter entscheidend zum Gelingen dieser Arbeit beigetragen hat. Ebenso gilt Herrn Professor Dr. Tüxen mein Dank für die vielen spannenden Diskussionen.
Ferner möchte ich mich herzlich bei allen Mitarbeitern des Lehrstuhls “Technik der Rechnernetze” bedanken, insbesondere bei Holger Bleul, Thomas Dreibholz und Birger Tödtmann, nicht nur für viele wertvolle Diskussionen, sondern auch dafür, dass sie mich zur rechten Zeit entlastet und unterstützt haben. Mein Dank gilt auch allen studentischen Hilfskräften, insbesondere Sebastian Rohde für seine hervorragende Mitarbeit an der *sctplib*.
Mein besonderer Dank gilt meiner Frau Sandra für ihre Geduld, ihr Verständnis und ihre Unterstützung, ohne die ich niemals fertig geworden wäre. Schließlich (last but not least) möchte ich mich herzlich bei meinen Eltern bedanken, die mir zu jedem Zeitpunkt alle denkbare Unterstützung haben zukommen lassen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

AAA	Authentifizierung, Autorisierung und Abrechnung
AIMD	additive increase, multiplicative decrease
AL	Adaptation Layer
AODV	Ad-hoc On-Demand Distance Vector Routing
ARL	Association Retransmission Limit
arwnd	Advertised Receiver Window
ASAP	Aggregate Server Access Protocol
ATM	Asynchronous Transfer Mode
BER	Bit Error Rate, Bitfehlerrate
BGP	Border Gateway Protocol
CACC	Changeover Aware Congestion Control
CBQ	Class Based Queueing
CBR	Constant Bit Rate
CMT	LS-SCTP mit Concurrent Multipath Transfer
CSMA	Carrier Sensing Multiple Access
CSS7	Common Channel Signaling System No. 7
CTSNA	Cumulative TSN Acknowledged
cwnd	Congestion Window
DRR	Distributed Resilient Routing
DSR	Dynamic Source Routing
DSS1	Digital Subscriber Signaling System No. 1
ECN	Early Congestion Notification
EIGRP	Enhanced Interior Gateway Routing Protocol
ENRP	Endpoint Name Resolution Protocol
FDDI	Fiber Distributed Data Interface
FIFO	First In, First Out
FR	Fast Retransmission
FRFR	Fast Retransmission - Fast Recovery

FTP	File Transfer Protocol
GPRS	General Packet Radio Service
GT	Global Title
HTTP	Hypertext Transfer Protocol
ICI	Interface Control Information
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IGRP	Interior Gateway Routing Protocol
IN	Intelligente Netze
IOS	Internet Operating System
IP	Internet Protocol
ISDN	Integrated Services Digital Network
ISP	Internet Service Provider
ISUP	ISDN User Part
LAR	Location-Aided Routing
LDP	Label Distribution Protocol
LS	Load Sharing
M2PA	SS7 MTP2-User Peer-to-Peer Adaptation Layer
M3UA	SS7 MTP3-User Adaptation Layer
MDTP	Multi-Network Datagram Transmission Protocol
MG	Media Gateway
MGC	Media Gateway Controller
MIB	Management Information Base
MPEG	Motion Picture Expert Group
MPLS	Multi-Protocol Label Switching
MSU	Message Signal Unit
MTP	Message Transfer Part
MTU	Maximum Transmission Unit
N-SAP	Network Service Access Point
NS	Name Server
OSI	Open Systems Interconnection
OSPF	Open Shortest Path First
oTCL	Object-oriented Tool Command Language
PE	Pool Element
PPP	Point-to-Point Protocol
PRL	Path Retransmission Limit
PSTN	Public Switched Telephony Network

PU	Pool User
QoS	Quality of Service
RAB	Radio Access Bearer
RADIUS	Remote Authentication Dial In User Service
RANAP	Radio Access Network Application Part
RED	Random Early Drop
RFC	Request for Comment
RIP	Routing Information Protocol
RNC	UMTS Radio Network Controller
RSVP	Resource Reservation Protocol
RSVP-TE	Resource Reservation Protocol - Extension for Traffic Engineered Tunnels
RTCP	RTP Control Protocol
RTO	Retransmission Timeout
RTP	Realtime Transport Protocol
RTT	Round Trip Time
RTTVAR	Round Trip Time Variation
rwnd	Receiver Window
SACK	Selective Acknowledgement
SCCP	Signaling Connection Control Part
SCP	Service Control Point
SCTP	Stream Control Transmission Protocol
SDL	Specification and Description Language
SEP	Signalisierendpunkt
SFQ	Stochastic Fair Queueing
SG	Signaling Gateway
SIGTRAN	Signaling Transport Working Group
SIP	Session Initiation Protocol
SL	Signaling Link
SMAP	System Management Application Part
SNMP	Simple Network Management Protocol
SP	Signalisierpunkt
SRTT	Smoothed Round Trip Time
SS7	(Common Channel) Signaling System No. 7
SS7	Zentrales Zeichengabesystem No. 7
SSH	Secure Shell
SSN	Stream Sequence Number
SSP	Service Switching Point

ssthresh	Slow Start Threshold
ST	Signalisiertransport
STP	Signalisiertransferpunkt
TCAP	Transaction Capabilities Application Part
TCP	Transmission Control Protocol
TFRC	TCP Friendly Rate Control
TLS	Transport Layer Security
TPAL	Transport Adaptation Layer
TSN	Transmission Sequence Number
TSVWG	Transport Area Working Group
TUP	Telephone User Part
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
UTRAN	UMTS Radio Access Network
WAN	Wide Area Network
WFQ	Weighted Fair Queueing
WLAN	Wireless LAN

Kapitel 1

Einleitung

Durch die steigende Verbreitung integrierter digitaler Dienste, basierend z.B. auf dem ISDN, oder der zunehmenden Nutzung neuer Funknetztechnologien wie GPRS (General Packet Radio Service) und UMTS (Universal Mobile Telecommunications System) müssen die Netzwerkarchitekturen, auf denen diese Technologien aufsetzen, Sprach-, Daten- und Multimedia-Dienste umfassend unterstützen. Dies führt zu einem zunehmenden Zwang und Trend zur Konvergenz von paketvermittelnden und leitungsvermittelnden Netzwerkarchitekturen.
Die effektive Kooperation zwischen herkömmlichen, auf ISDN-basierenden, und neueren IP-basierten Architekturen ist daher für heutige Netzbetreiber wesentlich, ebenso wie eine sinnvolle Evolution bestehender Kernnetze hin zu kostengünstigeren, einfacher und homogener strukturierten und gut wartbaren Netzen.
Der Trend zur Konvergenz der verschiedenen Technologien erfordert mithin, dass die Übertragung von Signalisierungsinformationen auch über IP-basierte Netze mit der erforderlichen Zuverlässigkeit und Dienstgüte erfolgen kann. In klassischen Telekommunikationsnetzen übernimmt den Transport von Signalisierungsinformationen das Zentrale Zeichengabesystem No. 7 (engl. Common Channel Signaling System No. 7, CSS7, oder kurz SS7) [30].
Um eine standardisierte Übertragung von Signalisierungsinformationen aus dem SS7 auch in IP-Netzen zu ermöglichen, begann die SIGTRAN Arbeitsgruppe der IETF 1998 damit, eine neue Protokollarchitektur für die Übertragung von SS7-basierten Signalisierungsinformationen zu definieren. Die Grundlage dieser Architektur bildet das im Jahre 2000 in dem Dokument [86] spezifizierte Stream Control Transmission Protocol (SCTP), ein neues, zuverlässiges Transportprotokoll, welches auf IP aufsetzt. Auf diesem Transportprotokoll wiederum setzen eine Reihe von Anpassungsschichten auf, die dazu bestimmt sind, Dienstprimiti-

ve und Nachrichten des zentralen Zeichengabesystems No. 7 zu übertragen (vgl. Abschnitt 2.1.2).
In dieser Arbeit wird das für diese Protokollarchitektur grundlegende Transportprotokoll SCTP eingehender dargestellt und untersucht. Die Protokollmechanismen, die entscheidend für das Zusammenspiel des SCTP mit etablierten Transportprotokollen wie TCP sind, werden ebenso betrachtet wie Modifikationen, die in bestimmten Situationen das Protokollverhalten optimieren. Diese Ergebnisse sind teilweise mit einer im Laufe der Arbeit entstandenen Protokollimplementierung in einem Labortestbett erarbeitet worden. Ferner werden Mechanismen untersucht, durch die SCTP gleichzeitig mehrere Netzwerkpfade effektiv zur Datenübertragung nutzen kann. Dies ist, insbesondere im Hinblick auf die Übertragung von Signalisierungsinformationen, eine wünschenswerte Eigenschaft, die das Standard-SCTP jedoch nicht vorsieht. Die entsprechenden Untersuchungen wurden an einer Reihe von ebenfalls im Laufe der Arbeit entstandenen Simulationsmodellen durchgeführt, die zuvor durch Vergleich mit den Ergebnissen aus dem Labortestbett validiert wurden. Dabei konnte gezeigt werden, dass die im Laufe dieser Arbeit entwickelten Algorithmen zur simultanen Pfadnutzung in bestimmten Szenarien den bisher von Iyengar et al. in [41] diskutierten Algorithmen in Bezug auf den möglichen Datendurchsatz überlegen sind.

1.1 Die Eigenschaften des SCTP

Das verbindungsorientierte Stream Control Transmission Protocol (SCTP) bietet einen zuverlässigen Transport von Nachrichten zwischen IP-basierten Endgeräten.
Eine SCTP-Verbindung, nachfolgend *Assoziation* genannt, kann ein Multiplex aus bis zu 65536 weitgehend unabhängigen Nachrichtenströmen übertragen, innerhalb derer Nachrichten in der Regel mit Reihenfolgesicherung übertragen werden. Neben der Sicherung der Reihenfolge der Nachrichten bei der Zustellung (d.h. die Reihenfolge bei der Zustellung gleicht der Absendereihenfolge) können Anwendungen Nachrichten auch in der Ankunftsreihenfolge zustellen lassen.
Ein weiteres Merkmal des SCTP ist die Unterstützung von IP-Multihoming, d.h. von mehreren IP-Adressen pro Assoziation. Dies ist relevant für Endgeräte, die mit mehreren Netzwerkadaptern ausgestattet sind (z.B. Router, Server etc.), oder für solche, die neben dem IPv4 [65] gleichzeitig auch das neuere IPv6 [18] als Netzwerkprotokoll unterstützen (sog. Dual-Stack Systeme [29]). Die Zahl dieser Endgeräte wird mit dem Wachstum datenorientierter Dienste in Mobilfunknetzen

in naher Zukunft stark zunehmen. Bei der Standardisierung des SCTP durch die IETF wurde hauptsächlich Rücksicht auf die bereits bestehenden und weit verbreiteten Protokollarchitekturen (insbes. TCP/IP) genommen: Protokollparameter sowie Sendealgorithmen wurden deshalb so gewählt, dass Standard-SCTP die hohen Anforderungen hinsichtlich Ausfallsicherheit und Fehlererkennung beim SS7-basierten Transport von Signalisierungsinformationen nicht erfüllen kann.
Die zentralen Teile dieser Arbeit befassen sich daher mit der Auswahl geeigneter Protokollparameter und Bewertung neuer Algorithmen für den höchst anspruchsvollen Einsatz des SCTP in modernen, IP-basierten Signalisiernetzen.

1.2 Anwendungsszenarien für das SCTP

Da SCTP auch im Hinblick auf spätere Erweiterbarkeit entwickelt wurde, sind zum gegenwärtigen Zeitpunkt noch nicht alle Anwendungsmöglichkeiten für dieses neue Protokoll abzusehen. Im Folgenden wird daher nur kurz auf einige der teilweise bereits in der Standardisierung befindlichen, wichtigen Anwendungen des SCTP eingegangen.

1.2.1 IP-basierter Transport von Signalisierungsnachrichten

In den gängigen Telekommunikationsnetzen werden Kontrollmeldungen vorwiegend basierend auf dem Signalisierungssystem No. 7 (SS7) [30] übertragen. Bestehende Dienste und Anwendungen unterstützen und nutzen die hohe Effizienz von SS7, die im Wesentlichen von der guten Fehlerkontrolle auf der Schicht 2 sowie den signalisierungsspezifischen Netzwerk-Managementprozeduren im MTP Level 3 herrühren.
Allerdings repräsentiert das SS7 Signalisierungsnetz ein logisch vom Nutzdatenverkehr (Sprachdaten) separiertes Netz, welches eine gesonderte Infrastruktur erfordert und nur wenige Ressourcen mit dem Nutzdatenverkehr teilt.
In der IETF Arbeitsgruppe ’Signaling Transport (SIGTRAN)’ wird seit einigen Jahren an einer Architektur zur Übertragung von Signalisierungsmeldungen auf eine andere Weise gearbeitet [60]. Diese Architektur basiert auf dem Transport von Signalisierungsinformationen durch das Stream Control Transmission Protocol (SCTP) [86], die über ein gemeinsames paketvermittelndes (IP-basiertes) Core-Netzwerk übertragen werden. Die Fluss- und Fehlerkontrolle werden Ende-zu-Ende durchgeführt und eine hohe Zuverlässigkeit erreicht man durch sog. ’Multihomed Nodes’ (das entspricht Signalisierungspunkten mit mehr

als einer Transport-Adresse) sowie durch Clustering (vgl. auch Abschnitte 1.2.2 und 2.2). Letztendlich ermöglicht die konsequente Einführung dieses Protokolls die Übertragung sowohl von Nutzdaten als auch Signalisierungsmeldungen über ein gemeinsames IP-basiertes Core-Netzwerk.

1.2.2 Reliable Server Pooling

SCTP ist als Transportprotokoll der neuen *Reliable Server Pooling*-Architektur [91] vorgesehen. Diese wird in der RSerPool-Arbeitsgruppe der IETF diskutiert und definiert Mechanismen für das zuverlässige Zusammenspiel von Serververbünden, die einen gemeinsamen Dienst erbringen, und Clients, die die Dienste eines Servers in Anspruch nehmen.
Während SCTP durch die Unterstützung des Multihoming für Redundanz in der Netzwerk- und Sicherungsschicht sorgt, definiert RSerPool einen Session-Layer, der auch bei Ausfällen von Servern dafür sorgt, dass Anwendungen einen Dienst transparent und ohne Unterbrechungen nutzen können. In Abschnitt 2.2 wird das Konzept des Reliable Server Pooling näher erläutert.

1.2.3 Weitere Anwendungen

In der IETF-Standardisierung wird die Nutzung des SCTP als Transportprotokoll einer ganzen Reihe von Anwendungsprotokollen geplant. Dabei tauchen immer wieder Bedenken auf, dass die mangelnde Verbreitung verfügbarer und getesteter Implementierungen einer schnellen Nutzung der entsprechenden Anwendungsprotokolle entgegensteht. Darüber hinaus sind viele Architekturelemente heutiger Netze (noch) nicht für SCTP vorbereitet (z.B. Firewalls). Daher wird neben SCTP in der Regel auch das etablierte TCP als Transportprotokoll für neue Anwendungsprotokolle berücksichtigt. Es ist jedoch davon auszugehen, dass mit zunehmender Verbreitung von nativen SCTP-Implementierungen in den aktuellen Betriebssystemen (vgl. auch Abschnitt 3.4) diese Bedenken hinfällig werden.

1.2.3.1 Authentifizierung, Autorisierung und Abrechnung (AAA)

Die AAA-Arbeitsgruppe der IETF standardisiert ein neues Protokoll unter dem Namen *Diameter* [13], welches Dienste für die Authentifizierung, Autorisierung und Abrechnung (engl. Authentication, Authorization and Accounting, AAA) zur Verfügung stellt. Nutzer dieser Dienste sind typisch Systeme für den Netzwerk-

zugang (etwa bei der Einwahl in das Zugangsnetz eines Internet Service Provider oder in ein WLAN-Funknetz).
Das Wachstum des Internets und die Einführung neuer Zugangsmöglichkeiten (z.B. DSL oder Kabelmodems) machen ein neues flexibles AAA-Protokoll erforderlich, das z.B. das etablierte *RADIUS*-Protokoll [70] ersetzt.
Die AAA-Arbeitsgruppe hat entschieden, dass in der Client-Server basierten *Diameter*-Architektur protokollkonforme Server sowohl TCP als auch SCTP als Transportprotokoll unterstützen *müssen*. Clients können entweder TCP oder SCTP unterstützen, wobei SCTP empfohlen wird. Diameter nutzt dabei die hervorragenden Eigenschaften des SCTP im Hinblick auf die mögliche Vermeidung von Blockierungen, indem alle möglichen Nachrichtenströme einer bestehenden SCTP-Assoziation verwendet werden (vgl. auch Abschnitt 2.3.5.2).

1.2.3.2 Session Initiation Protocol über SCTP

Das Session Initiation Protocol (SIP) [73] ist ein textbasiertes, HTTP-ähnliches Protokoll zum Aufbau interaktiver Kommunikationsbeziehungen (engl. Sessions) mehrerer Nutzer. Solche Kommunikationsbeziehungen können für interaktive Sprach- oder Videoverbindungen ebenso etabliert werden, wie für textbasierte Interaktionen (engl. Chat) oder Online-Spiele.
Als grundlegendes und flexibles IP-basiertes Signalisierungsprotokoll wird SIP gegenwärtig vor allem für IP-Telefonie eingesetzt. In den Spezifikationen für das IP-Multimedia Subsystem (IMS) des Release 5 der 3G-Mobilfunknetze [2] ist SIP als Transportprotokoll vorgesehen und erfordert dann insbesondere im UMTS-Kernnetz eine leistungsfähige Architektur, um eine Vielzahl simultaner SIP-Rufanforderungen verarbeiten und übertragen zu können.
Eine typische auf SIP basierende Architektur besteht aus Servern (Proxies) und Endgeräten, z.B. IP-Telefonen. Server dienen vor allem dazu, zu einer Domäne gehörige Teilnehmer zu registrieren und ankommende Anrufe entsprechend (gegebenenfalls über weitere Proxies) an diese Teilnehmer weiterzuleiten. Der Austausch von Mediendaten geschieht nachfolgend unabhängig von der SIP-Architektur basierend auf einer direkten IP-Verbindung der kommunizierenden Teilnehmer.
Da SCTP als Transportprotokoll für die Übermittlung von Signalisierungsdaten entwickelt wurde und SIP weitgehend unabhängig vom Transportprotokoll ist, profitiert SIP besonders von den Vorteilen des SCTP, wie z.B. der Vermeidung von Blockierungen bei Nachrichtenverlust. Dies wird durch erste Untersuchungen von De Marco et al. in [55] belegt. In der IETF-Standardisierung wird die

Nutzung von SCTP als Transportprotokoll für SIP-Signalisierung ebenfalls verfolgt; das entsprechende Dokument [72] von Schulzrinne et al. hat zurzeit noch den inoffiziellen ‘Draft’-Status.

1.2.3.3 Übertragung von Medienströmen

Mit der Erweiterung des SCTP durch den teilgesicherten Transportmodus (engl. Partially Reliable SCTP, PR-SCTP) [82] wird SCTP auch zum Anwärter für den flexiblen Transport von Multimedia- und Sensor-Daten (vgl. auch Abschnitt 2.4.1). So untersuchten Balk et al. in [5], sowie Molteni und Villari in [57] bereits die Übertragung von MPEG-4 [38] codierten Medienströmen über SCTP. Sie zeigen, dass bei möglichen Paketverlusten im Netz MPEG-4 Anwendungen, welche SCTP mit der PR-Erweiterung nutzen, durch selektive Neuübertragung bestimmter Schlüssel-Informationen (sog. MPEG-4 I-Frames) die Qualität von MPEG-4 basierten Video-Strömen im Vergleich zur Übertragung über RTP/UDP beträchtlich erhöhen können.

1.2.3.4 SCTP und Mobilität

Eine weitere Erweiterung des SCTP definiert Mechanismen, die das Hinzufügen und Löschen von Transportadressen in bestehenden SCTP-Assoziationen erlauben [83]. Diese Erweiterung kann auch genutzt werden, um es Endgeräten in mobilen Szenarien zu erlauben, Verbindungen ohne Unterbrechung aufrecht zu erhalten, während sie z.B. in einem Funknetz wechselnde Transportadressen besitzen. Entsprechende Untersuchungen wurden im Laufe dieser Arbeit durchgeführt und in [20] veröffentlicht.

1.3 Überblick über die Arbeit

Im Kapitel 2 wird zunächst die Motivation zur Entwicklung des SCTP erläutert, indem Anwendungsszenarien vorgestellt werden, die ein neues Transportprotokoll voraussetzen. Beispiele solcher Anwendungen sind die Übertragung von Meldungen des Zentralen Zeichengabesystems No. 7 über IP-basierte Netze oder das Reliable Server Pooling. Es folgt eine Diskussion des SCTP, die die Basis für die Untersuchungen in den folgenden Kapiteln bildet. Dabei werden u.a. Eigenschaften wie die Unterstützung für Multihoming, Überwachung von Pfaden, und die flexible Datenzustellung betrachtet. Anschließend folgt eine Diskussion der relevanten, standardisierten Protokollerweiterungen und ihrer Anwendungen.

Das anschließende Kapitel 3 befasst sich mit der im Laufe der Arbeit entstandenen SCTP-Implementierung, die unter einer Open Source-Lizenz für die Allgemeinheit verfügbar ist. Dabei werden die Entwicklungsstufen dieser Implementierung beginnend mit einer systemorientierten Spezifikation, über die funktionalen Anforderungen, bis hin zur existierenden Implementierung und ihrer Weiterentwicklung dokumentiert. Ferner werden die möglichen Schnittstellen für die Nutzung des SCTP vorgestellt, bevor ein Überblick über die Eigenschaften der wichtigsten derzeitig verfügbaren SCTP-Implementierungen gegeben wird.
Im Kapitel 4 wird eine Reihe von Untersuchungen vorgestellt, die mit der im Kapitel 3 beschriebenen SCTP-Implementierung im Labor in einer Testbett-Umgebung durchgeführt wurden. Zu diesen Untersuchungen gehört ein Vergleich des SCTP mit etablierten Transportprotokollen, der u.a. belegt, dass sich SCTP und TCP zur Verfügung stehende Netzwerkressourcen fair teilen. Damit konnte gezeigt werden, dass SCTP für einen breiten Einsatz im Internet geeignet ist. Zum anderen wurden Modifikationen des Protokolls untersucht, die unter bestimmten Gegebenheiten (z.B. hohe Verzögerung bei der Übertragung und verlustbehaftete Links) ein optimaleres Protokollverhalten bewirken. Schließlich werden Untersuchungen vorgestellt, in denen optimale Parametereinstellungen des SCTP für die Übertragung von Signalisierungsmeldungen aus SS7-Netzen vorgeschlagen werden.
Das Kapitel 5 befasst sich mit der ereignisorientierten Simulation des SCTP. Zunächst werden die Anforderungen an die Simulationsumgebung und an das SCTP-Modell erörtert, bevor auf die Struktur des im Laufe der Arbeit entwickelten Modells eingegangen wird. Zur Validierung dieses Modells werden mehrere der im Labortestbett untersuchten Szenarien durch entsprechende Simulationen nachgestellt und die Ergebnisse dieser Simulationen mit den Untersuchungen der SCTP-Implementierung verglichen. Durch die weitgehende Übereinstimmung zwischen dem simulierten und dem im Labor beobachteten Protokollverhalten ist gewährleistet, dass die im Kapitel 6 folgenden Ergebnisse auch in der Praxis relevant sind und Bestand haben. Dort werden Algorithmen untersucht, die das Verhalten des SCTP an die Anforderungen des Signalisierungstransports anpassen. Dazu gehören vor allem die simultane Nutzung aller zur Verfügung stehenden Netzwerkpfade einer SCTP-Verbindung. Damit wird zum einen eine gleichmäßigere Auslastung der Netzwerkpfade erreicht, zum anderen kann in Zeiten hoher Belastung ein insgesamt höherer Netzwerkdurchsatz erreicht werden als mit Standard-SCTP. In Kapitel 6 werden bisherige, in der Literatur diskutierte Lösungen untersucht, und mit neu entwickelten Algorithmen verglichen, die einige Nachteile der bekannten Lösungen beheben.

Die Arbeit schließt mit der Zusammenfassung in Kapitel 7 und gibt einen abschließenden Ausblick auf die zukünftige Entwicklung des SCTP.

Kapitel 2

Das Stream Control Transmission Protokoll (SCTP)

Die ursprüngliche Motivation für die Entwicklung von SCTP entstand aus dem zunehmenden Trend zur Konvergenz zwischen leitungsvermittelnden und paketvermittelnden Netzen, die es erfordert, dass Dienste wie z.B. IN-Dienste [95] von Servern in paketvermittelnden Netzen in leitungsvermittelnden Telekommunikationsnetzen angeboten werden können. So wurde Ende 1998 die SIGTRAN (Signaling Transport) Arbeitsgruppe der Internet Engineering Task Force (IETF) gegründet, die zum Ziel hatte, sich des Transports von PSTN-Signalisierung über IP-basierte, paketvermittelnde Netzinfrastrukturen – unter der Berücksichtigung funktionaler Aspekte und Leistungsanforderungen – anzunehmen. In dieser Gruppe wurden zunächst die Anforderungen an ein Transportprotokoll zu diesem Zweck diskutiert, bevor man mit der Standardisierung eines dafür geeigneten Protokolls begann.
Im Oktober 2000 wurde nach mehr als 14 Revisionen das Dokument zum Stream Control Transmission Protocol (SCTP) [86] von der IETF in den Status eines Standards-Track RFC erhoben und somit gültiger Internet-Standard. Damit war ein vorläufiger Schlusspunkt in einer Entwicklung erreicht, die zum Ziel hatte, ein neues Transportprotokoll zu standardisieren – neben dem UDP und dem TCP das dritte bisher von der IETF verabschiedete.
Im folgenden Kapitel werden zunächst einige typische Anwendungsszenarien erläutert, die sich auf die Dienste des SCTP stützen, bevor detaillierter auf seine Funktionen eingegangen wird. Im Anschluss daran folgt eine Diskussion neuerer Standards, die die grundlegenden Fähigkeiten des SCTP erweitern. Daraus ergeben sich neue Anwendungsmöglichkeiten, die im letzten Teil vorgestellt werden.

2.1 Transport von Signalisierungsnachrichten

Die SIGTRAN Arbeitsgruppe der IETF hat in dem Dokument [60] eine Rahmenarchitektur für den Transport von Signalisierungsdaten über paketvermittelnde Netze definiert. Diese setzt ein leistungsfähiges Transportprotokoll auf der Schicht 4 des Internet-Referenzmodells voraus. Um die Anforderungen an dieses Transportprotokoll verstehen zu können, wird hier das zentrale Zeichengabesystem No. 7 (Common Channel Signaling System No. 7 – nachfolgend kurz SS7 genannt) vorgestellt, auf welches sich größtenteils die Übertragung von Signalisierungsmeldungen in heutigen Telekommunikationsnetzen stützt. Eine detaillierte Darstellung des SS7 findet sich in [9].

2.1.1 Das zentrale Zeichengabesystem No. 7

Das ISDN Protokollreferenzmodell trennt die Funktionen der Nutzdatenübertragung von den Netzkontroll- und Signalisierungsfunktionen. Zur Übertragung von Signalisierungsinformationen zwischen Endteilnehmer und Vermittlungsstelle kommt das D-Kanal-Protokoll (Digital Subscriber Signaling System No. 1, DSS1) [36] zum Einsatz, während zur Zwischenamtsignalisierung zwischen Vermittlungsstellen das zentrale Zeichengabesystem No. 7 [30] (SS7) genutzt wird. Durch diese Architektur ergeben sich logisch und zum Teil auch physikalisch getrennte Netze für den Nutzdatentransfer (typisch mehrere Kanäle mit einer Kapazität von je 64 KBit/s) und den schmalbandigen, paketorientierten Transfer von Signalisierungsmeldungen.

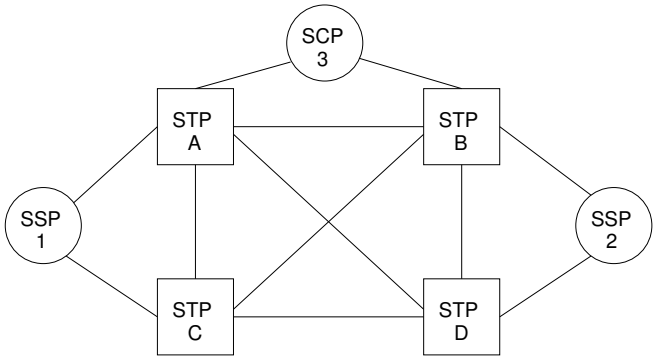

Abbildung 2.1: Einfaches Beispiel für ein SS7-Netz

In der SS7-Netztopologie unterscheidet man sog. Signalisierpunkte (SP) und Signalisiertransferpunkte (STP). Diese sind durch die zentralen Zeichengabekanäle (Signaling Links, SL) verbunden. SPs, die auch als Signalisierendpunkte (SEP) bezeichnet werden, sind Quellen und Senken der Signalisierungsmeldungen. Dabei unterscheidet man Vermittlungsstellen (Service Switching Points, SSP) und Dienststeuereinheiten (Service Control Points, SCP), wie z.B. Netzwerkdatenbanken (Location Register) oder Betriebs-und Wartungszentren. In ihnen sind typisch die Anwendungen, die Anwendungsprotokolle sowie die transportorientierten Protokollschichten implementiert. STPs leiten die Signalisierungsinformationen lediglich weiter und sorgen im Netz für Ausfallsicherheit (durch Redundanz) und optimale Wegewahl. In ihnen werden deshalb auch die oberen SS7-Protokollschichten nicht ausgewertet. Abbildung 2.1 zeigt ein einfaches SS7-Netz mit zwei Vermittlungsstellen SSP_1 und SSP_2, vier Signalisiertransferpunkten STP_A, STP_B, STP_C und STP_D, sowie einem SCP (SCP_3).
Die grundlegende Protokollarchitektur des SS7 zeigt Abbildung 2.2. Die transportorientierten Funktionsschichten 1-3 werden als “Nachrichtentransfer-Teil” (engl. Message Transfer Part, MTP) [32] bezeichnet. Die MTP-Schicht 1 (MTP1) realisiert typische Funktionen der OSI-Schicht 1, in der Regel auf TDM-basierten Signalisierkanälen. Die grundlegende Funktion der MTP-Schicht 2 ist die Übertragung von Nachrichteneinheiten (Message Signal Units, MSUs) über bidirektionale Signalisierkanäle mittels eines bitorientierten Sicherungsprotokolls. Dazu gehört auch die Fehlererkennung, Fehlerkorrektur, sowie eine Überwachung des Verbindungsstatus. Die MTP-Schicht 3 besteht aus zwei Funktionsblöcken : sie stellt einen einfachen, paketorientierten, verbindungslosen Dienst – eine Untermenge der OSI Schicht 3 – sowie Funktionen zum Management des Signalisierungsnetzes und der Signalisierverbindungen zur Verfügung (Wegewahl; alternative Wegewahl bei Ausfall eines SL; Überlastkontrolle). Der Signaling Connection Control Part (SCCP) [35] erweitert den MTP zum vollen Funktionsumfang der OSI Schicht 3 (OSI Network Service Access Point, N-SAP). So stellt er neben der Möglichkeit der Nutzung funktionaler Adressen (Global Titles, GT) auch vier verschiedene Dienste zur Verfügung:

1. verbindungsloser Dienst,
2. verbindungsloser, reihenfolgegesicherter Dienst,
3. verbindungsorientierter Dienst,
4. verbindungsorientierter Dienst mit Flusskontrolle.

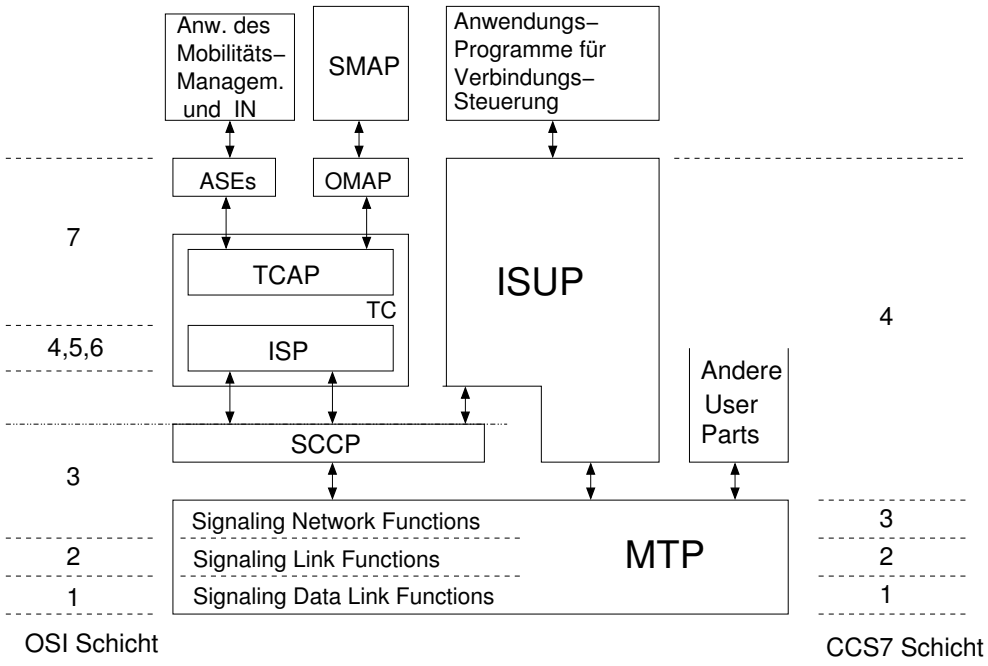

Abbildung 2.2: SS7 Protokoll-Architektur

Gemeinhin wird der SCCP in Kombination mit dem transaktionsorientierten Anwendungs-Teil (Transaction Capabilities Application Part, TCAP) eingesetzt, was es Anwendungen wie dem System-Management (SMAP) oder intelligenten Netzen (IN) [95] erlaubt, transaktionsbasierte Dienste zwischen zwei Netzknoten anzubieten. Darüber hinaus kommt SCCP beispielsweise im UTRAN als Transportprotokoll für den RANAP (Radio Access Network Application Part) zum Einsatz [1], welches den Aufbau des Radio Access Bearer (RAB) zwischen dem UMTS Kernnetz und Radio Network Controller (RNC) steuert.

Die Dienste des MTP und z.T. auch des SCCP werden von sog. Anwender-Teilen (engl. User Parts) in Anspruch genommen, die die OSI Schichten 4 bis 7 implementieren. Beispiele für diese Anwender-Teile sind z.B. der “Telephone User Part” (TUP), der die Rufsteuerung einfacher leitungsvermittelter Sprach- und Datenverbindungen durchführt, oder der ISDN User Part (ISUP) [34], der die Verbindungssteuerung leitungsvermittelter ISDN-Teilnehmerverbindungen realisiert.

2.1.2 Architekturen für den Signalisierungstransport

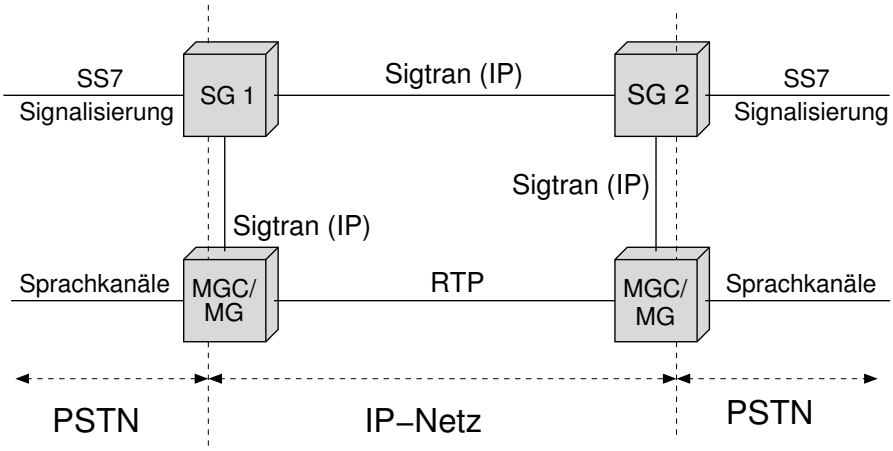

Abbildung 2.3: Rahmenarchitektur des IP-basierten Signalisierungstransports

Das Dokument [60] beschreibt eine Rahmenarchitektur für den Transport von Signalisierungsdaten zwischen leitungsvermittelnden und paketvermittelnden Netzen. Zentrale Elemente dabei sind Vermittlungsknoten, die sich an den Schnittstellen zwischen diesen Netzen befinden (engl. Gateways). Die Rahmenarchitektur sieht solche Gateways im Wesentlichen für die Umsetzung der Nutzdatenströme (engl. Media Streams) zwischen leitungsvermittelnden und paketvermittelnden Netzen vor, sowie für die Umsetzung und Übertragung von SS7-Signalisierungsmeldungen auf IP-basierte Signalisierungsprotokolle. Entsprechende Knoten heißen engl. Media Gateways (MG) und Signaling Gateways (SG). Des Weiteren beschreibt [60] Steuerknoten, die die Ressourcen der Media Gateways verwalten. Diese Steuerknoten heißen engl. Media Gateway Controller (MGC), und können als Signalisierungsendpunkte für Protokolle wie den SS7 ISDN User Part [34] oder Q.931/DSS1 [36] dienen.
Die logischen Funktionen von MGs, MGCs und SGs können auf verschiedene Art und Weise physikalisch implementiert werden; Abbildung 2.3 zeigt eine mögliche Realisierung, in der zwei SGs an SS7-Netze und zwei Knoten mit MG und MGC an herkömmliche Sprachkanäle eines leitungsvermittelnden Telekommunikationsnetzes angeschlossen sind. Die Kommunikation der SGs untereinander wird durch eine Variante des IP-basierten Signalisierungstransports rea-

lisiert (z.B. MTP3 über IP) und die Kommunikation zwischen SG und MG/MGC möglicherweise durch eine andere Variante (z.B. ISUP über IP). Zwei der möglichen Varianten des IP-basierten Signalisierungstransports werden exemplarisch in den folgenden Abschnitten näher erläutert.
Alle Varianten haben gemeinsam, dass ein einheitliches, auf IP aufsetzendes Transportprotokoll genutzt wird, welches durch geeignete Zwischenschichten (Adaptation Layers, AL) an das jeweilige Nutzerprotokoll angepasst wird. Daher muss die Funktionenkombination aus Anpassungsschicht (AL) und IP-basiertem Transportprotokoll so beschaffen sein, dass die oberen Protokollschichten (also je nach Anwendungsfall z.B. TCAP, ISUP, SCCP oder MTP3) ohne Modifikationen einsetzbar sind.

2.1.2.1 IP-basierte MTP2-Links

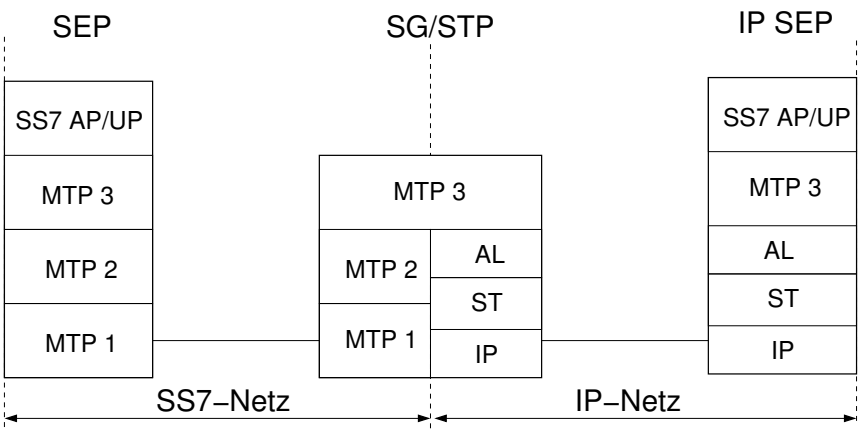

Abbildung 2.4: MTP2 Peer-to-Peer-Transport über IP-Netze

Abbildung 2.4 zeigt ein Anwendungsszenario einer Anpassungsschicht, mit der es möglich ist, STPs bzw. SGs im gemischten Betrieb mit IP-basierten Schnittstellen und traditionellen SS7-Verbindungen und SEPs oder auch STPs vollständig in IP-Netzen zu betreiben. Die Anpassungsschicht (AL) stellt in Verbindung mit dem IP-basierten Signalisiertransport (ST) die Funktionalität eines MTP2-Übertragungsabschnitts (Link) sowie die Dienstprimitive der MTP2-Schicht zur Verfügung. Für den Betrieb als vollwertiger, IP-basierter STP muss ein Knoten zu-

mindest MTP3 implementieren, für einen SEP wird neben dem MTP3 auch noch ein entsprechender Anwender-Teil (User Part) benötigt. Die notwendigen Funktionen der Anpassungsschicht stellt das in [28] definierte Protokoll “SS7 MTP2-User Peer-to-Peer Adaptation Layer” (M2PA) zur Verfügung.

2.1.2.2 MTP3-Anwender Anpassungsschicht

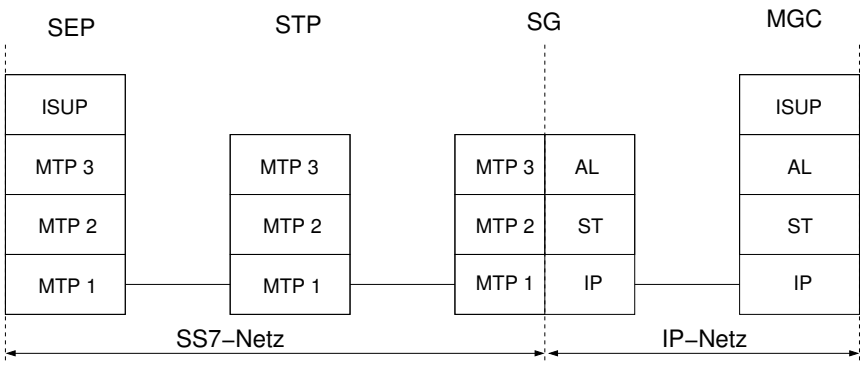

Abbildung 2.5: Transfer von MTP3-Primitiven über IP-Netze

Eine weitere Anwendungsmöglichkeit für den Signalisierungstransport über IP-Netze zeigt Abbildung 2.5. Dabei wird der Transport von MTP3-Anwender-Nachrichten am Beispiel des ISUP von einem SEP über ein SS7-Netz zu einem SG und weiter zu einem MGC in einem IP-Netz dargestellt. Die Anwender-Nachrichten, sowie Dienstprimitive zwischen MTP3 und MTP3-Anwender werden über die Anpassungschicht (AL) und Signalisiertransport (ST) an den MGC weitergeleitet. Das SG ist hier kein eigenständiger Knoten im SS7-Netz, erst die Kombination aus SG und MGC macht einen vollwertigen Signalisierendpunkt aus. Mithin wird also der MTP3 des SGs im MGC terminiert. Neben einem MGC als IP-basiertem Knoten sind natürlich auch z.B. IP-basierte Netzdatenbanken mit transaktionsorientierten Anwendungen (mittels TCAP über SCCP) denkbar. Ein Vorteil dieser Lösung ist die Tatsache, dass mehrere, vergleichsweise einfache Knoten im IP-Netz die Funktion eines SEPs in Kombination mit einem SG übernehmen können. Darüber hinaus können diese Knoten transparent über mehrere SGs mit dem SS7-Netz verbunden sein, was eine sehr flexible Nutzung ermöglicht. Mit dem Protokoll “SS7 MTP3-User Adaptation Layer” (M3UA) [76]

hat die SIGTRAN-Gruppe eine Anpassungsschicht definiert, mit der die oben beschriebenen Anwendungen implementiert werden können.

2.1.3 Anforderungen an das Transportprotokoll

In der ITU-Empfehlung Q.706 [31] werden folgende Anforderungen an SS7-Signalisierverbindungen gestellt:

- Durch Ausfall oder Fehler des MTP darf nicht mehr als eine von 10^7 Nachrichten verloren gehen.
- Nicht mehr als eine von 10^{10} Nachrichten darf dem MTP-Nutzer außer der Reihenfolge zugestellt werden. Das betrifft auch die Verdopplung von Nachrichten.
- Die Wahrscheinlichkeit, dass Fehler einer empfangenen Nachricht nicht erkannt werden, muss kleiner als 1 zu 10^{10} sein.
- Die Verfügbarkeit einer Signalisierbeziehung (zwischen zwei SPs) muss wenigstens 0,99998 betragen, das entspricht einer Ausfallzeit von höchstens 10 Minuten pro Jahr.

Die oben beschriebenen Protokollarchitekturen, in denen dem SS7-Netz Dienste aus einem IP-basierten Netz zur Verfügung gestellt werden, müssen daher so organisiert sein, dass diese Anforderungen erfüllt werden können. Das für den Signalisierungstransport (ST) eingesetzte Transportprotokoll spielt dabei eine entscheidende Rolle.
Diskussionen in der SIGTRAN-Gruppe ergaben, dass dieses folgenden Ansprüchen genügen sollte:

- Flexible, gesicherte Datenübertragung (z.B. mit oder ohne Reihenfolgesicherung).
- Erhaltung von Nachrichtengrenzen.
- Flexible Übertragung von Nachrichten (Übertragung eines Multiplexes aus mehreren Nachrichtenströmen).
- Unterstützung mehrerer Netzwerkpfade (Multihoming).
- Geeignete Fluss- und Überlastkontrollmechanismen für den Einsatz im Internet.

- Überwachungsmechanismen für Netzwerkpfade und Endpunkte.
- Konfigurierbare Timer.
- Implementierung aktueller Sicherheitsmechanismen und Vermeidung bekannter Sicherheitsprobleme.

In der SIGTRAN-Gruppe wurde ebenfalls diskutiert, inwieweit die bereits etablierten, IP-basierten Transportprotokolle User Datagram Protocol (UDP) [64] und Transmission Control Protocol (TCP) [66], die in bisherigen IP-basierten Lösungen Signalisierungsmeldungen übertragen, die nötigen Anforderungen für SS7-Protokolle erfüllen.

2.1.3.1 TCP als Transportprotokoll für den Signalisierungstransport

TCP ist ein verbindungsorientiertes Protokoll, das sowohl Fehlersicherung als auch Flusssteuerung zur Verfügung stellt. Es hat jedoch folgende Nachteile:

- TCP ist stream-orientiert. Das bedeutet, dass es nicht einzelne Nachrichten, sondern einen Strom einzelner Bytes transportiert. Sollen jedoch Nachrichten übertragen werden, so muss die Anwendung Nachrichtengrenzen einfügen (Record Marking). Das sofortige Absenden vollständiger Nachrichten muss durch die Anwendung mittels des Push-Mechanismus erzwungen werden.
- Transaktionsorientierte Anwendungen erfordern lediglich eine teilweise Reihenfolgesicherung von Meldungen. TCP erbringt einen strikt reihenfolgegesicherten Dienst, wodurch es unnötigerweise zu einer Blockierung bereits angekommener Daten kommen kann (engl. Head-of-Line Blocking). Dies kann bei Paketverlusten – je nach Rundlaufverzögerung (engl. Round Trip Time, RTT) – zu inakzeptablen Wartezeiten für Signalisierungsanwendungen führen.
- TCP unterstützt keine Redundanzmechanismen auf der Netzwerkschicht. Eine TCP-Verbindung wird durch genau zwei Transportadressen charakterisiert, das sind Kombinationen aus einer IP-Adresse und einer Portnummer. Eine Unterstützung für mehrere redundante Wege (engl. Multihoming) müsste in der Anwendungsschicht durch mehrere parallele TCP-Verbindungen implementiert werden.

- Typische TCP-Implementierungen erlauben nicht, spezifische Protokollparameter zu kontrollieren. Das ist allerdings für Szenarien erforderlich, in denen Signalisierungsdaten mit zeitkritischen Anforderungen übertragen werden.

- Viele TCP-Implementierungen sind anfällig für Angriffe, die die Dienstgüte beeinträchtigen (engl. Denial-of-Service Attacks).

Insbesondere die ersten drei Aspekte führten dazu, dass TCP nicht ernsthaft als Transportprotokoll für die SIGTRAN-Protokollarchitekturen in Erwägung gezogen wurde.

2.1.3.2 UDP als Transportprotokoll für den Signalisierungstransport

UDP erbringt einen unzuverlässigen, ungeordneten und nachrichtenorientierten verbindungslosen Transportdienst. Ferner bietet es keine Flusssteuerungs- und Überlastkontrollmechanismen. Daher ist es als Transportprotokoll für den Signalisiertransport ungeeignet.
Da UDP allerdings ein sehr einfaches Protokoll ist, wurde in der SIGTRAN-Gruppe als Signalisiertransport zunächst der Protokollvorschlag Multi-Network Datagram Transmission Protocol (MDTP) [84] diskutiert, der sich auf die Dienste des UDP stützt und folgende Mechanismen definiert:

- Zuverlässiger Transport von Nachrichten (durch Vergabe von Sequenznummern, Quittierung erhaltener Nachrichten und gegebenenfalls Neuübertragung verlorener Nachrichten).

- Flusssteuerungs- und Überlastkontrollmechanismen.

- Unterstützung von mehreren Transportadressen pro Protokollendpunkt (Multihoming).

- Überwachung der Verfügbarkeit von Netzwerk-Pfaden und assoziierten Protokollendpunkten (durch periodische sog. Heartbeats).

MDTP wurde lange als Lösung für den Signalisiertransport gehandelt, mehrfach überarbeitet und modifiziert, mündete schließlich aber in die Definition des IP-basierten Transportprotokolls SCTP [86], das ausführlich in Abschnitt 2.3 erläutert wird.

2.2 Reliable Server Pooling

Die RSerPool-Arbeitsgruppe der IETF arbeitet an einer Rahmenarchitektur [91] für die Nutzung und Verwaltung von zuverlässigen Serververbünden. Diese Architektur definiert Server (engl. Pool Elements, PEs), die in einem Verbund einen Dienst anbieten und Nutzer dieses Dienstes (engl. Pool Users, PUs). Des Weiteren gehört zur RSerPool-Architektur ein Namensdienst, der von einem besonderen Server-Verbund, den Namens-Servern (engl. Name Servers, NSs) angeboten wird. Er liefert den PUs einen zu einem Dienstnamen (z.B. “Video Service”) gehörigen Zeiger (Pool Handle, PH). Die oben erwähnte Rahmenarchitektur beschreibt die Nutzung der zwei folgenden Protokolle:

1. das “Endpoint Name Resolution Protocol” (ENRP) [93] dient der Kommunikation zwischen Namens-Servern und erlaubt ihnen den Austausch von Informationen zu allen Server-Verbünden, d.h. zu allen registrierten PEs sowie deren angebotenen Diensten.

2. das “Aggregate Server Access Protocol” (ASAP) [87] dient der Kommunikation folgender Architekturelemente:

   (a) PU und NS: Zur Abfrage von zu einem PH gehörigen Kombinationen aus Transportprotokoll und Transportadressen. Diese adressieren PEs, die den entsprechenden Dienst erbringen.

   (b) PE und NS: Zur Registrierung (und Deregistrierung) von PEs bei ihren NSs.

   (c) PU und PE: Zum Austausch der eigentlichen Nutzdaten, sowie relevanter Daten zur Organisation eines Serverwechsels im Fehlerfall (Failover).

   (d) PE und PE: Dies ist ein Spezialfall der Kommunikation zwischen PU und PE.

Bei allen diesen Kommunikationsbeziehungen ist SCTP als das ENRP bzw. ASAP stützende Transportprotokoll vorgesehen, um eine größtmögliche Flexibilität und Redundanz auf der Netzwerkschicht erzielen zu können. Lediglich im Falle der Kommunikation zwischen PU und NS (2.a) sowie zwischen PU und PE (2.c) ist ebenfalls vorgesehen, dass TCP anstatt SCTP (wenn auch mit gewissen funktionalen Einschränkungen) genutzt werden kann. Dadurch sollen einer größtmöglichen Anzahl von Nutzern die RSerPool-Dienste zugänglich gemacht werden.

2.3 Protokolleigenschaften des SCTP

Das Stream Control Transmission Protocol (SCTP) ist ein nachrichtenorientiertes Vielzweck-Transportprotokoll, welches durch die Trennung von zuverlässigem Transport und Reihenfolgesicherung einen flexibleren Datentransport erlaubt als TCP und durch die Unterstützung von Redundanzmechanismen in der Netzwerkschicht (Unterstützung von Multihoming) eine erhöhte Fehlertoleranz aufweist.
SCTP ist in den Dokumenten [86, 89] standardisiert. Der ursprüngliche Standard wurde im Oktober 2000 von der SIGTRAN-Gruppe verabschiedet und von der IETF zum Standards-Track RFC erhoben. Seither werden alle Erweiterungen und Fortentwicklungen zum Thema SCTP in der Transport Area-Arbeitsgruppe der IETF (TSVWG) diskutiert. Im September 2002 wurde mit [89] der vom SCTP zu verwendende Prüfsummen-Algorithmus (bis dahin Adler-32) geändert. Seither ist die Verwendung des CRC32C-Algorithmus durch den Standard vorgeschrieben (siehe auch Abschnitt 2.3.2).

2.3.1 Nachrichtenorientierter Transport

SCTP ist ein verbindungsorientiertes Protokoll. Eine SCTP-Verbindung wird *Assoziation* (engl. Association) genannt; sie besteht zwischen zwei Protokollendpunkten, die jeweils durch eine Menge von Transportadressen (d.h. Kombinationen von IP-Adressen und SCTP Portnummern) charakterisiert sind. Alle Transportadressen eines SCTP-Endpunktes haben identische Portnummern.
Beim Verbindungsaufbau werden zwischen den Endpunkten alle Transportadressen ausgetauscht, die Quell- oder Zieladressen eines SCTP-Pakets innerhalb einer Assoziation sein dürfen. Ferner handeln die beiden Endpunkte eine Anzahl von unidirektionalen logischen Übertragungskanälen für Nachrichtenströme aus, innerhalb derer dem Empfänger Nachrichten in der Regel reihenfolgegesichert zugestellt werden, es sei denn der Anwender verlangt eine ungeordnete Zustellung. Diese Kanäle werden nachfolgend mit “Streams” bezeichnet.
Im Gegensatz zum TCP bewahren die Sende- und Empfangsprimitive des Protokolls die Nachrichtengrenzen. Dies hat einen enormen Vorteil für Anwendungen, die nachrichtenbasiert agieren, denn sie müssen nicht mehr umständlich die Markierung von Nachrichten und deren Längen vornehmen.
Das Protokoll kann mehrere kurze Nachrichten gebündelt in einem SCTP-Paket übertragen, welches von einem verbindungslosen, datagrammorientierten Netzwerk-Dienst (wie z.B. dem IP) zum Ziel-Endpunkt transportiert wird. Im Folgenden wird von IP als Netzwerk-Dienst ausgegangen. SCTP vermeidet eine

Fragmentierung von Daten, die über IP transportiert werden, durch die Anwendung der sog. “Path MTU Discovery”, die in [56] ausführlich beschrieben wird.

2.3.2 Paketformate

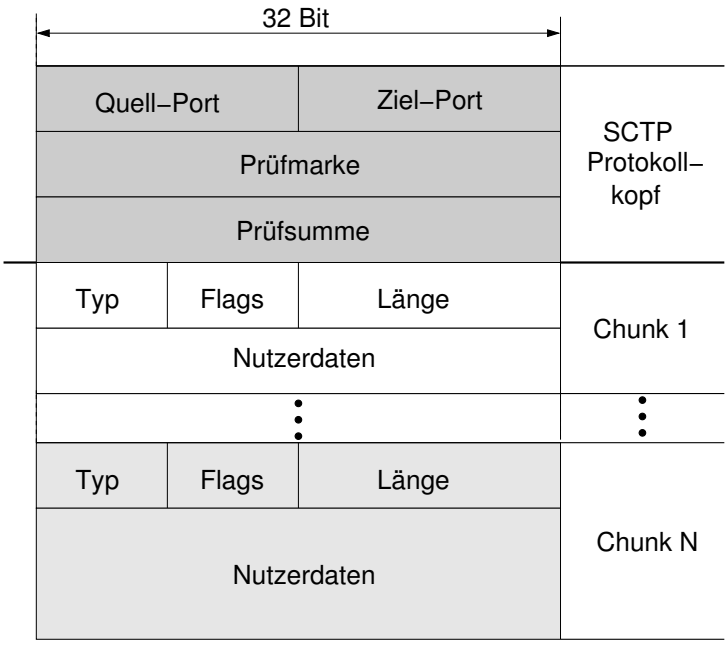

Abbildung 2.6: SCTP Paketformat mit Common Header und Chunks

Wie in Abbildung 2.6 dargestellt, bestehen SCTP Pakete aus einem einheitlichen, 96 Bit langen Protokollkopf (engl. Common Header), gefolgt von einer variablen Anzahl an Protokolleinheiten, nachfolgend engl. *Chunks* genannt. Der Protokollkopf enthält, analog zu TCP und UDP, zwei 16 Bit lange Quell- und Zielportnummern, eine 32 Bit lange Prüfmarke (engl. Verification Tag) und eine 32 Bit lange CRC32C-Prüfsumme [89]. Die Prüfmarke ist ein zufällig gewählter Wert, der beim Verbindungsaufbau zwischen beiden Protokollendpunkten ausgetauscht wird. Mit Hilfe der Prüfmarke wird unterbunden, dass gefälschte Pakete durch sog. blinde Angriffe (bei denen der Angreifer die Verbindung nicht belauschen kann) in bestehende Assoziationen eingeschleust werden können.

Wenn ein Endpunkt ein SCTP Paket empfängt, wird zunächst verifiziert, ob es eine korrekte Prüfsumme trägt. Ist dies nicht der Fall, so wird das Paket verworfen, da es bei der Übertragung verfälscht wurde. Dann überprüft der empfangende Endpunkt, ob er anhand der Quell- und Zieladressen (also der IP-Adressen) und der Quell- und Zielports des Pakets dieses zu einer bestehenden Verbindung zuordnen kann, oder ob es sich um ein Paket handelt, das zu einer neuen Verbindung gehört. Im ersten Fall (Paket gehört zu einer bestehenden Verbindung) wird nun überprüft, ob die Prüfmarke zu der bestehenden Verbindung passt. Ist dies nicht der Fall, so wird das Paket verworfen und ein Abort-Chunk an die Gegenseite gesendet. Ansonsten werden die empfangenen Chunks der Reihe nach bearbeitet. Im zweiten Fall (Paket gehört zu einer neuen Verbindung) wird die Prüfmarke vom Sender auf 0 gesetzt und beim Empfänger ignoriert (siehe auch Abschnitt 2.3.3). Im Fehlerfall können Endpunkte allerdings bestimmte Meldungen auch dann bearbeiten, wenn sie die entsprechenden Prüfmarken nicht zuordnen können (siehe auch Abschnitt 2.3.4).
Chunks bestehen zumindest aus einem Header und sind mindestens 32 Bit lang. Darin sind enthalten:

- der Typ (8 Bit),
- optional genutzte Flags (8 Bit),
- und das Längenfeld (16 Bit).

Man unterscheidet zwei Chunktypen: Daten-Chunks (engl. Data Chunks) und Kontroll-Chunks (engl. Control Chunks). Daten-Chunks übertragen die eigentlichen Nutzerdaten (siehe auch Abschnitt 2.3.5). Kontroll-Chunks werden genutzt

- zum Verbindungsaufbau,
- zum Verbindungsabbau,
- für Bestätigungen empfangener Nutzdaten,
- für die Überwachung der Erreichbarkeit des assoziierten Endpunkts,
- für die Überwachung des Status von Netzwerkpfaden zum assoziierten Endpunkt,
- für die Übertragung von Fehlermeldungen,
- für optionale Protokollerweiterungen.

Daten-Chunks, sowie eine Vielzahl von Kontroll-Chunks, haben eine Reihe vorgeschriebener Protokollfelder, deren Format in [86] definiert ist. Darüber hinaus führen einige Kontroll-Chunks auch noch eine Zahl vorgeschriebener und/oder optionaler Parameter variabler Länge mit, deren Format gemäß Abbildung 2.7 durch ein Typfeld, ein Längenfeld und ein Datenfeld bestimmt ist. Chunks und Parameter, die nicht ein Vielfaches von 32 Bit lang sind, werden mit so genannten Füllbytes (engl. Padding) mit dem Binärwert 0 aufgefüllt.

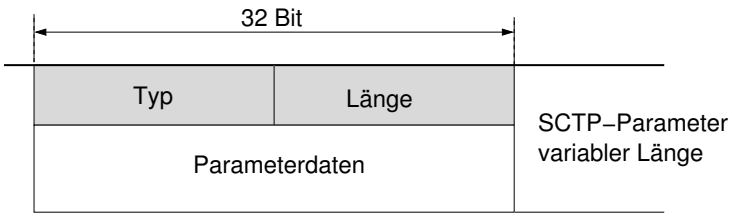

Abbildung 2.7: SCTP Parameterformat

2.3.3 Verbindungsaufbau

SCTP-Assoziationen werden (im einfachsten Fall) durch Austausch von vier SCTP Paketen mit Kontroll-Chunks etabliert, wie in Abbildung 2.8 dargestellt. Der Protokollendpunkt, der die Verbindung initiiert, wird nachfolgend “Client” genannt, der passive Protokollendpunkt, der den Verbindungsaufbauwunsch akzeptiert, wird “Server” genannt. Der Client sendet zunächst einen INIT-Chunk, der als vorgeschriebene Felder neben der Größe des Sendefensters die vom Server zu verwendende Prüfmarke (engl. Initiate Tag), die Anzahl der maximal möglichen Streams in Sende- und Empfangsrichtung ($\leq 2^{16}$) und die erste Sendereihenfolgenummer (engl. Initial Transmission Sequence Number) enthält. Als variable Parameter sendet der Client eine Liste aller von ihm innerhalb der betreffenden Assoziation verwendeten IP-Adressen mit (dies ist vorgeschrieben, sofern es mehr als eine Adresse ist). Gleichzeitig wird ein Timer gestartet, der bei einem möglichen Verlust des INIT-Chunks eine Neuübertragung initiiert.
Der Server sendet eine Bestätigung des INIT-Chunks, einen sog. INIT-ACK-Chunk. Dieser enthält die gleichen Informationen wie ein INIT-Chunk und zusätzlich noch einen Parameter, der als engl. Cookie bezeichnet wird. Das Cookie enthält sämtliche Informationen aus INIT- und INIT-ACK-Chunk und zusätzlich

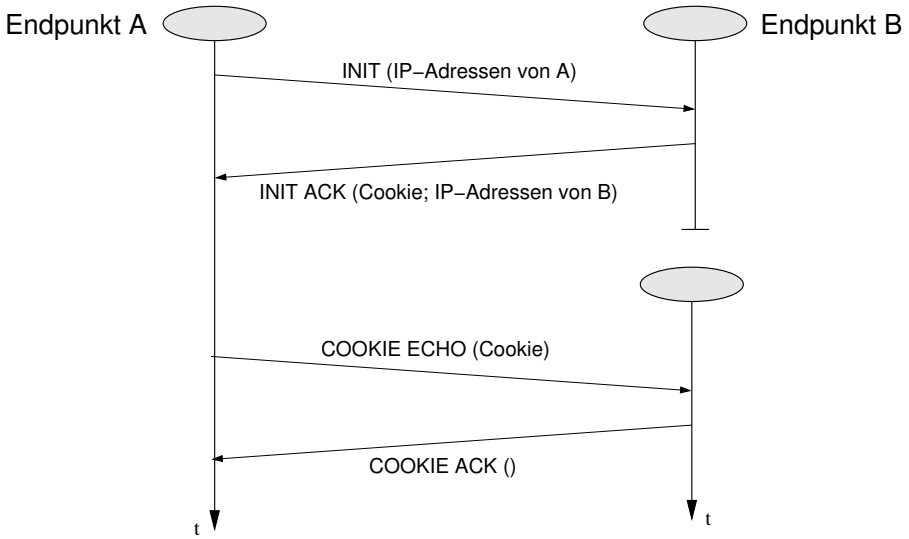

Abbildung 2.8: Aufbau einer SCTP-Assoziation

noch eine kryptographisch sichere Signatur (z.B. SHA-1 [21]) aller Daten des Cookie-Parameters. Diese Signatur wird mit einen nur dem Server bekannten geheimen Schlüssel erzeugt. Mit der Signatur kann der Server – und nur der Server – die Korrektheit des Cookies verifizieren. Nach dem Aussenden des INIT-ACK-Chunks alloziert der Server keine Ressourcen und löscht alle zu der betreffenden, im Aufbau befindlichen Assoziation gehörenden Informationen. Dadurch werden Angriffe, die frühzeitig Serverressourcen blockieren können (engl. Denial of Service, DoS) ausgeschlossen.
Der Client sendet ein erhaltenes Cookie mit dem dritten Kontroll-Chunk, dem sog. COOKIE-ECHO zurück. Erst wenn der Server dieses COOKIE-ECHO erhält und die Korrektheit der Signatur des Cookies verifiziert hat, wird er die nötigen Ressourcen für eine Assoziation belegen und in den Zustand einer ordentlich etablierten Assoziation wechseln. Er sendet zur Bestätigung ein COOKIE-ACK-Chunk für das erhaltene COOKIE-ECHO an den Client und kann danach weitere Daten vom Client akzeptieren. Der Client wechselt nach dem Erhalt des entsprechenden COOKIE-ACK-Chunks in den Zustand einer etablierten Assoziation.
Die Abbildung 2.9 zeigt ein Zustandsdiagramm des Verbindungsaufbaus, in dem der Client über die Zustände CLOSED, COOKIE-SENT und COOKIE-ECHOED den Zustand ESTABLISHED (entspricht der etablierten Assoziation) erreicht, während der Server vom Zustand CLOSED erst nach Erhalt des gültigen Cookies

direkt in den etablierten Zustand wechselt. Die Übergangszustände des Clients sind entsprechend durch Timer gegen Nachrichtenverlust abgesichert. SCTP-Implementierungen der Clients, wie auch der Server, erzeugen bei dem Übergang vom Zustand CLOSED zu ESTABLISHED eine Datenstruktur, die alle relevanten Daten einer Assoziation enthält. Diese ist in Abbildung 2.9 und nachfolgend mit TCB (engl. Transmission Control Block) bezeichnet und wird in Abschnitt 12 von [86] näher erläutert.

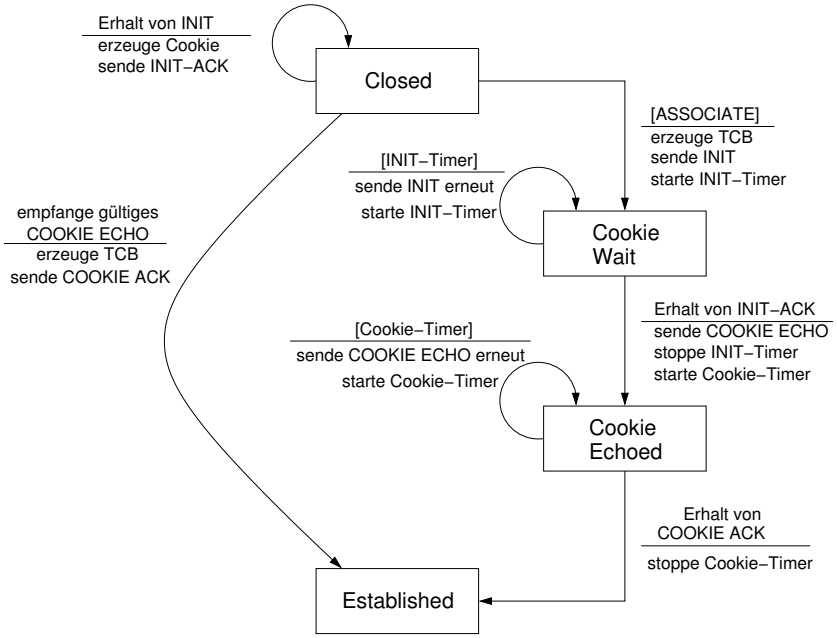

Abbildung 2.9: Zustandsdiagramm des Aufbaus einer SCTP-Assoziation

2.3.4 Verbindungsabbau

Für den Verbindungsabbau einer etablierten SCTP-Assoziation definiert der RFC 2960 zwei Methoden:

1. den Abbruch (engl. Abort). Er wird durch Aussenden eines ABORT-Chunks initiiert und in besonderen Fällen eingesetzt, wenn der sichere Austausch

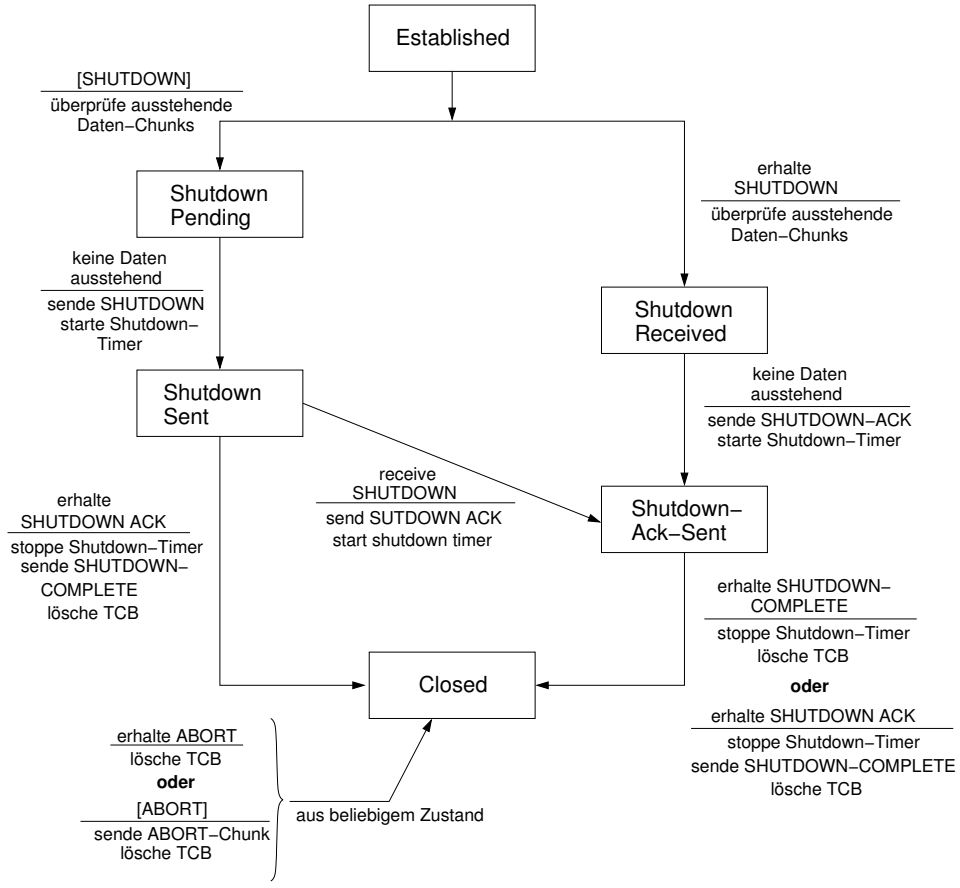

Abbildung 2.10: Zustandsdiagramm des Abbaus einer SCTP-Assoziation

noch im Transfer oder in Pufferspeichern befindlicher Dateneinheiten vernachlässigbar oder nicht mehr möglich ist. Dies kann z.B. vor dem Herunterfahren eines Rechners der Fall sein, der noch aktive Assoziationen etabliert hat.

2. die ordentliche Beendigung (engl. Graceful Termination). Dabei wird von beiden Seiten sicher gestellt, dass noch ausstehende Dateneinheiten gesendet und von der Gegenseite bestätigt wurden.

Bei letzterem Verfahren leitet der SCTP-Nutzer den Verbindungsabbau durch Aufruf des SHUTDOWN-Primitivs ein, und der Endpunkt wechselt in den Zu-

stand “Shutdown Pending” (vgl. Abbildung 2.10). Sind alle Daten von der Gegenseite bestätigt worden, so sendet er einen SHUTDOWN-Chunk und wechselt in den Zustand “Shutdown Sent”. Dieser Zustand ist durch einen Timer gesichert, bei dessen Ablauf erneut ein SHUTDOWN-Chunk gesendet würde. Bei Erhalt des SHUTDOWN-Chunks wechselt die Gegenseite in den Zustand “Shutdown Received”.
Hat sie nun ihrerseits alle Dateneinheiten gesendet und bestätigt bekommen, so sendet sie als Bestätigung des SHUTDOWN einen SHUTDOWN-ACK-Chunk, startet einen Timer, und wechselt in den Zustand “Shutdown-Ack Sent”. Der Endpunkt, der den Verbindungsabbau einleitet, erhält den SHUTDOWN-ACK-Chunk und sendet einen SHUTDOWN-COMPLETE-Chunk zurück. Danach löscht er alle Assoziationsdaten und wechselt in den Zustand “Closed”.
Wenn die Gegenseite den SHUTDOWN-COMPLETE-Chunk erhält, löscht sie ebenfalls alle Assoziationsdaten und nimmt den Zustand “Closed” an. Bei Verlust dieser letzten Meldung würde sie nach Ablauf des Timers eine Neuübertragung des SHUTDOWN-ACK-Chunks durchführen, was der Empfänger wieder mit einem SHUTDOWN-COMPLETE-Chunk beantworten muss (auch wenn er alle Assoziationsdaten bereits gelöscht hat; siehe auch Abschnitt 8.4.5. von [86]).
Situationen, in denen beide Endpunkte die Assoziation zu einer ähnlichen Zeit beenden wollen, werden durch die übrigen Zustandsübergänge in Abbildung 2.10 abgedeckt.

2.3.5 Übertragung von Nutzdaten

Das SCTP unterstützt zunächst einen zuverlässigen, gesicherten Transport zwischen zwei Endpunkten. Dieser wird erreicht durch die Nummerierung von übertragenen Dateneinheiten (Daten-Chunks) mit sog. Übertragungssequenznummern (engl. Transmission Sequence Numbers, TSN) und Quittierung des Erhalts durch die Gegenseite mittels selektiver Bestätigungen (engl. Selective Acknowledgement, SACK) in SACK-Chunks. Für TSNs steht ein Feld von 32 Bit zur Verfügung; der erste übertragene Daten-Chunk erhält einen beim Verbindungsaufbau vereinbarten Wert.
Der SACK-Chunk enthält neben der für die Flusssteuerung wichtigen Fenstergröße (siehe auch Abschnitt 2.3.8) die größte in fortlaufender Reihenfolge empfangene TSN (engl. Cumulative TSN Acknowledged, CTSNA), sowie zwei weitere mögliche Angaben:

- Bereiche von TSNs, die nicht in fortlaufender Reihenfolge empfangen wur-

den. Die nicht erhaltenen, also fehlenden TSNs nennt man “Lücken” (engl. Gaps).

- Werte von TSNs, die mehrfach erhalten wurden. Diese heißen “Duplikate”.

Quittungen werden verzögert gesendet, spätestens aber 200 ms nach dem Erhalt eines Datenpakets oder sofort nach dem Erhalt des jeweils zweiten Datenpakets. In einem möglichen Fehlerfall, wie etwa bei Auftauchen von Lücken in den SACK-Chunks, wird für jedes Datenpaket sofort eine Quittung gesendet.
Da SCTP insbesondere als Transportprotokoll für kurze Signalisierungsnachrichten entworfen wurde, existiert die Möglichkeit, effektiv mehrere kurze Nachrichten (Daten-Chunks) mit Kontroll-Chunks (wie etwa einem SACK-Chunk) in einem SCTP-Paket zusammenzufassen (engl. Bundling). Dabei ist in [86] festgelegt, dass Daten-Chunks nach den Kontroll-Chunks im Paket angeordnet sind und der Reihenfolge nach verarbeitet werden.

2.3.5.1 Verlust eines Daten-Chunks

Das SCTP reagiert auf den Verlust eines Datenchunks – je nach Protokollsituation – mit Veränderungen der Fluss- und Überlastkontroll-Parameter, was weiter unten in Abschnitt 2.3.8 erläutert wird, sowie mit der Neuübertragung der entsprechenden Daten.
Für einen zu sendenden Daten-Chunk wird ein sog. T3-Timer gestartet – falls noch keiner läuft – und erst wieder gestoppt, wenn entsprechende Quittungen für alle noch nicht bestätigten Daten-Chunks eingegangen sind. Treffen neue Quittungen ein, ohne dass *alle* ausstehenden Daten bestätigt wurden, so wird der entsprechende Timer neu gestartet. Läuft der Timer aus, so wurde für einen Daten-Chunk keine passende Quittung empfangen, und der Daten-Chunk muss neu übertragen werden. Dies nennt der RFC 2960 “timer-basierte Neuübertragung”. Sie kann auftreten, wenn kaum Daten gesendet werden und ein Paket verloren geht, oder wenn der entsprechende SACK-Chunk für einen gesendeten Daten-Chunk verloren geht und der T3-Timer ausläuft, bevor weitere SACKs eintreffen.
Neben der timer-basierten Neuübertragung definiert der RFC 2960 noch die “schnelle Neuübertragung” (engl. Fast Retransmission), die bei Verlusten während laufender Datenübertragung insgesamt einen günstigeren Protokollverlauf erlaubt. Dabei wird bei jedem ankommenden SACK-Chunk nach wiederholt auftretenden Lücken bei den bestätigten TSNs gesucht. Für jeden Daten-Chunk wird ein Zähler (engl. Gap Counter) geführt, der anzeigt, wie oft die TSN des

Chunks in einer Lücke enthalten war. Wird eine TSN viermal durch eine Lücke als nicht erhalten markiert, so muss der Endpunkt diese neu übertragen.

2.3.5.2 Flexible Nachrichtenzustellung durch den Stream-Mechanismus

Beim Verbindungsaufbau vereinbaren beide SCTP-Endpunkte eine Anzahl von ein- und ausgehenden unidirektionalen Nachrichtenkanälen, die Nachrichtenströme (Streams). Innerhalb der Streams können Nachrichten entweder geordnet oder aber in Ankunftsreihenfolge an den SCTP-Nutzer ausgeliefert werden. Letztere können also die geordneten Nachrichten überholen.
Dieser Mechanismus benutzt zwei 16 Bit-Felder im Daten-Chunk, die zum einen die Nummer des Nachrichtenstroms und zum anderen eine weitere, von der TSN unabhängige Sequenznummer, die Strom-Sequenznummer (engl. Stream Sequence Number, SSN), tragen. Er hat zum Zweck, unnötige Blockierungseffekte zu vermeiden, die dadurch zustande kommen können, dass einzelne Nachrichten verloren gehen und neu übertragen werden müssen. Bei strikt geordneter Auslieferung wird dann die Zustellung nachfolgender Daten verzögert. Der Stream-Mechanismus erlaubt es nun, voneinander unabhängige Daten innerhalb verschiedener Streams zu transportieren, so dass verlorene Daten eines Streams nicht die Zustellung von Daten an den SCTP-User eines anderen Streams verzögern. Aus der Feldgröße ergibt sich eine maximale Anzahl von $2^{16} = 65536$ Streams, und ein Bereich für Strom-Sequenznummern von 0 bis 65535.

2.3.5.3 Segmentierung von Nachrichten

Da SCTP ein Vielzweck-Transportprotokoll ist, können Nachrichten verschiedener Längen transportiert werden. Um die ineffektive IP-Fragmentierung zu vermeiden, schreibt [86] vor, dass SCTP-Implementierungen die Größe der MTU auf benutzten Netzwerkpfaden gemäß [56] feststellen.
Übersteigt die Länge einer zu sendenden Nachricht die maximale Länge, die aufgrund der festgestellten Pfad-MTU in einem Daten-Chunk als Nutzdaten transportiert werden kann, so muss die Protokoll-Implementierung diese Daten in mehrere Segmente aufteilen. Diese Segmente werden nachfolgend in separaten, mit aufeinander folgenden TSNs versehenen Daten-Chunks übertragen. In diesem Fall stimmt also die Zuordnung Nachricht $\leftrightarrow$ Daten-Chunk nicht mehr. Um die Segmente wieder korrekt zusammensetzen zu können, werden die entsprechenden Daten-Chunks mit Markierungen (Flags) versehen, die Beginn, Mitte und Ende eines Segments eindeutig kennzeichnen. Diese Flags und die direkt aufeinander

folgenden TSNs der einzelnen Segmente erlauben dem Empfänger eindeutig, die Nachricht aus den Nachrichtenfragmenten zu rekonstruieren.

2.3.6 Unterstützung von Multihoming

Das SCTP unterstützt das sog. Multihoming, d.h. Endpunkte, die mehr als eine IP-Adresse besitzen. Der RFC 2960 bezeichnet den Weg von einem Endpunkt zu einer IP-Adresse des assoziierten Endpunktes als Netzwerk-Pfad oder einfach *Pfad*. Da IP-Implementierungen die zu einer Zieladresse gehörige Quelladresse durch den Prozess der Wegewahl (engl. Routing) festlegen, besitzt ein Endpunkt A, der n IP-Adressen besitzt, zu einem Endpunkt B mit m IP-Adressen genau m gültige Pfade. IP-Pakete, die A zu B sendet, können genau m mögliche Kombinationen aus Quell- und Zieladressen aufweisen. Ebenso besitzt B natürlich n Pfade zum Endpunkt A. Dabei wird davon ausgegangen, dass die Endpunkte die Wegewahl-Tabelle (engl. Routing Table) nicht beeinflussen können. Sollte eine SCTP-Implementierung die Wegewahl-Tabelle beeinflussen können, so besitzt im obigen Beispiel Endpunkt A gemäß Definition immer noch m Pfade zu Endpunkt B, IP-Pakete können aber mehr als m Kombinationen aus Quell- und Zieladressen aufweisen und zwar maximal $m \cdot n$ Kombinationen.

Jeder Endpunkt führt pro Pfad einen separaten Parametersatz für die Fluss- und Überlastkontrolle (siehe auch Abschnitt 2.3.8). Somit wird eine Kompatibilität zu etablierten Transportprotokollen wie TCP sichergestellt.

Nach dem Aufbau einer Assoziation kennen beide Protokollendpunkte alle möglichen Pfade zu ihrem assoziierten Endpunkt. Jeder Endpunkt wählt einen der Pfade aus, der nachfolgend der sog. *Primärpfad* wird. Der Primärpfad trägt die Hauptverkehrslast der Assoziation, kann aber vom SCTP-Nutzer jederzeit neu festgelegt werden. Andere Pfade werden nur für den Datentransfer genutzt, wenn Neuübertragungen durchgeführt werden (vgl. Abschnitt 2.3.5). Ein Endpunkt soll, wenn möglich, Neuübertragungen nur über Pfade vornehmen, die vom ursprünglich gewählten Pfad (in der Regel vom Primärpfad) verschieden sind. Zusätzlich kann der SCTP-Nutzer explizit verlangen, dass eine Nachricht über einen bestimmten Pfad gesendet wird.

Bestätigungen (also SACK-Chunks) für empfangene Daten sollen an diejenige Adresse zurückgesendet werden, von der die Daten kamen. Werden durch einen SACK-Chunk Daten bestätigt, die von mehreren unterschiedlichen Adressen empfangen wurden, so steht es der Implementierung frei, einen geeigneten Pfad für die Quittungen auszuwählen.

2.3.7 Fehlererkennung und Fehlermanagement

Der RFC 2960 definiert Mechanismen zur einfachen Überwachung des Pfadzustands innerhalb einer etablierten Assoziation und damit einhergehend die Überwachung der Erreichbarkeit eines assoziierten Endpunkts. Dies geschieht durch zwei bisher noch nicht erwähnte Kontroll-Chunks:

1. den HEARTBEAT-Chunk und
2. den HEARTBEAT-ACK-Chunk.

Endpunkte senden periodisch HEARTBEAT-Chunks auf allen inaktiven Pfaden zum assoziierten Endpunkt. Pfade sind *inaktiv*, wenn während einer bestimmten Zeit, z.B. innerhalb der letzten Heartbeat-Periode, keine Chunks gesendet wurden, die zur neuerlichen Bestimmung der RTT genutzt werden können.
Die HEARTBEAT-Chunks enthalten einen von der jeweiligen Implementierung frei definierbaren Parameter (vgl. Abbildung 2.7). Bei Erhalt eines HEARTBEAT-Chunks muss dieser Parameter vom Endpunkt in einem HEARTBEAT-ACK-Chunk an die Quelladresse des HEARTBEAT-Chunks zurückgesendet werden. Jeder Endpunkt führt einen Zähler, dessen Wert jeweils um eins erhöht wird, wenn ein HEARTBEAT-Chunk nicht innerhalb einer bestimmten Zeit bestätigt wurde. Wenn dieser Fehlerzähler einen Schwellwert *Path.Max.Retrans* ([86] empfiehlt 5) überschreitet, so wird der Pfadzustand auf *unerreichbar* gesetzt. Der Endpunkt wird weiterhin versuchen, HEARTBEAT-Chunks auf diesem Pfad zu senden, bis entweder eine gültige Bestätigung empfangen wird und der Pfadzustand wieder auf *erreichbar* gesetzt werden kann, oder die Assoziation beendet ist.
Da Quittungen (SACK-Chunks) für erhaltene Daten-Chunks an die Adresse zurückgesendet werden sollen, von der die Daten kamen, benötigen Pfade, die aktiv für die Datenübertragung genutzt werden, keine Kontrolle der Erreichbarkeit durch den Heartbeat-Mechanismus. Bei einer timer-basierten Neuübertragung wird der Fehlerzähler für den entsprechenden Pfad ebenfalls erhöht, was für die Fehlererkennung auf diesem Pfad genügt.
Wird ein HEARTBEAT-ACK-Chunk als Bestätigung eines HEARTBEAT-Chunks, oder ein SACK-Chunk als Quittung eines Daten-Chunks empfangen, so wird für den betreffenden Pfad der entsprechende Fehlerzähler zurückgesetzt.
Pfade, die als *unerreichbar* gekennzeichnet sind, werden nicht für die Datenübertragung benutzt, und zusätzlich wird dem SCTP-Nutzer die Änderung von Pfadzuständen angezeigt. Insbesondere kann der SCTP-Nutzer einen neuen Primärpfad auswählen, wenn der alte Primärpfad unerreichbar wird. Der Datentransfer

wird trotz möglichem Pfadausfall über sekundäre Pfade fortgeführt, falls solche vorhanden sind. Siehe auch Abschnitt 4.4 für weitere Informationen zum Thema “Pfadausfall”.

2.3.7.1 Überwachung der Pfadeigenschaften

Für jeden Pfad einer SCTP-Assoziation werden bei Eintreffen neuer SACK- oder HEARTBEAT-ACK Chunks kontinuierlich geglättete Schätzwerte für die Rundlaufzeit (engl. Smoothed RTT, *SRTT*) und deren Schwankung (engl. RTT Variation, *RTTVAR*) bestimmt. Aus diesen Schätzwerten wird gemäß der Gleichung

$$\mathrm{RTO} = \mathrm{SRTT} + 4 \cdot \mathrm{RTTVAR} \tag{2.1}$$

der neue Wert für den Neuübertragungs-Timer Timeout (engl. Retransmission Timer Timeout Value, RTO) des betreffenden Pfades bestimmt. Ist der neu berechnete Wert kleiner als der Protokoll-Parameter RTO_{min}, so wird RTO_{min} angenommen. Ist er größer als der Protokoll-Parameter RTO_{max}, so wird RTO_{max} angenommen. [86] schlägt $\mathrm{RTO}_{min} = 1$ s und $\mathrm{RTO}_{max} = 60$ s vor. Nach einer timer-basierten Neuübertragung (vgl. Abschnitt 2.3.5.1) wird der aktuelle RTO-Wert für den betreffenden Pfad verdoppelt, oder, falls der neue Wert größer als RTO_{max} wäre, auf diesen Wert gesetzt. Dieser Mechanismus (engl. *Exponential Backoff*) dient dazu, im Überlast-Fall eine zusätzliche Netzbelastung, die durch sukzessive Neuübertragungen entstehen würde, zu vermeiden.

2.3.7.2 Überwachung des assoziierten Endpunktes

Jeder Endpunkt überwacht auch mit einem Zähler die Anzahl aufeinander folgender Neuübertragungen von Daten- oder HEARTBEAT-Chunks zum assoziierten Endpunkt ohne erfolgreiche Quittung. Dieser Zähler wird für *jede* erfolgreich empfangene Quittung zurückgesetzt (also SACK- oder HEARTBEAT-ACK-Chunk). Wenn der Fehlerzähler den Schwellwert *Assoc.Max.Retrans* (in [86] wird ein Wert von 10 empfohlen) überschreitet, so ist der assoziierte Endpunkt nicht mehr erreichbar, und die Assoziation wird geschlossen.

2.3.8 Fluss- und Überlastkontrolle

Analog zur Fluss- und Überlastkontrolle beim TCP [3] wird die Übertragung von Daten-Chunks durch das SCTP durch zwei Regelmechanismen gesteuert:

1. Die fensterbasierte Flusssteuerung und

2. die Überlastkontrolle.

Die Flusssteuerung benutzt zum einen die beim Verbindungsaufbau zwischen den Endpunkten ausgetauschten Maximalwerte für die Größen der Sendefenster und bestimmt zum anderen die maximale Datenmenge, die gesendet werden darf, dynamisch nach dem Empfang neuer SACK-Chunks. Diese enthalten eine aktuelle Angabe des Empfängerfensters und damit des freien Puffers des Empfängers (engl. Advertised Receiver Window, $arwnd$).
Nachdem ein Endpunkt alle durch ein SACK bestätigten Daten-Chunks markiert hat, berechnet er die Länge aller noch ausstehenden (d.h. gesendeten, aber noch unbestätigten) Daten-Chunks. Diese wird von dem im SACK empfangenen $arwnd$ subtrahiert und ergibt die Datenmenge, die augenblicklich gesendet werden darf. Erhält der Sender ein $arwnd$ von Null, so darf er normalerweise keine weiteren Daten mehr senden. Da SACK-Chunks bei der Übertragung verloren gehen können, ist es ihm zu jeder Zeit erlaubt, durch Aussenden eines neuen Daten-Chunks einen neuen SACK-Chunk anzufordern, der eine Änderung des $arwnd$ anzeigen kann. Gilt allerdings $arwnd = 0$, so darf er jeweils nur maximal einen Daten-Chunk ausstehend haben.
Für die Überlastkontrolle (engl. Congestion Control) wird ein Parametersatz pro Pfad zum assoziierten Endpunkt benötigt. Dieser umfasst zwei Werte: das Überlast-Sendefenster (engl. Congestion Window, nachfolgend $cwnd$ genannt) und die Überlast-Schwelle (engl. Slow Start Threshold, $ssthresh$). Die aktuellen Werte dieser Parameter entscheiden über den Sendezustand des betreffenden Pfades. Abschnitt 7.2 von [86] definiert die folgenden zwei Zustände:

1. Langsamer Anstieg (engl. Slow Start), für $cwnd \leq ssthresh$

2. Überlast-Vermeidung (engl. Congestion Avoidance), für $cwnd > ssthresh$.

Abbildung 2.11 stellt die Entwicklung des $cwnd$ über der Zeit, sowie die entsprechenden Zustände Slow Start und Congestion Avoidance anschaulich dar. Die Datenübertragung erfordert zu Beginn ein langsames Erproben der Bedingungen des Netzwerks, damit dieses nicht überlastet wird. Daher werden die Parameter wie folgt initialisiert:

$$ssthresh \Leftarrow arwnd \quad \text{und} \tag{2.2}$$

$$cwnd \Leftarrow min(4 \cdot MTU, max(2 \cdot MTU, 4380Bytes)) \tag{2.3}$$

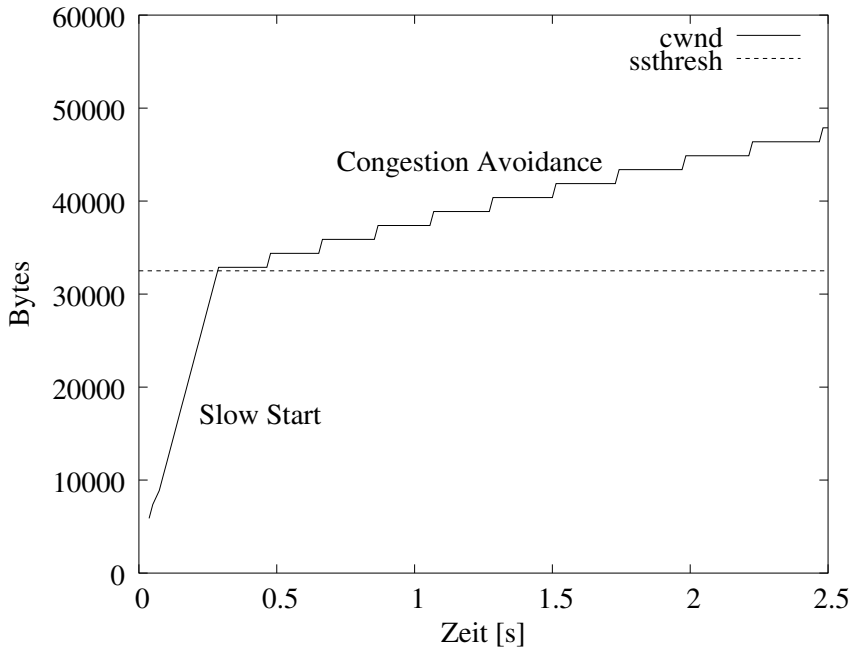

Abbildung 2.11: Das Wachstum des Congestion Window: Slow Start und Congestion Avoidance

Das Sendefenster wird im Zustand “Slow Start” relativ schnell erhöht, denn für jedes neue SACK wird das $cwnd$ um die MTU, oder falls dies weniger ist, um die Anzahl der bestätigten Bytes, erhöht. Im Idealfall (keine Verluste) wächst dadurch das $cwnd$ exponentiell mehrmals pro RTT (vgl. auch Wachstum des $cwnd$ in Abbildung 2.11 für $t \leq 0,25$ s). Wird anfangs $ssthresh \Leftarrow arwnd$ gesetzt, so bleibt der Sender im “Slow Start”, bis das gesamte Sendefenster ausgenutzt ist und erreicht nie den Zustand der Überlast-Vermeidung. In Abbildung 2.11 gilt allerdings zum dargestellten Zeitpunkt $ssthresh = \frac{1}{2} \cdot arwnd$.
Im Zustand der “Congestion Avoidance” erfolgt das weitere Wachstum des $cwnd$ langsamer, maximal einmal pro gemessener RTT. Wird ein Pfad längere Zeit (eine RTO) nicht genutzt, so werden seine Parameter zurückgesetzt, und er beginnt gegebenenfalls erneut im “Slow Start”.
Ein Wachstum des $cwnd$ darf in jedem Fall nur erfolgen, wenn der neue SACK-Chunk das CTSNA avanciert, und das Wachstum muss jeweils dem $cwnd$ des-

jenigen Pfades gutgeschrieben werden, über den die entsprechenden bestätigten Daten gesendet wurden.
Werden während der Übertragung Datenverluste festgestellt, so wird die Größe des Sendefensters halbiert und damit die Senderate drastisch reduziert (AIMD, Additive Increase, Multiplicative Decrease)[1]. Dies hängt damit zusammen, dass Datenverluste einer Netzüberlast-Situation zugeschrieben werden, in der Router beginnen, Pakete zu verwerfen. Im Einzelnen werden bei einer schnellen Neuübertragung die folgenden Zuweisungen vorgenommen:

$$ssthresh \Leftarrow max(\frac{cwnd}{2}, 2 * MTU) \text{ und } cwnd \Leftarrow ssthresh \tag{2.4}$$

Bei einer timer-basierten Neuübertragung hingegen wird wieder ganz langsam begonnen:

$$ssthresh \Leftarrow max(\frac{cwnd}{2}, 2 * MTU) \text{ und } cwnd \Leftarrow MTU \tag{2.5}$$

2.3.9 Protokoll-Sicherheit

SCTP wurde entworfen mit den Erfahrungen, die durch den jahrelangen Einsatz des TCP gewonnen wurden. Daher spezifiziert der Standard [86] Mechanismen, die bestimmte Angriffe auf Protokollendpunkte bzw. bestehende Assoziationen erschweren. Ein Angreifer, dem es möglich ist, die Mehrzahl aller Pakete zwischen zwei Endpunkten zu überwachen, kann als sog. “Man in the Middle” Daten in eine bestehende Assoziation einschleusen. In dieser Hinsicht ist SCTP (ebenso wie TCP) anfällig für Angriffe.
Erheblich schwieriger ist es jedoch für Angreifer, *blind* Daten in eine Assoziation einzuschleusen, d.h. ohne dass er die Mehrzahl der übertragenen Daten abhören kann. Dies wird erreicht durch den Einsatz der zufällig gewählten 32 Bit langen Prüfmarken (Tags), die beim Verbindungsaufbau ausgetauscht werden. Dadurch wird die Chance, dass ein von einem Angreifer erzeugtes Paket von einem Endpunkt akzeptiert wird, um den Faktor $1 : 2^{32}$ reduziert. Zusätzlich müsste ein solches Paket natürlich noch eine gültige Kombination von Quell- und Zieladressen sowie Quell- und Zielports aufweisen.
Ein weiterer Sicherheitsmechanismus, der Einsatz des Cookies (siehe auch Abschnitt 2.3.3), stellt sicher, dass ein Server erst dann Ressourcen alloziert, wenn

[1] Beschreibt die Überlast-Kontrolle bei Protokollen wie TCP und SCTP. Dabei wird das Congestion Window (*cwnd*) schrittweise maximal um eine MTU erhöht und bei Feststellen von Verlusten (durch Überlast im Netz) halbiert.

der Client das vom Server erzeugte Cookie an diesen zurückgesendet hat. Dadurch wird verifiziert, dass der Absender des INIT-Chunks wirklich eine Verbindung initiieren wollte. Unter der Voraussetzung, dass der Server einen wirklich zufälligen Schlüssel für die Erzeugung der Signatur (HMAC, siehe [53]) verwendet, macht dieser Mechanismus die unerwünschte Allokation von Ressourcen nahezu unmöglich.

SCTP erhöht durch die Unterstützung von Multihoming potentiell die Verfügbarkeit der Netzwerkverbindung zwischen zwei Endpunkten, unterstützt aber keine inhärenten Sicherheitsprotokolle, um die Sicherheit (Vertraulichkeit, Authentizität usw.) der übertragenen Daten zu gewährleisten. Für diesen Zweck ist der SCTP-Benutzer gegenwärtig entweder auf die Funktionen von IPSec [51], oder der Transport Layer Security (TLS) [19] angewiesen. Erstere Lösung sichert, wie in [79] beschrieben, die Protokolldaten des SCTP in der Netzwerkschicht.

TLS hingegen setzt eine verbindungsorientierte, reihenfolgegesicherte, bidirektionale Kommunikation voraus. Die sich daraus ergebenden Anforderungen an Implementierungen des SCTP wurden von uns in dem Dokument “Transport Layer Security over Stream Control Transmission Protocol” [48] als Standards-Track RFC in die IETF-Standardisierung eingebracht. Darin werden u.a. bidirektionale Kanäle definiert, die je einen ein- und ausgehenden Stream pro TLS-Verbindung nutzen. Eine SCTP-Assoziation kann dann im Multiplex mehrere TLS-Verbindungen transportieren.

2.4 Erweiterungen des SCTP

Bei der Standardisierung des SCTP wurde Wert darauf gelegt, ein Protokoll zu definieren, welches leicht erweiterbar ist. So definiert [86] 15 Chunk-Typen und sieht dafür ein 8 Bit-Feld vor. Die verbleibenden Werte sind für Erweiterungen gedacht, von denen einige in der TSVWG der IETF bereits diskutiert werden. Zwei dieser Erweiterungen werden nachfolgend vorgestellt.

Die Erweiterungen profitieren von einem SCTP-Mechanismus, der einfach auf- und abwärtskompatible Protokollerweiterungen ermöglicht. Die jeweils ersten zwei Bits neu definierter Chunk- oder Parametertypen entscheiden darüber, wie eine SCTP-Implementierung bei Verarbeitung dieser Chunks oder Parameter vorgeht, wenn ihr diese unbekannt sind:

Bits	Aktion
00	Beende die Verarbeitung des SCTP-Pakets/Chunks; verwerfe es/ihn.
01	Beende die Verarbeitung des SCTP-Pakets/Chunks, verwerfe es/ihn, und sende einen Fehlerbericht zurück (Fehler: nicht erkannter Chunk/Parameter).
10	Überspringe den Chunk/Parameter und setze die Verarbeitung beim nächsten Chunk/Parameter fort.
11	Überspringe den Chunk/Parameter und setze die Verarbeitung beim nächsten Chunk/Parameter fort und sende einen Fehlerbericht zurück (Fehler: nicht erkannter Chunk/Parameter).

Dieser Mechanismus erlaubt es einer Implementierung, bereits beim Verbindungsaufbau festzustellen, ob die Gegenseite einen bestimmten Parameter unterstützt. Des Weiteren kann eine Implementierung zu jeder Zeit einen neu definierten Chunk an seinen assoziierten Endpunkt senden. Dieser wird entsprechend reagieren, woraus sich entweder ergibt, dass dieser Chunk benutzt werden kann, oder von der Gegenseite nicht unterstützt wird.

2.4.1 Der teilgesicherte Transportmodus (PR-SCTP)

Die in RFC 3758 [82] spezifizierte Erweiterung des SCTP um einen nur teilweise gesicherten Übertragungsmodus (engl. Partially Reliable SCTP, PR-SCTP) erlaubt es einer Implementierung, statt einer Neuübertragung eines oder mehrerer möglicherweise verloren gegangener Daten-Chunks einen neuen Chunk-Typen zu senden, der der Gegenseite anzeigt, dass sie ihren CTSNA-Zähler erhöhen kann. Dieser Chunk-Typ wird engl. Forward-TSN-Chunk bezeichnet (“erhöhe die TSN”), und er enthält u.a. die TSN, auf die der Empfänger seinen CTSNA-Wert setzen soll. Beim Verbindungsaufbau werden Parameter ausgetauscht, die der Gegenseite anzeigen, dass die PR-SCTP Protokollerweiterung unterstützt wird. Nur wenn beide Seiten diese Erweiterung unterstützen, dürfen Forward-TSN-Chunks innerhalb einer Assoziation gesendet werden.
Der Erhalt eines Forward-TSN-Chunks ist für den Empfänger äquivalent mit dem Erhalt aller bisher noch nicht vom assoziierten Endpunkt erhaltenen Daten-Chunks mit einer TSN kleiner oder gleich der darin enthaltenen TSN. Abbildung 2.12 zeigt, dass neben dem neuen CTSNA für den Empfänger noch eine Reihe von Streams und zugehörigen Stream-Sequenznummern übertragen werden. Diese ermöglichen es, für jeden Stream festzustellen, welche Nachrichten dort nach Erhalt des Forward-TSN-Chunks sofort ausgeliefert werden können,

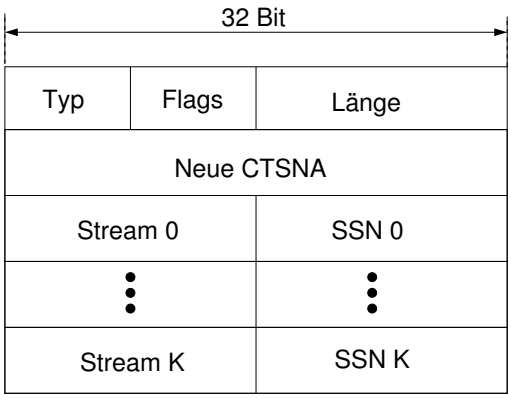

Abbildung 2.12: Aufbau eines Forward-TSN Chunks

bzw. welche Nachrichten-Fragmente nicht mehr zu kompletten Nachrichten zusammengesetzt werden und daher verworfen werden können.
Die PR-SCTP-Erweiterung ermöglicht es einem Endpunkt, einen Dienst anzubieten, welcher übertragene Daten mit einer Lebensdauer versieht, nach deren Ablauf in keinem Fall eine Neuübertragung der entsprechenden Nachricht vorgenommen wird. Dem Sender steht es dabei frei, bei jeder Nachricht individuell zu bestimmen, mit welcher Qualität (geordnet oder ungeordnet, zuverlässig oder unzuverlässig) und Lebensdauer sie gesendet wird.

2.4.2 Dynamische Adressänderung (Add-IP)

Die Möglichkeit, dynamisch Adressen einer bestehenden Assoziation zu ändern, eröffnet für die Anwendbarkeit des SCTP neue Perspektiven. Erreicht wird dies durch zwei neue Chunks, die eine Änderung der Adresskonfiguration anzeigen (ASCONF-Chunk) und den Erhalt einer solchen Anzeige bestätigen (ASCONF-Ack-Chunk) [83]. Mögliche Konfigurationsänderungen sind im Wesentlichen:

- das Hinzufügen einer neuen Adresse,
- das Löschen einer alten Adresse und
- das Setzen eines neuen Primärpfads auf dem assoziierten Endpunkt.

Die Add-IP-Erweiterung verwendet Sequenznummern, um die ASCONF-Chunks zu nummerieren. Regel ist, dass genau ein ASCONF-Chunk gesendet werden darf und dieser erst bestätigt werden muss, bevor ein weiterer ASCONF-Chunk gesendet werden darf. Der Chunk wird durch einen Timer gegen Verlust abgesichert. Die Sequenznummer wird entsprechend für jede neue ASCONF-Nachricht um eins erhöht. Die jeweilige Konfigurationsänderung wird als Parameter variabler Länge mit dem ASCONF-Chunk übertragen, wobei dieser mehrere Parameter gleichzeitig enthalten kann (z.B. eine neue Adresse und das Setzen des neuen Primärpfades auf diese Adresse). Parameter werden in der Reihenfolge abgearbeitet, in der sie im Chunk auftreten. Dabei darf ein Endpunkt nie seine letzte Adresse löschen.
Jeder ASCONF-Chunk enthält zusätzlich eine Adresse, die der assoziierte Endpunkt vor Empfang des ASCONF-Chunks einer Assozation zuordnen kann. In [83] wird der normale Algorithmus, um Pakete zu erkennen, die nicht zu einer etablierten oder neuen Assoziation gehören, modifiziert: zunächst wird versucht, das SCTP-Paket anhand seiner Quell- und Zieladressen und Quell- und Zielports zu einer bestehenden Assoziation zuzuordnen. Falls das nicht gelingt, wird versucht, die Kombination aus der im ASCONF-Chunk enthaltenen (Absender-)Adresse mit den Ports des SCTP-Headers zu einer Assozation zuzuordnen. Falls auch dies misslingt, so wird das Paket verworfen und ein Abort-Chunk an die Gegenseite gesendet. Andernfalls wird die Prüfmarke (Tag) überprüft, und wenn diese fehlerfrei ist, werden die im Paket enthaltenen Chunks (und insbesondere der ASCONF-Chunk) weiterverarbeitet.

2.5 Eignung für zusätzliche Anwendungen

Mit den beiden in den Abschnitten 2.4.1 und 2.4.2 beschriebenen Erweiterungen ergeben sich, neben den bisher schon erwähnten Applikationen, neue Anwendungsgebiete für das Vielzweck-Transportprotokoll SCTP. Diese Gebiete werden zurzeit intensiv in der Forschung diskutiert.

2.5.1 Transport von Multimedia-Strömen

Die Erweiterung von SCTP um die Möglichkeit, Nachrichten mit einer variablen Lebenszeit zu versehen, ist eine ideale Voraussetzung für ein echtzeitfähiges Transportprotokoll für Multimedia-Daten. Hier kann die Anzahl von Neuübertragungen so limitiert werden, dass Daten, die veraltet sind, nicht mehr

übertragen werden müssen und so die Bandbreite einer Verbindung für aktuellere Daten zur Verfügung steht. Die Tatsache, dass SCTP effektive Neuübertragungen durchführt (selektive Quittierung), sorgt dafür, dass SCTP zur Übertragung von Multimedia-Datenströmen, die beim Empfänger zunächst gepuffert werden, geeignet ist. Informationseinheiten (z.B. MPEG-4 Frames, vgl. [5]) können in verschiedenen Streams mit unterschiedlichen Zuverlässigkeitsstufen (zuverlässig, maximal eine Neuübertragung, keine Neuübertragung, also unzuverlässig, o.ä.) übertragen werden. Untersuchungen, bei denen MPEG-4 Daten über das PR-SCTP übertragen wurden [5, 57], zeigen, dass Multimedia-Anwendungen nicht nur von den Eigenschaften des SCTP wie der gebündelten Übertragung von kurzen Kontroll- und Anwendungs-Nachrichten in SCTP-Paketen, oder der flexiblen Zustellung von Daten profitieren, sondern auch von der Variabilität der Zuverlässigkeit, die fein granular pro Nachrichteneinheit bestimmt werden kann.

2.5.2 Mobilitätsunterstützung

Ein Charakteristikum IP-basierter Mobilitätslösungen (Mobile IP [63], Mobile IPv6 [44]) ist die Tatsache, dass ein sich durch ein Netz bewegendes Endgerät wechselnde IP-Adressen erhält. Unter diesen Bedingungen versuchen die oben erwähnten Lösungen, Pakete zumindest temporär über einen fixen Endpunkt, den sog. Home-Agent, umzuleiten, bei dem sich das mobile Endgerät nach jedem Adresswechsel registrieren muss.
Die in Abschnitt 2.4.2 beschriebene Add-IP Erweiterung des SCTP eröffnet eine neue Möglichkeit, mit wechselnden Adressen umzugehen. Geht man von einem feststehenden Server mit unveränderlichen Adressen und einem mobilen Endgerät mit sich potentiell ändernden Adressen aus, so bietet die oben beschriebene Protokollerweiterung eine einfache Möglichkeit, persistente Ende-zu-Ende Verbindungen auf der Transportschicht zu realisieren. Diese wird als “Mobile SCTP” in [69] ausführlich erläutert. Im Falle eines simultanen Wechsels von Adressen zweier SCTP-Endpunkte wird bei ausschließlicher Nutzung von Mobile-SCTP jedoch eine bestehende Assoziation unterbrochen.
In [52] wird eine weiterführende Lösung erörtert, die als Kombination von Mobile-IP und Mobile-SCTP eine Mobilitätslösung bietet, die auch in einem solchen Fall die Assoziation aufrecht erhält. Dabei bleibt beim simultanen Handoff die Erreichbarkeit jedes Endpunktes über die jeweilige Adresse seines Home-Agents gewährleistet. Eine weitere Lösung für dieses Problem wurde in [20] ausführlich vorgestellt. Dabei nutzt jeder Endpunkt die Session-Layer-Funktionalität des Reliable Server Pooling, um im Falle des simultanen Han-

doffs zweier Mobile-SCTP Endpunkte die neue Adresse per Namensanfrage in Erfahrung zu bringen und eine abgebrochene Assoziation neu zu etablieren. Das Reliable-Server Pooling sorgt in diesem Fall für eine nahtlose Fortsetzung der Sitzung.

2.6 Weiterführende Informationen

Für weiterführende Informationen sei besonders das Buch von Randall Stewart und Qiaobing Xie [85] empfohlen, in dem die Autoren ausführliche Erläuterungen zu den Eigenschaften und Mechanismen des SCTP geben. Insbesondere ist hier auch interessant, dass die Diskussionen, die in den IETF Arbeitsgruppen um die Protokolleigenschaften geführt wurden und somit die Hintergründe für manche Designentscheidungen näher erläutert werden.
Ein weiterer guter Ausgangspunkt ist die Webseite “SCTP for Beginners” [45], die der Autor erstellt hat, um Lesern schnell einen strukturierten Überblick über die Eigenschaften des SCTP zu geben. Hier findet man auch aktuelle Verweise auf verschiedene SCTP-Implementierungen, Standards, Veröffentlichungen und nützliche Software. Die Zugriffsstatistik zeigt, dass dieses Angebot wahrgenommen und seit einigen Jahren mehr als 1000 Mal im Monat genutzt wird.

Kapitel 3

Überblick über eine Protokollimplementierung

Ein beträchtlicher Teil der Forschungsarbeiten zum Thema SCTP wurde mit Hilfe einer am Lehrstuhl “Technik der Rechnernetze” erstellten Protokollimplementierung angefertigt, die zunächst in Kooperation mit der Fa. Siemens AG (ICN), München, entwickelt wurde.
Im folgenden Kapitel werden die während der Abwicklung dieses Projekts durchlaufenen Entwicklungsphasen und Ergebnisse dokumentiert. Zu Projektbeginn wurde eine formale Grobspezifikation erstellt, die nicht nur eine Projektdokumentation darstellen sollte, sondern auch als Grundlage einer funktionalen Gliederung des SCTP und der nachfolgenden Implementierung diente.
Darüber hinaus werden zwei Möglichkeiten für standard-konforme Schnittstellen näher erläutert, bevor schließlich eine Auswahl der wichtigsten zurzeit verfügbaren SCTP-Implementierungen vorgestellt und erörtert werden.

3.1 SDL-basierter Systementwurf

Bei der Entwicklung der Protokollimplementierung wurde im Wesentlichen nach einem einfachen und dem Umfang des Softwareprojekts angemessenen Entwicklungsmodell vorgegangen, dem bekannten Wasserfall-Modell [8]. Dabei werden mehrere Entwicklungsphasen in einem Softwareprojekt beschrieben, die von einer groben Aufgabenstellung über eine detaillierte Anforderungsanalyse zu einer formalen Spezifikation reichen, wobei letztere als Basis für eine Implementierung dient. Ist die Implementierung erfolgt, so folgen als abschließende Prozessstufen

der Test, die Fehlerbeseitigung und Wartung.
In der ersten Projektstufe wurde als Ziel die Entwicklung einer prototypischen, d.h. einer nicht unbedingt auf hohe Leistung hin optimierten, SCTP-Implementierung festgelegt. Als erster Schritt sollte dazu – ganz gemäß der oben erwähnten Methodik – eine Anforderungsanalyse für den Einsatz der Software, sowie eine formalisierte Beschreibung der Implementierungsstruktur durchgeführt werden, wie im folgenden Abschnitt erläutert.

3.1.1 Entwicklungsmethodik

Bei der Spezifikation wurde eine Top-Down Methodik gewählt. Das bedeutet, es wurde zunächst die grobe Funktionalität der zu erstellenden Software in den System-Kontext eingeordnet, in dem sie zu funktionieren hat. Dabei wurde die Spezifikationssprache SDL verwendet. SDL ist eine von der CCITT (ITU) in [37] standardisierte Spezifikationssprache und Entwurfsmethode. Sie ist als formalisierte Sprache besonders geeignet, Kommunikationsprotokolle und Datenverarbeitungssysteme als sog. erweiterte endliche Automaten zu beschreiben. Dabei existieren drei verschiedene Beschreibungsmöglichkeiten:

1. Grafisch. Dabei werden Prozesse als Zustandsablaufdiagramme mit Ein- und Ausgaben, Entscheidungen, sowie Prozeduren beschrieben. Die Beschreibung höherer Hierarchiestufen eines Systems wird durch die Komposition miteinander kommunizierender Prozesse gebildet. Bei dieser Variante werden i.A. nicht die zeitlichen Eigenschaften eines Systems beschrieben, sondern nur das Ein-/Ausgabeverhalten.

2. Textuell. Zu einer grafischen Darstellung äquivalente, aber maschinenlesbare Form. Erlaubt eine vollständige Verhaltensspezifikation.

3. Nachrichtenflüsse. Sie erlauben, das Verhalten auch in zeitlich korrektem Kontext darzustellen, beschreiben i.A. aber nicht sämtliche Verhaltensmöglichkeiten eines Prozesses.

Bei der Protokollspezifikation kommt die block- und modulorientierte Hierarchie des SDL voll zum Tragen. Die Funktionalitäten der einzelnen Teilsysteme und Blöcke können in funktionale Module aufgeteilt, und deren Interaktion mit Hilfe von Ein-/Ausgaben durch sog. Signale verdeutlicht werden.
Bei der Einteilung in Blöcke gilt der Grundsatz der funktionalen und informativen Kohäsion: es werden Blöcke/Module durch Teilprozesse gebildet, die funktional

zusammen gehören, oder die intern auf die gleichen Daten zugreifen. Die Kommunikation zwischen den Blöcken erfolgt so, dass eine lose Kopplung entsteht, bei der interne Zustände und Daten nicht geändert oder abgefragt werden müssen. Ferner gilt der Grundsatz der Modularisierung, bei der jedes Modul ausgetauscht werden kann, ohne dass andere Module davon beeinflusst werden.

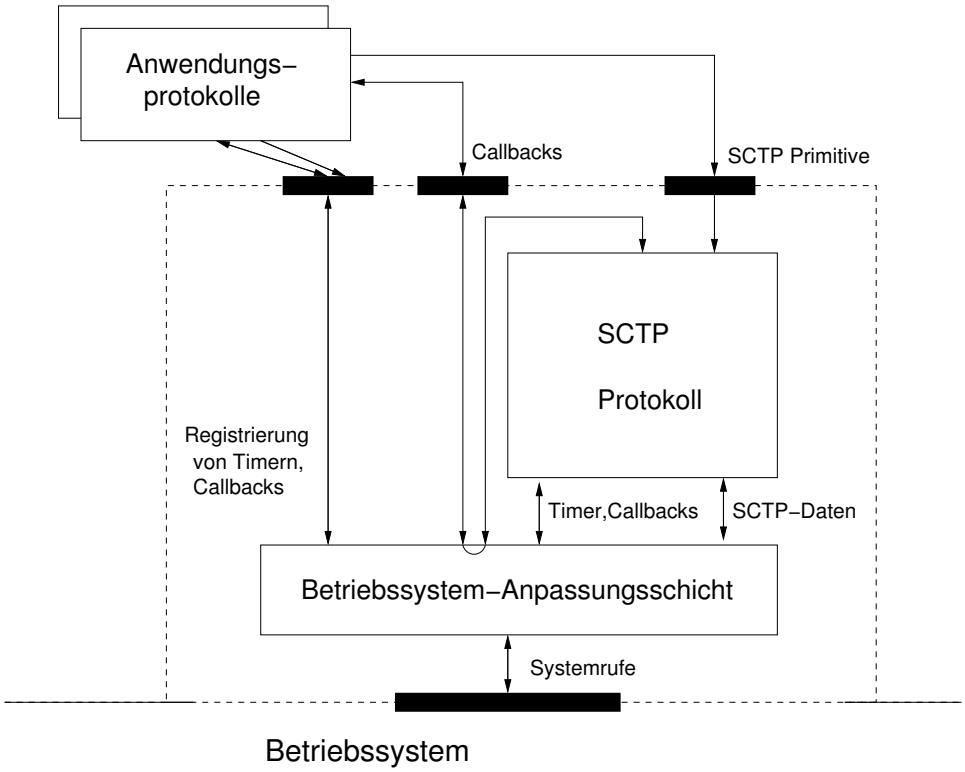

Abbildung 3.1: Systemstruktur der SCTP-Implementierung

3.1.2 Spezifikation auf Systemebene

Abbildung 3.1 zeigt die grundlegende Struktur, die das Ergebnis der Top-Down Spezifikation eines SCTP-Protokollsystems darstellt. Dieses System enthält neben der Anwendung die Implementierung des Transportprotokolls und eine Anpassungsschicht an das jeweilige Betriebssystem. Hinter dieser Architektur steht die

Idee der Kapselung betriebssystemnaher Funktionen wie Ein-/Ausgabe über Netzwerkprotokolle und Timerfunktionen in einer Schicht mit festgelegter Schnittstelle und einer weitgehend betriebssystemunabhängigen Protokollimplementierung, die eine standardisierte Schnittstelle für unterschiedliche Anwendungen bietet.
Die Anwendung registriert sog. Callbacks, d.h. Funktionen, die die Protokollimplementierung zur asynchronen Kommunikation mit der Anwendung nutzen kann und aufruft. Ferner stellt die Implementierung eine Schnittstelle mit Dienstprimitiven zur Verfügung, über die die Anwendung z.B. Assoziationen aufbauen und beenden sowie Daten senden kann. Die Anwendung nutzt die Dienste der Anpassungsschicht, um z.B. Timer zu implementieren, Daten über die Standardeingabe zu lesen oder über UDP zu kommunizieren. Die eigentliche SCTP-Implementierung reicht SCTP-Pakete über die Anpassungsschicht an die Netzwerkschicht weiter und benutzt ebenfalls die Timerfunktionen, um einen korrekten Protokollablauf zu gewährleisten.

3.1.3 Funktionale Gliederung des Protokollmoduls

Der nächste Schritt bei der Top-Down Spezifikation ist die Untergliederung der Protokollfunktionalität in funktionale Module. Die Abbildung 2 aus [86] diente hierbei lediglich als Grundlage. SDL stellt jedoch auch die Art und Richtung der Kommunikation zwischen Prozessen und Modulen dar, deshalb wurde die im Ablauf des Projekts erstellte Grobspezifikation detaillierter als Abbildung 2 aus [86] und zeigt insbesondere die Trennung von Daten- und Kontrollpfad innerhalb des Protokollmoduls auf. Abbildung 3.2 zeigt die entstandene Struktur und den Daten- und Kontrollfluss exemplarisch, sowie deren Verknüpfung mit den SDL-Prozessen, die das eigentliche Protokollverhalten bestimmen.
Eine zentrale Funktion kommt dem SCTP-Controller zu: Er beinhaltet den endlichen Automaten, der den Zustand des Protokolls beschreibt, d.h. den Übergang vom Zustand *Closed* zum Zustand *Established* beim Verbindungsaufbau (siehe auch Abb. 2.9), bzw. vom Zustand *Established* zum Zustand *Closed* beim Verbindungsabbau (siehe Abb. 2.10). Dabei erzeugt der SCTP-Controller die entsprechenden Kontroll-Chunks mit allen Parametern und sendet sie über den Paketierungs- und den Validierungsprozess an die Anpassungsschicht, die den weiteren Transport über das Netzwerk übernimmt.
Der Paketierungs-Prozess hat zwei Funktionen: zum einen fasst er ausgehende Chunks zu SCTP-Paketen zusammen, fügt den entsprechenden Common Header hinzu und reicht das Paket an den Validierungsprozess weiter. Zum anderen akzep-

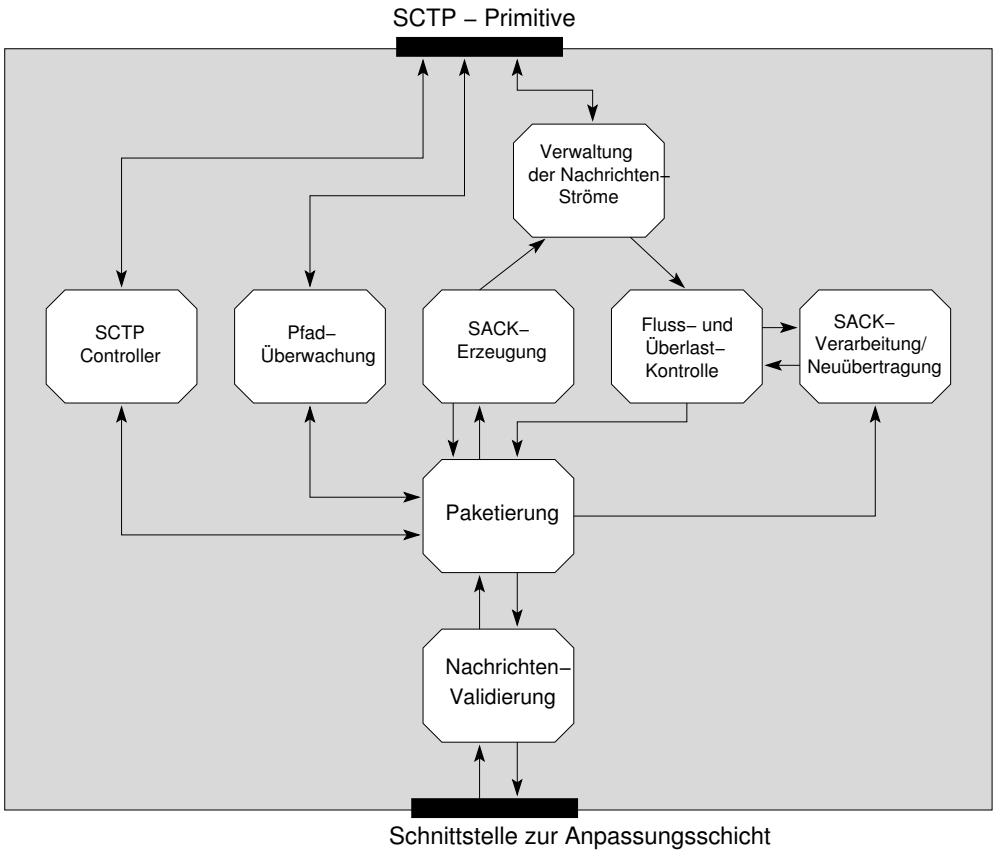

Abbildung 3.2: Funktionale Struktur des SCTP-Protokollmoduls

tiert er von diesem SCTP-Pakete und zerlegt sie in einzelne Chunks, die er an empfangende Prozesse weiterleitet. Der Validierungsprozess berechnet die CRC32C Prüfsumme und schreibt sie in den Common Header ausgehender SCTP-Pakete, bzw. überprüft diese Prüfsumme in ankommenden Paketen. Stimmt die berechnete Prüfsumme eines ankommenden Pakets nicht mit der im Paketkopf enthaltenen Prüfsumme überein, so wird das Paket verworfen. Der Pfadüberwachungs-Prozess sendet – timergesteuert – regelmäßig HEARTBEAT-Chunks zu allen ansonsten unbenutzten Pfaden. Er verarbeitet auch ankommende HEARTBEAT- und HEARTBEAT-ACK-Chunks und überwacht mit Fehlerzählern den Status aller Pfade, sowie der gesamten Assoziation.
Die restlichen Prozesse verarbeiten Daten- und SACK-Chunks. Sendet der Nutzer

Nachrichten, so werden aus diesen in dem Prozess, der die Verwaltung von Nachrichtenströmen übernimmt, Daten-Chunks erzeugt. Dies kann z.B. auch eine Segmentierung langer Nachrichten beinhalten. Die Daten-Chunks werden an den Fluss- und Überlastkontroll-Prozess (vgl. Abbildung 3.2) weitergeleitet. Dieser verwaltet neben einer Warteschlange die notwendigen Parameter, um zu entscheiden, wann ein Daten-Chunk auf einem bestimmten Pfad gesendet werden darf. Erstmalig übertragene Chunks werden im angrenzenden Prozess solange gespeichert, bis ihr Empfang durch ankommende SACK-Chunks bestätigt wird und sie gelöscht werden können. Gegebenenfalls werden die Chunks auch neu ausgesandt, wenn sie nicht rechtzeitig bestätigt werden, oder eine schnelle Neuübertragung angezeigt ist (vgl. Abschnitt 2.3.5.1).
Ankommende Daten-Chunks werden an den SACK-Erzeugungs-Prozess weitergeleitet, der überprüft, ob diese Daten neu sind und gegebenenfalls zur Bestätigung SACK-Chunks sendet. Neue Daten-Chunks werden an den Prozess der Nachrichtenstrom-Verwaltung weitergeleitet, der die gegebenenfalls segmentierten Daten wieder zu Nachrichten zusammensetzt und sie in der richtigen Reihenfolge an den SCTP-Nutzer ausliefert.
Die Funktion einzelner zentraler Prozesse wurde detaillierter in SDL-GR spezifiziert, wobei aber viele Abläufe nur als Prozeduren angeführt wurden, deren Funktionen nicht weiter ausgearbeitet wurden. Die somit formalisiert beschriebenen Abläufe dienten als direkte Grundlage für die nachfolgend von Hand erzeugte Protokollimplementierung.
Bei dieser Vorgehensweise ist es wichtig festzustellen, dass die formalisierte Beschreibung nicht die vollständige Interaktion aller beteiligten Prozesse aufgeführt hat. So fehlen beispielsweise in Abbildung 3.2 die Erzeugung von Prozessen beim Systemstart, die Aufrufe von Timern in der Anpassungsschicht und die Kommunikation zwischen Pfadüberwachungs- und Flusskontroll-Prozess sowie zwischen Pfadüberwachungs- und SACK-Verarbeitungs-Prozess. Letztere hat z.B. Einfluss auf den Status von Pfaden, sowie auf deren Nutzung.

3.2 Architektur der SCTP-Implementierung

Nach Abschluss der Grobspezifikation wurde auf der Basis dieser Aufteilung der Protokollfunktionalitäten mit der Implementierung der einzelnen Prozesse in der Programmiersprache C begonnen. Im nun folgenden Abschnitt werden die Grundprinzipien erläutert, die bei der Erstellung der Implementierung zugrunde gelegt wurden.

3.2.1 Vorüberlegungen

Grundsätzlich gibt es zwei Möglichkeiten, ein neues Protokollsystem auf einem mehrprozessfähigen Betriebssystem zu implementieren:

1. als normalen oder privilegierten Prozess, der vom Betriebssystem (in der Regel nebenläufig zu anderen Prozessen) abgearbeitet wird.

2. als Erweiterung des bestehenden Betriebssystems, je nach System als festes Bestandteil des Betriebssystems, als dynamisches Modul oder als zusätzlicher Treiber.

Jede dieser Möglichkeiten hat Vorzüge und Nachteile. Im ersten Fall ist die Erstellung der Implementierung an die – wohldokumentierten – Schnittstellen des Betriebssystems gebunden, und erlaubt die Fehlersuche und Optimierung mit komfortablen Werkzeugen wie Debuggern, Speicherprüfern (vergl. z.B. das Programm `valgrind`) und Profilern. Darüber hinaus kann auf umfangreiche Bibliotheken zurückgegriffen werden, die häufig benötigte Funktionen wie z.B. die Verwaltung von Listen zur Verfügung stellen, ohne dass diese noch einmal neu implementiert werden müssen. Fehler in der Implementierung führen i.A. zum Abbruch des Protokoll-Prozesses und beeinträchtigen die Funktion des Gast-Betriebssystems nicht. Änderungen können damit schneller in eine solche Implementierung eingebracht und getestet werden. Auf der anderen Seite müssen transportierte Daten mehrfach kopiert werden, z.B. von der Anwendung zum Protokoll-Prozess und von dort noch einmal zum Puffer der Netzwerkschicht des Betriebssystems (entsprechend beim Empfänger), was den Datendurchsatz beeinträchtigen kann. Timer eines normalen Prozesses können nicht mit derselben feinen Granularität laufen, wie Timer innerhalb eines Betriebssystems. Auch dies kann Einfluss auf die Leistung und den Durchsatz einer Protokollimplementierung haben. Des Weiteren skaliert eine Implementierung als Prozess schlechter, d.h. es können gleichzeitig weniger Verbindungen aufrecht erhalten werden, als dies bei einer Implementierung innerhalb des Betriebssystem der Fall ist. Schließlich erfordert die Implementierung eines neuen, auf IP-basierenden Protokolls bei den meisten Betriebssystemen (Linux, Solaris, BSD-Unix) die Nutzung eines sog. Raw-Sockets. Dieser erlaubt es zwar privilegierten Prozessen, alle SCTP-Pakete zu empfangen, er ermöglicht aber nicht die Diskriminierung nach Ports. Damit können mehrere gleichzeitig aktive SCTP-Prozesse auch alle bei einem Endgerät ankommenden SCTP-Pakete empfangen und werden auf diese reagieren, was zu einem scheinbar nicht protokollkonformen Verhalten eines Endgeräts führen

kann. Um diese Problematik zu lösen, ist eine Interprozess-Kommunikation erforderlich, wenn mehrere SCTP-Prozesse gleichzeitig aktiv sind.
Im zweiten Fall, der Betriebssystem-Implementierung, ist die Entwicklungszeit für eine ausgereifte und stabile Protokollschicht erfahrungsgemäß wesentlich höher anzusetzen, da die Komplexität des jeweiligen Betriebssystems wesentlich größer ist als die der funktionalen Schnittstellen für einen Prozess eines Betriebssystems. Fehler bei der Programmierung erfordern in der Regel eine umständliche Fehlersuche und beeinträchtigen oft die Systemstabilität, so dass auch die Entwicklungsumgebung komplexer ist als im ersten Fall. Ferner erfordern auch kleinere Änderungen oft eine aufwändige Neuübersetzung des Betriebssystemkerns. Eine Betriebssystem-Implementierung wird immer nur auf einem bestimmten Betriebssystem laufen, da die internen Unterschiede verschiedener Betriebssysteme z.T. sehr stark ausgeprägt sind. Eine Prozess-Implementierung hingegen kann portabel programmiert werden, so dass sie auf einer Vielfalt von Betriebssystemen lauffähig ist. Ein unbestrittener Vorzug von Betriebssystem-Implementierungen ist hingegen die Leistungsfähigkeit, die im Idealfall nur noch das einmalige Kopieren der zu sendenden Daten von dem erzeugenden Anwendungsprozess in den Betriebssystem-Kern erfordert. Ihre Integration in bestehende Betriebssysteme ist hingegen schwieriger, da die Funktionalität nur über eine komplexe, standardisierte Schnittstelle (siehe auch Abschnitt A.2) angeboten werden kann, die noch nicht von den Systembibliotheken (z.B. `libc`) unterstützt wird. Dies erfordert zumindest in der ersten Übergangszeit zusätzliche Bibliotheken, mittels derer die Anwendungen die neuen Betriebssystem-Schnittstellen nutzen können.

3.2.2 Rahmenbedingungen der Implementierung

Die Zielsetzung zu Beginn der Entwicklung unserer Implementierung war es, möglichst schnell eine umfangreiche Protokollfunktionalität zur Verfügung zu stellen, um Untersuchungen des Protokollverhaltens durchführen zu können. Dabei war die Leistungsfähigkeit der Implementierung zweitrangig. Aufgrund der Vorgaben unserer Projektpartner sollte die Implementierung des Protokolls in einer Funktionsbibliothek in der Programmiersprache C realisiert werden, die neben einer Schnittstelle gemäß Abschnitt 10 von [86] auch die Funktionen der Betriebssystem-Anpassungsschicht (vgl. Abbildungen 3.1 und 3.3) zur Verfügung stellen sollte. Als `sctplib` wurde diese Implementierung im Juni 2000 unter der GNU-Lizenz [27] der Allgemeinheit zur Verfügung gestellt und seither in mehr als 15 Versionen weiter gepflegt. Aus den zahlreichen Rückmeldungen der für

die Implementierung eingerichteten Mailing-Liste [74] ist zu entnehmen, dass die `sctplib` weltweit von vielen Forschungsgruppen und Unternehmen eingesetzt wird.
Durch das Linken des Objektcodes der Bibliothek zu dem einer Anwendung entsteht ein einziger Prozess, der die jeweilige Anwendung und die Protokollfunktionalität des SCTP integriert. Die Nachteile, die durch die Nutzung des Raw-Sockets entstehen, wurden dabei in Kauf genommen: werden mehrere dieser Prozesse gleichzeitig auf einem Endgerät ausgeführt, so verhalten sich diese möglicherweise nicht protokollkonform. So kann z.B. eine Server-Instanz auf einen ankommenden INIT-Chunk mit einem INIT-ACK-Chunk antworten, während eine zweite Server-Instanz diesen Chunk nahezu gleichzeitig mit einem ABORT-Chunk ablehnt. Um auch in diesem Fall Tests durchführen zu können, kann das Aussenden von ABORT-Chunks für einen Prozess deaktiviert werden.
Durch die Kapselung der Anpassungsschicht in einem einzigen Quellcode-Modul konnte die Implementierung portabel programmiert und schnell an neue Betriebssysteme angepasst werden. Zum gegenwärtigen Zeitpunkt werden als Betriebssysteme Linux, Windows, FreeBSD, Solaris und das BSD-basierte Mac OS 10 unterstützt. Abbildung 3.3 zeigt die funktionalen Module der auf der SDL-Spezifikation aus Abschnitt 3.1.3 beruhenden SCTPLIB-Implementierung, deren Funktionen im Folgenden näher erläutert werden.

3.2.3 Die Betriebssystem-Anpassungsschicht

Die Betriebssystem-Anpassungsschicht kapselt alle Funktionen, die abhängig vom jeweiligen Betriebssystem implementiert werden müssen. Hierzu zählen im Wesentlichen die Ein- und Ausgabe über Dateisystem-Deskriptoren sowie Timer. Die Ein- und Ausgabe umfasst dabei das Senden und Empfangen von SCTP-Paketen mittels eines sog. Raw-Sockets, der alle IP-Pakete mit der IP-Kennung 132 für das SCTP-Protokoll empfängt. Zum Öffnen dieses Raw-Sockets und zum Setzen entsprechender Socket-Parameter (wie Größe des Empfangspuffers etc.) werden bei Posix-kompatiblen Betriebssystemen Administrator-Privilegien benötigt.
Darüber hinaus wird auch das Lesen von der Standard-Eingabe zur interaktiven Kommunikation mit einem Benutzer in dieser Schicht implementiert. Um eine einfache Interprozess-Kommunikation mit anderen Prozessen zu ermöglichen, wurden ferner Schnittstellen-Funktionen zur Nutzung des UDP-Protokolls geschaffen.
Zur Verwaltung und Steuerung zeitbasierter Ereignisse (Timer) wird eine Liste

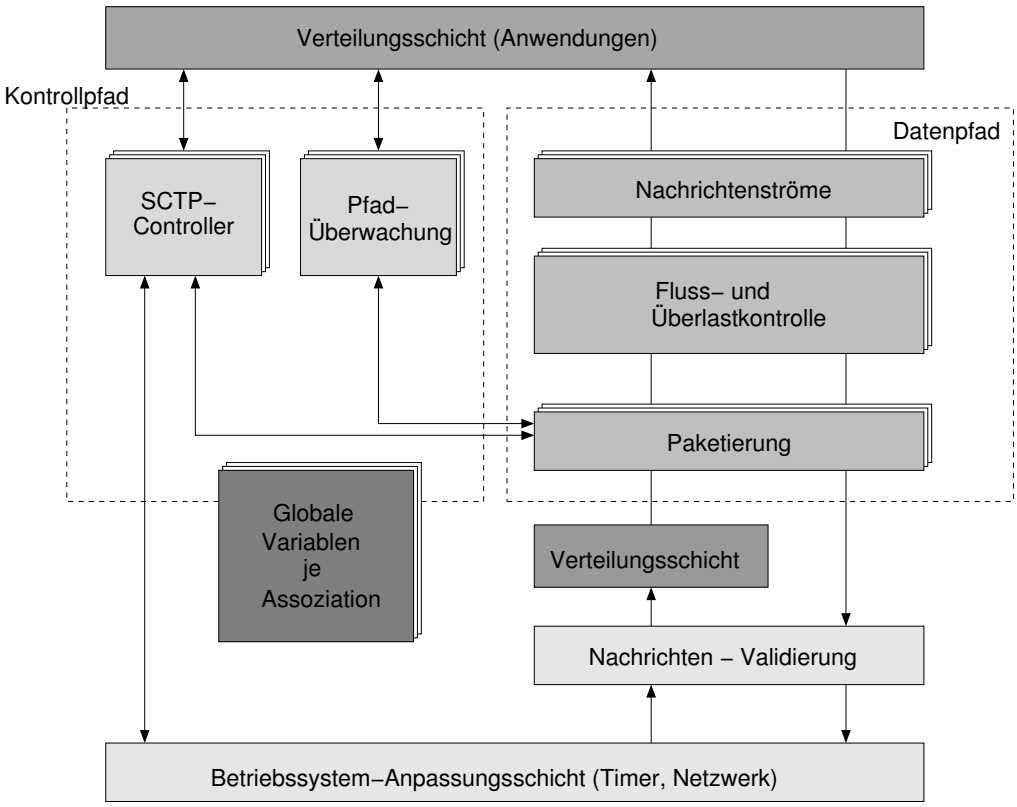

Abbildung 3.3: Funktionale Module der SCTPLIB

mit zukünftigen Ereignissen geordnet nach deren Ausführungszeit geführt. Zeigt die Systemzeit an, dass ein solches Ereignis ausgeführt werden soll, so wird eine zu dem entsprechenden Timer gehörige Funktion mit vorher definierten Parametern aufgerufen. Die Timer werden zur Steuerung des Protokollablaufs eingesetzt, können aber auch vom Anwender genutzt werden.

Die `sctplib` wartet im normalen Betrieb in einer Schleife fortlaufend auf das Eintreten des nächsten Ereignisses, das mit der Betriebssystemfunktion `select()` abgefragt wird. Dies ist entweder ein Ereignis auf einem der registrierten Dateisystem-Deskriptoren (SCTP Eingabe, UDP Eingabe, oder Standardeingabe), oder der Ablauf des nächsten Timers. Alle diese Ereignisse können darin münden, dass die `sctplib` über eine Rückruf-Funktion dem Nutzerpro-

gramm die Ablaufkontrolle übergibt. Dieses Programm muss dann sicherstellen, dass die Ablaufkontrolle nicht für eine zu lange Zeit dort verbleibt, da ansonsten der ordentliche Protokollablauf nicht mehr gewährleistet werden kann. Dies kann man als kooperatives Multitasking auffassen. Die Implementierung dieser beiden Prozesse als nebenläufige Threads wäre eine denkbare Alternative, da hierbei die Steuerung der Ablaufkontrolle dem Scheduler der Threads übertragen würde. Aufgrund des oben bereits erwähnten Projektumfelds wurde davon jedoch Abstand genommen.

3.2.4 Die Verteilungsschicht

Die Verteilungsschicht ermöglicht einem einzelnen Nutzerprogramm die Verwaltung mehrerer nebenläufiger SCTP-Assoziationen und ordnet die eintretenden Ereignisse der jeweils korrekten Assoziation zu. Diese Verwaltungsarbeit ist an zwei Schnittstellen zu leisten (vgl. Abbildung 3.3): zum einen an der Schnittstelle zum Benutzerprogramm, zum anderen an der Schnittstelle zur Betriebssystem-Anpassungsschicht. Für die erstere sieht der Abschnitt 10 von [86] einen Index vor, der die jeweilige Assoziation pro Endpunkt eindeutig bezeichnet. Für ankommende SCTP-Pakete ist es notwendig, aus Absende- und Zieladresse, sowie aus Absende- und Zielport die zugehörige Assoziation zu bestimmen. Die entsprechende Kombination muss für jeden Host eindeutig den Endpunkt der jeweiligen Assoziation bestimmen (vgl. auch Abschnitt 1.4, S.14 von [86]). Timer enthalten als Parameter einen Zeiger auf die zugehörige Assoziation und können so entsprechend ausgeführt werden.
Bei der Implementierung der Assoziationsverwaltung profitiert die `sctplib`-Implementierung von der objektorientierten Struktur, die sich aus der in Abschnitt 3.1.2 beschriebenen anfänglichen Spezifikation ergab. So können alle zu einer Assoziation gehörigen Parameter in den jeweiligen funktionalen Blöcken durch Zeiger auf eine für jeden Block spezifische Datenstruktur abgefragt und modifiziert werden. Die objektorientierte Kapselung erlaubt dabei den Zugriff auf blockinterne Daten nur über definierte Schnittstellenfunktionen, auch wenn dies nicht durch die verwendete Programmiersprache C erzwungen wird.
In der Verteilungsschicht werden auch die wesentlichen Mechanismen implementiert, die die Behandlung von SCTP-Paketen durchführen, die nicht zu einer etablierten Assoziation gehören. Dies können Pakete sein, die den Aufbau einer neuen Assoziation veranlassen (z.B. ein INIT-Chunk) oder aber Pakete, die nicht zugeordnet werden können. In diesem Fall wird in der Verteilungsschicht eine entsprechende standardkonforme Antwort generiert und gesendet, z.B. ein ABORT-

Chunk (vgl. Abschnitt 8.4 von [86]).

3.2.5 Der Kontrollpfad

Zum Kontrollpfad gehören alle Funktionen, die den Auf- und Abbau von Assoziationen steuern, sowie die Überwachung der Pfadzustände. Diese Funktionen sind in zwei funktionalen Modulen implementiert: dem SCTP-Zustandsautomaten (SCTP-Controller) und der Pfadkontrolle, die in Abbildung 3.3 links dargestellt sind.
Der SCTP-Zustandsautomat enthält im Wesentlichen Funktionen, mit denen er auf Dienstprimitive des SCTP-Nutzers reagiert und z.B. den Aufbau oder Abbau von SCTP-Assoziationen initiiert. Als Reaktion auf ankommende Kontroll-Chunks ändert er den Zustand einer Assoziation (also bei INIT, INIT-ACK, COOKIE-ECHO, COOKIE-ACK, ABORT, SHUTDOWN, SHUTDOWN-ACK, oder SHUTDOWN-COMPLETE). Im Fehlerfall wird eine entsprechende Antwort erzeugt (in der Regel Verwerfen von Chunks oder Senden eines ABORT-Chunks). Andere externe Ereignisse können ebenfalls eine Änderung des Assoziationszustands herbeiführen, wie z.B. der Ablauf eines Protokoll-Timers, durch den ein Fehlerzähler erhöht wird. Überschreitet dieser den Schwellwert, dann kann die Assoziation beendet werden (und in den Zustand CLOSED wechseln). Die Implementierung des SCTP-Zustandsautomaten erfolgte auf der Grundlage der vorher erstellten SDL-Grobspezifikation, die alle möglichen Zustandsübergänge erfasste.
Das Pfadkontroll-Modul steuert die Verarbeitung von HEARTBEAT- und HEARTBEAT-ACK-Chunks sowie die Überwachung von Rundlaufzeiten und Pfadzuständen für alle Pfade einer Assoziation. Dazu werden HEARTBEAT-Chunks in regelmäßigen, durch zufällige Jitter verzögerten, Abständen auf allen inaktiven Pfaden gesendet. Für aktive Pfade werden die Bestätigungen empfangener Daten-Chunks für die Messung aktueller Rundlaufzeiten herangezogen. Nach Karn's Algorithmus [50] werden dabei Neuübertragungen ausdrücklich von der Messung ausgeschlossen, da bei diesen nicht eindeutig festzustellen ist, ob eine Bestätigung durch einen erstmalig übertragenen oder einen neu übertragenen Daten-Chunk ausgelöst wurde. Gemäß [86] werden die Messungen der Rundlaufzeiten maximal einmal pro aktueller Rundlaufzeit durchgeführt. Werden auf einem Pfad über einen längeren Zeitraum keine Bestätigungen empfangen und deshalb bei den resultierenden Neuübertragungen die Fehlerzähler des Pfades inkrementiert, so wechselt der Zustand eines Pfades von *aktiv* zu *inaktiv*, wenn der Fehlerzähler einen Schwellwert (`path.max.retrans`) überschreitet. In die-

sem Fall wird der SCTP-Nutzer durch eine Rückruf-Funktion benachrichtigt. Sind entweder alle Pfade inaktiv oder überschreitet der Fehlerzähler für die gesamte Assoziation einen zweiten Schwellwert (`assoc.max.retrans`), so wird die Assoziation terminiert.
Zusammen mit den minimalen und maximalen Timeouts (`path.rto.min` und `path.rto.mx`) für die Neuübertragungstimer sind diese Schwellwerte Protokollparameter, die die Geschwindigkeit und Sicherheit, mit der ein tatsächlicher Ausfall eines SCTP-Pfades erkannt werden kann, entscheidend beeinflussen. Weitere Informationen zu diesem Thema finden sich in Abschnitt 4.4.

3.2.6 Der Datenpfad

Die Implementierung des Datenpfades ist analog zu der in Abschnitt 3.1.2 erläuterten Struktur realisiert. Dabei wird in dem Modul zur Verwaltung der Nachrichtenströme (engl. Stream Engine) die Segmentierung und Wiederherstellung von Nachrichten durchgeführt. Für jeden vom SCTP zu sendenden Daten-Chunk wird eine Datenstruktur erstellt, die neben den Nutzdaten alle relevanten Informationen, wie Anzahl der Sendeversuche, letzte Zieladresse, Lebensdauer, Status der Bestätigung etc. enthält. Zur effizienten Verwaltung dieser Strukturen werden innerhalb der `sctplib` nur Zeiger benutzt. Lediglich bei der endgültigen Paketierung werden die Nutzdaten in einen zusammenhängenden Sendepuffer kopiert. Alle Nutzdaten werden in der Stream Engine in Warteschlangen verwaltet, bis die Flusskontrolle ein Aussenden erlaubt. Erstmalig gesendete Nutzdaten werden entsprechend in einer Liste in dem Modul zur Neuübertragung solange vorgehalten, bis eine gültige Bestätigung eintrifft und der CTSNA-Wert (vgl. Abschnitt 2.3.5) anzeigt, dass die Daten beim assoziierten Endpunkt angelangt sind. Erneut zu übertragende Daten-Chunks werden in eine Sendewarteschlange im Flusskontroll-Modul eingereiht und haben dort Vorrang vor neuen (noch nicht gesendeten) Chunks. Da die `sctplib` die in Abschnitt 2.4.1 beschriebene Erweiterung (PR-SCTP) voll unterstützt, kann der SCTP-Nutzer zu sendenden Nachrichten eine Lebensdauer zuordnen. Ist diese überschritten, so werden die zu der entsprechenden Nachricht gehörigen Daten-Chunks nicht mehr gesendet und auch nicht mehr neu übertragen. Wurden die entsprechenden Daten-Chunks bereits gesendet (und somit TSNs vergeben), so sendet der SCTP-Endpunkt einen Forward-TSN Kontroll-Chunk und zeigt dem assoziierten Endpunkt damit an, dass Chunks übersprungen wurden und nicht länger erwartet werden müssen.
In Empfangsrichtung überwacht das Modul zur SACK-Erzeugung die ankommenden Daten-Chunks. Nur neue, d.h. erstmalig empfangene Daten-Chunks werden

an die Stream Engine weitergeleitet. In dem Modul werden auch Strukturen und Listen für mehrfach empfangene Daten und die sog. Lücken in der Ankunftsreihenfolge verwaltet, d.h. für Daten-Chunks, die nicht in sequenzieller Reihenfolge empfangen wurden. Aus diesen Strukturen werden zu geeigneten Zeitpunkten die Bestätigungen in Form von SACK-Chunks generiert. Dieses Modul verarbeitet ankommende Forward-TSN Kontroll-Chunks. Ein solcher Chunk hat prinzipiell eine ähnliche Wirkung wie der Empfang aller Daten-Chunks mit einer TSN kleiner oder gleich der Forward-TSN. Der einzige Unterschied besteht darin, dass das Modul zur Verarbeitung der Nachrichtenströme alle Warteschlangen überprüfen muss, um etwaige Nachrichtenfragmente, die dann nicht mehr zusammengesetzt werden können, zu verwerfen und fertige Nachrichten zuzustellen.

3.2.7 Weiterentwicklung der SCTP-Implementierung

Die Entwicklung der `sctplib` war und ist ein kontinuierlicher Prozess, in dem neu erkannte Fehler beseitigt und neue Funktionen hinzugefügt werden. Die aktuelle Entwicklung geschieht in zwei getrennten Versionen. Die Version 1.0 unterstützt den oben beschriebenen Funktionsumfang, liefert also eine Implementierung die im Wesentlichen konform zu [86, 89, 80] ist und die Netzwerkprotokolle IPv4 und IPv6 unterstützt. Ferner wird die PR-SCTP Erweiterung [82] unterstützt. Die Version 1.0 ist zurzeit für die Betriebssysteme Linux, FreeBSD, Solaris, Mac OS 10 und Windows verfügbar und wird als stabile Version weiter gepflegt, d.h. es werden neu erkannte Probleme beseitigt, ohne dass der Funktionsumfang wesentlich erweitert wird.
Daneben gibt es eine Entwicklerversion 1.3, die neben dem von der Version 1.0 unterstützten Funktionsumfang noch weitere Funktionalitäten aufweist. Zunächst wurde in dieser Version die interne Adressverwaltung umstrukturiert und die Erkennung der Änderung lokaler Adressen hinzugefügt. Dies ermöglicht es beispielsweise der Implementierung zu erkennen, wenn ein Interface in einem Funknetz eine neue IP-Adresse erhält, oder aber die Verbindung verliert. Darauf aufbauend wurde ein Modul hinzugefügt, das die SCTP-Erweiterung zur dynamischen Adresskonfiguration (vgl. Abschnitt 2.4.2) realisiert.
Ein weiterer Punkt ist die im RFC 2960 geforderte Pfad-MTU Erkennung (vgl. auch Abschnitt 2.3.1). Diese wird nicht in der Version 1.0 unterstützt und wurde in der Entwicklerversion 1.3 ergänzt. Als maximale MTU einer Assoziation wird dazu zunächst die kleinste MTU aller lokalen Interfaces angenommen. Auf allen Pfaden wird bei Paketen, die die der MTU entsprechende Länge erreichen, regelmäßig das “Don’t Fragment”-Bit (DF-Bit) des IP-Protokollkopfes gesetzt. Sind

Pfade unbenutzt, so werden regelmäßig (z.B. alle 10 Minuten) HEARTBEAT-Chunks in der der MTU entsprechenden Länge gesendet. In [10] wird gefordert, dass Router, die ein IP-Paket der entsprechenden Länge nicht ohne IP-Fragmentierung weiter senden können, das Paket bei gesetztem DF-Bit verwerfen und eine ICMP-Fehlermeldung (Typ: “Endpoint Unreachable”, Code: “Fragmentation Needed”) zurücksenden müssen. Diese Fehlermeldung enthält auch einen MTU-Wert, der die maximale Länge einer Nachricht anzeigt, die ohne Fehler hätte übertragen werden können. Die `sctplib` dekodiert die ankommenden ICMP-Fehlermeldungen und passt die jeweilige Pfad-MTU der Assoziationen nachfolgend entsprechend an.
Schließlich wurde das bereits in Abschnitt 3.2.1 angesprochene Problem der Interprozesskommunikation gelöst. Dazu wurde die SCTP-Implementierung in ein Client-/Server-System aufgespalten. Der Server, nachfolgend *SCTP-Server* genannt, ist als zentraler, privilegierter Hintergrundprozess einmal pro Endgerät aktiv und führt die Ein- und Ausgabe über den SCTP-Netzwerksocket durch. Der SCTP-Server reagiert auf ankommende SCTP-Pakete und weist diese gegebenenfalls ab, wenn keine dazu passende Client-Instanz registriert ist. Ist eine passende Client-Instanz registriert, so leitet er das ankommende Paket durch eine Interprozesskommunikation an diese Instanz weiter. Diese Interprozesskommunikation bedient sich der Unix-Domain Sockets, die einen zuverlässigen, datagrammorientierten lokalen Transport zur Verfügung stellen.
SCTP-Clients führen die eigentliche Verarbeitung der für sie bestimmten SCTP-Pakete durch. Die Schnittstelle zwischen SCTP-Clients und SCTP-Server liegt also in der Verteilungsschicht. Diese wurde um die Funktionalität der Interprozesskommunikation erweitert und stellt den Clients und dem Server Funktionen zur Registrierung, Deregistrierung, zum Empfang und Versenden von SCTP-Paketen sowie für sonstige Ereignisse (z.B. Änderung der Pfad-MTU, Änderung der lokalen Adressen, Überwachung des Status von Client-Programmen) zur Verfügung. Diese interne Änderung der `sctplib`-Implementierung wurde so durchgeführt, dass vorhandene Programme, die auf die stabile Version 1.0 aufsetzen, prinzipiell unverändert weiter eingesetzt werden können und ordnungsgemäß funktionieren, sofern ein SCTP-Server aktiv ist.
Auch vom Standpunkt der Systemsicherheit ist diese Änderung der Implementierungsstruktur äußerst vorteilhaft, da die Funktionalität des privilegierten Prozesses auf ein notwendiges Minimum beschränkt werden kann, und die Nutzer des SCTP auf einem System mehrere SCTP Client-Prozesse auch mit niedrigen Privilegien starten können. Zudem ist es bei der Client-/Server-Struktur möglich, eine korrekte Behandlung mehrerer nebenläufiger SCTP-Prozesse auf einem Sy-

stem durchzuführen, da ein einzelner zentraler Prozess die Verteilung von SCTP-Paketen koordiniert.

3.3 SCTP Benutzerschnittstellen

Für die Nutzung des SCTP existieren derzeitig zwei mögliche Benutzerschnittstellen. Die erste entspricht den von OSI bekannten Dienstprimitiven, die in Form von Schnittstellen- und Rückruffunktionen (engl. Callbacks) implementiert werden. Abschnitt 10 von [86] stellt beispielhaft diesen Dienst-Zugangspunkt für das SCTP vor. Er wurde als Grundlage für die Entwicklung der Schnittstelle zur `sctplib` genommen. Darin existieren entsprechende Funktionen für den Auf- und Abbau von Verbindungen, zum Senden und Empfangen von Nachrichten und für die Status-Überwachung. Protokollereignisse (z.B. ein erfolgreicher Aufbau einer Assoziation) werden der jeweiligen Anwendung über Rückruffunktionen mitgeteilt. Die verfügbaren Dienstprimitive und ihre Implementierung werden ausführlicher im Anhang A.1 erläutert.
Die zweite Schnittstelle ist entsprechend der in den gängigen Betriebssystemen verfügbaren, POSIX-kompatiblen Socket-Schnittstelle modelliert. Dabei enthält sie Modifikationen, um die Möglichkeiten des SCTP auszuschöpfen. Diese Socket-Schnittstelle ist in [88] definiert und ähnelt der Standard-Schnittstelle für die auf einer Vielzahl von Betriebssystemen etablierten Transport-Protokolle TCP oder UDP. Die weitgehende Kompatibilität verspricht eine einfache Portierung der meisten Anwenderprogramme, in der Regel aber ohne die – gegenüber den Protokollen TCP und UDP – stark erweiterten Möglichkeiten des SCTP auszunutzen. Die Nutzung von Möglichkeiten wie dem Multiplexen von Nachrichten in unterschiedliche Datenströme oder dem Multihoming wird durch eine Erweiterung bestehender Funktionen der Socket-Schnittstelle erreicht.
Durch die Festlegung von Daten-Strukturen und Funktionen in dem Dokument [88] bereits in einer frühen Phase der Entwicklung von neuen Protokollimplementierungen (siehe auch Abschnitt 3.4) kann sichergestellt werden, dass eine Vielzahl von SCTP-Implementierungen der verschiedensten Betriebssysteme eine identische Schnittstelle aufweisen werden. So können betriebssystemübergreifend SCTP-spezifische, portable Anwendungen geschrieben werden.
Das Dokument [88] unterscheidet zwei Socket-Typen:

- TCP-ähnliche Sockets, die eine 1-zu-1 Beziehung zwischen einem Socket-Deskriptor und einer Assoziation aufweisen.

- UDP-ähnliche Sockets, die eine 1-zu-N Beziehung zwischen einem Socket-Deskriptor und N Assoziationen aufweisen können.

Nähere Erläuterungen zu diesen Socket-Typen und ihren Unterschieden, sowie zur typischen Anwendung und zu SCTP-spezifischen neuen Funktionen finden sich im Anhang A.2.

3.4 Vergleich verschiedener Implementierungen

Nachfolgend wird ein kurzer Überblick über die gängigsten Implementierungen (Stand: Mai 2005) gegeben, die zurzeit für die etablierten Standard-Betriebssysteme zur Verfügung stehen. Abschließend zeigt Tabelle 3.1 die wichtigsten Eigenschaften dieser Implementierungen noch einmal im Überblick.
Alle hier erwähnten Implementierungen, inklusive der `sctplib`, wurden auf regelmäßig stattfindenden Treffen erfolgreich auf Interoperabilität hin getestet.

3.4.1 Die SCTP-Implementierung für KAME

KAME ist eine Erweiterung des Netzwerk-Protokollstacks für die Betriebssysteme FreeBSD, OpenBSD und NetBSD. Diese Erweiterung liefert für die erwähnten Systeme insbesondere eine vollwertige und stabile Unterstützung von IPv6, der nächsten Generation des Internet-Protokolls, und von IPSec.
Der von der Firma Cisco beschäftigte Randall Stewart, einer der Autoren des RFC 2960, begann im Jahr 2001 mit der Erstellung einer unter der BSD-Lizenz stehenden Kernel-Implementierung von SCTP im KAME-Stack. Diese wurde vorwiegend unter FreeBSD entwickelt. Zum gegenwärtigen Zeitpunkt ist dies wahrscheinlich die stabilste und umfangreichste Implementierung, die erhältlich ist. Die RFCs 2960 und 3309 [86, 89] sowie die Erweiterung des SCTP-Implementer's Guide [80] werden vollständig unterstützt. Ebenso werden die Erweiterungen PR-SCTP und Add-IP [82, 83] und darüber hinaus die Early Congestion Notification (ECN) [67] unterstützt. Außerdem werden noch zwei Erweiterungen unterstützt: Hochgeschwindigkeits-SCTP, eine Anpassung des Drafts [23] an das SCTP und der Draft [81] zum SCTP Packet-Drop Report. Dies erlaubt es Netzwerkelementen, einem SCTP-Sender Paketverluste zu melden, die nicht durch Überlast (engl. Congestion), sondern z.B. durch Verluste auf einer Funkstrecke zustande kommen. In diesem Fall muss die Überlastkontrolle die Senderate nicht durch den AIMD-Algorithmus so stark reduzieren, wie in Abschnitt 2.3.8 beschrieben.

Eine Unterstützung für ECN-Nonce [78], eine robustere Version der ECN, ist noch nicht vorhanden, soll aber in naher Zukunft implementiert werden. Eine Bibliothek ruft die Schnittstellenfunktionen des Betriebssystemkerns, um die in [88] beschriebenen Funktionen wie z.B. `sctp_sendmsg()` zu implementieren. Alle grundlegenden Funktionen, wie z.B. `socket()`, stehen natürlich im System zur Verfügung. Da die KAME-Erweiterung IPv6 unterstützt, werden – im Gegensatz zu den meisten IPv4-Netzwerkstacks – auch mehrere Default-Einträge beim Routing möglich, was es dieser SCTP-Implementierung erlaubt, z.B. im Falle einer Neuübertragung oder eines Pfadausfalls eine sekundäre Default-Route zu benutzen.
Da KAME unter FreeBSD, NetBSD und OpenBSD läuft, werden alle Architekturen unterstützt, unter denen diese Betriebssysteme laufen. Das sind zumindest ix86, Intel64/AMD64, Alpha, PowerPC und Sparc64. Auf allen diesen Systemen kann SCTP nach Installation des KAME-Stacks statisch in den Betriebssystemkern einkompiliert werden.

3.4.2 SCTP im Linux-Kern (LKSCTP)

Seit 2001 arbeitet eine Gruppe von Entwicklern verschiedener Firmen (u.a. Motorola, IBM und Nokia) an einer nativen Kernel-Implementierung für das Betriebssystem Linux (LKSCTP). Seit der Kernel-Version 2.5.43 ist SCTP Bestandteil des Standard-Betriebssystemkerns und wurde auch in der stabilen Kernel-Version 2.6 als experimentelles Netzwerkmodul integriert.
Gegenwärtig unterstützt diese Implementierung die RFCs 2960, 3309 und den Implementer's Guide, sowie die Early Congestion Notification (ECN). Erweiterungen wie PR-SCTP und Add-IP werden noch nicht unterstützt; es ist jedoch geplant, diese in Zukunft zu implementieren.
Ähnlich wie bei der KAME-Implementierung werden sowohl TCP-ähnliche als auch UDP-ähnliche Socket-Schnittstellen unterstützt, und alle anderen Funktionen von [88] werden in einer Bibliothek zur Verfügung gestellt. LKSCTP unterstützt auch einen Teil der SCTP-MIB [62], aus der über das Protokoll SNMP [17, 54] bestimmte Protokollparameter und Statistikwerte abgefragt werden können.
Die LKSCTP Implementierung für Linux-Version 2.6 wurde auf den Plattformen ix86, PowerPC und Intel64 validiert, andere werden sicherlich folgen. Aufgrund der Modularität des Linux-Kerns kann SCTP entweder statisch in den Kern einkompiliert werden oder aber zur Laufzeit als Kern-Modul wie ein Treiber in ein System geladen werden. Wie auch der Linux-Kern wird die LKSCTP-

Implementierung unter der GPL veröffentlicht, während die Bibliothek unter der LGPL freigegeben ist (damit auch kommerzielle Programme diese nutzen dürfen, ohne ihren Quellcode offenlegen zu müssen). Die Stabilität des LKSCTP wird von den Entwicklern als zuverlässig, aber noch nicht ausreichend eingestuft und die Komplexität der Implementierung liegt bei etwa 30000 Zeilen C-Code.

3.4.3 SCTP im Cisco IOS

Cisco spielte eine wichtige Rolle bei der Standardisierung des SCTP und integriert dieses Protokoll auch in seine IOS-Betriebssysteme. Cisco war auch eine der ersten Firmen, die ihren SCTP-Stack in Produkten auf den Markt brachten (Cisco IP-Transferpunkte). Die IOS-Implementierung wird u.a. auch von Randall Stewart mitentwickelt.
Sie unterstützt die RFCs 2960 und 3309 sowie den Implementer's Guide und auch die PR-SCTP Erweiterung. Die Add-IP Erweiterung wird noch nicht unterstützt; lediglich ein kleiner Teil der Spezifikation, der die Übertragung von speziellen Anwendungsdaten (engl. Adaptation Layer Indication) beim Verbindungsaufbau erlaubt, wurde implementiert. ECN [67] wird ebenfalls nicht unterstützt
Cisco-intern wird eine über SNMP abzufragende MIB integriert, jedoch ist dies noch nicht Teil der Standard SCTP-Implementierung für das IOS. Die Implementierung unterstützt darüber hinaus den SCTP Packet-Drop Report [81].

3.4.4 SCTP in IBM's AIX-Unix

Neben seinem Engagement beim Linux-System hat IBM noch eine weitere SCTP-Implementierung entwickelt, die in das IBM-eigene kommerzielle AIX Unix integriert ist. Ebenso wie die anderen Implementierungen unterstützt die AIX-Version die RFCs 2960 und 3309 sowie den Implementer's Guide (bis zur Version 04, Stand Oktober 2002). Eine weitere Entwicklung der Implementierung ist nach Aussagen der Entwickler zunächst nicht mehr vorgesehen.
Die Early Congestion Notification wird unterstützt, die Erweiterungen PR-SCTP und Add-IP oder die SCTP-MIB aber nicht. Die Implementierung bietet nur ein UDP-ähnliches Interface, bei dem alle Funktionen über Standard-Betriebssystem-Aufrufe (wie z.B. `socket()` oder `sendmsg()`) zugänglich sind.
Unterstützt wird nur das System AIX auf der PowerPC-Architektur. SCTP auf AIX ist modular und kann zur Laufzeit in das System geladen werden. Die Implementierung ist stabil und hat einen Umfang von ca. 15000 Zeilen C-Code, der von 3 Personen entwickelt wurde.

	KAME	LKSCTP	IOS	AIX	SCTPLIB
RFC 2960	Ja	Ja	Ja	Ja	Ja
Impl. Guide	Ja	Ja	Ja	Ja (bis Version 04)	Ja
RFC 3309	Ja	Ja	Ja	Ja	Ja
PR-SCTP	Ja	Nein	Ja	Nein	Ja
ADD-IP	Ja	Nein	z.T.	Nein	Ja
ECN	Ja	Ja	Nein	Ja	Nein
Socket-API (1-zu-1)	Ja	Ja	Nein	Nein	Ja (s.o.)
Socket-API (1-zu-N)	Ja	Ja	Nein	Ja	Ja (s.o.)
Andere Erweiterungen	Ja[1,2]	Nein	Ja[1]	Nein	Nein
Lizenz	BSD	GPL	kommerziell	kommerziell	LGPL
Betriebssysteme	BSD-Unices	Linux 2.6	IOS	AIX	Linux, FreeBSD, Solaris, Windows
Architekturen	fast alle gängigen	ix86,i64, PowerPC	MIPS, PowerPC	PowerPC	ix86, PowerPC, Sparc, Arm
Modular	Nein	Ja	Nein	Ja	Keine Kernel-Implementierung
Stabilität	Hoch	Mittel	Hoch	Mittel	Hoch
Code-Zeilen (ca.)		30000		15000	20000

Tabelle 3.1: Vergleich verschiedener SCTP-Implementierungen ([1]Packet Drop Extension; [2] High-Speed SCTP)

Kapitel 4

Protokolluntersuchungen im Labor-Testbett

Im folgenden Kapitel werden die Forschungsergebnisse vorgestellt, die u.a. mit SCTP Implementierungen im Laborbetrieb erarbeitet wurden. Zunächst wird der grundsätzliche Aufbau des Testbetts erläutert, in dem diese Versuche realisiert wurden.
Das Ziel der ersten Untersuchungen, die zur Evaluierung des SCTP insgesamt und verschiedener, standardkonformer Protokollausprägungen im Besonderen angefertigt wurden, war es, das Verhalten der untersuchten Implementierungen in Bezug auf die Qualität der Fluss- und Überlastkontrolle zu untersuchen. Da der Einsatz eines neuen Protokolls insbesondere im öffentlichen Internet ähnlich hohe Anforderungen stellt, wie der Einsatz etablierter Protokolle (insbesondere des TCP), muss die Verträglichkeit aller in diesem Umfeld benutzten Protokolle gewährleistet sein. An dieser Stelle ist besonders zu untersuchen, ob der allgemeine Einsatz des SCTP eine Verdrängung oder Qualitätseinbußen bestehender, auf anderen Protokollen basierender, Anwendungen zur Konsequenz haben kann. Die im Kapitel 2.4 erwähnten Erweiterungen des SCTP dürfen ebenfalls nicht zu solchen Auswirkungen führen.
In weiterführenden Untersuchungen wurde eine Modifikation des Algorithmus zum Senden von Empfangsbestätigungen (SACK-Chunks) vorgeschlagen, der in bestimmten Szenarien zu einem wesentlich günstigeren Protokollverhalten führt, als der entsprechende, im Standard RFC 2960 [86] geforderte Algorithmus. Die Ergebnisse dieser Untersuchungen werden in Abschnitt 4.3 zusammengefasst.
Mit dem Einsatz von SCTP als das vorrangige, den IP-basierten Transport von Signalisierungsmeldungen stützende, Protokoll sind besondere Anforderungen

verbunden, damit auch bei widrigen Umständen Fehlersituationen schnell erkannt und behoben werden können. Inwieweit und unter welchen Bedingungen SCTP diese Anforderungen erfüllen kann, wird in Abschnitt 4.4 näher erläutert.

4.1 Aufbau des Testbetts

Das Testbett besteht aus einem einfach strukturierten Netz, das Untersuchungen des SCTP sowohl im Singlehoming-Betrieb (mit einer Adresse pro Endgerät), als auch im Dualhoming-Betrieb (mit je zwei Adressen pro Endgerät) erlaubt. Durch den zusätzlichen Einsatz von IPv6 kann die Zahl der Adressen pro Endpunkt noch einmal um 4 erhöht werden (je eine site-lokale und globale Adresse pro Interface), so dass Konfigurationen mit 6 Adressen pro Endpunkt möglich sind.

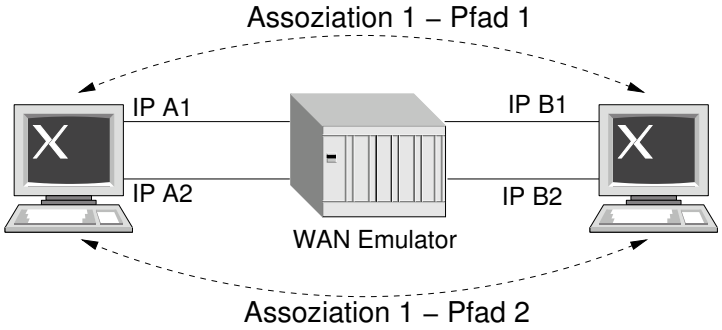

Abbildung 4.1: Testbett-Konfiguration für Dualhoming

Abbildung 4.1 stellt eine einfache Netzkonfiguration für zwei Endpunkte dar, die über die Adressen A1 und A2 sowie B1 und B2 erreichbar sind. Zentrales Element dieses Szenarios ist ein sog. WAN-Emulator, der es erlaubt, verschiedene einfache Netzkonfigurationen einzustellen. Dieser WAN-Emulator ist prinzipiell ein einfacher Router, der zusätzlich Funktionen wie (deterministische oder zufällige) Verzögerung transportierter IP-Pakete, beschränkte Paket-Warteschlangen und die Beschränkung der Netz-Bandbreite zur Verfügung stellt. Basis dieses WAN-Emulators waren die Systeme Dummynet [71] unter FreeBSD, sowie Nistnet [16] unter Linux.
Mit diesem Testbett ist es möglich, das Protokollverhalten in Abhängigkeit von der festgelegten Rundlaufzeit und Verlustwahrscheinlichkeit zu untersuchen.

Darüber hinaus können Fehlerszenarien untersucht werden, in denen ein Pfadausfall simuliert wird. Dafür kann z.B. auf den Endgeräten oder auf dem WAN-Emulator mit Firewall-Regeln der Verlust aller IP-Pakete erzwungen werden. Paketüberholungen können durch variable Verzögerung aufeinander folgender Pakete ebenfalls erzeugt werden.
Um den maximalen Durchsatz der untersuchten Protokolle TCP und SCTP unter den jeweilig durch den WAN-Emulator erzwungenen Bedingungen zu ermitteln, werden Programme wie das bekannte `ttcp` [77] für das TCP und ein entsprechendes Programm für das SCTP benutzt. Diese Programme können in den Modi *Quelle* oder *Senke* arbeiten und erzeugen einen unidirektionalen Verkehrsstrom zwischen der Quelle und der Senke. Da die Quelle in einem Modus arbeitet, in dem sie immer gesättigt ist, wird bei Einsatz der Testprogramme der maximale Durchsatz der verwendeten Transportprotokolle erzielt, der durch die Fluss- und Überlastkontrollmechanismen gesteuert wird. Weitere Testprogramme erzeugten bidirektionalen Verkehr unter SCTP. Die benutzten Testprogramme wurden so geschrieben, dass sie periodisch den Protokolldurchsatz bezogen auf die jeweilige, konfigurierte Zeitperiode (z.B. mehrmals pro Sekunde) ausgeben können.

4.2 Untersuchung des Verhaltens der Fluss- und Überlastkontrolle

Sowohl TCP als auch SCTP sind Transportprotokolle mit einer *fensterbasierten* Flusskontrolle. Sie können in einem gegebenen Zeitraum zwischen Aussenden von Daten und Eintreffen einer zugehörigen Bestätigung ($= 1 \cdot RTT$) maximal eine Datenmenge senden, die dem Sendefenster des Kommunikationspartners entspricht. Für kurze Rundlaufzeiten beschränkt die zur Verfügung stehende Bandbreite der Netzwerkschicht den maximalen Nutzdurchsatz des Transportprotokolls, für längere der Quotient aus der Größe des Empfangsfensters des Partners und der Rundlaufzeit.

4.2.1 Eine TCP-Verbindung und eine SCTP-Assoziation

Das transiente Verhalten des SCTP wurde untersucht, wenn bei bestehender SCTP-Verbindung eine weitere SCTP- oder TCP-Verbindung aktiviert wird. Diese konkurriert mit der ersten SCTP-Verbindung um die zur Verfügung stehende Bandbreite. Abbildung 4.2 zeigt das transiente Verhalten beim Einschalten einer

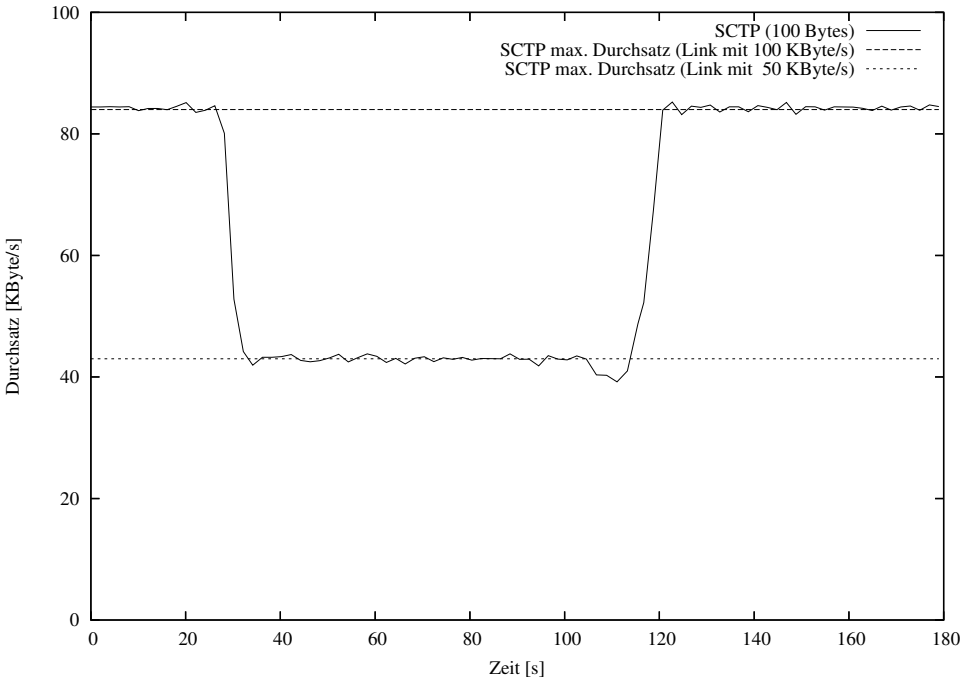

Abbildung 4.2: Protokolldurchsatz einer SCTP-Assoziation beim Einschalten einer TCP-Verbindung

kurzzeitig aktiven zweiten TCP-Verbindung. Dabei erhalten nach dem Abklingen des Einschaltvorgangs die Anwendungen auf beiden Verbindungen gleiche Anteile des Gesamtnutzdurchsatzes.

4.2.2 Mehrere TCP-Verbindungen und eine SCTP-Assoziation

Im Weiteren wurde das Verhalten der Fluss- und Überlastkontrolle des SCTP untersucht, wenn sich eine Assoziation mit mehreren konkurrierenden TCP-Verbindungen einen gemeinsamen, bandbreitenbeschränkten Link teilt. Bei dieser Untersuchung ergab sich, dass auf Links mit einer zur Verfügung stehenden Bandbreite (bezogen auf den Nutzdurchsatz) von B bei insgesamt n Verbindungen mit einem Bandbreitenbedarf von x, für $B \leq (n \cdot x)$, jede Verbindung einen Anteil von B/n erhält. Abbildung 4.3 stellt das Ergebnis anschaulich dar.
Diese Untersuchungen werden in der Simulation bestätigt (siehe auch Abschnitt 5.4.1).

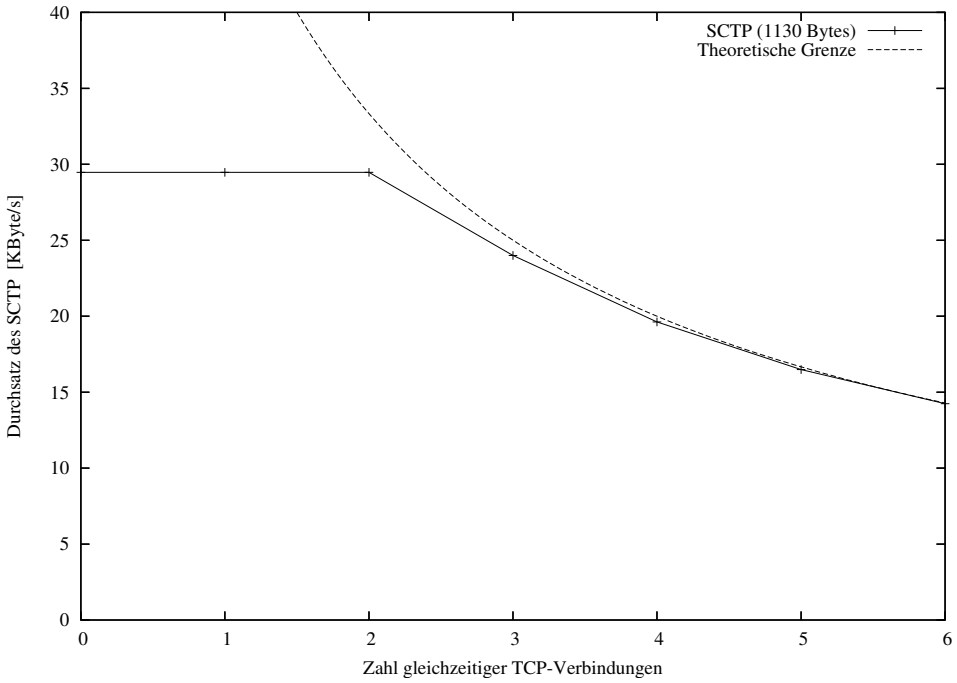

Abbildung 4.3: Protokolldurchsatz einer SCTP-Assoziation beim Zuschalten mehrerer TCP-Verbindungen

4.2.3 Vergleich zwischen TCP und SCTP

Diese ersten Ergebnisse zum Verhalten des SCTP in Zusammenhang mit dem TCP wurden in [49] vorgestellt, wobei das oben erwähnte Programm `ttcp` für Durchsatz-Messungen des TCP zur Anwendung kam und eine entsprechende Anwendung für das SCTP. Bei den Messungen wurden Nachrichten unterschiedlicher Längen übertragen.

TCP minimiert durch die Zusammenfassung von Nachrichten zu einem Bytestrom den Protokolloverhead, da die übertragenen Segmentlängen maximiert werden. Es erfordert aber bei der Übertragung von (kurzen) Nachrichten zusätzlich die Einfügung von Markierungen (engl. Record Marking), die es der Anwendung erlauben, die Nachrichtengrenzen zu bestimmen und die Nachrichten wiederherzustellen. SCTP hingegen weist in der Regel einen höheren Protokolloverhead auf, macht diesen jedoch in der Praxis oft durch die gewonnene Flexibilität bei der Zustellung angekommener Nachrichten, insbesondere im Falle von Paketverlusten

im Netz und bei mehreren voneinander unabhängigen Nachrichtenströmen, wett. Für nachrichtenorientierte Anwendungen ist die Anwendung von SCTP auch vorteilhaft, da Nachrichten effektiv in einem SCTP-Paket zusammengefasst werden können (vgl. Abschnitt 2.3.5).
Die in [49] vorgestellten Messungen ergeben, dass sich die Protokolle TCP und SCTP beim Wettbewerb um limitierte Bandbreitenressourcen der Netzwerkschicht in einer einfachen Netzwerktopologie ähnlich verhalten und dass die Bandbreite prinzipiell fair auf die Anzahl der Verbindungen verteilt wird. Kleine Unterschiede zwischen TCP und SCTP ergeben sich jedoch dadurch, dass die Berechnung der Fluss- und Überlastkontrollparameter (insbes. $cwnd$, $rwnd$) beim SCTP auf Basis der gesendeten Bytes, beim TCP aber auf Basis der gesendeten MTUs geschieht.

4.2.3.1 Weiterführende Untersuchungen

In der Veröffentlichung [11] untersuchten Brennan und Curran die Überlastkontrollmechanismen von SCTP und TCP an einem Simulationsmodell und stellten Mängel beim SCTP im Vergleich zur TCP-Reno Variante fest, die bei einzelnen Paketverlusten zum Tragen kommen. Es war insbesondere Tatsache, dass die schnelle Neuübertragung von Daten-Chunks beim SCTP nach einem einzelnen Verlustereignis mehrfach erfolgen konnte, wodurch das $cwnd$ unnötig verringert wurde. Ferner wurde vorgeschlagen, bei Neuübertragungen den aktuellen Wert des $cwnd$ nicht zu berücksichtigen, was in jedem Fall eine Neuübertragung ermöglicht (selbst wenn bei maximal reduziertem $cwnd$ noch nicht alle Daten von der Gegenseite bestätigt sind). So wurden in [11] entsprechende Modifikationen vorgeschlagen, die Eingang in die Standardisierung und in den “SCTP Implementer’s Guide”[80] fanden.

4.3 Ein modifizierter Algorithmus für Empfangsbestätigungen

Das Verhalten des SCTP wurde in einem anwendungsorientierten Szenario mit TCP verglichen (vgl. auch [46]). Dabei wurde davon ausgegangen, dass dem TCP und dem SCTP eine bidirektionale Satellitenverbindung mit der Bandbreite W' eines T1-Links ($W' = 1.544$ MBit/s) zur Verfügung steht, die eine hohe Verzögerung von $D' = 250$ ms aufweist. Abbildung 4.4 zeigt schematisch die untersuchte Netztopologie.

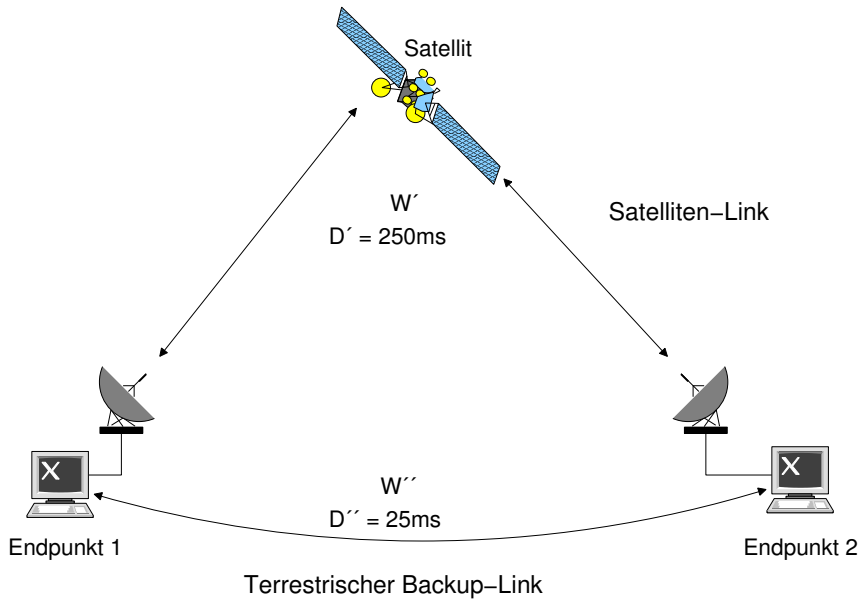

Abbildung 4.4: Durchsatzuntersuchung bei TCP und SCTP in einer Topologie mit Satellitenstrecke

Satellitenverbindungen können temporär, z.B. bei atmosphärischen Störungen, eine recht hohe Verlustwahrscheinlichkeit aufweisen, was bei TCP und SCTP dazu führt, dass der Flusskontrollmechanismus von Netzüberlast ausgeht und die Senderate (über das $cwnd$) reduziert. Dabei verhalten sich TCP und SCTP prinzipiell ähnlich.

Das erwähnte Szenario wird nun mit einer zusätzlichen Backup-Leitung (z.B. einem dedizierten ISDN-Link) erweitert, die nur das SCTP aufgrund der Multihoming-Unterstützung nutzen kann. Diese Leitung soll eine Bandbreite von $W'' = 128$ KBit/s zur Verfügung stellen und eine wesentlich kürzere Linkverzögerung $D'' = 25$ ms aufweisen als die Satellitenstrecke.

In dem untersuchten Szenario wurde angenommen, dass ein Sender mit einem mittleren (konstanten) Ankunftsabstand von $\Delta t_{msg} = 33$ ms Nachrichten sendet, die jeweils eine Nutzlast von 1 KByte enthalten. Damit ergibt sich ein Nettodurchsatz von 30 KByte/s, was sowohl TCP als auch SCTP in dem erwähnten Szenario im fehlerfreien Fall leicht erreichen. In den folgenden Experimenten wurde ange-

nommen, dass während einer signifikanten Zeitdauer bei der Übertragung 1% aller Pakete Übertragungsfehler erleiden (entspricht einer BER von etwa $1.2 \cdot 10^{-6}$).
In den nachfolgend erläuterten Untersuchungen werden die Zustellzeiten aller Nachrichten zwischen Sender und Empfänger betrachtet, die sich durch die Überlagerung mehrerer Effekte ergeben:

- Wartezeit in der Absender-Warteschlange,
- Verzögerung in der Zugangs-Warteschlange der genutzten Übertragungsstrecke,
- die Übertragungszeit (oder Emissionszeit) auf dieser Strecke,
- die Laufzeitverzögerung auf der Strecke,
- Wartezeit zur Wiederherstellung der korrekten Reihenfolge im Empfänger (bei verloren gegangenen Paketen).

Die ersten beiden Wartezeiten sind bei dem geringen Nachrichtenangebot von 30 KByte/s in diesem Szenario zu vernachlässigen.
Bei den nachfolgenden Abschätzungen (Abschnitte 4.3.1 - 4.3.3) wird davon ausgegangen, dass neu übertragene Daten nicht erneut verloren gehen, was in der Praxis nicht immer gegeben ist. Hier kann es immer dann zu Mehrfachverlusten kommen, wenn Daten mehrfach über den fehlerhaften Pfad gesendet werden.

4.3.1 Abschätzung des TCP-Protokollverhaltens

Bei einer Link-Verzögerung D' auf dem primären (Satelliten-)Pfad und einem mittleren Ankunftsabstand von Δt_{msg} der gesendeten Nachrichten beim Sender gilt für die maximale Übertragungszeit $t_{lost,TCP}$ einer verlorenen Nachricht bei Neuübertragung über das TCP:

$$t_{lost,TCP} \approx 3 \cdot D' + 4 \cdot \Delta t_{msg} + 2 \cdot t_{et'} \approx 893 \text{ ms} \tag{4.1}$$

wobei $t_{et'} \leq 5$ ms die Emissionszeit der übertragenen Nachrichten auf dem Satelliten-Link bezeichnet.

4.3.2 Abschätzung des SCTP Standard-Algorithmus

Der beim SCTP zusätzlich zur Verfügung stehende Pfad weist nur eine geringe Bandbreite auf. Seine Nutzung im Falle einer schnellen Neuübertragung (Fast Retransmission, FR) durch Übertragungsfehler auf den ersten Pfad kann jedoch sehr vorteilhaft sein, da er eine wesentlich geringere Verzögerung aufweist. Da bei kleinen Verlustraten die FR ein seltenes Ereignis ist, wird nur wenig zusätzliche Bandbreite auf dem sekundären Pfad belegt.
Ausgehend von einer Link-Verzögerung D' auf dem primären Satellitenpfad, einer Verzögerung von D'' auf dem ISDN-basierten Pfad und einem mittleren Ankunftsabstand von Δt_{msg} der gesendeten Nachrichten beim Sender gilt für die maximale Übertragungszeit $t_{lost,SCTP}$ einer verlorenen Nachricht bei Neuübertragung:

$$t_{lost,SCTP} \approx 2 \cdot D' + D'' + 4 \cdot \Delta t_{msg} + t_{et'} + t_{et''} \approx 729 \text{ ms} \qquad (4.2)$$

wobei $t_{et''} \leq 66$ ms die Emissionszeit einer neu übertragenen Nachricht auf dem Sekundär-Link bezeichnet.
Diese Werte ergeben sich durch die Rundlaufzeit zwischen den Endpunkten über die Satellitenstrecke ($2 \cdot D'$) und der Zeit, die vergeht, bis 4 aufeinander folgende SACKs nach einem Paketverlust gesendet werden ($4 \cdot \Delta t_{msg}$). Hinzu kommt die Zeit, die die neue Übertragung der Nachricht auf dem Sekundärlink benötigt. Die Emissionszeiten der SACK-Chunks auf dem Primärlink werden hierbei vernachlässigt, da sie wesentlich kleiner sind als die der Nachrichten auf Primär- oder Sekundärlink.
Damit ergibt sich durch die zusätzliche Nutzung des Sekundärlinks beim SCTP eine Verkürzung der maximalen Nachrichtenverzögerung gegenüber TCP von

$$t_{lost,TCP} - t_{lost,SCTP} = \Delta t_{max,lost} = D' - D'' + t_{et'} - t_{et''} \approx 164 \text{ ms} \qquad (4.3)$$

Dies führt auch zu einer geringeren Wartezeit in der Empfänger-Warteschlange, da schneller die richtige Reihenfolge wiederhergestellt werden kann.

4.3.3 Abschätzung der Standard-Modifikation

Schließlich wurde in [46] eine SCTP-Modifikation untersucht, die nach dem Feststellen einer nicht in Sendereihenfolge zugestellten Nachricht alle SACK-Chunks mit Berichten über mögliche Verluste (d.h. Lücken bei den in Reihenfolge eingetroffenen und bestätigten Sequenznummern) über den Link mit der kürzesten Rundlaufzeit sendet.

Durch diese Modifikation ist es möglich, die Zustellzeit von Paketen relativ unabhängig von der Fehlerrate auf dem Primärpfad zu halten. Damit bietet sich das Verfahren insbesondere für Anwendungen an, in denen eine starke Variation der Zustellzeit kritisch ist.
Da die Bestätigungsmeldungen auch benutzt werden, um die Umlaufverzögerung pro Pfad zu bestimmen, ist bei der Modifikation darauf zu achten, dass nur diejenigen Meldungen für diese Berechnung benutzt werden, die keine Lücken melden und tatsächlich über den Primärpfad laufen.
Bei dieser Protokollmodifikation erfolgt nun eine maximale Verzögerung der Zustellung bei verloren gegangenen Nachrichten um die Zeit

$$t_{mod,lost} \approx D' + 2 \cdot D'' + 4 \cdot \Delta t_{mesg} + t_{et'} + t_{et''} \approx 504 \text{ ms} \tag{4.4}$$

mit $t_{et''}$ der Übertragungszeit über den sekundären ISDN-Link. Die Übertragungszeit der SACK-Chunks wird wieder vernachlässigt.
Im Vergleich zu TCP verringert sich die maximale Zustellverzögerung von Nachrichten um

$$\Delta t_{max,mod,tcp} = 2 \cdot D' - 2 \cdot D'' + t_{et'} - t_{et''} \approx 389 \text{ ms}. \tag{4.5}$$

Im Vergleich zum Standard-SCTP verringert sich die maximale Zustellverzögerung von Nachrichten bei diesem modifizierten SCTP-SACK Algorithmus um

$$\Delta t_{max,mod,sctp} = D' - D'' \approx 225 \text{ ms}. \tag{4.6}$$

4.3.4 Abschließende Bewertung

Die hier kurz dargestellten Ergebnisse wurden in [46] anhand einer Testbett-Umgebung verifiziert, und es konnte gezeigt werden, dass sich SCTP generell günstiger verhält als TCP, wenn verlustbehaftete Links mit stark unterschiedlichen Übertragungseigenschaften vorhanden sind. Die in Abschnitt 4.3.3 diskutierte Optimierung erlaubt darüber hinaus eine weitere Verbesserung des Protokollverhaltens bei fehlerbehafteten Links. Diese Ergebnisse sind nicht nur in dem erwähnten Szenario relevant, in dem eine Satellitenstrecke mit hoher Bandbreite aber auch hoher Linkverzögerung zur Verfügung steht, sondern spielen auch in Mobilitätsszenarien mit heterogenen, IP-basierten Funkzugangsnetzen, in denen Endgeräte gleichzeitig über mehrere IP-Adressen erreichbar sind, eine wesentliche Rolle.

4.4 Untersuchung des SCTP im Falle eines Failover

Da SCTP auch definiert wurde, um den Transport von SS7-Signalisierungsnachrichten in IP-Netzen zu ermöglichen, lag es nahe, zu untersuchen, wie Protokollparameter eingestellt werden müssen, damit die besonderen Anforderungen an Ausfallsicherheit und Redundanz in SS7-Netzen erfüllt werden.
Die ITU-T Empfehlung Q.706 [31] stellt sehr hohe Anforderungen an das Übertragungssystem des MTP in Bezug auf Verlustwahrscheinlichkeit, Reihenfolgesicherung und Verfügbarkeit. Insbesondere spezifiziert sie auch die Anforderungen an eine sog. Changeover-Prozedur, die bei Ausfall eines MTP2-Links durch den MTP3 ausgeführt wird. MTP2-Links werden aus Gründen der Ausfallsicherheit immer in redundanten, teilweise physikalisch getrennten Paaren implementiert. So kann der MTP3 bei Ausfall eines Links alle Nachrichten über den noch aktiven MTP2-Link aussenden. Nach [31] muss diese Changeover-Prozedur nach spätestens 800 ms abgeschlossen sein. Bei Nutzung des M2PA (vgl. auch [28]) ersetzen eine oder mehrere SCTP-Assoziationen zusammen mit entsprechenden M2PA-Layern die MTP2-Links. Dabei muss sichergestellt werden, dass die oben erwähnten Anforderungen bezüglich des Changeover erfüllt werden können.
Das Changeover-Verhalten einer Anwendung, die Teilfunktionen des MTP3 modelliert, wurde untersucht. Diese sendet zunächst Daten über einen primären Breitband-Signalisierlink bzw. Pfad. Fällt dieser Pfad aus, so führt die Anwendung einen Changeover durch und überträgt zunächst die verbleibenden Nachrichten aus der Warteschlange des ausgefallenen Pfads und danach alle neuen Nachrichten über den sekundären Link bzw. Pfad. Es wurde das Verhalten zweier möglicher Anordnungen untersucht, denen beiden eine identische Netztopologie und physikalische Versuchsanordnung (vgl. auch Abschnitt 4.1) zugrunde liegt:

1. zwei SCTP-Assoziationen (jeweils single-homed), entsprechend zwei MTP2-Links mit je einem SCTP-Pfad, und

2. eine SCTP-Assoziation (dual-homed), entsprechend einem Link mit zwei SCTP-Pfaden.

In dem betrachteten Netzaufbau wurde angenommen, dass Pakete bei der Übertragung zwischen den beiden Endpunkten eine Verzögerung von je 10 ms erfahren. Die Verzögerung wurde so gewählt, um ein Szenario zu untersuchen, in dem ein auf IP/SCTP/M2PA basierender Signalisierendpunkt nicht weit vom

zugehörigen Signalisierungs-Gateway entfernt ist (z.B. weniger als 100 km). In diesem Fall ist die gewählte Nachrichtenverzögerung angemessen. Die Bandbreite zwischen den beiden Endpunkten war auf 2,048 MBit/s beschränkt, d.h. es wurde die Übertragung auf einem E1-Link emuliert. Eine Quelle erzeugte Nachrichten, deren Ankunftsabstände einer negativen Exponentialverteilung mit einer mittleren Ankunftsrate von 100 Nachrichten pro Sekunde gehorchten. Alle Nachrichten hatten eine konstante Länge von 500 Bytes, womit sich (incl. Overhead) eine mittlere Netzlast von ca. 22% ergab.
Das Verhalten des SCTP beim Pfadausfall ist im Wesentlichen abhängig von den folgenden Protokollparametern des SCTP (vgl. auch Abschnitt 2.3.7.1):

- RTO_{max}, der maximale Wert, den ein T3-Timer (für Neuübertragungen) annehmen kann. Spätestens nach Ablauf dieser Zeit findet eine timer-basierte Neuübertragung statt, wenn für einen bereits gesendeten Daten-Chunk kein SACK empfangen wurde.

- RTO_{min}, der minimale Wert, den ein T3-Timer (für Neuübertragungen) annehmen kann. Frühestens nach Ablauf dieses Timers kann eine timer-basierte Neuübertragung erstmals stattfinden.

- RTO_{init}, der Startwert für den T3-Timer Timeout.

- dem Schwellwert für Fehler auf einem Pfad (Path Retransmission Limit, PRL).

- dem Schwellwert für Fehler pro Assoziation (Association Retransmission Limit, ARL).

Die beiden untersuchten Anordnungen wurden bei verschiedenen Einstellungen dieser Parameter miteinander verglichen. Dabei wurden insbesondere die Werte für PRL/ARL von 2 bis 5 und die Werte von RTO_{max} von 100 ms bis 250 ms variiert, also in dem für den SS7 Signalisierungstransport interessanten Wertebereich. Für kleinere Werte dieser Parameter ergab sich häufig ein instabiles Verhalten; sie wurden daher nicht weiter berücksichtigt.
Neben der maximalen und der mittleren Nachrichtenverzögerung während der Changeover-Prozedur wurde auch noch die Gesamtdauer des Changeovers bestimmt, da diese als Kriterium für die Eignung als Transportmedium der SS7-Signalisierung entscheidend ist. Es wurde angenommen, dass die jeweilige Changeover-Prozedur beendet war, nachdem die Nachrichtenverzögerung nach

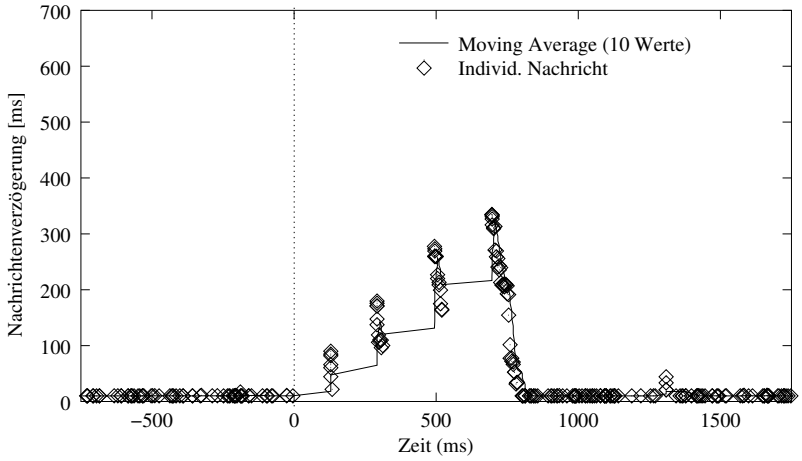

Abbildung 4.5: Nachrichtenverzögerung bei erfolgreicher Neuübertragung über Sekundärpfad

dem simulierten Pfadausfall wieder den Wert angenommen hatte, den sie vor diesem Ereignis aufwies.
Für jeden Parameter-Punkt wurden 25 Messungen durchgeführt, deren Ausgaben nachfolgend mittels geeigneter Skripte so ausgewertet wurden, dass die im Folgenden präsentierten Ergebnisse ein Konfidenzintervall von 99% zeigen. Aus den Messungen wurde die individuelle Nachrichtenverzögerung (nach einer Synchronisierung zwischen Sender und Empfänger) mit einer Genauigkeit von ca. 1 ms ermittelt.

4.4.1 Pfadausfall bei einer Assoziation mit zwei Pfaden

Fällt der Primärpfad einer Assoziation mit zwei Pfaden aus, so werden die ältesten Daten-Chunks nach Ablauf des Retransmission Timers (RTO) über den Sekundärpfad (erfolgreich) neu übertragen und bestätigt. Alle von der Anwendung neu gesendeten Nachrichten werden zunächst wieder (erfolglos) über den Primärpfad übertragen und können frühestens beim nächsten Ablauf des Retransmission Timers erfolgreich über den Sekundärpfad gesendet werden. Die neuen Nachrichten verbleiben also in der Sendewarteschlange, bis schließlich der bei jeder Neuübertragung inkrementierte Fehlerzähler des Pfades den Schwellwert PRL erreicht. Dann wird der Anwendung signalisiert, dass der assoziierte

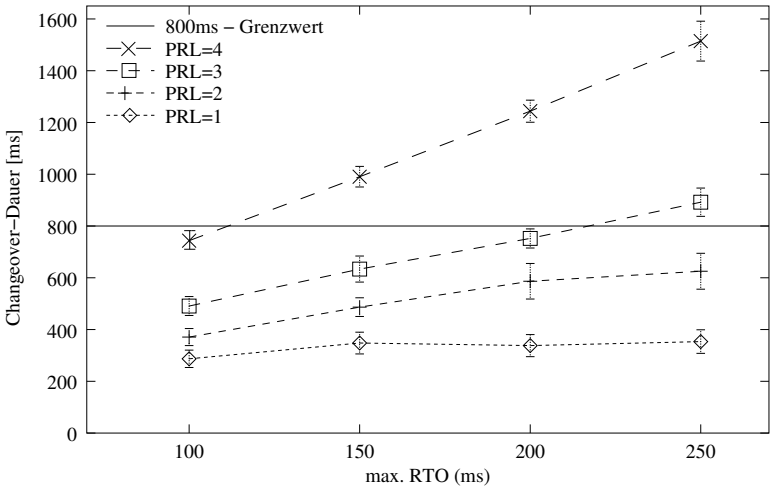

Abbildung 4.6: Dauer der Failover-Prozedur (Assoziation mit zwei Pfaden)

Endpunkt nicht mehr über den Primärpfad erreicht werden kann. Die Anwendung wird in der Regel darauf mit einer Änderung des Primärpfads reagieren (vgl. SETPRIMARY im Abschnitt A.1) und den bisherigen Sekundärpfad zum neuen Primärpfad machen. Diese Aktion wird nach einer Zeit t_{fail} durchgeführt, die mit

$$\mathrm{PRL} \cdot \mathrm{RTO}_{min} \leq \mathrm{t}_{fail} \leq \mathrm{PRL} \cdot \mathrm{RTO}_{max} \tag{4.7}$$

abgeschätzt werden kann.
Die Changeover-Prozedur wird frühestens nach der Zeit t_{fail} eingeleitet und ist beendet, wenn alle in der Warteschlange befindlichen alten Daten-Chunks über den neuen Pfad gesendet wurden und die durchschnittliche Wartezeit in der Sendewarteschlange wieder den ursprünglichen Wert erreicht (bei identischen Charakteristika der beiden Pfade).
Die maximale Nachrichtenverzögerung erleiden Nachrichten, die beim letzten Ablauf des Neuübertragungs-Timers übertragen werden (siehe auch Abbildung 4.5).
Die folgenden Abbildungen 4.6 und 4.7 zeigen die Ergebnisse der Untersuchungen [47]. Sie dokumentieren das Verhalten des SCTP beim Pfadausfall in der Konfiguration mit zwei Pfaden für verschiedene Werte des Parameters PRL und RTO_{max} und geben die im Experiment gemessene maximale Verzögerung von

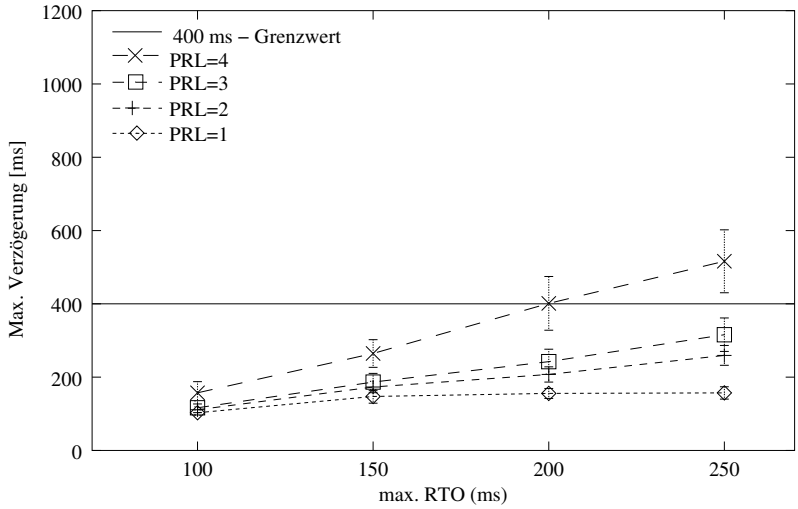

Abbildung 4.7: Maximale Nachrichtenverzögerung (Assoziation mit zwei Pfaden)

Nachrichten und die Dauer der Changeover-Prozedur wieder. Eine abschließende Bewertung findet sich in Abschnitt 4.4.3.

4.4.2 Pfadausfall bei zwei Assoziationen mit einem Pfad

Ab dem Zeitpunkt des Pfadausfalls werden alle Daten-Chunks innerhalb der ersten Assoziation über den ausgefallenen Pfad neu übertragen. Da alle diese Neuübertragungen erfolglos sind, verbleiben alle Daten zunächst in der Warteschlange. Bemerkt die Anwendung den Pfadausfall, so wird gleichzeitig die zugehörige Assoziation beendet, da deren einziger Pfad nicht mehr verfügbar ist. Die nicht erfolgreich übertragenen Daten werden von der Anwendung zurückgeholt und über die zweite Assoziation übertragen. Den typischen Verlauf der Nachrichtenverzögerung über der Zeit bei diesem Verfahren zeigt Abbildung 4.8.
Abbildung 4.9 gibt anschaulich für die die entsprechenden Parameter PRL/ARL$=1\ldots4$ und $\text{RTO}_{min} = 40$ ms die maximale Verzögerung der übertragenen Nachrichten in Abhängigkeit vom Parameter RTO_{max} wieder. Die ersten erfolgreich übertragenen Daten können in diesem Fall erst nach dem Ausfall der Pfades bzw. der Assoziation durchgeführt werden. Die maximale Nachrichtenverzögerung

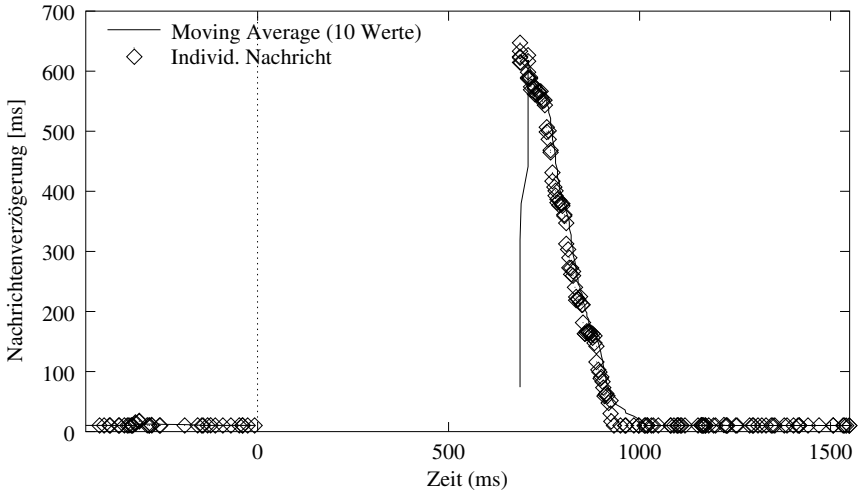

Abbildung 4.8: Verlauf der Nachrichtenverzögerung (zwei Assoziationen mit einem Pfad)

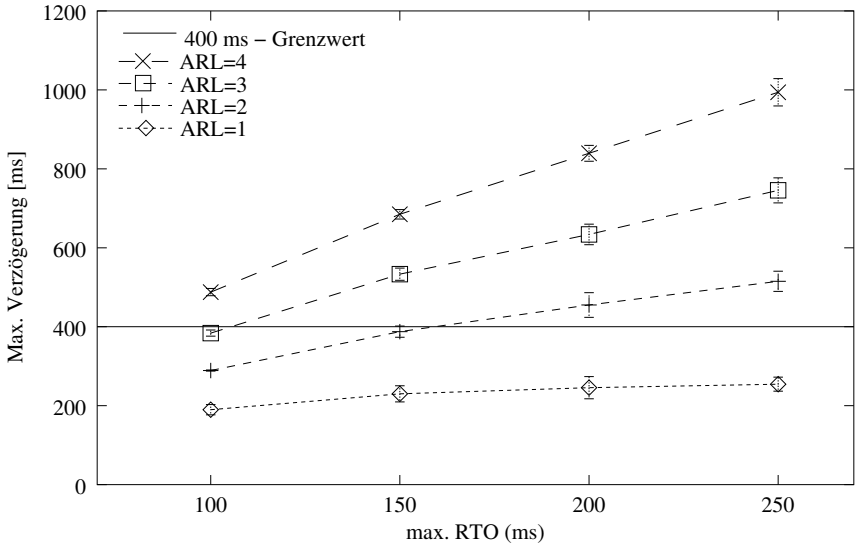

Abbildung 4.9: Maximale Nachrichtenverzögerung (zwei Assoziationen mit einem Pfad)

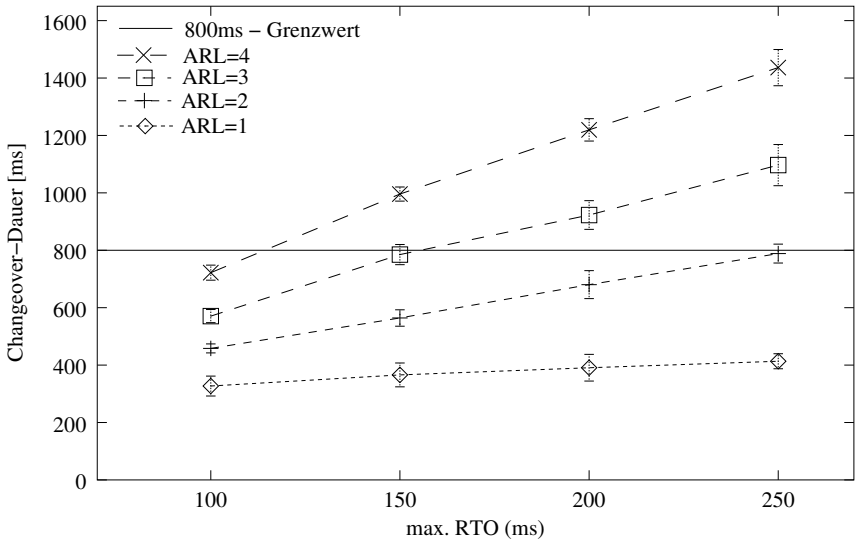

Abbildung 4.10: Dauer der Failover-Prozedur (zwei Assoziationen mit einem Pfad)

$t_{delay,max}$, die die ältesten Nachrichten erleiden, kann abgeschätzt werden (unter der Annahme, dass die Nachrichten auf dem sekundären Link eine Verzögerung von $t_{delay,2}$ erleiden):

$$\mathrm{ARL} \cdot \mathrm{RTO}_{min} + t_{delay,2} \leq t_{delay,max} \leq \mathrm{ARL} \cdot \mathrm{RTO}_{max} + t_{delay,2} \tag{4.8}$$

Abbildung 4.10 zeigt die resultierenden Zeiten für die Dauer der gesamten Changeover-Prozedur.

4.4.3 Bewertung der Ergebnisse

Die in den vorigen Abschnitten vorgestellten Ergebnisse dokumentieren, dass SCTP in einer Konfiguration mit zwei Pfaden in der Regel zu einem günstigeren Verhalten (geringere durchschnittliche und maximale Nachrichtenverzögerung) im Falle eines Pfadausfalls führt, als eine Konfiguration mit zwei Assoziationen, die single-homed betrieben werden [47]. Dies ist um so bedeutsamer, als beide Konfigurationen mögliche Realisierungen eines MTP2-Linksets darstellen können.

In den gegebenen untersuchten Szenarien war über einen weitreichenden Parameterraum ($\text{RTO}_{max} \leq 200$ ms und $\text{PRL} \leq 3$) die Gesamtdauer des Changeovers für die Konfiguration einer SCTP-Assoziation mit zwei Pfaden geringer als 800 ms und erfüllt damit die Anforderungen an Changeover-Prozeduren nach [31]. In einem kontrollierten Netz, in dem zusätzliche Nachrichtenverzögerungen durch Überlast und nachfolgende Neuübertragungen ausgeschlossen werden können, genügt also SCTP in dieser Konfiguration den Anforderungen an den Transport von MTP2-Primitiven.
In der zweiten untersuchten Konfiguration müssen die Parameter restriktiver gewählt werden ($\text{RTO}_{max} \leq 200$ ms und $\text{PRL} \leq 2$), damit die entsprechenden Anforderungen erfüllt werden.

Kapitel 5

Modellierung und Bewertung des SCTP

In diesem Kapitel werden zunächst die Randbedingungen für die Implementierung eines zur Simulation geeigneten Modells des SCTP erläutert. Es folgt eine Darstellung der Struktur und der Implementierung des Modells, bevor in den Abschnitten 5.5ff ausgewählte Simulationsergebnisse vorgestellt werden. Das im Laufe dieser Arbeit entstandene Modell wurde im Quellcode von einer Forschungsgruppe eines international orientierten europäischen Netzbetreibers lizensiert und wird dort zur Bewertung des SCTP-Protokollverhaltens eingesetzt.

5.1 Zielsetzung der Protokollmodellierung

Ein wesentlicher Bestandteil dieser Arbeit sind Untersuchungen des Protokollverhaltens des SCTP, die an einem Simulationsmodell durchgeführt wurden. Der Schwerpunkt dieser Untersuchungen liegt auf dem Verhalten des SCTP-Datenpfads, sowie den Mechanismen für die Erkennung der Pfadausfälle.
Modelle von Kommunikationssystemen sind immer zweckgebunden und müssen die für die Untersuchungen relevanten Aspekte der Systeme korrekt widerspiegeln. Ein weiterer wichtiger Punkt ist die Möglichkeit, auf Parameter eines Systems Einfluss zu nehmen, um das Systemverhalten geeignet quantitativ analysieren zu können. Diese Eigenschaften sollten sowohl Modelle, als auch die zugehörigen Umgebungen und Werkzeuge, in denen diese Modelle untersucht werden, unterstützen. Vor Beginn der Untersuchungen wurden daher die verfügbaren Simulationswerkzeuge untersucht (vgl. Abschnitt 5.2) und auf ihre

Tauglichkeit für die Simulation des SCTP hin überprüft.
Ferner wurden die notwendigen Elemente des Protokolls sowie Datenformate festgelegt, die zur Simulation der relevanten Teile des Protokollverhaltens notwendig sind, wobei auf die bereits durchgeführte funktionale Spezifikation zurückgegriffen wurde, die in Abschnitt 3.1.2 erläutert wurde.
Da die Untersuchungen auf das Verhalten des Datenpfades und die Mechanismen der Ausfallerkennung beschränkt sein sollen, wird der SCTP-Controller (vgl. auch Abbildung 3.2) nicht in der Simulation implementiert, und die unterstützten Datenelemente sind neben dem SCTP-Paket generell SCTP-Chunks und im speziellen die folgenden Chunks:

- Daten-Chunks,
- SACK-Chunks,
- HEARTBEAT-Chunks und
- HEARTBEAT-ACK-Chunks.

Dies bedeutet, dass der komplexe Assoziationsauf- und abbau nicht modelliert wird, und Assoziationen direkt den Zustand *Established* einnehmen. Jedoch muss eine Möglichkeit bestehen, den Ausfall einer Assoziation zu modellieren, der gegebenenfalls durch den Ausfall aller Pfade oder die Überschreitung der Schwellwerte für Neuübertragungen auftreten kann.
Ferner soll in der Simulation das Multihoming unterstützt werden und ein realistisches Modell des IP-Routings zur Verfügung stehen, so dass insbesondere die Effekte untersucht werden können, die SCTP von TCP und UDP unterscheiden. Des Weiteren wird angenommen, dass die Berechnung und Prüfung der SCTP Prüfsummen gegenüber Laufzeiten und Warteschlangenverzögerungen zu vernachlässigen sind und dass im Netz Paketverluste, Verzögerungen und Paketüberholungen auftreten können.

5.2 Die ereignisorientierte Simulation

Bei der ereignisorientierten Simulation wird der Zustand eines Systems durch die Menge aller Variablen beschrieben, die ein System zu einem gegebenen Zeitpunkt definieren. Die Repräsentation durch ein Simulationssystem erfolgt typisch durch eine Menge von Datenstrukturen, die sich in Abhängigkeit von der Simulationszeit ändern (vgl. auch [59]). Ereignisse werden definiert als Zeitpunkte, zu denen

sich der Zustand des Systems ändert, d.h. bei dem Systemvariablen geändert werden. Die Simulationszeit ist dabei ein für die Simulation globaler Parameter. Die Simulation beruht auf der Modellierung der korrekten zeitlichen Reihenfolge aller Änderungen des Systemzustands, d.h. aller für ein System relevanten Änderungen der Systemvariablen und der dazugehörigen Zeitpunkte. Bei der ereignisorientierten Simulation wird angenommen, dass alle Systemvariablen zwischen diskreten Ereignissen konstant bleiben.
Ein Spezialfall der ereignisorientierten Simulation ist die prozessorientierte Simulation, die in einer Vielzahl von Simulationswerkzeugen zum Einsatz kommt (vgl. auch Abschnitt 5.2.2): dabei durchlaufen Entitäten (z.B. Datenstrukturen, die Nachrichten modellieren) ein Netz von Stationen, die durch einen oder mehrere nebenläufige Prozessmodelle beschrieben werden. Beim Durchlauf durch diese Stationen (z.B. Schichten eines Protokolls) können sowohl die Entitäten als auch die Prozesse ihre Attribute ändern. Daher ist die prozessorientierte Simulation besonders geeignet für die Simulation von Kommunikationsprotokollen, bei denen z.B. bereits beim Senden eines Datagramms aufgrund der Übertragungsrate des zugehörigen physikalischen Mediums und des Abstands zwischen Sender und Empfänger der Ankunftszeitpunkt eines Datagramms berechnet werden kann.
Die Menge der Systemereignisse wird typisch in einer Liste verwaltet, die nach der zugehörigen Simulationszeit der Ereignisse geordnet ist. Das Simulationswerkzeug übernimmt die Verwaltung der Ereignisliste und führt typischerweise bei jedem Ereignis einen zugehörigen Prozess aus. Das Systemverhalten wird also repräsentiert durch eine Abfolge von Prozessen, die die Systemvariablen beeinflussen und denen jeweils ein Ausführzeitpunkt bezogen auf die globale Simulationszeit zugeordnet ist. Die Prozesse werden aktiviert, wenn die globale Simulationszeit den dem Prozess zugeordneten Ausführzeitpunkt erreicht und laufen bis zu ihrem Ende ab, ohne dass die Simulationszeit fortschreitet.
Darüber hinaus definiert das Simulationswerkzeug eine Schnittstelle für Prozesse, mit der diese die Ereignisliste manipulieren können. Typische Funktionen sind z.B. das Hinzufügen oder Löschen eines Ereignisses zu einem bestimmten Zeitpunkt (entspricht im Wesentlichen dem Starten oder Stoppen eines Timers bei einem Protokoll).
Simulationswerkzeuge unterstützen unterschiedliche Varianten der Interprozesskommunikation, die entscheidend für die Simulation von Kommunikationsprotokollen sind. Dazu gehören z.B. das Senden von Nachrichten an andere Prozesse, die sofort oder zu einem späteren Zeitpunkt bei diesem Prozess eintreffen und ein neues Ereignis darstellen.
Ferner müssen auch Schnittstellen zur Ausgabe relevanter, ereignisbezogener Da-

ten angeboten werden. Dies dient dem Verfolgen des Systemverhaltens über der Zeit (transientes Verhalten) oder dem Festhalten statistisch interessanter Daten für eine spätere Auswertung.

5.2.1 Statistische Auswertung von Simulationsergebnissen

Ausgaben einer ereignisorientierten Simulation sind typisch entweder Daten, die das Verhalten des simulierten Systems in Abhängigkeit der Simulationszeit (also transient) widerspiegeln, oder eindimensionale Werte, die das Verhalten des Systems in Abhängigkeit von bestimmten Systemparametern mit einer einzigen Kenngröße beschreiben. Auf das SCTP bezogen sind solche Systemparameter z.B. die Anzahl der durchgeführten Neuübertragungen über einen gegebenen Pfad, bevor dieser den Zustand *unerreichbar* annimmt (vgl. Parameter PRL in 4.4), oder die maximale RTO (vgl. auch RTO_{max} in Abschnitt 2.3.7.1).
Die Ergebnisse einer Simulation erlauben die Untersuchung eines bestimmten Aspektes des Systems in Abhängigkeit von gewählten Parametern und in Abhängigkeit von einer bestimmten Systemtopologie. Dazu werden in diesen Simulationsläufen typische Leistungsmaße eines Systems bestimmt, wie z.B. der Durchsatz eines Übertragungsprotokolls, Last auf einem Link oder Warte- und Übertragungszeiten von Nachrichten. Es wird davon ausgegangen, dass die Kennzahlen für die Leistungsmaße aus den einzelnen Simulationsläufen unabhängige Stichproben aus einer Grundgesamtheit darstellen, die durch eine gleiche Verteilungsdichtefunktion gekennzeichnet ist.
Aussagen über Parameter einer Stichprobe, wie z.B. über Erwartungswerte von Leistungsmaßen aus mehreren Simulationsläufen, werden durch Stichprobenfunktionen getroffen, deren Funktionswerte Schätzungen für die Parameter und somit selbst Zufallsvariablen darstellen. Als Schätzwert für den Erwartungswert einer Grundgesamtheit bietet sich beispielsweise das empirische Mittel einer (repräsentativen) Stichprobe aus dieser Grundgesamtheit an.
Die Angabe solcher Stichprobenparameter ist aber nur aussagekräftig, wenn auch gleichzeitig Aussagen über ihre Genauigkeit bzw. ihre Sicherheit gemacht werden. Gemäß [12] gilt bei der Konfidenzschätzung für das arithmetische Mittel μ einer Stichprobe bei unbekannter Standardabweichung σ aus einer nach $N(\mu, \sigma)$ normalverteilten Grundgesamtheit, dass die Größe

$$\frac{\bar{X} - \mu}{S}\sqrt{n} \tag{5.1}$$

einer t-Verteilung mit $n - 1$ Freiheitsgraden genügt. Dabei ist n der Stichprobe-

numfang, $\bar{X}$ der Erwartungswert (also der tatsächliche Mittelwert) der Zufallsvariablen X in der Grundgesamtheit, μ das arithmetische Mittel aus der Stichprobe, und S die Standardabweichung von X in der Stichprobe.
Es gilt dann für das Konfidenzniveau $p = 1 - \alpha$, bzw. für die vorgegebene Irrtumswahrscheinlichkeit α:

$$P\left(\bar{X} - \frac{S}{\sqrt{n}} t_{\alpha,n-1} < \mu < \bar{X} + \frac{S}{\sqrt{n}} t_{\alpha,n-1}\right) = p = 1 - \alpha \tag{5.2}$$

Typische Werte für α sind z.B. $\alpha = 0,05$, $\alpha = 0,02$ oder $\alpha = 0,01$. Die korrespondierenden Werte $t_{\alpha,n-1}$ der t-Verteilung des Freiheitsgrades $n - 1$ und der Irrtumswahrscheinlichkeit α können dann aus einer Tabelle entnommen werden (vgl. auch Abschnitt 1.1.2.11 von [12]). Mittels dieser Formel kann also ein Intervall angegeben werden, in dem der tatsächliche Erwartungswert einer Zufallsvariablen mit einer gewissen Sicherheit liegt (z.B. “Mit einer Wahrscheinlichkeit von 99 % liegt der mittlere Durchsatz einer bestimmten Konfiguration eines Übertragungssystems in dem Intervall 1,2 MBit/s bis 1,4 MBit/s”).
Nach Analyse der Auswirkung von Parametereinstellungen auf die relevanten Leistungsmaße eines Systems kann der Anwender bei der Wahl geeigneter Systemparameter unterstützt und ein System in Hinblick auf bestimmte Leistungsmaße optimiert werden. Typische Leistungsmaße bei Kommunikationsnetzen sind z.B. Wartezeit, Verweilzeit, Antwortzeit, Dienstabwicklungsdauer, Warteschlangenlänge, Ressourcenauslastung und Durchsatz [59].

5.2.2 Simulationswerkzeuge

Im Bereich der ereignisorientierten Simulation von Kommunikationsprotokollen und Netztopologien gibt es eine ganze Reihe von Simulationswerkzeugen, die mit unterschiedlichen Schwerpunkten konzipiert wurden. Im Folgenden werden die relevanten Eigenschaften der wichtigsten dieser Programme kurz erläutert. Zu diesen Eigenschaften zählen u.a.

- die Art der Simulation (z.B. ereignisorientiert, hybride Mischung aus ereignisorientierter Simulation und stochastischen Methoden und Analyse von realen Traces),
- Möglichkeit zur Spezifikation von komplexen und hierarchischen Topologien,
- Unterstützung einfacher und komplexer Schichtmodelle,

- Einfache Spezifikation neuer Protokollschichten und unterstützte Spezifikationssprachen,
- Möglichkeiten zur Analyse der Simulationsergebnisse,
- mitgelieferte und erweiterbare Modellbibliotheken und Quellenmodelle,
- Skalierbarkeit und
- unterstützte Rechner-Plattformen, Lizensierung.

5.2.2.1 BONeS

BONeS (für Block Oriented Network Simulator) ist eine Simulationsumgebung mit grafischer Benutzeroberfläche, die es erlaubt, Systeme modular zu spezifizieren. Die Module bestehen entweder aus endlichen Zustandsautomaten, aus in C oder C++ programmierten Funktionen oder aus hierarchisch angeordneten Blockdiagrammen. Blöcke können auf Eingaben reagieren und Ausgaben erzeugen. Die oben erwähnten Grundbausteine (C/C++ Funktionen, endliche Zustandsautomaten und hierarchische Blockdiagramme) können zu weiteren Blöcken zusammengesetzt werden, was prinzipiell eine beliebige Schachtelungstiefe und damit eine hohe Abstraktion von Funktionalitäten erlaubt. Innerhalb eines zusammengesetzten Moduls werden die Blöcke, die die Bausteine des Moduls darstellen, als Signalfluss-Blockdiagramme miteinander verbunden. Über diese Verbindungen werden Datenstrukturen übertragen, die z.B. Netzwerkpakete modellieren können.
Durch die generische Struktur von Modellen können so verschiedenste Anwendungen und Datenübertragungssysteme (wie Protokolle, Prozessorsysteme und verteilte Rechnerarchitekturen) modelliert werden. Die Entwicklungsumgebung von BONeS bringt neben generischen Quellen- und Link-Modellen auch Modelle für Standard-Protokolle wie TCP mit.
BONeS wurde im Rahmen der Kooperation mit Siemens benutzt, um anfänglich das Verhalten des Protokolls MDTP [84], dem Vorläufer des SCTP, im Vergleich zu TCP zu untersuchen. Dabei wurden Unzulänglichkeiten bei den Flusskontroll-Algorithmen festgestellt und als Ergebnis in die IETF-Standardisierung eingebracht. In Konsequenz führte dies dazu, dass SCTP als Nachfolger des MDTP einen dem TCP ähnlichen Flusskontroll-Algorithmus erhalten hat. Seit Anfang 2000 ist BONeS nicht mehr verfügbar.

5.2.2.2 Network Simulator 2 (NS-2)

Der Network Simulator Version 2 (NS-2) ist ein ereignisorientierter Simulator, der von der Universität Berkeley entwickelt wurde, um insbesondere Varianten von IP-basierten Netzen zu untersuchen. Simulationen werden in NS-2 durch Interpretation von oTcl-Skripten [92] ausgeführt, in denen Netzwerktopologien und Anwendungsverhalten beschrieben werden. Der eigentliche Kern des Simulators besteht aus Modellen, deren Verhalten in C++ beschrieben wird.
Wird in einem oTcl-Skript ein Objekt instantiiert, so führt der Simulator entsprechende C++ Funktionen aus, durch die ein solches Objekt im Speicher angelegt wird. Ferner bieten oTcl-Prozeduren die Möglichkeit, eine Simulation flexibel zu steuern, indem Variablen und Schleifen zur Simulation verschiedener Parameter benutzt werden, Anwendungen gestartet oder gestoppt werden, Links ausfallen, oder zu überwachende Statistiken angegeben werden.
Interessant bei NS-2 ist insbesondere der objektorientierte Ansatz, der es einfach erlaubt, von einem Modell abgeleitete Klassen zu erzeugen, die Varianten des zugrunde liegenden Modells sind. In diesen Varianten können einzelne Funktionen auf unterschiedliche, aber ähnliche Weise implementiert (überladen) sein, was eine einfache Analyse von Modellvarianten erlaubt.
NS-2 stellt eine umfangreiche Modellbibliothek zur Verfügung, in der überwiegend IP-basierte Modelle, aber auch Modelle für drahtlose Funkzugangsnetze zu finden sind. Dazu gehören z.B. IP und verschiedene TCP-Varianten, UDP, RTP, RTCP, Routing-Protokolle, verschiedene Queueing-Mechanismen in Routern (FIFO, RED, WFQ, CBQ, SFQ, DRR), MPLS, und Multicast-Protokolle. Auch Anwendungsprotokolle wie HTTP, FTP, sowie Quellenmodelle, die CBR-Verkehr über UDP erzeugen, sind implementiert.
Die Elemente zu untersuchender Modelle sind Links und Knoten, deren Topologie in einer Textdatei als oTcl-Skript beschrieben wird. Für die Erzeugung sehr komplexer Topologien existieren Tools, die die entsprechenden Skripte generieren.
Die Implementierung und Simulation eines neuen Protokolls in NS-2 geschieht typisch in mehreren Schritten: zunächst wird die Protokollimplementierung in Form von C++ Klassen und oTcl-Funktionen zur Code-Basis des Simulators hinzugefügt. Knoten, die das das entsprechende Protokoll einsetzen, werden in einem Simulationsskript instantiiert und die Simulation ausgeführt. Abschließend können die ausgegebenen Ergebnisse analysiert werden.
Die Ergebnisse gibt NS-2 u.a. in Form eines sog. Tracefiles aus, in dem alle Ereignisse einer Simulation festgehalten werden. Aus diesem Tracefile erzeugt das

dazu gehörige Programm NAM (Network Animator) auf Wunsch eine animierte Darstellung des Paketflusses, in der auch Parameter wie die Längen von Warteschlangen in Routern angezeigt werden können.
Von allen hier vorgestellten Simulationstools bietet NS-2 aufgrund des hohen Speicherbedarfs während der Simulation die schlechteste Skalierbarkeit. Die internen Schnittstellen, auf die zurückgegriffen werden muss, um neue Modelle hinzuzufügen, sind komplex und unzulänglich dokumentiert. Eine Nachbearbeitung der Ergebnisse erfolgt durch separate Statistikprogramme, die nicht zur Distribution von NS-2 gehören.
Im Hinblick auf die Modellierung des SCTP ist anzumerken, dass es zumindest schwierig ist, einen Knoten auf einfache Weise so zu modellieren, dass er das IP-Multihoming mit entsprechendem IP-Routing unterstützt.
Schließlich ist NS-2 ein Programm, das als Open Source der Allgemeinheit als Quellcode zur Verfügung steht und daher von sehr vielen Forschungsgruppen weltweit benutzt wird. Dies trug zur erheblichen Verbreitung und Akzeptanz dieses Simulators bei.

5.2.2.3 QualNet/GloMoSim

QualNet basiert auf dem für Universitäten frei erhältlichen Simulator GloMoSim, der an der Universität von Kalifornien (UCLA) entwickelt wurde. Der Fokus dieser mittlerweile nicht mehr gepflegten Simulationsbibliothek war es, skalierbare Netzwerkmodelle drahtloser Netze zu schaffen, die parallel auf heterogenen Architekturen simuliert werden können. Die Sprache, die zur Implementierung von Modellen benutzt wird, ist C-basiert und nennt sich PARSEC. Ihr Hauptmerkmal ist die funktionale Trennung der Modellbeschreibung vom Simulationsprotokoll, damit sowohl sequenziell als auch parallel simuliert werden kann. Aufgrund dieser strikten Trennung sind Simulationen auf Basis von GloMoSim/QualNet hochgradig skalierbar, d.h. es können Systeme in der Größenordnung mehrerer tausend Knoten simuliert werden. Protokolle sind in Schichten implementiert, was es bei Einhaltung der Schnittstellen zwischen den Schichten ähnlich wie bei OSI einfach erlaubt, einzelne Schichten durch Modifikationen zu ersetzen, ohne andere Schichten ändern zu müssen. GloMoSim kommt zwar nahezu ohne Dokumentation, aber es gibt eine ganze Reihe von Protokollen, die implementiert wurden:

- Anwendungsprotokolle wie Telnet, FTP, HTTP und CBR-Quellen,
- die Transportprotokolle TCP in den Varianten FreeBSD und Tahoe, sowie UDP,

- Routing-Protokolle, insbesondere auch für Wireless/AdHoc-Protokolle (AODV, DSR, OSPF, LAR, u.a.),
- Layer2-Protokollen wie CSMA und Wireless LAN (IEEE 802.11).

Darüber hinaus gibt es eine Reihe von Mobilitäts-Modellen sowie Modelle für die Ausbreitung elektromagnetischer Wellen.
QualNet, das kommerziell vertrieben wird und auf GloMoSim basiert, ist eine Weiterentwicklung, welche die meisten der Schwächen von GloMoSim behebt: so steht neben ausführlicher Dokumentation kommerzieller Support zur Verfügung, und QualNet liefert nahezu alle relevanten Protokollmodelle in einer Standard-Bibliothek mit dem Simulator. Diese Modelle können durch Parameter konfiguriert werden, ihr Verhalten aber nicht grundsätzlich geändert werden.
Die QualNet Developer Suite besteht aus fünf Modulen:

1. Animator, ein Tool zur Erstellung von Netz- und Protokoll-Topologien und zur Animation von Simulationen.
2. Designer, ein Programm zur Erstellung von Modellen, die auf endlichen Automaten basieren.
3. Analyzer, ein Programm zur Analyse von Statistik-Werten und Simulationsausgaben.
4. Tracer, ein Programm zur Analyse des Paketflusses (auch beim Durchlauf durch die einzelnen Protokollschichten) und
5. Simulator, der eigentliche Simulations-Kern, der Ereignisse verwaltet.

QualNet wird nach NS-2 von den meisten Forschungsgruppen weltweit eingesetzt [58] und gilt somit als gut akzeptiertes Werkzeug. Es liefert umfangreiche Modellbibliotheken, ist hochgradig skalierbar und gut dokumentiert. Die Nutzung ist aber grundsätzlich kostenpflichtig (wenn man vom einfacheren GloMoSim absieht).

5.2.2.4 OMNeT++

OMNeT++ (Objective Modular Network Testbed in C++) ist ein objektorientierter, modularer, ereignisorientierter Simulator, mit dessen Entwicklung 1998 begonnen wurde. Er bietet die Möglichkeit, das Verhalten von Modulen in der Sprache C++ zu beschreiben, und liefert grafische Tools zur Erstellung von Topologien

und zur Animation von Simulationen. OMNeT++ kann zur Simulation sämtlicher Systeme eingesetzt werden, die Nachrichten austauschen, d.h. seine Anwendung ist nicht nur auf Netzwerksysteme beschränkt. Module können parametrisiert und prinzipiell in beliebigen Hierarchien geschachtelt werden. Die Module der untersten Hierarchiestufe werden in C++ implementiert, wobei auch hier einfach von den Vererbungsmechanismen dieser Sprache Gebrauch gemacht werden kann und so Varianten bereits bestehender grundlegender Modelle einfach erzeugt werden können. OMNeT++ ist kein kommerzielles Tool und wird erst seit wenigen Jahren entwickelt. Daher existieren noch keine so umfangreichen Modell-Bibliotheken, wie bei allen anderen Simulationsprogrammen. Eine Bibliothek mit IP-basierten Protokollen ist von der University of Technology, Sydney (UTS) und der CTIE, Monash University, Melbourne, erhältlich. Die sog. IPSuite und IPv6Suite kommen mit Modellen des IP (Versionen 4 bzw. 6), TCP, UDP und RTP, darüber hinaus existieren frei erhältliche Modelle der Protokolle MPLS, LDP und RSVP-TE. Zu den zur Verfügung stehenden Layer2-Modellen gehören neben PPP-Links Modelle des Ethernet (10/100/1000 MBit/s sowie Busse, Hubs und Switches), Token-Ring, FDDI, Hiperlan/2 und Wireless LAN. OMNeT++ erlaubt das Linken einer grafischen Schnittstelle zu Simulationen, so dass diese auch interaktiv ausgeführt werden können. Dabei können z.B. Variablen sichtbar gemacht und verändert werden bzw. ausgetauschte Nachrichten beobachtet werden. Die Ausgabe von Ergebnissen erfolgt i.A. als Textdatei, wobei verschiedene OMNeT++ Klassen Statistikfunktionen bieten, mit denen Daten vor der Ausgabe bereits bearbeitet werden können. Ansonsten kann die Nachbearbeitung einfach durch Übernahme der Textdateien in gängige Statistik-Tools erfolgen (die nicht Teil von OMNeT++ sind). Da OMNeT++ auf C++ und das portable Tcl/Tk aufsetzt, ist es sehr portabel und läuft auf allen gängigen Unix-Systemen, auf MacOS und auf Windows-Systemen. Es ist als Open Source verfügbar und für den nicht-kommerziellen Einsatz sowie den Einsatz in der Lehre kostenfrei. Für den Test und die Fehlersuche können viele verfügbare und sehr leistungsfähige Werkzeuge genutzt werden (z.B. `valgrind` [75]).

Abschließend ist zu sagen, dass OMNeT++ ein viel versprechendes und wohl strukturiertes Programm ist, das mittlerweile in der Forschung eine immer weitere Verbreitung findet. Die Erstellung von neuen Modellen ist intuitiv und einfach in der Sprache C++ möglich. Einzig die fehlende Vielfalt an verfügbaren Modellen ist ein Nachteil.

5.2.2.5 OPNET Modeler

OPNET Modeler ist eine mächtige kommerzielle Simulationsumgebung für die ereignisorientierte und hybride Simulation von Protokollen und Kommunikationsnetzen. Das Verhalten von Modellen wird anhand von erweiterten Zustandsautomaten spezifiziert, die bei Zustandsänderungen Funktionen ausführen können. Die dabei verwendete Sprache ist Proto-C, eine um Bibliotheksfunktionen erweiterte Variante der Sprache C (auch die Nutzung von C++ ist grundsätzlich möglich).

OPNET unterstützt die Hierarchiestufen Prozess (auf der untersten Ebene), Knoten und Subnetz bzw. Netz. Das eigentliche Verhalten wird dabei auf Prozessebene festgelegt; in einem Knoten können verschiedene Prozesse nur noch miteinander verbunden (und parametrisiert), aber nicht mehr grundsätzlich verändert werden. In einem Knoten werden so z.B. auch die einzelnen Protokollschichten miteinander verbunden. Dies erlaubt es für ein schnelles Prototyping auch, nach geringfügigen Modifikationen eines zugrunde liegenden Modells, tiefer gelegene Protokollschichten auszulassen und z.B. gleich zwei benachbarte Instanzen eines Transportprotokolls direkt miteinander zu verbinden. Mit dem Netzwerk-Editor schließlich werden die zu simulierenden Netztopologien durch Verknüpfen von Knotenmodellen mit Links erstellt.

Zum OPNET Modeler gehört eine umfangreiche Bibliothek unterstützter Protokolle und Modelle für Netzwerkgeräte wie Router, Switches etc. unterschiedlicher Hersteller. Neben den Standard-Protokollen IP (Version 4 und 6), TCP, UDP gehören Routing-Protokolle (RIP, OSPF, IGRP, EIGRP, BGP) und Layer-2 Protokolle wie Ethernet, FDDI, TokenRing, FrameRelay und PPP zum Standardumfang. Auch Modelle des ATM oder für IP-basierte QoS-Protokolle werden standardmäßig mitgeliefert. Für Simulation von Mobilitätsprotokollen und MPLS müssen kostenpflichtige Erweiterungen beschafft werden.

Für jede der erwähnten Hierarchiestufen besitzt der Modeler einen eigenen grafischen Editor, was eine komfortable Spezifikation von Systemen erlaubt. Des Weiteren besitzt der OPNET Modeler ein Analysemodul, welches zur schnellen Auswertung von Statistiken einer Simulation dient, aber auch die Erfassung von Daten mehrerer Simulationsläufe mit unterschiedlichen Parametern zulässt. Dabei können z.B. Konfidenzintervalle einfacher Ergebnisse gebildet und dargestellt werden. Der Modeler setzt einen C/C++ Compiler auf der verwendeten Plattform voraus und unterstützt Solaris, HP-UX und Windows (2000 und XP). Eine Spezifikation und Integration eigener Transportprotokolle ist generell möglich, ebenso wie die Erweiterung bzw. Änderung bestehender Protokolle und Modelle, da alle Modelle der Funktionsbibliothek im Quellcode vorliegen.

5.2.3 Diskussion geeigneter Simulationswerkzeuge

Zum Zeitpunkt der Modellierung des SCTP stand das Werkzeug BONeS (vgl. Abschnitt 5.2.2.1) nicht mehr zur Verfügung, da Herstellung und Support seit Anfang 2000 eingestellt worden war. Für NS-2 existiert bereits seit 2003 eine von einer Gruppe an der Universität Delaware entwickelte SCTP-Implementierung [15].
Da eine unabhängige Bewertung des SCTP gewünscht war, schied eine Nutzung dieser Implementierung als Grundlage für nachfolgende Untersuchungen aus diesem Grund aus. Darüber hinaus unterstützt NS-2 keine Knoten, die natives Multihoming zur Verfügung stellen. Dieses Problem wurde in der NS-2 Implementierung von Caro et. al. [15] dadurch umgangen, dass ein SCTP-Agent definiert wurde, der ein Konglomerat mehrerer Knoten darstellt. Zu diesen gehören Interface-Knoten, die den eigentlichen Verkehr erzeugen, und ein Kern-Knoten, der Verbindungen zu mehreren Interface-Knoten besitzt und bei der Übertragung von Nachrichten für die entsprechende Auswahl ausgehender Interfaces sorgt. Das ausgewählte Simulationswerkzeug sollte daher eine möglichst realistische Implementierung der IP-Protokollschichten liefern und natives Multihoming unterstützen.
OMNeT++ schien zum Zeitpunkt des Beginns der Arbeit noch nicht genügend ausgereift, besitzt aber durchaus das notwendige Potenzial zu einem ausreichenden Simulationswerkzeug. Insbesondere der überlegte und klar strukturierte Aufbau überzeugt; ferner existieren Implementierungen aller im IP-Umfeld notwendigen Protokollschichten und Netzwerkelemente.
So reduzierte sich die Auswahl möglicher Simulationswerkzeuge auf den OPNET Modeler, der bereits am Lehrstuhl erfolgreich eingesetzt wurde und auf die Kombination QualNet/GloMoSim. Aus Mangel an Erfahrungen mit letzterer Umgebung und aus Kostengründen schied diese Kombination jedoch ebenfalls aus, und es wurde entschieden, die Implementierung mit dem OPNET Modeler durchzuführen. Dieses Werkzeug erlaubt zumindest eine einfache Integration des neuen Transportprotokolls in bestehende Host-Modelle, die auch die volle Funktionalität der IP-Protokollschichten (inklusive Multihoming und Routing) bieten.

5.3 Struktur des Simulationsmodells

In Abschnitt 5.1 wurde bereits erwähnt, dass die Modellierung des SCTP mit dem Ziel verfolgt wurde, den Datenpfad und relevante Teile der Pfadüberwachung zu implementieren. Die Implementierung erfolgte unter folgenden Randbedingungen:

- Festlegung auf den OPNET Modeler als Simulator.
- Ähnlichkeit der SCTP-Implementierung zu der bestehenden TCP-Implementierung wird angestrebt (insbesondere bezüglich der Anwendungsschnittstellen).
- Unterstützung von Knoten mit nativem IP (Version 4) und Multihoming.
- SCTP-Assoziationen zwischen Endpunkten werden statisch konfiguriert.

Es zeigte sich im Laufe der Arbeit, dass insbesondere die zweite Randbedingung (Ähnlichkeit der Anwendungsschnittstellen) aufgrund der komplexen Anwendungsmodelle des OPNET Modeler nicht eingehalten werden konnte. Im Modeler benutzen Anwendungsmodelle, die auf ein Transportprotokoll aufsetzen (mithin auf TCP oder UDP) eine Abstraktionsschicht, den sog. Transport Adaptation Layer (TPAL). Da SCTP aber Dienste zur Verfügung stellt, die die Fähigkeiten von UDP oder TCP übersteigen (z.B. die teilweise Reihenfolgesicherung, oder Nachrichtenströme), stellt der TPAL eine unzureichende Schnittstelle dar, die die neuen Eigenschaften des SCTP nicht zu nutzen vermag. Daher wurden für die Simulationen mit dem im Laufe dieser Arbeit erstellten SCTP-Modell eigene Anwendungen definiert, die weiter unten im Abschnitt 5.3.4 erläutert werden.

5.3.1 Überblick über das SCTP-Modell

Das im Laufe der Arbeit erstellte SCTP-Modell besteht aus zwei OPNET Prozessen. Der erste ist ein statisch erzeugter Basisprozess (sctp_dispatcher), welcher in einem Netzknoten genau einmal existiert. Er verarbeitet Nachrichten und Dienstprimitive, die von der Anwendungsschicht an das SCTP-Protokoll gesendet werden, sowie Datagramme der IP-Schicht, die den Protokoll-Typ SCTP aufweisen. Er wird im Folgenden auch mit *Verteiler-Prozess* bezeichnet.
Der zweite Prozess wird von dem Basisprozess zur Simulationszeit dynamisch erzeugt. Er repräsentiert eine (etablierte) SCTP-Assoziation und erhält vom Verteiler-Prozess die für die jeweilige Assoziation bestimmte Nutzlast der empfangenen IP-Datagramme, sowie Anwendungsnachrichten und von der Anwendung aufgerufene Dienstprimitive. Im Folgenden wird dieser Prozess auch mit *Assoziations-Prozess* bezeichnet.
Abbildung 5.1 zeigt die Einbindung des SCTP Verteiler-Prozesses in einen Netzknoten. Er besitzt zum einen ein direktes Interface zur SCTP-Anwendung, zum anderen ist er, ähnlich wie die Prozesse, die TCP und UDP implementieren, mit

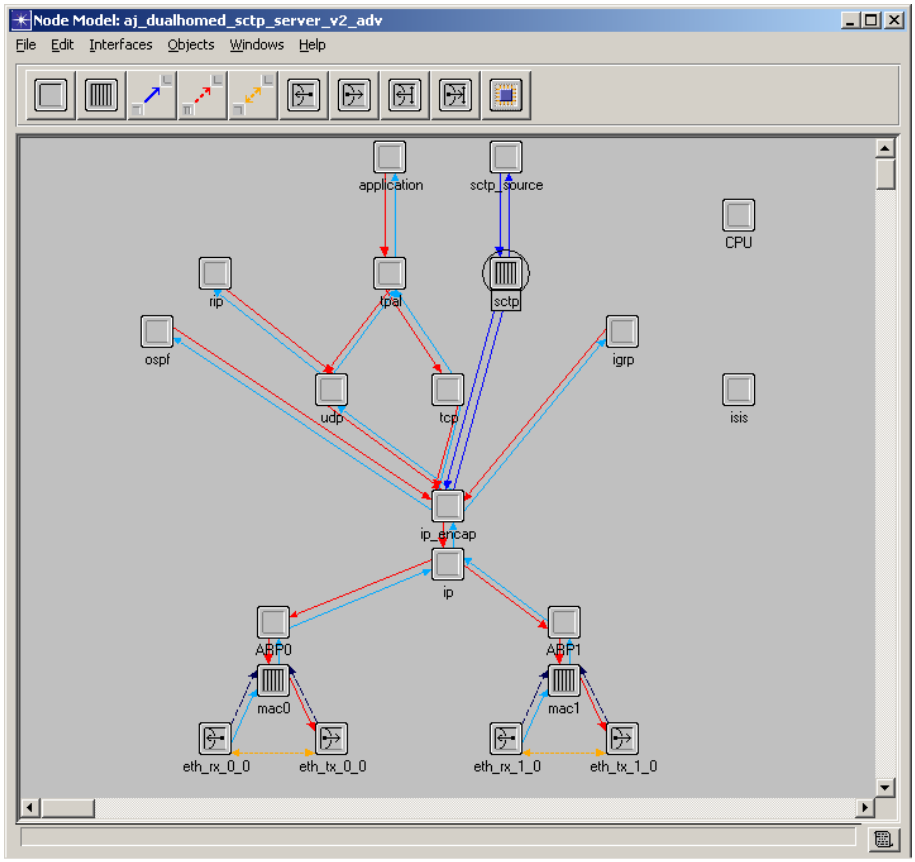

Abbildung 5.1: Netzknoten mit SCTP-Unterstützung

der IP-Schicht verbunden. Ein Assoziations-Prozess sendet empfangene Nutzdaten direkt an die Anwendung und reicht ausgehende SCTP-Pakete an den IP-Prozess weiter.

5.3.2 Der SCTP Datenpfad

Das Verhalten einer Assoziation wird praktisch vollständig in dem Assoziations-Prozess realisiert, der in Abbildung 5.2 dargestellt ist. Sie stellt anschaulich die Ereignisse dar, auf die der Prozess reagiert. Im Einzelnen sind dies:

- Initialisierung einer Assoziation (ASSOC_CREATE),

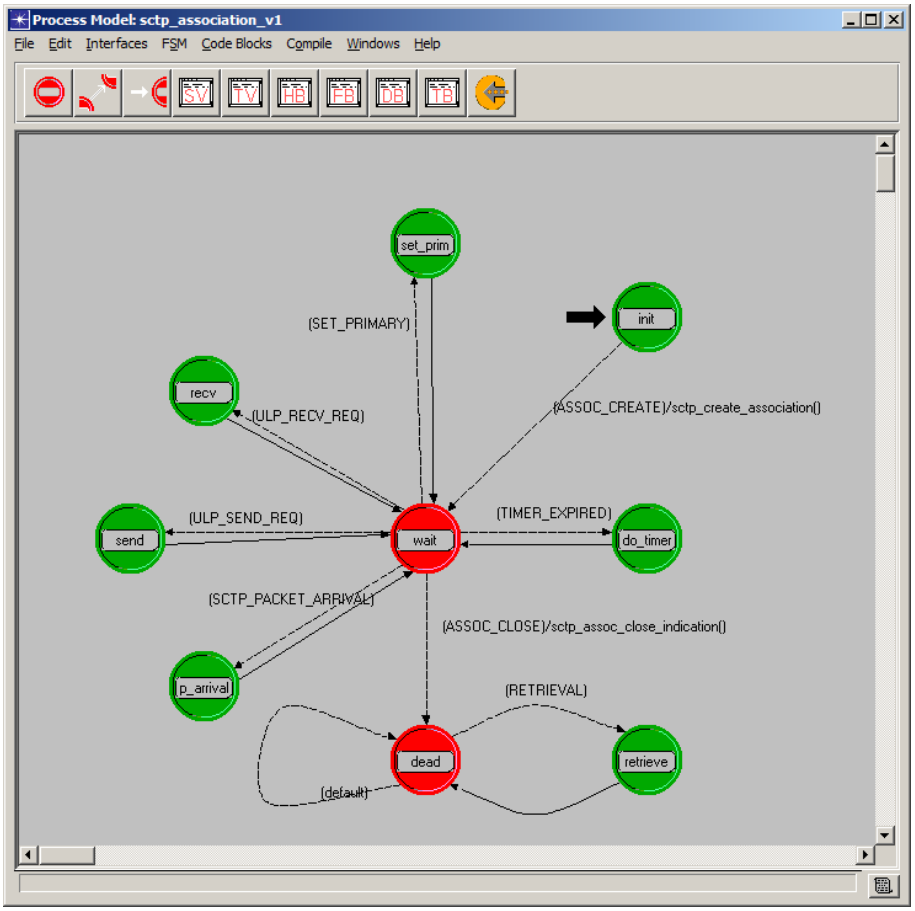

Abbildung 5.2: Prozess-Modell einer SCTP-Assoziation

- Erhalt einer zu sendenden Nachricht von der Anwendung (ULP_SEND_REQ),
- Abholen empfangener und zustellbereiter Nachrichten durch die Anwendung (ULP_RECV_REQ),
- Setzen eines (neuen) Primärpfads durch die Anwendung (SET_PRIMARY),
- Ankunft eines IP-Datagramms, in dem ein SCTP-Paket mit Chunks enthalten ist (SCTP_PACKET_ARRIVAL),

- Terminierung einer Assoziation durch Überschreiten einer Fehlerschwelle (ASSOC_CLOSE) und
- Ablauf eines Protokoll-Timers (TIMER_EXPIRED).

Ist die Assoziation beendet, so kann die Anwendung Nachrichten, deren Erhalt nicht vom anderen Kommunikationsendpunkt bestätigt wurde, oder die gar nicht erst gesendet werden konnten, zurückholen und gegebenenfalls anderweitig bearbeiten oder übertragen (RETRIEVAL).

5.3.2.1 Senden von SCTP-Nachrichten

Der Sendepfad des Modells reagiert auf die folgenden Ereignisse:

- Erhalt einer zu sendenden Nachricht von der Anwendung,
- Ankunft eines SCTP-Pakets mit einem SACK-Chunk,
- Überschreiten der Fehlerschwelle *path.max.retrans* eines Pfades und anschließende Aktivierung/Deaktivierung dieses Pfades,
- Überschreiten einer Fehlerschwelle *assoc.max.retrans* und anschließende Terminierung der Assoziation und
- Ablauf eines Neuübertragungs-Timers (T3 Retransmission Timer).

Wenn die Anwendung eine Nachricht über einen Stream senden will, so wird diese über einen OPNET-Paketstrom als Paket an das SCTP-Modul übertragen. Die Nachricht wird von einer Datenstruktur begleitet, die ICI (Interface Control Information) genannt wird. ICIs enthalten die zu den Nachrichten gehörigen Protokollinformationen, wie z.B. Assoziation und Index des Nachrichtenstroms, über den die Nachricht gesendet werden soll.
Bei Erhalt der Nachricht generiert der Assoziations-Prozess eine Datenstruktur, in der die Nachricht, sowie alle relevanten Protokolldaten (Strom-Sequenznummer, TSN, Flags, benutzte Pfade, Zeit des letzten Sendeversuchs etc.) gespeichert werden. Diese Datenstruktur (nachfolgend Chunk-Struktur genannt) wird im Folgenden in verschiedenen Warteschlangen des Assoziations-Prozesses gespeichert, bis sie nach erfolgreicher Übertragung und Bestätigung durch den assoziierten SCTP-Endpunkt gelöscht werden kann.

Die Schicht, in der die SCTP-Nachrichtenströme verwaltet werden, besteht aus mehreren Warteschlangen und dazugehörigen Datenstrukturen, die alle zu sendenden Nachrichten eines Stroms verwalten: diese werden zunächst in einer FIFO-Warteschlange gespeichert, nachdem sie mit einer Strom-Sequenznummer versehen wurden. Ferner werden auch noch relevante Statistik-Werte gespeichert.
Die Senderseite des Datenpfads besteht aus zwei weiteren wichtigen Warteschlangen:

1. Die Speicherliste. In ihr werden alle Chunk-Strukturen gespeichert, nachdem sie erstmalig an den assoziierten Endpunkt gesendet wurden. Die Chunk-Strukturen werden aus dieser Liste entfernt und endgültig gelöscht, nachdem ihr Erhalt endgültig vom assoziierten Endpunkt bestätigt wurde (oder die Assoziation terminiert ist).

2. Die Übertragungsliste. In ihr werden Chunk-Strukturen temporär gespeichert, wenn sie erneut an den assoziierten Endpunkt übertragen werden.

In diesen beiden Warteschlangen werden die Chunk-Strukturen nach TSNs sortiert. TSNs werden Chunks erstmalig zugeordnet, wenn eine Chunk-Struktur aus einer Strom-Warteschlange entnommen und gesendet wird. Nun werden Nachrichten nach dem folgenden Prozess gesendet:

- Bestimmung der nächsten zu sendenden Chunk-Struktur: dazu wird zunächst die Übertragungsliste überprüft und falls diese nicht leer ist, wird die erste Chunk-Struktur dieser Liste betrachtet.

- Ansonsten wird eine Scheduler-Funktion aufgerufen, die eine Chunk-Struktur eines Nachrichtenstroms auswählt.

- Überprüfung der Sendebereitschaft: Erlauben die Regeln und Parameter der Fluss- und Überlastkontrolle das Aussenden der ausgewählten Chunk-Struktur, so wird diese (gegebenenfalls mit einem SACK-Chunk) in einem neuen SCTP-Paket gesendet.

- Erstmalig gesendete Chunk-Strukturen werden in die Speicherliste eingetragen.

- Referenzen auf Chunk-Strukturen werden nach dem Senden aus der Übertragungsliste oder der Strom-Warteschlange entfernt.

Durch diese Regeln wird sichergestellt, dass Chunk-Strukturen mit älteren TSNs vor neuen Nachrichten übertragen werden (wie im RFC 2960 [86] verlangt). Wird die Assoziation beendet, bevor alle Warteschlangen leer sind, so kann die Anwendung die nicht bestätigten oder nicht gesendeten Chunk-Strukturen zurückholen, und anderweitig behandeln (engl. Message Retrieval).
Wenn ein SACK-Chunk eintrifft, wird zunächst überprüft, ob dieser neue Daten bestätigt. Ist dies nicht der Fall, wird der SACK-Chunk verworfen. Wird der CTSNA-Wert des assoziierten Endpunkts erhöht, so können die entsprechenden Chunk-Strukturen aus der Speicherliste entfernt, und der zugeordnete Speicher freigegeben werden. Anschließend werden die Algorithmen der Überlastkontrolle (vgl. auch Abschnitt 2.3.8) ausgeführt, die in Abhängigkeit der Parameter $ssthresh$, $cwnd$ und $partial\_bytes\_acked$ sowie der Anzahl der bestätigten Bytes den $cwnd$-Wert des Pfades erhöhen, auf dem die bestätigten Daten gesendet wurden.

5.3.2.2 Empfang von SCTP-Nachrichten

Der Empfangspfad des Assoziations-Modells reagiert auf die folgenden Ereignisse:

- Ankunft eines SCTP-Pakets mit Daten-Chunks,
- Abholen empfangener und zustellbereiter Nachrichten durch die Anwendung,
- Terminierung einer Assoziation durch Überschreiten einer Fehlerschwelle und
- Ablauf des SACK-Timers.

Die Verwaltung von empfangenen Daten-Chunks und zugehörigen Nachrichten erfolgt mittels mehrerer Datenstrukturen. Zunächst wird die TSN eines neu empfangenen Daten-Chunks mit der Liste aller bisher empfangenen TSNs verglichen. Wurde der Chunk bereits zu einem früheren Zeitpunkt empfangen, so wird seine TSN in eine Liste von Duplikaten eingetragen und mit dem nächsten SACK-Chunk an den assoziierten Endpunkt gesendet. Die Nachricht wird verworfen. Ist der empfangene Daten-Chunk neu, so wird sein Erhalt durch einen an den assoziierten Endpunkt gesendeten SACK-Chunk bestätigt, der entweder einen inkrementierten CTSNA-Wert oder einen angepassten Gap-Report (vgl. auch Abschnitt 2.3.5) enthält.

Nachfolgend wird die neue Nachricht an ihren zugehörigen Nachrichten-Strom weitergeleitet, der u.a. durch Datenstrukturen mit drei Warteschlangen beschrieben wird. Die erste dieser Warteschlangen (mit *Unordered Queue* bezeichnet) nimmt Nachrichten, deren Daten-Chunks ohne Reihenfolgesicherung gesendet wurden (mit gesetztem *unordered*-Flag), in Ankunftsreihenfolge auf. Die zweite (als *Reordering Queue* bezeichnete) nimmt Nachrichten auf, deren Daten-Chunks mit Reihenfolgesicherung und folglich mit gültiger Strom-Sequenznummer, gesendet wurden und sortiert diese nach TSNs. Nachdem alle Daten-Chunks eines neu angekommenen SCTP-Pakets verarbeitet und deren zugehörige Nachrichten in entsprechende Warteschlangen eingereiht wurden, wird überprüft, welche dieser Nachrichten an die Anwendung weitergereicht werden können. Dies sind alle Nachrichten aus der ersten Warteschlange, sowie alle in Reihenfolge erhaltenen Nachrichten der zweiten Warteschlange. Sie werden in eine dritte Warteschlange (die *Delivery Queue*) übertragen, aus der die Anwendung sie nach dem FIFO-Prinzip abholen kann. Das bedeutet grundsätzlich, dass Nachrichten, die ohne Reihenfolgesicherung gesendet werden, solche, die mit Reihenfolgesicherung gesendet wurden, überholen können.
Die Anwendung signalisiert der Assoziation durch ein Dienstprimitiv ihre Bereitschaft, eine bestimmte Anzahl neuer Nachrichten eines ausgewählten Stroms zu empfangen. Der Assoziations-Prozess wird dann die angegebene Anzahl von Nachrichten an die Anwendung senden, nachdem diese in die *Delivery Queue* übertragen wurden.

5.3.3 SCTP Fehler-Management und Pfadüberwachung

Bei der Erzeugung des Assoziations-Prozesses werden entsprechend der Anzahl der Netzwerkpfade zum assoziierten Endpunkt Strukturen angelegt, in denen alle Pfad-Parameter und Timer für jeden einzelnen Pfad verwaltet werden.
Das Pfadüberwachungsmodul des Assoziations-Modells reagiert auf die folgenden Ereignisse:

- Setzen eines (neuen) Primärpfads durch die Anwendung,
- Ankunft eines SCTP-Pakets mit HEARTBEAT- oder HEARTBEAT-ACK-Chunks,
- Bestätigung eines erstmalig übertragenen Daten-Chunks,
- Überschreiten der Fehlerschwelle eines Netzwerk-Pfades,

- Überschreiten der Fehlerschwelle der Assoziation,
- Ablauf eines Heartbeat-Sende-Timers und
- Ablauf des Heartbeat-Timeouts.

Regelmäßig werden bei Ablauf des Heartbeat-Sende-Timers eines Pfades, auf dem keine Daten übertragen werden (normalerweise also auf allen Pfaden außer dem Primärpfad), HEARTBEAT-Chunks gesendet, auf die der assoziierte Endpunkt mit Bestätigungen reagiert. Trifft eine solche Bestätigung vor Ablauf des Heartbeat-Timeouts ein, so wird sie für die Bestimmung der Rundlaufzeit des jeweiligen Pfades benutzt, und die Fehlerzähler des Pfades und der Assoziation werden zurückgesetzt. Andernfalls werden die entsprechenden Fehlerzähler bei Ablauf des Heartbeat-Timeouts inkrementiert.
Auf dem Primärpfad wird die Rundlaufzeit häufiger bestimmt: jedes Mal wenn ein Daten-Chunk, der erstmalig an den assoziierten Endpunkt gesendet wurde, bestätigt wird, kann diese Bestätigung zur Bestimmung der Rundlaufzeit herangezogen werden. Diese Messung wird gemäß [86] nicht öfter als einmal pro Rundlaufzeit zur Bestimmung der geglätteten Rundlaufzeit (Smoothed Roundtrip Time, SRTT) benutzt. Die Schätzfunktion für die Rundlaufzeit ist daher für den Primärpfad in der Regel genauer als für die alternativen Pfade.

5.3.4 Anwendungsmodelle

Zum Test der Funktionalität der Protokollimplementierung, sowie zur Erarbeitung der im Folgenden vorgestellten Ergebnisse wurden im Wesentlichen die in diesem Abschnitt näher erläuterten Anwendungsprozesse genutzt. Diese Prozesse erzeugen während ihrer Initialisierungs-Phase eine oder mehrere SCTP-Assoziationen und weisen Funktionen auf, um Nachrichten über das SCTP zu senden und zu empfangen sowie auf die Ereignisse der Assoziationen zu reagieren.

5.3.4.1 Quelle mit einer Assoziation

Das erste und einfachste Anwendungsmodell, das verwendet wurde, besteht aus einer Modifikation des Opnet Modells der *Simple Source*, einer einfachen Quelle. Es wird nachfolgend auch *SCTP Applikation* genannt. Dieses Modell erzeugt zunächst auf dem angeschlossenen SCTP-Modul einen neuen Assoziations-Prozess, der durch eine etablierte SCTP-Assoziation mit einem zweiten Endpunkt

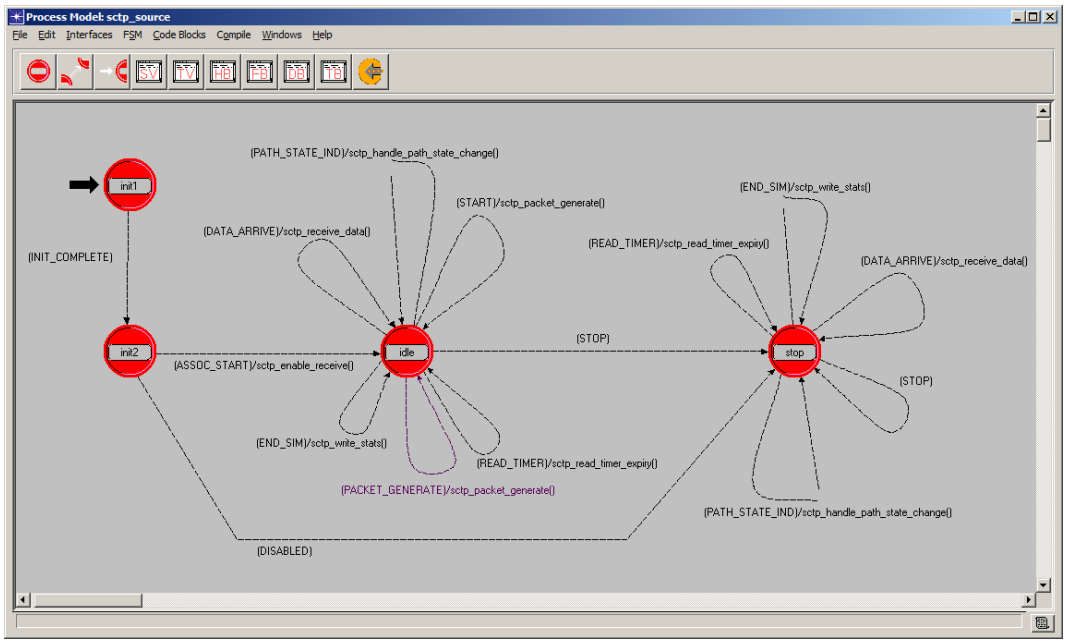

Abbildung 5.3: Prozessmodell einer einfachen Anwendung

verbunden ist. Diese Assoziation kann je nach der IP-Konfiguration des Netzknotens multi-homed oder single-homed sein. Auf dem assoziierten Endpunkt wird ein identischer Prozess ausgeführt, so dass nach der Initialisierung beide Endpunkte über die etablierte SCTP-Assoziation miteinander kommunizieren können. Der Anwendungsprozess erzeugt ab einer konfigurierbaren Startzeit der Quelle Nachrichten, deren Zwischenankunftszeiten und Längen jeweils konfigurierbaren Verteilungen gehorchen. Beispiele für typische, von OPNET unterstütze Verteilungen sind :

- Konstante Verteilung (Parameter: c). Die Funktion erzeugt jedes Mal den Wert c.
- Gleichverteilung (Parameter: min, max). Jeder Wert zwischen min und max ist bei dieser Verteilung gleich wahrscheinlich.
- Negative Exponentialverteilung (Parameter λ). Die von dieser Funktion erzeugten Werte x gehorchen der Verteilung mit der Verteilungsdichtefunktion $f(x) = \lambda \cdot e^{-\lambda \cdot x}$. Der Erwartungswert $E(X)$ ist $1/\lambda$.

- Normalverteilung (Parameter: μ, σ). Die von dieser Funktion erzeugten Werte x gehorchen der Verteilung mit der Verteilungsdichtefunktion

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \tag{5.3}$$

  Der Erwartungswert $E(X)$ ist in diesem Fall μ, die Standardabweichung σ.

- Paretoverteilung (Parameter: Location a, Shape c). Die von dieser Funktion erzeugten Werte x gehorchen der Verteilung mit der Verteilungsdichtefunktion

$$f(x) = c \cdot \frac{a^c}{x^{c+1}} \quad \text{mit } a \leq c \leq \infty \text{ und } a, c > 0 \tag{5.4}$$

  Der Erwartungswert $E(X)$ ist in diesem Fall $ca/(c-1)$ für $c > 1$.

- Dreiecksverteilung (Parameter min, max). Die Wahrscheinlichkeit, dass ein Wert von $\mu = (max - min)/2$ erzeugt wird, ist am höchsten. Die Wahrscheinlichkeitsdichte nimmt zu den Rändern min und max hin linear ab. Werte kleiner als min oder größer als max werden durch diese Verteilung nicht erzeugt.

Darüber hinaus ist es auch möglich, in einer Tabelle eine Folge von Werten anzugeben, die die Verteilung bestimmen, oder eine Verteilung mittels einer Wahrscheinlichkeitsdichtefunktion selbst zu definieren.
Das SCTP-Anwendungsmodell der *SCTP Applikation* fungiert auch als Nachrichtensenke. Der Anwender kann konfigurieren, wie viele Nachrichten in einer vorgegebenen Zeiteinheit vom SCTP akzeptiert werden. Wird diese Rate überschritten, so verbleiben Nachrichten im Pufferspeicher des Assoziations-Prozesses, der dies dem assoziierten Endpunkt durch kleiner werdende $arwnd$-Werte in SACK-Chunks signalisiert. Durch diese Eigenschaft kann der Effekt eines geschlossenen Sendefensters einfach simuliert werden.
Da die Assoziation in der Regel alle Netzwerk-Pfade zur Übertragung nutzt, muss die Anwendung auch auf den Ausfall eines Pfades, insbesondere des Primär-Pfades, reagieren. Der Zustand aller Pfade wird überwacht, und bei Ausfällen wird gegebenenfalls ein neuer aktiver Pfad zum Primär-Pfad gewählt. Bei Terminierung der Assoziation durch Ausfall aller Pfade oder Überschreiten des Schwellwerts für Assoziations-Fehler wird die Übertragung beendet.

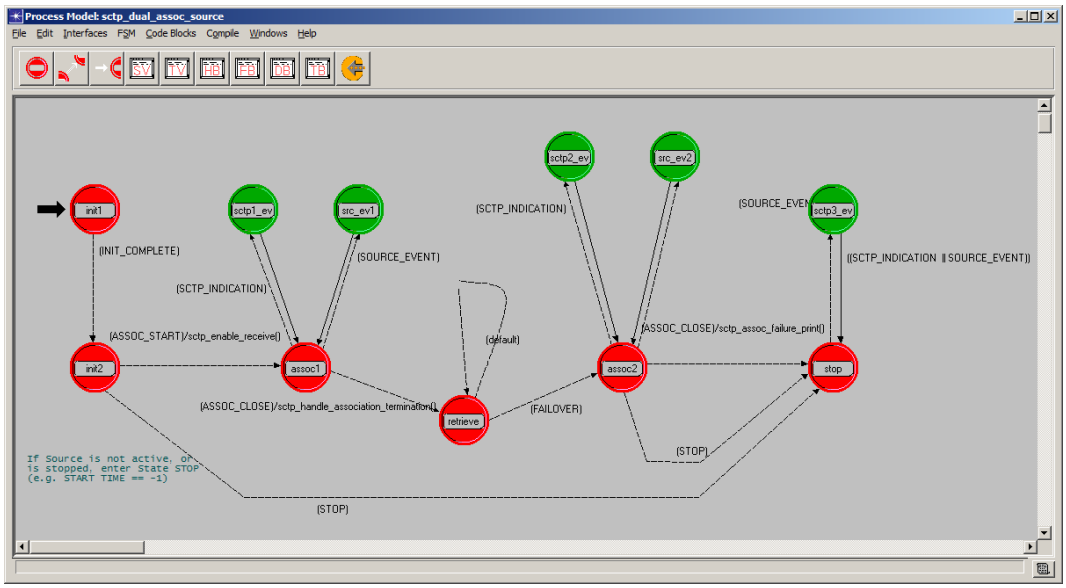

Abbildung 5.4: Prozessmodell einer Anwendung, die Failover unterstützt

5.3.4.2 Quelle mit mehreren Assoziationen

Das einfache Modell der *SCTP Applikation* wurde erweitert, um auch den realistischen Anwendungsfall des Failover (vgl. Abschnitt 4.4) unter ähnlichen Bedingungen wie im Testbett zu untersuchen.
Zunächst erzeugt die Anwendung auf einem Netzknoten mit zwei IP-Schnittstellen zwei SCTP-Assoziationen, die single-homed sind. Auf einem zweiten Netzknoten läuft ein identisches Anwendungsmodell. Die Quelle beginnt, Daten über die erste der beiden Assoziationen zu senden. Die zweite Assoziation wird zu diesem Zeitpunkt der Simulation nicht benutzt; auf ihr werden nur HEARTBEAT- und HEARTBEAT-ACK-Chunks gesendet. Während der Simulation wird der zur ersten Assoziation gehörige Pfad unterbrochen und die Assoziation nach Überschreiten der Fehlerschwelle terminiert. Sobald dieses Ereignis der Anwendung mitgeteilt wird, wechselt sie vom Zustand *assoc1* zum Zustand *retrieve* (vgl. auch Abbildung 5.4). In diesem werden zunächst Daten-Chunks, die noch in der Speicherliste der terminierten Assoziation verblieben sind, ausgelesen und entweder verworfen (wenn die Duplikation von Nachrichten möglichst ausgeschlossen werden soll) oder über die zweite Assoziation erneut gesendet (dabei kann eine Duplizierung von Nachrichten u.U. nicht ausgeschlossen wer-

den). Anschließend werden Daten-Chunks, die noch nicht gesendet wurden, aus den Stream-Listen ausgelesen und über die zweite Assoziation (und damit über den zweiten Netzwerkpfad) gesendet. Nachfolgend nutzt die Quelle nur noch die zweite Assoziation zur Datenübertragung. Ergebnisse der Untersuchungen mit dem hier beschriebenen Anwendungsprozess sind unter Abschnitt 5.6 zu finden.

5.3.4.3 Echo-Server

Der Echo-Server Prozess erzeugt zu Simulationsbeginn eine konfigurierbare Anzahl von Assoziationsprozessen mit Assoziationen zu einem oder mehreren Endgeräten. Auf diesen laufen Anwendungen wie die *SCTP Applikation*, die unidirektionalen Verkehr erzeugen. Jede beim Echo-Server ankommende Nachricht wird von diesem sofort wieder über den gleichen Nachrichtenstrom an den Absender zurückgesendet. Die Eigenschaft der aktivierten oder deaktivierten Reihenfolgesicherung der jeweils empfangenen Nachricht wird dabei beibehalten.
Die Differenz aus Absendezeit der Nachricht durch die assoziierte Anwendung und Ankunftszeitpunkt beim Echo-Server Prozess kann als Statistik ausgeben werden; der Absender kann entsprechend die Rundlaufzeit der Nachricht bestimmen.

5.3.4.4 Gesättigte Quelle

Um den maximalen Durchsatz unter den gegebenen Bedingungen testen zu können, wurde ein Quellenmodell erstellt, welches beim Simulationsstart eine SCTP-Assoziation initialisiert und eine konfigurierbare Anzahl von (z.B. n) Nachrichten an den assoziierten Endpunkt sendet. Das SCTP-Assoziationsmodell wurde um eine Funktion erweitert, die der Anwendung signalisiert, wenn die Sendewarteschlange leer ist. Die gesättigte Quelle erzeugt bei Erhalt dieses Signals erneut n Nachrichten und sendet diese an den assoziierten Endpunkt. Damit ist die Sendewarteschlange immer gefüllt und die SCTP-Assoziation sendet die maximal mögliche Datenmenge.

5.3.5 Modelle der Netzinfrastruktur

Die der weiter unten beschriebenen Untersuchung zugrunde gelegten Modelle für die Netzinfrastruktur sind grundsätzlich so einfach wie möglich gehalten. Typische Infrastrukturmodelle, wie das von vielen Veröffentlichungen über das TCP

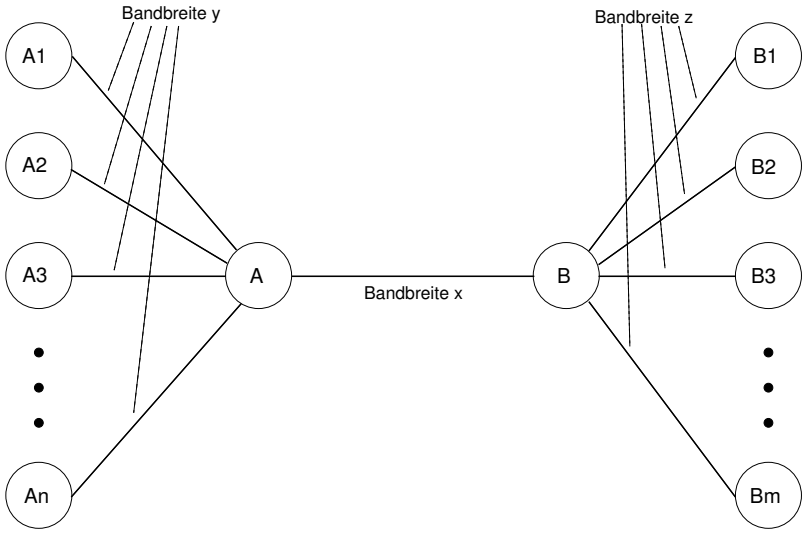

Abbildung 5.5: Einfaches Infrastrukturmodell mit Barbell-Topologie

bekannte Barbell-Modell (siehe auch Abbildung 5.5) und dessen Varianten erlauben Vergleiche des SCTP mit etablierten Protokollen wie TCP. Bei dieser einfachen Topologie werden ein Reihe von z.B. unidirektionalen Verbindungen über eine gemeinsam genutzte, knappe Ressource etabliert und treten in Wettbewerb zueinander. Im idealen Fall teilen sich n Verbindungen zwischen den Knoten A1 bis An und den Knoten B1 bis Bm in Abbildung 5.5 die Bandbreite x so auf, dass jede Verbindung eine Bandbreite von x/n belegt (vorausgesetzt, es gilt $y \geq \frac{x}{n}$, $n \leq m$ und $z \geq \frac{x}{n}$ und die Knoten von A_1 bis A_n senden die Daten).
Wie in [4] erwähnt, spiegelt solch eine einfache Topologie allerdings nicht die Situation wider, die in typischen Zugangsnetzen heutiger Internet-Infrastrukturen überwiegt: dort befindet sich der sprichwörtliche *Flaschenhals* (engl. Bottleneck Link) typisch nicht im Backbone-Bereich, der regelmäßig überdimensioniert ist, sondern eher im Access-Bereich, bzw. am Übergang zwischen Access und Backbone. Die in [4] für Simulationen vorgeschlagenen Topologie-Modelle weisen daher auch mehrfache Überlastbereiche auf, deren Effekte sich überlagern können.
Bei einer zu hohen Komplexität der Netztopologie sind die vielfältigen Einflüsse verschiedener Parameter im Bereich der Infrastrukturelemente auf das Verhalten der Simulation nur schwer einzuschätzen und systematisch zu analysieren. Die Arbeit beschränkt sich daher im Folgenden auf möglichst einfache Topologien

mit Netzen auf Basis des Ethernet (10Base-T, 100Base-T und 1000Base-T) sowie PPP-Links verschiedener Bandbreiten (ISDN, T1, E1) mit konfigurierbarer Link-Verzögerung. Verwendete Netzelemente sind Hubs, Switches, Router und Endgeräte mit mehreren Netzschnittstellen, die das Multihoming unterstützen.

5.4 Untersuchung der Fluss- und Überlastkontrollmechanismen

Die an der Prototypimplementierung im Labor durchgeführten Untersuchungen (vgl. auch Abschnitt 4.4 und [49]) konnten mit Hilfe des im Laufe dieser Arbeit angefertigten Simulationsmodells des SCTP bestätigt werden. Damit konnte gezeigt werden, dass das Simulationsmodell realistische Ergebnisse im Hinblick auf zentrale Protokollmechanismen wie Neuübertragung, Pfadüberwachung und Fehlererkennung liefert. Die erzielten Ergebnisse werden nachfolgend kurz vorgestellt.

5.4.1 Fairness zwischen TCP und SCTP

Ähnlich dem in Abschnitt 4.2.3 geschilderten Szenario wurden zur Untersuchung der Fairness zwischen TCP und SCTP Simulationen durchgeführt, die einen Vergleich des Verhaltens des SCTP mit dem des TCP in Bezug auf die Nutzung knapper Ressourcen wie der Netzwerk-Bandbreite zulassen.

5.4.1.1 Metriken für die Beschreibung der Fairness

In der Literatur werden verschiedene Metriken angegeben, um die Fairness zwischen Protokollen quantitativ vergleichbar zu machen. Mit Hilfe solcher Metriken lassen sich die Eigenschaften der Kurzzeit-Fairness (engl. Short Term Fairness) und der Langzeit-Fairness (engl. Long Term Fairness) beschreiben.
Die Kurzzeit-Fairness beschreibt vor allem das dynamische Protokollverhalten, z.B. bei plötzlichen Änderungen der Gegebenheiten und das Antwortverhalten der Regelmechanismen der jeweiligen Protokollausprägungen. In [6] schlagen Bansal et al. beispielsweise folgende Metriken vor :

- Die *Stabilisierungszeit*. Nach plötzlicher Verringerung der verfügbaren Bandbreite bezeichnet sie die Zeit, nach der der Paketverlust wieder ein akzeptables Maß annimmt (z.B. das 1,5-fache der normalen Verlustrate).

- Die *Aggressivität*. Bei plötzlicher Erhöhung der zur Verfügung stehenden Bandbreite bezeichnet sie die maximale Erhöhung der Senderate innerhalb einer RTT.

- Die *Glätte*. Bei burstartigen Verlusten bezeichnet die Glätte das Verhältnis der Senderaten in aufeinander folgenden RTT-Perioden.

Die Metriken, die die Kurzzeit-Fairness eines Protokolls quantifizieren, werden im Folgenden nicht weiter betrachtet, da die Hauptanforderung an das neue Transportprotokoll SCTP zunächst die faire Bandbreitenverteilung langlebiger Verbindungen im Zusammenspiel mit etablierten Protokollen wie TCP ist.
Die Metriken für die Langzeit-Fairness beschreiben im Wesentlichen das Protokollverhalten über einen längeren Zeitraum, wobei der jeweilige Protokolldurchsatz über diesen Zeitraum entscheidend ist. Um vergleichbare Aussagen treffen zu können, wird das Verhältnis verschiedener Protokolle betrachtet. So schlagen Floyd et al. in [24] vor, den mittleren Protokolldurchsatz mehrerer Verbindungen unterschiedlicher Protokollausprägungen auf einem Link ohne Überlast und nachfolgende Verluste miteinander in Relation zu setzen (z.B. den Durchsatz von 10 TCP-Verbindungen und 10 TFRC-Verbindungen, d.h. eines Protokolls mit TCP-freundlicher Ratenkontrolle). Eine ähnliche Metrik wird auch von Resaie et al. in [68] vorgeschlagen und dort als *Interprotokoll-Fairness* bezeichnet.
Die in Abschnitt 5.4.1.4 erläuterten Untersuchungen beschränken sich auf die grundlegende Darstellung der Interprotokoll-Fairness zwischen einer SCTP-Assoziation und einer variablen Anzahl von TCP-Verbindungen, um den Vergleich der Leistungsfähigkeit der Überlastkontroll-Algorithmen des SCTP-Modells zu der realen Implementierung zu ermöglichen. Die modellierten Szenarien entsprechen mithin denjenigen im Abschnitt 4.2.

5.4.1.2 Untersuchte Szenarien

In den durchgeführten Simulationen wird ein unidirektionaler Datentransfer zwischen zwei SCTP-Instanzen (nachfolgend Sender und Empfänger genannt) sowie zwischen mehreren TCP-Instanzen modelliert, die alle in einer einfachen Barbell-Topologie (vgl. Abbildung 5.5) angeordnet sind. Der Flaschenhals der Topologie wird durch den zentralen T1-Link modelliert, über den zwei Router miteinander verbunden sind. An jedem dieser Router ist ein LAN (100BaseT-Ethernet) angeschlossen, das aus mehreren Endgeräten besteht, die TCP- und SCTP-Verkehr erzeugen.

Es werden mehrere Szenarien untersucht, in denen zwischen 0 und 5 TCP-Verbindungen parallel zu einer SCTP-Verbindung Daten über den T1-Link übertragen. Alle Verbindungen werden in Sättigung betrieben, um den maximalen Protokolldurchsatz eines langen Datentransfers zu modellieren.

5.4.1.3 Eine SCTP-Assoziation mit variabler Linkverzögerung

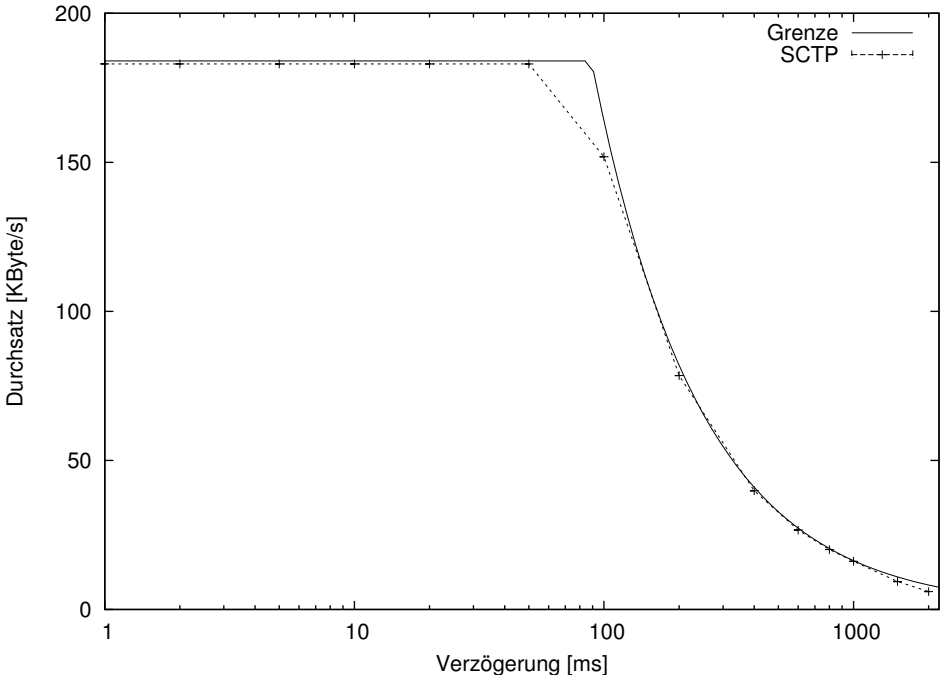

Abbildung 5.6: Durchsatz einer SCTP-Verbindung bei variabler Linkverzögerung

Zunächst wurde in der Simulation das Verhalten des SCTP auf einem Link untersucht, wobei alle relevanten Protokoll-Eigenschaften im Modell implementiert und berücksichtigt wurden, die sich aktuell in der Standardisierung befinden. Das sind z.B.

1. die schnelle Wiederherstellung des $cwnd$-Wertes nach einer schnellen Neuübertragung (engl. Fast Retransmission - Fast Recovery, FRFR), ein Mechanismus, der bei mehrfachem Paketverlust innerhalb einer RTT die mehrfache Reduktion des $cwnd$ vermeidet oder

2. die Erhöhung des $cwnd$-Startwertes auf mindestens 4380 Bytes [80].

Dabei wurde der Durchsatz in Abhängigkeit von der Linkverzögerung bestimmt. Abbildung 5.6 zeigt das Verhalten der SCTP-Verbindung. Das SCTP befindet sich bei dieser Simulation in Sättigung, d.h. die Sendewarteschlange ist ständig gefüllt. Die Nachrichtenlängen gehorchen in der in Abbildung 5.6 gezeigten Simulation einer Dreiecksverteilung mit einem Mittelwert von $l_{avg} = 700$ Bytes, einer maximalen Länge von $l_{max} = 1300$ Bytes und einer minimalen Länge von $l_{min} = 100$ Bytes. In allen Simulationen wurden Konfidenzintervalle von 99% bestimmt, die maximale Abweichungen von ± 65 Byte/s vom Mittelwert ergaben und daher vernachlässigbar klein waren. Wie der Abbildung 5.6 zu entnehmen ist, erzielt das SCTP bis zu einer Linkverzögerung von ca. 100 ms den maximalen Durchsatz, der auf dem T1-Link möglich ist. Bei größeren Verzögerungen beschränkt das Sendefenster die Quelle, ähnlich anderen fensterbasierten Transportprotokollen wie z.B. TCP.

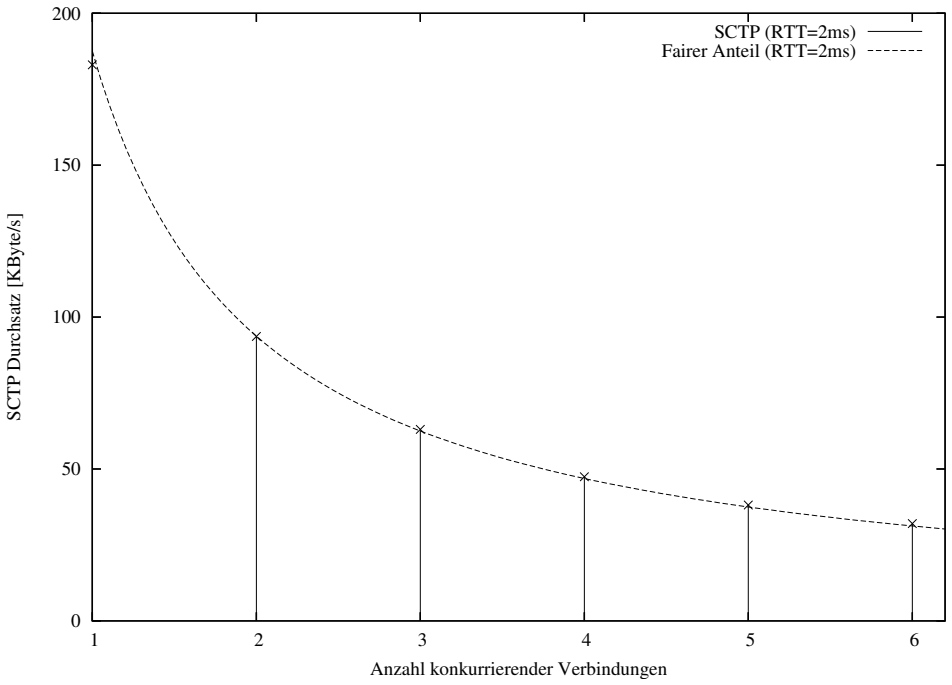

Abbildung 5.7: Durchsatz des SCTP in Konkurrenz zu mehreren TCP-Verbindungen

5.4.1.4 Mehrere TCP-Verbindungen und eine SCTP-Assoziation

Zum Vergleich mit der Bewertung des SCTP in [49] (vgl. auch Abschnitt 4.2) wurden einige Simulationsläufe durchgeführt, in denen die Fairness des SCTP zum TCP bestimmt wurde. Dabei wurde ebenfalls die in Abbildung 5.5 dargestellte Barbell-Topologie zugrunde gelegt und die Zahl der TCP-Verbindungen, die in Konkurrenz zu einer SCTP Assoziation treten, variiert. Abbildung 5.7 stellt den mittleren Protokoll-Durchsatz der SCTP-Assoziation in Abhängigkeit von der Anzahl der gleichzeitig aktiven TCP-Verbindungen dar. Der Durchsatz der SCTP-Assoziation verhält sich umgekehrt proportional zu der Anzahl der insgesamt aktiven (SCTP und TCP) Verbindungen. Somit bestätigt sich auch für das OPNET-Modell, dass sich SCTP die Bandbreite fair mit mehreren TCP-Verbindungen teilt, da die Regeln zur Flusskontrolle und Überlastvermeidung bei beiden Protokollen sehr ähnlich sind (entsprechend dem RFC 2914 - Congestion Control Principles [22]).

5.4.2 Vorteile und Nachteile des SCTP

Ein zentraler Unterschied zwischen SCTP und TCP in Bezug auf die Flusskontrolle ist die Tatsache, dass die Erhöhung des $cwnd$ bei SCTP basierend auf der individuellen Länge der bestätigten Nachrichten erfolgt, d.h. byte-basierend, im Gegensatz zum TCP, wo diese Erhöhung basierend auf der Anzahl eingetroffener Bestätigungen, in der Regel um das Vielfache einer MTU, durchgeführt wird.
Ferner verlässt sich SCTP implizit auf die SACK-Chunks zur Feststellung verloren gegangener Nachrichten. Dies ist beim TCP nur dann der Fall, wenn beide Seiten die SACK-Erweiterung [26] unterstützen, was allerdings bei so gut wie allen verbreiteten TCP-Implementierungen der Fall ist. Die Anzahl der Fragmente ist beim TCP-SACK auf vier limitiert, bzw. bei Nutzung weiterer TCP-Optionen, wie z.B. den TCP-Zeitstempeln (vgl. [43]), auf maximal drei Fragmente. Der SACK-Chunk des SCTP kann hingegen eine wesentlich größere Zahl an Fragmenten bestätigen, bei einer MTU von 1500 Bytes für IP-Datagramme bis zu einer theoretischen Grenze von 363 Fragmenten. Darüber hinaus können auch mehrfache Duplikate (also fälschlich neu übertragene TSNs) in einem SACK-Chunk angezeigt werden. In [80, 86] wird derzeitig nicht angegeben, wie diese Informationen genutzt werden sollten. Einen entsprechenden Algorithmus beschreiben Floyd et al. in [25] für die DSACK-Erweiterung des TCP. Dabei werden die Informationen über Duplikate dazu genutzt, die Reduktion des $cwnd$ nach verfrühten, unnötigen Neuübertragungen (engl. Spurious Retransmissions) wieder rückgängig zu ma-

chen. Blanton und Allman weisen in ihren Untersuchungen [7] aber darauf hin, dass die Anwendung dieses Algorithmus zu Netzüberlast führen kann.
In einer neueren Untersuchung [14] stellen Camarillo et al. fest, wie sich SCTP im Vergleich zu TCP beim IP-basierten Signalisierungstransport für SIP-Nachrichten verhält. Sie weisen darauf hin, dass SCTP durch die Flexibilität bei der Nachrichtenzustellung (vgl. Abschnitt 2.3.5.2) zwar Vorteile bezüglich der Vermeidung von Blockierungseffekten im Falle von Paketverlusten haben kann, machen aber deutlich, dass sich die Fragmentierung von Nachrichten auf der Transportschicht nachteilig auswirken kann, wenn sich auf einem Pfad die MTU ändert. Darüber hinaus ist es beim SCTP potentiell schwieriger, eine optimale Fragmentierung längerer Nachrichten vorzunehmen, da die Länge von zu sendenden SACK-Chunks bei fragmentierter Nutzlast nicht in jedem Fall vorhergesagt werden kann. Implementierungen müssen aus diesem Grund u.U. SACK-Chunks separat senden, oder suboptimale Größen für Daten-Chunks mit Nachrichtenfragmenten wählen, was ineffizient ist.

5.5 Protokolloptimierung bei asymmetrischen Pfaden

In Abschnitt 4.3 wurde bereits eine Motivation für die Nutzung des SCTP in einem Szenario mit Multihoming gegeben. Das normale und das modifizierte Protokollverhalten wurden in einer Testbett-Umgebung untersucht [46]. Die qualitative Abschätzung dieser Untersuchung wurde in den Abschnitten 4.3.2 und 4.3.3 vorgenommen. Im Folgenden sollen diese Ergebnisse durch eine Untersuchung eines geeigneten Simulationsmodells verifiziert werden. Dazu wurde das bestehende OPNET SCTP-Modell zunächst um die Option erweitert, SACK-Chunks nicht nur über den Pfad zurückzusenden, auf dem Daten-Chunks erhalten wurden (wie Abschnitt 6.4 in [86] gefordert), sondern auf dem Pfad mit der kürzesten Rundlaufzeit. Diese Option kann in der Modellimplementierung vom Nutzer zur Simulationszeit entsprechend konfiguriert werden.

5.5.1 Untersuchte Topologie

Abbildung 5.8 zeigt die der Simulation zugrunde liegende Topologie. Dabei nutzt eine TCP-Verbindung zwischen den Systemen `tcp_client` und `tcp_server` die Satellitenstrecke, die durch einen Link mit einer Verzögerung von 250 ms

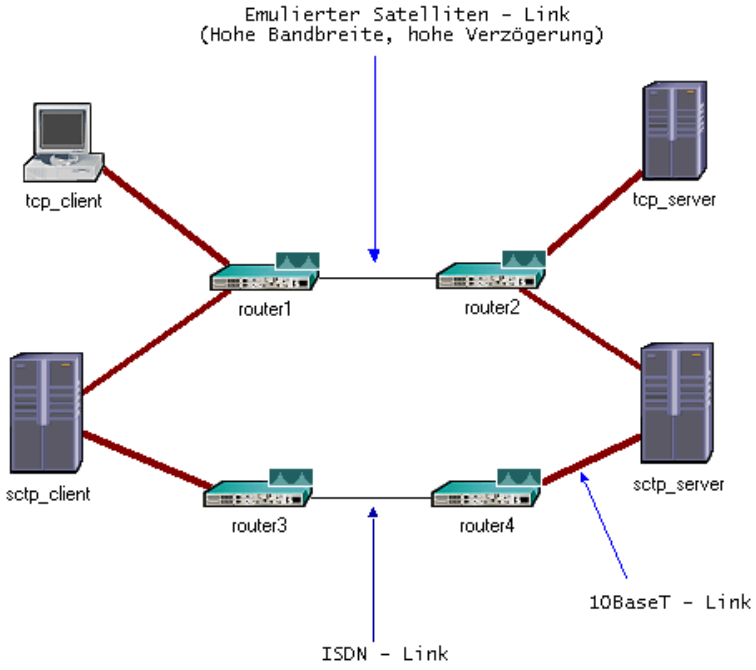

Abbildung 5.8: Netztopologie, die Satellitenübertragung modelliert

und einer Bandbreite von 1,544 MBit/s (entsprechend einem T1-Link) modelliert wird. Parallel dazu existiert eine SCTP-Assoziation zwischen den Systemen `sctp_client` und `sctp_server`, die neben der Satellitenstrecke als Primärpfad auch noch einen Backup-Link als Sekundärpfad nutzt, der mit geringer Verzögerung (z.B. 10ms) und geringer Bandbreite (64 KBit/s) einen dedizierten ISDN-Link modelliert. Das Verhalten der TCP-Verbindung wurde in der weiteren Untersuchung nicht detaillierter betrachtet. Die SCTP-Server überträgt Nachrichten an den SCTP-Client, deren Länge einer Dreiecksverteilung mit einer mittleren Länge von 710 Bytes gehorcht, und die minimal 20 Bytes sowie maximal 1400 Bytes lang sind. Die Ankunftsabstände der Nachrichten gehorchen einer negativ exponentiellen Verteilung mit einer mittleren Rate von etwa 136 Nachrichten pro Sekunde ($\lambda = 136$). Die Quelle erzeugt mithin ein mittleres Verkehrsaufkommen von 96,467 KByte/s, was den T1-Link zu ca. 50% auslastet.

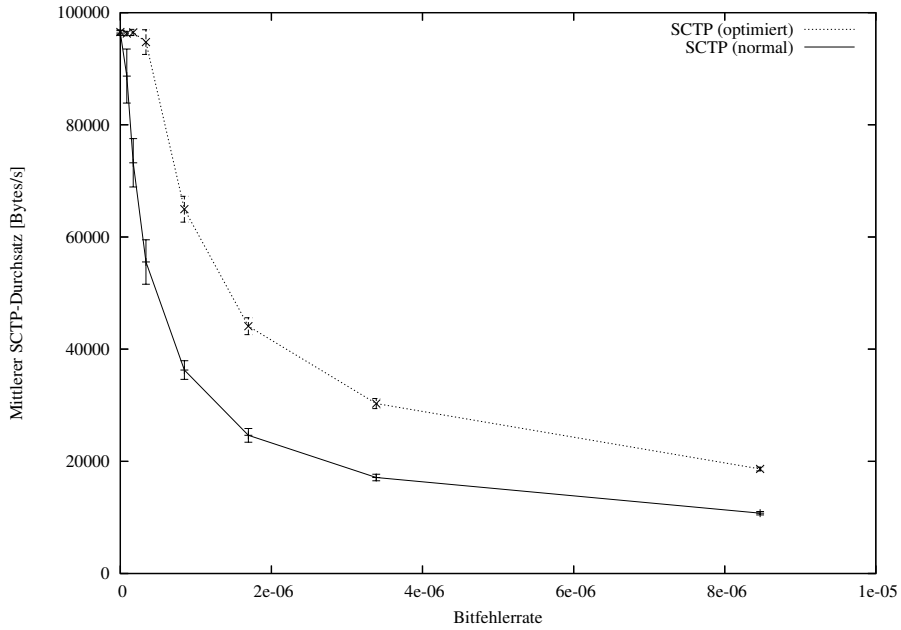

Abbildung 5.9: Vergleich des Durchsatzes der Protokollvarianten – Überblick über die gesamte Simulation

5.5.2 Untersuchung des Protokoll-Verhaltens

In der Simulation wurden zwei SCTP-Protokollvarianten untersucht. Dabei wurde angenommen, dass auf der Satellitenstrecke während der Übertragung eine konstante Bitfehlerrate vorherrscht. Dies ist insofern realistisch, als durch atmosphärische Phänomene (Wolken, Regen etc.) über einen längeren Zeitraum Pakete fehlerhaft übertragen werden, die in der Regel auf Schicht 2 oder Schicht 3 verworfen werden.

Abbildung 5.10 zeigt die erzielten Durchsätze als Funktion der Bitfehlerrate im Bereich von $0..2 \cdot 10^{-6}$ sowohl für Standard-SCTP wie auch für die Protokollmodifikation, die SACKs immer über den Pfad mit der kürzesten RTT sendet. Unter den gegebenen Bedingungen erzeugt die Bitfehlerrate im Mittel eine Anzahl von fehlerhaften Paketen, die aus der folgenden Tabelle zu entnehmen ist:

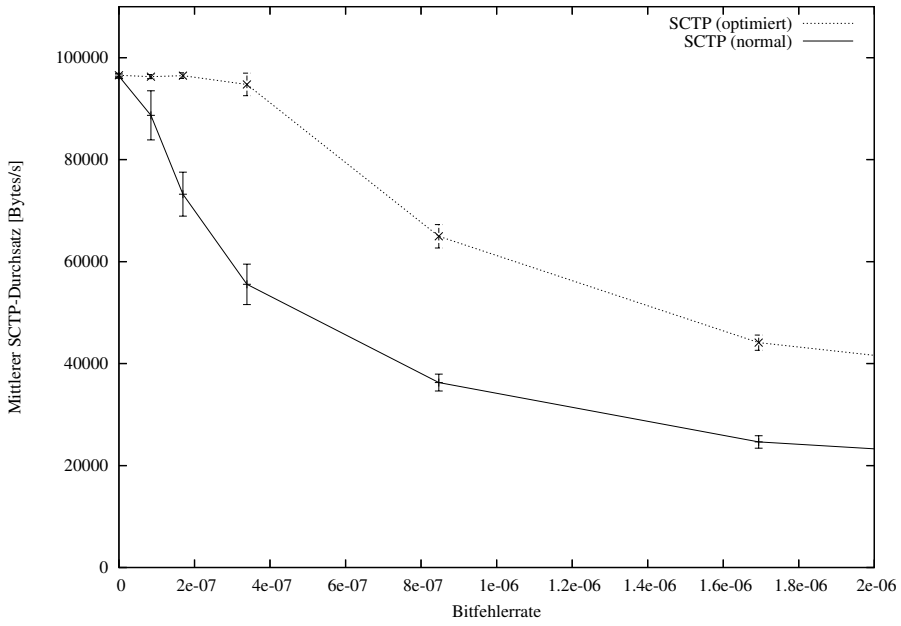

Abbildung 5.10: Vergleich des Durchsatzes der Protokollvarianten – Ausschnittvergrößerung

BER	**Fehlerhafte Pakete**
$1 \cdot 10^{-7}$	1 aus 1650
$2 \cdot 10^{-7}$	1 aus 825
$4 \cdot 10^{-7}$	1 aus 412
$6 \cdot 10^{-7}$	1 aus 274
$8 \cdot 10^{-7}$	1 aus 206
$1 \cdot 10^{-6}$	1 aus 165
$2 \cdot 10^{-6}$	1 aus 82
$4 \cdot 10^{-6}$	1 aus 41
$6 \cdot 10^{-6}$	1 aus 27
$8 \cdot 10^{-6}$	1 aus 20
$1 \cdot 10^{-5}$	1 aus 16

Im fehlerfreien Fall erzielen beide Protokollrealisierungen einen identischen Durchsatz; bis zu einer Fehlerrate von $4 \cdot 10^{-7}$ bleibt der Durchsatz des modifizierten SCTP nahezu konstant, während das Standard-SCTP bereits einen signi-

fikanten Durchsatzrückgang verzeichnet. Abbildung 5.9 zeigt die entsprechenden Durchsatzwerte im Bereich der Bitfehlerrate von $0..1 \cdot 10^{-5}$. Wieder ist ersichtlich, dass sich das modifizierte SCTP selbst bei hohen Bitfehlerraten wesentlich günstiger verhält.

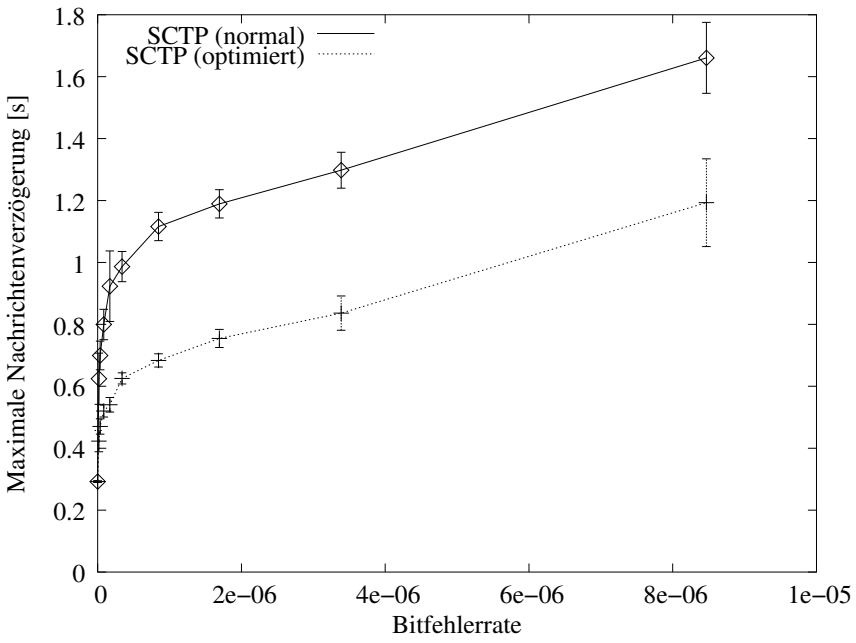

Abbildung 5.11: Vergleich der Verzögerung der Protokollvarianten – Überblick über die gesamte Simulation

Ebenso wie bei der im Labor-Testbett durchgeführten Untersuchung (vgl. Abschnitt 4.3) wurde in der Simulation auch die Nachrichtenverzögerung untersucht. Interessant ist hierbei vor allem die maximale Nachrichtenverzögerung, die bei Paketverlust und strikter Reihenfolgesicherung (Übertragung erfolgte nur über einen Nachrichtenstrom) durch die Wartezeit bei der Neuübertragung bedingt ist. Abbildung 5.12 zeigt die maximale Nachrichtenverzögerung für Bitfehlerraten im Bereich von $0..2 \cdot 10^{6}$ sowohl für Standard-SCTP als auch für das modifizierte SCTP, Abbildung 5.11 zeigt die entsprechenden Werte für Bitfehlerraten zwischen $0..1 \cdot 10^{-5}$. Für höhere Bitfehlerraten BER $> 10^{-6}$ ist die Differenz zwischen der maximalen Nachrichtenverzögerung beim Standard-SCTP und beim modifizier-

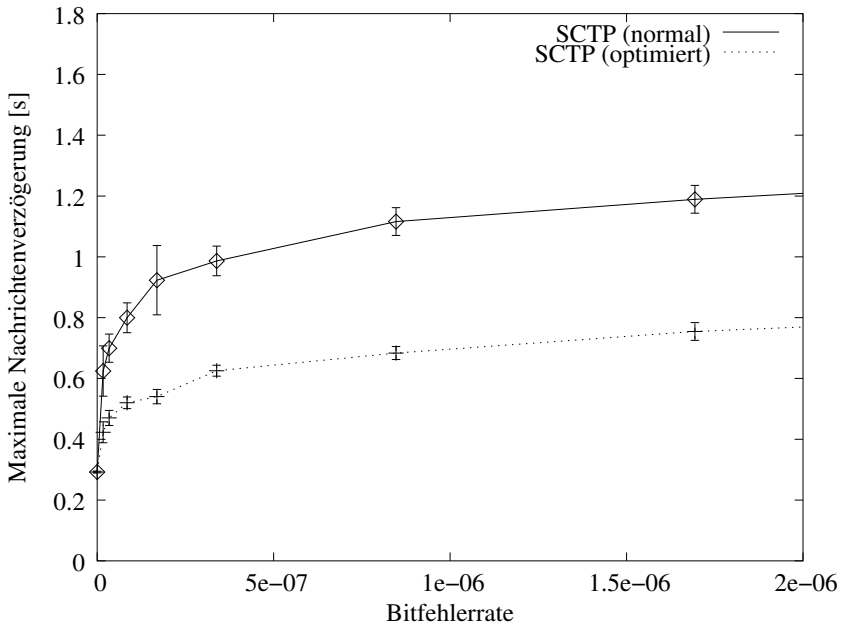

Abbildung 5.12: Vergleich der Verzögerung der Protokollvarianten – Ausschnittvergrößerung

ten SCTP nahezu konstant und beträgt, im Gegensatz zu der in Abschnitt 4.3.3 angeführten Formel 4.6, ca. 460 ms. Da hier die maximalen Nachrichtenlaufzeiten im Mittel betrachtet werden, lässt dies darauf schließen, dass bei der Simulation im Gegensatz zu der Berechnung immer einige Nachrichten mehrfach durch Bitfehler auf dem Link verfälscht und nachfolgend verworfen werden. Bei der Neuübertragung können natürlich wieder Fehler auftreten. Die mittlere Nachrichtenverzögerung, die hier in den Simulationen nicht betrachtet wurde, wird jedoch im Mittel um den in Gleichung 4.6 berechneten Wert von ca. 225 ms verringert.

Eine weitere Modifikation des Standard-SCTP ist denkbar, in der der Empfänger SACKs nur über den Pfad mit der kürzesten RTT an den Sender zurückschickt, solange er Lücken in der Ankunftsreihenfolge der erhaltenen TSNs feststellt. Wenn tatsächlich ein Paketverlust festgestellt wurde und der Sender erfolgreich die Neuübertragung abgeschlossen hat, werden die SACK-Chunks wieder über den Pfad gesendet, auf dem neue Nachrichten empfangen werden. Diese Protokoll-Modifikation kann leicht in zukünftigen Versionen des OPNET SCTP-Modells

implementiert und eingehender untersucht werden. Interessant sind dabei insbesondere die Auswirkungen dieser Modifikation auf die Schätzwerte für die geglättete RTT (SRTT, vgl. Abschnitt 2.3.7.1).

5.6 Untersuchung der Failover-Mechanismen in der Simulation

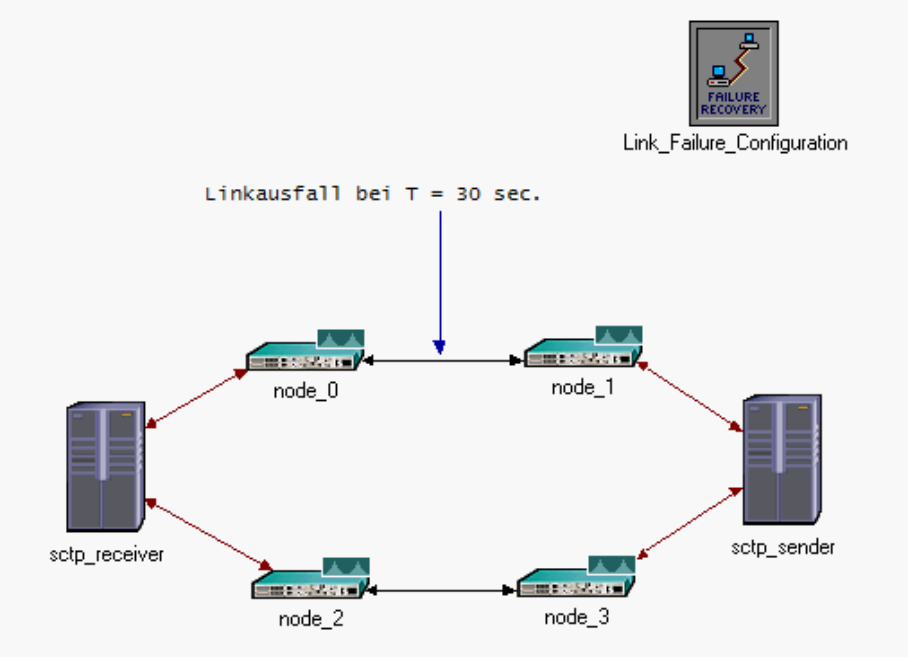

Abbildung 5.13: Modelltopologie zur Untersuchung des Failover-Verhaltens

Die Untersuchungen aus Abschnitt 4.4 hinsichtlich der Parametereinstellungen in Failover-Szenarien wurden für einen größeren Parameterbereich (ARL/PRL$=1\ldots5$, und $\text{RTO}_{max} = 100\ldots500$ ms) an einem geeigneten, realistischen Simulationsmodell (vgl. auch Abbildung 5.13) durchgeführt.
In der Simulation wurden die beiden untersuchten Szenarien – wie im Testbett – durch unterschiedliche Anwendungsmodelle realisiert, von denen nach Feststellen des Pfadausfalls, bzw. nach Feststellen des durch den Pfadausfall verursachten

Assoziationsausfalls, Daten über den Sekundärpfad bzw. die sekundäre Assoziation übertragen wurden. Im Gegensatz zu den Untersuchungen in Abschnitt 4.4 wurde das Ende der Failover-Prozedur angenommen, nachdem die Sendewarteschlangen wieder einen Zustand eingenommen hatten, der auch vor dem Failover bestand (im Wesentlichen den leeren Zustand). Wie in 4.4 beschrieben, betrug die Linkbandbreite auf dem Bottleneck-Link 2,048 MBit/s und die Quelle erzeugte Nachrichten der Länge 500 Bytes, deren Ankunftsabstände einer negativen Exponentialverteilung mit einer mittleren Ankunftsrate von 100 Nachrichten pro Sekunde gehorchten. Die Datenübertragung setzte nach einer Simulationszeit von t_{start} =0 s ein, und bei einer Zeit von t_{fail} =30 s wurde eine Linkunterbrechung simuliert.
Im Laufe der Untersuchungen wurde von uns eine Inkonsistenz in der Spezifikation des RFC 2960 [86] entdeckt, nach der nicht genau festgelegt war, wann der Anwendung ein Pfadausfall signalisiert wird (bei Erreichen oder bei Überschreiten der festgelegten Fehlerschwelle *Path.Max.Retrans*, vgl. auch Abschnitt 2.3.7). Diese Inkonsistenz wurde im SCTP Implementer's Guide [80] behoben. Danach wird nunmehr der Anwendung ein Pfadausfall signalisiert, nachdem die Fehlerschwelle überschritten wurde. Die Änderung der Spezifikation wurde bereits in den in den Abschnitten 4.4.1 und 4.4.2 beschriebenen Auswertungen berücksichtigt.

5.6.1 Pfadausfall bei einer Assoziation mit Dualhoming

Die in Abbildung 5.14 dargestellten Ergebnisse zeigen die Dauer eines Failovers nach Ausfall des Primärpfads bei einer SCTP-Assoziation mit zwei Pfaden. Nachdem der Anwendung der Ausfall von der SCTP-Assoziation angezeigt wird, schaltet diese den Primärpfad um. Die Ergebnisse bestätigen die experimentellen Untersuchungen an der realen Implementierung in unserem Testbett. Einzig die in Abbildung 4.6 dargestellte Kurve für den Parameter PRL= 4 weicht von den Ergebnissen der Simulation ab, da bei den Simulationen ein anderes Kriterium für das Ende des Failovers angenommen wurde, als bei den Messungen an der Implementierung im Testbett (vgl. auch die Erläuterung in Abschnitt 5.6.2).
Die Gesamtdauer des Failovers liegt in dem erwähnten Szenario für die Parameter RTO_{min} =40 ms, PRL=1...2 und RTO_{max} =100...500 ms unter 800 ms, und ist daher prinzipiell für den Transport von SS7-Nachrichten über SCTP geeignet. Dabei ist allerdings zu berücksichtigen, dass bei einer höheren mittleren Last die Längen der Warteschlangen auf der Senderseite schneller anwachsen und damit die Dauer des Failovers zunimmt.

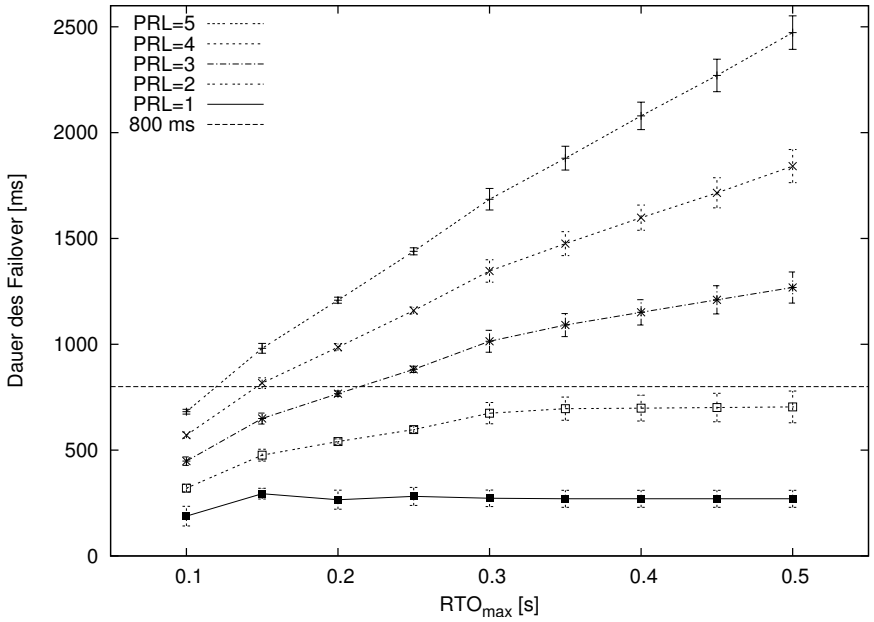

Abbildung 5.14: Failover-Zeiten einer Assoziation mit zwei Pfaden

Die auf den Abbildungen 5.14 und 5.15 dargestellten Graphen zeigen für die simulierten Parameterpunkte Konfidenzintervalle von 99%. Dabei weist die Dauer des Failovers regelmäßig Konfidenzintervalle im Bereich von mindestens ± 18ms bis maximal ± 80ms auf. Die entsprechenden Konfidenzintervalle für die maximale Nachrichtenverzögerung liegen im Bereich von mindestens ± 12ms bis maximal ± 86ms.

Die besonders großen Konfidenzintervalle für die Einstellung des Parameters PRL$= 1$ in Abbildung 5.15 sind bedingt durch die Tatsache, dass bei dieser Einstellung in einer Vielzahl von Simulationen fälschlicherweise ein Pfadausfall erkannt wurde, teilweise auch auf dem Sekundärpfad. Daher konnten von 20 ausgeführten Simulationen pro Parameterpunkt im Mittel bei dieser Einstellung nur 5 über den Simulationszeitraum von 120 Sekunden zu Ende geführt werden, ohne dass die Assoziation vorher beendet wurde (unabhängig von der Einstellung des Parameters RTO$_{max}$). Die entsprechende Parameter-Einstellung PRL$= 1$ kann deshalb nicht empfohlen werden.

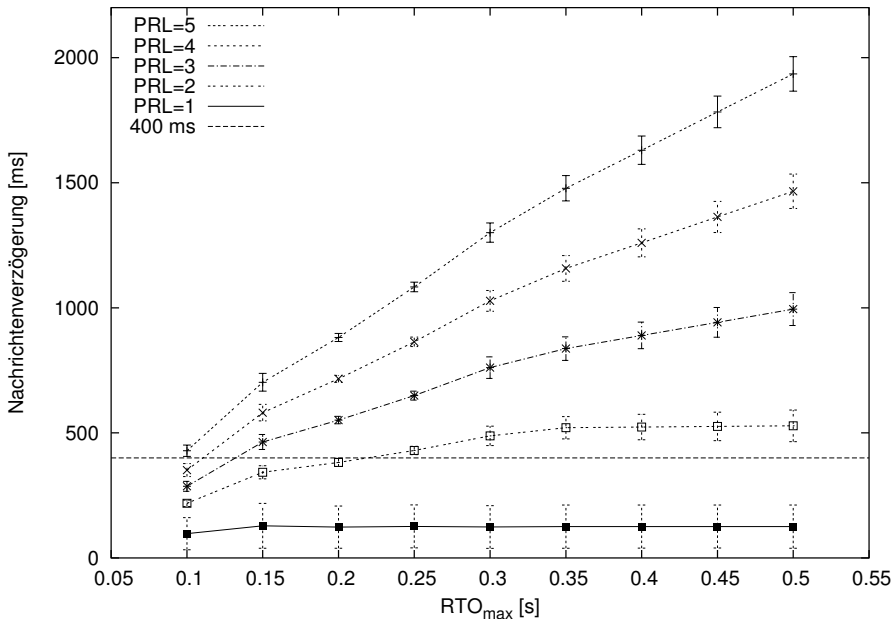

Abbildung 5.15: Maximale Nachrichtenverzögerung einer Assoziation mit zwei Pfaden

5.6.2 Pfadausfall bei zwei Assoziation mit Singlehoming

Auch die in der Abbildung 5.17 dargestellten Ergebnisse bestätigen die experimentellen Untersuchungen an der realen Implementierung in unserem Testbett (vgl. Abschnitt 4.4.2). Die Ergebnisse stimmen nahezu exakt mit den in Abbildung 4.9 dargestellten Ergebnissen überein. Die in der Simulation festgestellten 99%-igen Konfidenzintervalle für die Verteilung der maximalen Nachrichtenverzögerung liegen alle im Bereich ± 6 ms.

Im Gegensatz dazu weichen die Ergebnisse aus Abbildung 4.10 von den Simulationsergebnissen, die in Abbildung 5.17 dargestellt sind, ab. Die Bestimmung der jeweiligen Dauer eines Failovers basiert in den Simulationsszenarien, wie bereits oben erwähnt, auf der Betrachtung der Länge der Sendewarteschlangen. Nach Erkennung des Assoziationsausfalls werden alle Nachrichten, die nicht über die erste Assoziation übertragen werden konnten, in die Sendewarteschlange der zweiten Assoziation geschrieben, von der sie nachfolgend übertragen werden. Wenn die Länge dieser Warteschlange unter einen Schwellwert sinkt, wird in der Simulati-

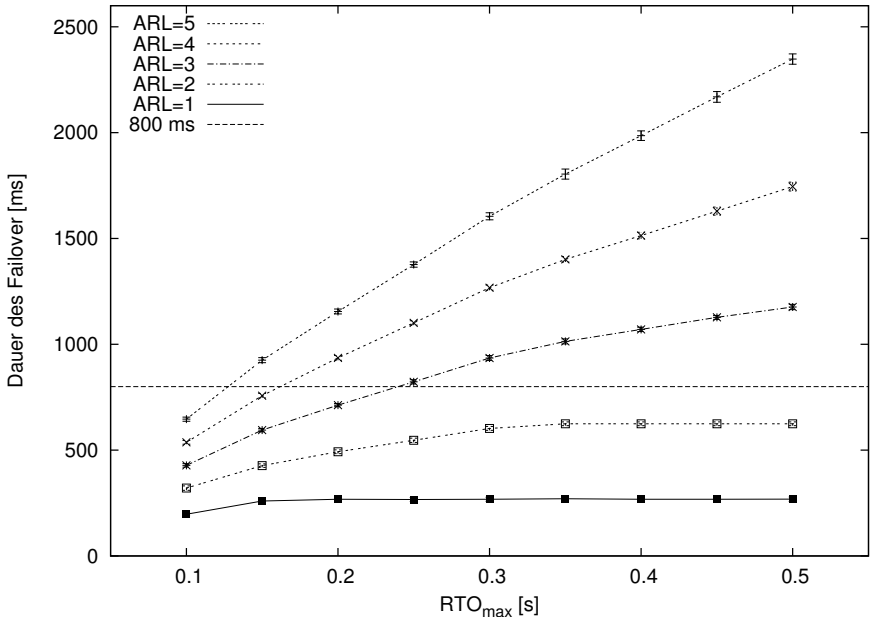

Abbildung 5.16: Failover-Zeiten bei zwei Assoziationen (single-homed)

on davon ausgegangen, dass die Failover-Prozedur abgeschlossen ist und die entsprechende Dauer des Failovers berechnet. Im Gegensatz dazu wurde in den Untersuchungen in Abschnitt 4.4 das Protokollverhalten bezogen auf den Empfänger betrachtet. Dabei wurde angenommen, dass die Failover-Prozedur abgeschlossen ist, nachdem die mittlere Nachrichtenverzögerung wieder auf das vor dem Failover festgestellte Maß zurückgegangen war.

Auffällig ist, dass das Protokollverhalten in diesem Szenario reproduzierbarer scheint, und entsprechend die 99%-igen Konfidenzintervalle wesentlich kleiner (zwischen ± 7 ms und ± 25 ms) sind, als beim Szenario mit Dualhoming und zwei Pfaden. Dies ist dadurch bedingt, dass der Zustand bei nur einem Pfad wesentlich genauer definiert ist, wenn alle Neuübertragungen über den einzigen Pfad der ersten Assoziation erfolglos bleiben und erst nach Erkennung des Assoziationsausfalls und einer erfolgreichen Neuübertragung der Daten über die zweite Assoziation, diese ausschließlich genutzt wird.

Abschließend bleibt zu sagen, dass in beiden Szenarien die Failover-Prozedur innerhalb von weniger als 800 ms erfolgreich abgeschlossen werden kann, vorausgesetzt, dass entweder der Parameter RTO_{max} niedrig genug (kleiner als 150 ms),

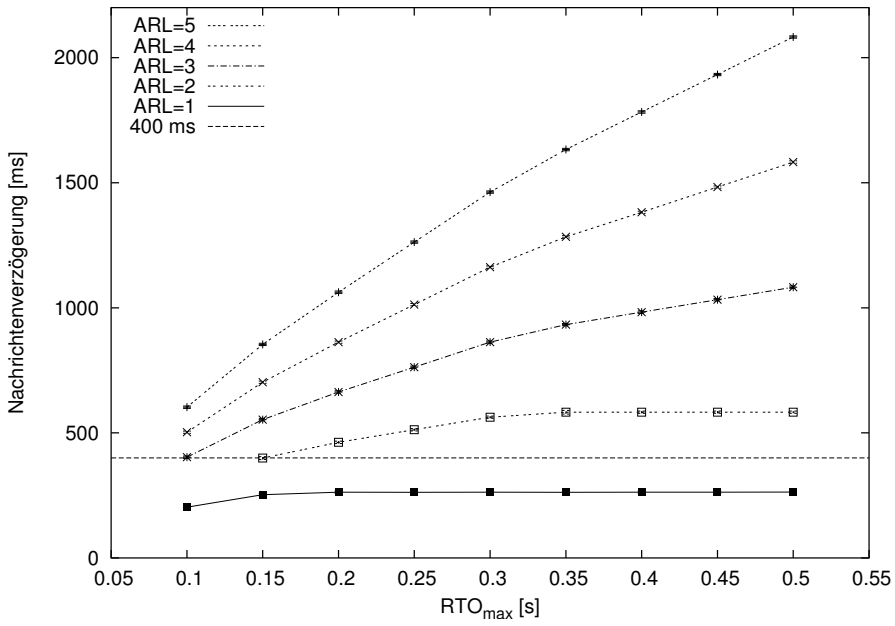

Abbildung 5.17: Maximale Nachrichtenverzögerung bei zwei Assoziationen (single-homed)

oder aber die Parameter ARL/PRL klein genug gewählt werden (z.B. ARL/PRL ≤ 2). Darüber hinaus werden im Szenario mit einer dual-homed Assoziation etwas geringere maximale Nachrichtenverzögerungen erzielt, während im Szenario mit zwei single-homed Assoziationen die Gesamtdauer der Failover-Prozedur etwas geringer ausfällt.

Kapitel 6

Untersuchung von Algorithmen zur simultanen Pfadnutzung

In diesem Kapitel werden SCTP-Varianten diskutiert, die bei einer Assoziation mit Multihoming die Daten auf mehrere vorhandene Pfade verteilen. Im Gegensatz dazu nutzt das Standard-SCTP nur einen Primärpfad zur Datenübertragung und alternative Pfade ausschließlich als Backup sowie zur schnellen Neuübertragung.
Damit die Verteilung der Daten auf mehrere Pfade effektiv gelingt, müssen einige der zentralen Algorithmen des RFC 2960 [86] für die Fluss- und Überlastkontrolle, sowie für die schnelle Neuübertragung an die veränderten Gegebenheiten der simultanen Pfadnutzung angepasst werden. Dabei ist insbesondere der Effekt der Paketüberholung durch unterschiedliche Pfadcharakteristika zu berücksichtigen. Paketüberholungen *können* zwar auch auftreten, wenn Daten nur über einen Pfad übertragen werden [61], bei der simultanen Nutzung mehrerer Pfade ist dies aber die Regel und somit ein zu erwartendes und normales Phänomen.
Im Folgenden wird aufgezeigt, welches Verhalten eine standardkonforme Implementierung des SCTP bei Nutzung mehrerer Pfade durch periodische Änderung des Primärpfades aufweist und welche Probleme dabei auftreten können. In Abschnitt 6.3 werden einige Lösungsstrategien für diese Effekte erörtert, die von Iyengar et al. in [39] vorgeschlagen werden. In Abschnitt 6.4 werden einige Vorschläge zur Verbesserung der von Iyengar et al. diskutierten SCTP-Varianten gemacht.
Die verschiedenen Protokollvarianten werden in Abschnitt 6.5 mit Hilfe des in Kapitel 5.3 beschriebenen und entsprechend modifizierten Simulationsmodells

untersucht. Abschnitt 6.6 schließt mit einer Bewertung der vorgeschlagenen Modifikationen hinsichtlich ihrer Eignung für einen speziellen oder universellen Einsatz.

6.1 Überblick über Lastverteilungsszenarien

In der gegenwärtigen Form sehen der RFC 2960 und der "SCTP Implementer's Guide" [80, 86] nicht vor, dass mehrere Pfade einer SCTP-Assoziation nebeneinander zur erstmaligen Übertragung neuer Daten genutzt werden. Danach werden neue Daten erstmalig ausschließlich über den Primärpfad und nur erneut übertragene Daten über sekundäre Pfade gesendet.
Allerdings sieht der RFC 2960 vor, dass die Anwendung (prinzipiell zu einem beliebigen Zeitpunkt) den Primärpfad ändern kann. Außerdem kann bei jedem Aufruf eines SEND-Primitivs für eine zu sendende Nachricht ein Zielpfad oder eine Zieladresse ausgewählt werden (vgl. auch die Funktionen `sctp_send()` in Abschnitt A.1 oder `sendmsg()` in Abschnitt A.2).
Dies hat in der Regel aber einen negativen Einfluss auf das Protokollverhalten, denn aufgrund von unterschiedlichen Pfadcharakteristika (z.B. bei unterschiedlichen Pfadverzögerungen) kommt es beim Empfänger regelmäßig zur Bildung von Lücken in der Ankunftsreihenfolge der TSNs. In diesem Fall sendet der Empfänger gemäß RFC 2960 pro neuer Nachricht einen SACK-Chunk zur Bestätigung an den Sender. Sobald der Empfänger bei vorhandenen Lücken in der Ankunftsreihenfolge mehr als vier Pakete mit Daten-Chunks erhalten und diese entsprechend mit vier SACK-Chunks, die Gap-Reports enthalten, beantwortet hat (vgl. Abschnitt 2.3.5), kommt es beim Sender in der Regel zu einer schnellen Neuübertragung (vgl. Abschnitt 2.3.5.1). Die dabei vorgenommene Reduktion des Überlast-Sendefensters $cwnd$ und der Überlast-Schwelle $ssthresh$ (vgl. auch Abschnitt 2.3.8) wird also durch Paketüberholung – und nicht Paketverluste – verursacht und ist daher in der Regel unnötig. Bei unterschiedlichen Pfadcharakteristika kommt es insbesondere am Anfang einer Verbindung zu mehrfachen Reduktionen dieser Parameter, was nachfolgend den Durchsatz weit unter das normale Maß verringert.
Neben dem Phänomen der übermäßigen Reduktion der Parameter $cwnd$ und $ssthresh$ bei naiver Auswahl der alternierenden Zielpfade diskutieren Iyengar et al. in [40] das Verhalten des Standard-SCTP beim Umschaltvorgang auf einen neuen Primärpfad. Sie nehmen an, dass bei diesem Umschaltvorgang eine Reihe von Daten-Chunks noch über den alten Pfad und gleichzeitig bereits neue Daten-

Chunks über den neuen Primärpfad gesendet werden. Auch hier wird der Sender nach Erhalt von vier aufeinander folgenden SACK-Chunks, welche Lücken in der Ankunftsreihenfolge der erhaltenen TSNs anzeigen, fälschlicherweise eine schnelle Neuübertragung über einen alternativen Pfad durchführen (vgl. auch Abschnitt 2.3.5.1). Dieser ist bei einem Szenario mit Dualhoming gleichzeitig der neue Primärpfad. In der Folge werden die Steuerungs-Parameter $cwnd$ und $ssthresh$ des alten Primärpfads gemäß den Regeln aus Abschnitt 2.3.8 reduziert, so dass das $cwnd$ halbiert wird und $ssthresh$ auf den neuen $cwnd$-Wert gesenkt wird. Damit befindet sich der alte Primärpfad mit reduziertem Sendefenster im Bereich der Überlast-Vermeidung.
Erhält der Sender nachfolgend eine Bestätigung der fälschlicherweise neu übertragenen TSNs, so wird die Anzahl der darin bestätigten Bytes dem neuen Primärpfad zugeschrieben (über den die TSNs unnötigerweise neu gesendet wurden), und sein Sendefenster $cwnd$ wird zusätzlich erhöht. Caro et al. nennen dieses Verhalten engl. Congestion Window Overgrowth (übermäßiges Wachstum des Überlast-Sendefensters) und schlagen in [40] Maßnahmen vor, um diesem Effekt zu begegnen. Diese werden in Abschnitt 6.3 ausführlicher diskutiert.

6.2 Mechanismen bei gleichzeitiger Nutzung mehrerer Pfade

Im Folgenden werden die zu lösenden Probleme auf der Sender- und Empfängerseite diskutiert, bevor im weiteren Verlauf dieses Kapitels Mechanismen zur optimalen Nutzung mehrerer Pfade einer SCTP-Assoziation mit Multihoming erörtert werden. Dabei wird davon ausgegangen, dass zentrale Algorithmen des Standard-SCTP modifiziert werden können. Insgesamt sollen aber die Funktionen zur Fluss- und Überlastkontrolle so beibehalten werden, dass ein Einsatz des modifizierten SCTP in bestehenden Netzen in Konkurrenz zu etablierten Protokollen wie TCP oder Standard-SCTP, die die Größe des Sendefensters durch AIMD-Algorithmen (vgl. auch Abschnitt 2.3.8) regeln und dem RFC 2914 (“Congestion Control Principles”) [22] entsprechen, möglich ist.

6.2.1 Verteilung der Daten beim Sender

Prinzipiell gibt es für den Sender zwei Möglichkeiten, wechselnde Zielpfade auszuwählen: zum einen kann er es der Anwendung überlassen, den gewünschten

Zielpfad anzugeben, und passt nachfolgend die Algorithmen der Fluss- und Überlaststeuerung geeignet an die Nutzung alternierender Pfade an, um die in Abschnitt 6.1 beschriebenen Effekte zu vermeiden. Diese Methode ist jedoch ineffektiv, da die Wahl der Zieladresse jeweils von der Anwendung getroffen werden muss, die nicht in jedem Fall die notwendige Kenntnis der relevanten Parameter (wie z.B. $cwnd$, Rundlaufzeit etc.) besitzt und auch nicht besitzen soll.

Um die Anwendung von dieser Problematik zu entlasten, kann andererseits das SCTP Mechanismen zur Pfadauswahl implementieren. Diese Variante wird im Folgenden detaillierter untersucht und allgemein mit dem Namen LS-SCTP bezeichnet, wobei LS engl. für *Load Sharing*[1] steht, was so viel wie Lastverteilung bedeutet.

LS-SCTP kann alternierend den Primärpfad umschalten, z.B. wenn das Sendefenster auf dem aktuellen Primärpfad voll ausgeschöpft ist und auf diesem Pfad zu einem gegebenen Zeitpunkt keine neuen Daten mehr gesendet werden können. Bezeichnet p den aktuellen Primärpfad, $osb(p)$ die Anzahl der auf dem Pfad p gesendeten, aber noch unbestätigten Bytes und $cwnd(p)$ das aktuelle (Überlast-) Sendefenster für den Pfad p, so ist die Bedingung für den Umschaltvorgang in diesem Fall $osb(p) \geq cwnd(p)$. Der Umschaltvorgang auf einen neuen Primärpfad i sollte allerdings nur geschehen, wenn auf diesem Pfad i gilt : $osb(i) < cwnd(i)$. Dadurch ist gewährleistet, dass nach dem Umschalten auch neue Daten gesendet werden können. Ansonsten wird kein Umschaltvorgang vorgenommen.

Existieren n Pfade zum Empfänger, so gilt bezüglich des Sendefensters $rwnd$ des Empfängers:

$$\sum_{j=1}^{n} osb(j) \leq rwnd \tag{6.1}$$

Geht man von einem gesättigten Sender aus, so resultiert aus diesem Algorithmus, dass der Sender über jeden Pfad solange Daten sendet, wie es das Sendefenster (bzw. Empfängerfenster) $rwnd$ erlaubt und bis das Sendefenster des jeweiligen Pfades ausgeschöpft ist. Anschließend schaltet er dann auf einen Pfad um, dessen Sendefenster das weitere Senden neuer Daten erlaubt. Das Sendefenster wird demnach, entsprechend der jeweils aktuellen $cwnd$-Werte der einzelnen Pfade, aufgeteilt.

[1] Der Begriff *Load Sharing* wird in verschiedenen Zusammenhängen benutzt. Hier ist die simultane Nutzung mehrerer SCTP-Pfade gemeint, mit dem Zweck, insgesamt einen höheren Protokolldurchsatz zu erreichen und die Last auf diese Pfade zu verteilen.

6.2.2 Reaktion des Empfängers

Der Empfänger sendet beim Standard-SCTP nach Erhalt von je zwei Paketen mit Daten-Chunks einen SACK-Chunk zur Bestätigung, in der Regel an die Adresse, von der er das letzte Paket erhielt. Stellt er Lücken (oder Duplikate) in der Ankunftsreihenfolge der empfangenen TSNs fest, so wird, solange diese Lücken vorhanden sind, für jedes empfangene Paket mit Daten-Chunks sofort ein SACK-Chunk an den Sender geschickt.
Im Falle des LS-SCTP sind Lücken in der Ankunftsreihenfolge in der Regel kein Zeichen für eine Vertauschung oder Verlust von Paketen, sondern resultieren aus den unterschiedlichen Charakteristika der verwendeten Pfade. Sie kommen daher regelmäßig vor und sind keine Ausnahme. Aus diesem Grund sollte ein Empfänger, der LS-SCTP unterstützt, andere Mechanismen verwenden und SACK-Chunks auch bei auftretenden Lücken in der Ankunftsreihenfolge der empfangenen TSNs nur nach zwei Paketen mit Daten-Chunks senden. Darüber hinaus kann er eine Heuristik anwenden, nach der z.B. ein Paketverlust angenommen wird, nachdem eine bestimmte Anzahl von Daten-Chunks den im SACK-Chunk übertragenen Wert des CTSNA nicht erhöht haben (paketbasierend), oder nachdem der CTSNA-Wert eine bestimmte Zeit lang nicht erhöht wurde, obwohl weitere Pakete mit Daten-Chunks empfangen wurden (zeitbasierend).
Eine weitere entscheidende Möglichkeit des Empfängers, das Verhalten des Senders zu beeinflussen, liegt in der Wahl der Adresse, an die der SACK-Chunk gesendet wird. Dabei sind mehrere Optionen denkbar:

1. Damit der Sender die Änderungen des $cwnd$-Wertes an die Charakteristik des jeweilig verwendeten Pfades anpassen kann (so wie dies auch bei ausschließlicher Nutzung eines einzigen Primärpfades geschieht), sollte der Empfänger nach Erhalt zweier Pakete mit Daten-Chunks auf *einem* Pfad sofort einen SACK-Chunk an die zugehörige Absende-Adresse der Daten-Pakete zurücksenden, unabhängig von auftretenden Lücken). Im Gegensatz dazu betrachtet Standard-SCTP die Anzahl der erhaltenen Daten-Pakete *pro Assoziation* und nicht pro Pfad.

2. Der Empfänger kann alle SACK-Chunks an den Pfad mit der kürzesten gemessenen Rundlaufzeit (RTT) senden, was, wie bereits in Abschnitt 4.3.3 gezeigt, unter bestimmten Bedingungen einen positiven Einfluss auf das Protokollverhalten haben kann.

3. Der Empfänger kann die SACK-Chunks über alle Pfade verteilen und reihum über alle Pfade an den Sender schicken. Dies würde dazu führen,

dass der Sender Messungen der Rundlaufzeit für alle möglichen Adress-Kombinationen durchführen könnte. Die Ankunftsreihenfolge der SACK-Chunks beim Sender ist in diesem Falle jedoch noch stärker abweichend von der Sendereihenfolge der Daten, als dies ohnehin der Fall sein würde. Aus diesem Grund wird diese Variante nachfolgend nicht weiter betrachtet.

6.2.3 Reaktionen des Senders auf Bestätigungen

Die Nutzung von mehreren Pfaden bei der Datenübertragung mit Standard-SCTP führt dazu, dass die Daten überlappend und nicht in der Absendereihenfolge beim Empfänger eintreffen. Die Bestätigungen des Empfängers enthalten daher Lücken (vgl. Abschnitt 2.3.5), die vom Sender nachfolgend als Paketverlust interpretiert werden können (wenn vier aufeinander folgende SACK-Chunks identische TSNs als *nicht erhalten* markieren). Als Reaktion auf den vermeintlichen Paketverlust reduziert der Standard-SCTP Sender fälschlich das Sendefenster, bzw. die Parameter $cwnd$ und $ssthresh$.
LS-SCTP muss daher einerseits Strategien implementieren, die es erlauben, Änderungen der Ankunftsreihenfolge beim Empfänger, die aufgrund der Übertragung über unterschiedliche Pfade zustande kommen, von tatsächlichen Paketverlusten zu unterscheiden. Darüber hinaus müssen die Regelmechanismen für die entscheidenden Pfadparameter $cwnd$ und $ssthresh$ beim Sender so angepasst werden, dass sie möglichst Werte annehmen, die sie auch annehmen würden, wenn der zugehörige Pfad als ausschließlicher Primärpfad genutzt würde, oder, falls dies nicht möglich ist, Werte, die den Nutzdurchsatz optimieren oder die Last optimal verteilen.

6.3 Lastverteilung mit der Changeover-Aware Congestion Control

Iyengar et al. diskutieren in [41] Strategien zur gleichzeitigen Nutzung mehrerer Pfade in einer SCTP-Assoziation mit Multihoming. Dort definieren sie verschiedene Algorithmen, die nachfolgend detaillierter beschrieben werden. Diese werden insgesamt engl. als *Changeover Aware Congestion Control* (CACC) bezeichnet. Die CACC-Algorithmen nutzen zusätzliche Variablen, mit denen Zustände gespeichert werden können, die einen bestimmten Pfad betreffen. Bezeichnet *d(i)* den *i*-ten Pfad zum Ziel und *k* eine solche Variable, so steht *d(i).k* die zu diesem

Zielpfad *d(i)* zugehörige Variable.
Die im Folgenden benutzten Variablen sind:

1. *d(i).saw_new_path_ack*. Bool'sche Variable, die anzeigt, ob ein empfangener SACK-Chunk neue, über den Pfad *d(i)* gesendete, TSNs bestätigt, oder nicht.

2. *d(i).highest_path_tsn_acked*. Enthält die höchste durch einen neu empfangenen SACK-Chunk bestätigte TSN für den Pfad *d(i)*.

3. *d(i).pCTSNA*. Für den Pfad *d(i)* wird die höchste TSN gesucht, die ohne Unterbrechung der (auf *d(i)* gesendeten) Reihenfolge der TSNs bestätigt wurde. Der gefundene Wert wird in diesem Parameter gespeichert.

4. *d(i).new_pseudo_cumack*. Bool'sche Variable, die anzeigt, ob der *pCTSNA*-Wert des Pfades *d(i)* durch einen empfangenen SACK-Chunk erhöht wurde.

5. *d(i).find_expected_pseudo_cumack*. Flag, welches angibt, ob weiter nach einem neuen, höheren erwarteten *pCTSNA*-Wert für den Pfad *d(i)* gesucht wird, oder ob die Reihenfolge der ohne Unterbrechung auf diesem Pfad gesendeten und bestätigten TSNs bereits unterbrochen wurde.

Im Folgenden werden kurz die in [41] beschriebenen Algorithmen vorgestellt und diskutiert.

6.3.1 CACC mit Vermeidung unnötiger Neuübertragungen

Split Fast Retransmit Changeover Aware Congestion Control (SFR-CACC) [41, 42] bezeichnet einen Algorithmus zur weitgehenden Vermeidung/Reduktion von falschen bzw. unnötigen Neuübertragungen (engl. Spurious Retransmissions). Dazu wird der Algorithmus, nach dem der ursprüngliche Sender von Daten-Chunks (und Empfänger von SACK-Chunks) entscheidet, ob der Gap Counter (vgl. Abschnitt 2.3.5.1) eines Daten-Chunks erhöht wird oder nicht, modifiziert. Dieser Algorithmus greift immer dann, wenn die TSN des Chunks durch eine Lücke im SACK-Chunk als nicht erhalten markiert wurde. Es wird davon ausgegangen, dass Chunks in Gruppen über einen Pfad übertragen werden, bevor ein Umschaltvorgang auf einen neuen Primärpfad durchgeführt wird.
Wird ein SACK-Chunk empfangen, der Lücken anzeigt, so werden alle Daten-Chunks der Speicherliste (engl. Retransmission Queue, vgl. auch Abschnitt 5.3.2) betrachtet, die durch diesen SACK-Chunk *erstmalig* bestätigt werden.

- Für alle Pfade *i*, initialisiere das Flag *d(i).saw_new_path_ack* = FALSE.

- Für alle Pfade *i*, über die ein Daten-Chunk gesendet, welcher durch das SACK neu bestätigt wurde, setze *d(i).saw_new_path_ack* = TRUE.
- Für alle Pfade *i*, für die gilt: (*d(i).saw_new_path_ack* == TRUE), bestimme die höchste TSN, die neu bestätigt wurde. Diese wird in dem Pfad-Parameter *d(i).highest_path_tsn_acked* gespeichert.
- Untersuche alle TSNs der Speicherliste, ob sie in einer Lücke enthalten sind: ist der Chunk mit der TSN t in einer Lücke enthalten und d_t der Zielpfad, über den er gesendet wurde, so wird der Gap Counter für t erhöht, wenn gilt: d_t.*saw_new_path_ack* == TRUE und $t < d_t$.*highest_path_tsn_acked*.

Die Anwendung dieses Algorithmus führt dazu, dass bei Bildung von Lücken in der Reihenfolge der bestätigten TSNs diese tatsächlich nur dann für die Neuübertragung berücksichtigt werden, wenn auf dem zugehörigen Pfad schon Daten mit höheren TSNs bestätigt wurden. Damit können überflüssige Neuübertragungen beim LS-SCTP vermieden werden.

6.3.2 CACC mit Aktualisierungen des $cwnd$

Die alleinige Anwendung der SFR-CACC führt dazu, dass das Wachstum des $cwnd$, das zum Erreichen eines optimalen Durchsatzes erforderlich ist, nicht für alle Pfade gleichmäßig erfolgt. Bei diesem Algorithmus wird nämlich nur dann ein $cwnd$-Update durchgeführt, wenn der CTSNA-Wert der Assoziation durch ein SACK erhöht wird. Dabei wächst in der Regel das $cwnd$ eines Pfades (in der Regel das $cwnd$ des Pfades mit der höchsten Rundlaufzeit), während andere Pfade weniger am $cwnd$-Wachstum teilhaben.
Daher schlagen Iyengar et al. in [41] eine Erweiterung für SFR-CACC vor, die einen Parameter *pCTSNA* einführt, der den CTSNA-Wert *bezogen auf einen Pfad* angibt. Dies erlaubt es einer Implementierung festzustellen, ob durch eine neue Bestätigung der *pCTSNA*-Wert des Pfades erhöht wurde und in diesem Fall das Überlastsendefenster $cwnd$ dieses Pfades entsprechend der Menge der neu bestätigten Daten anzupassen. Diese Erweiterung wird in [41] als SFR-CACC mit $cwnd$-Update (SFR-CACC-CU) bezeichnet. Iyengar et. al. nennen diese Variante aber auch SCTP mit “Concurrent Multipath Transfer”; im Folgenden wird sie daher einfach mit *CMT* bezeichnet.
Ein kurzes Beispiel soll die Funktion des CMT Algorithmus verdeutlichen: Man stelle sich eine Assoziation mit zwei Pfaden *d(0)* und *d(1)* vor. Der Sender sendet die Daten-Chunks mit den TSNs 1, 2 und 3 sowie 7, 8 und 9 über den langsameren

Pfad *d(0)* und die Daten-Chunks mit den TSNs 4, 5 und 6 sowie 10, 11 und 12 über den schnelleren Pfad *d(1)*. Zu einem späteren Zeitpunkt erhält der Sender einen SACK-Chunk mit dem Wert *ctsna*=7 und dem Gap-Report $(10, 11)$. Dann ist der Parameter *d(0).pCTSNA*=7 und der Parameter *d(1).pCTSNA*=11.
Der CMT Algorithmus nutzt zur Bestimmung der jeweiligen *pCTSNA*-Werte aller Pfade die Flags *new_pseudo_cumack* und *find_expected_pseudo_cumack*, sowie die Historie der Zielpfade aller Chunks in der Speicherliste. Zu Beginn wird für alle *i* das Flag *d(i).find_expected_pseudo_cumack* = TRUE gesetzt. Bei Empfang eines SACK-Chunks werden zusätzlich zu den im vorigen Abschnitt 6.3.1 beschriebenen Schritten die folgenden Aktionen ausgeführt:

- Setze für alle Pfade *i* die Variable *d(i).new_pseudo_cumack* = FALSE.
- Wenn durch den SACK-Chunk ein neuer CTSNA-Wert angezeigt wird, setze für jeden Daten-Chunk mit der TSN n, der durch diesen SACK-Chunk erstmalig bestätigt wurde, das Flag p_n.*find_expected_pseudo_cumack* = TRUE und das Flag p_n.*new_pseudo_cumack* = TRUE. (p_n bezeichnet dabei den Pfad, über den der Chunk mit der TSN n ursprünglich gesendet wurde).
- Enthält der SACK-Chunk auch Gap-Reports, so werden alle Chunks r der Speicherliste (Retransmission Queue) untersucht; dabei sei p_r der Pfad, über den ein Chunk r gesendet wurde.
  - Wenn r bisher noch nicht bestätigt wurde und gilt:

    (p_r.*find_expected_pseudo_cumack* == TRUE),

    dann setze

    p_r.*find_expected_pseudo_cumack* = FALSE,

    und

    p_r.*pCTSNA* = r.
  - Wenn r erstmalig bestätigt wurde und gilt:

    p_r.*pCTSNA* == r,

    dann setze die Flags

    p_r.*find_expected_pseudo_cumack* = TRUE,

    und

    p_r.*new_pseudo_cumack* = TRUE.

- Für alle i, für die gilt:

  (*d(i).new_pseudo_cumack* == TRUE),

  führe ein Update des $cwnd$ durch, entsprechend der Längen der neu bestätigten Chunks.

6.3.3 CACC mit verzögerter Bestätigung

Standard-SCTP fordert, dass der Empfänger bei Feststellen von Lücken in der Ankunftsreihenfolge empfangener Daten pro ankommendem Paket einen SACK-Chunk sendet, um den Sender möglichst schnell zu informieren und gegebenenfalls eine schnelle Neuübertragung auszulösen (vgl. Abschnitt 2.3.5.1). Sind keine Lücken vorhanden, so wird ein SACK-Chunk pro zwei ankommenden Paketen mit Daten-Chunks gesendet.
Da bei LS-SCTP Lücken in der Ankunftsreihenfolge jedoch die Regel sind, schlagen Iyengar et al. in [41] schließlich noch einen Algorithmus vor, nach dem der Empfänger auch bei Paketüberholungen und nachfolgenden Lücken in der Ankunftsreihenfolge Bestätigungen in SACK-Chunks verzögert sendet, d.h. je zwei ankommenden Paketen mit Daten-Chunks wird ein SACK-Chunk gesendet. Zusätzlich berücksichtigen sie in ihrem Algorithmus die Möglichkeit tatsächlicher Paketverluste, die durch zusätzliche Flags in den SACK-Chunks schneller erkannt werden können. Damit benötigt der Empfänger nicht in jedem Fall vier SACKs zur Entscheidung für eine schnelle Neuübertragung, sondern kann diese u.U. auch schon nach zwei SACKs mit entsprechend gesetzten Flags durchführen.

6.4 Modifikationen des CACC

Im Folgenden werden weitere im Laufe dieser Arbeit entstandene Varianten zur Optimierung der von Iyengar et al. in [41] diskutierten Algorithmen erläutert. Sie basieren auf der Idee, die Behandlung von Daten- und SACK-Chunks sowohl auf der Sender- als auch auf der Empfängerseite an die Nutzung mehrerer Pfade anzupassen.

6.4.1 Pfadbezogene Bestätigungen

Um ein optimales, an die Gegebenheiten der jeweiligen Pfade angepasstes, Verhalten und Wachstum der $cwnd$-Parameter aller Pfade beim Sender zu erreichen,

wird im Folgenden eine Modifikation des Verhaltens sowohl bei der Senderinstanz als auch bei der Empfängerinstanz vorgeschlagen. Der Sender benutzt den in Abschnitt 6.2.1 für das LS-SCTP beschriebenen Algorithmus, um neu zu sendende Daten-Chunks auf die gegebenen Pfade zu verteilen. Der Empfänger führt *pro Pfad zum Sender* einen Zähler, nachfolgend *path_sack_counter* genannt. Dabei bezeichnet *d(i).path_sack_counter* den entsprechenden Zähler für den Pfad *d(i)*. Er wird mit *Null* initialisiert.
Erhält der Empfänger ein SCTP-Paket mit Daten-Chunks auf dem Pfad *d(i)*, so startet er für diesen Pfad den Timer *d(i).path_sack_timer*, sofern dieser noch nicht läuft und inkrementiert den Zähler *d(i).path_sack_counter* um eins. Nimmt ein Zähler *d(i).path_sack_counter* den Wert zwei an, oder läuft der Timer *d(i).path_sack_timer* ab, so wird der entsprechende *d(i).path_sack_counter* auf den Wert *Null* zurückgesetzt und sofort ein SACK-Chunk über den Pfad *d(i)* an den Sender zurückgeschickt. Bei einem Sender gemäß Standard-SCTP, der nur einen einzigen Primärpfad nutzt, und einem Netz, in dem keine Paketverluste oder Paketüberholungen vorkommen, entspricht dieser Algorithmus dem RFC 2960 [86]. Im Gegensatz zu den in Abschnitt 6.3 beschriebenen Algorithmen wird hier also ein SACK für zwei auf einem Pfad empfangene Pakete mit Daten-Chunks gesendet und nicht für zwei empfangene Pakete mit Daten-Chunks pro Assoziation. Die Ergebnisse des hier beschriebenen Algorithmus werden in Abschnitt 6.5.3 vorgestellt und mit den Ergebnissen des CMT Algorithmus verglichen.

6.4.2 Senden von SACKs über den schnellsten Pfad

Eine weitere Variante, die bereits in Abschnitt 6.2.2 erwähnt wurde, ist der Algorithmus, bei dem der Empfänger alle Bestätigungen über den Pfad mit der kürzesten gemessenen Rundlaufzeit sendet. Dadurch sinkt die Rundlaufzeit für alle Pakete mit Daten-Chunks, und das $cwnd$ aller Pfade kann schnell wachsen.
Andere Varianten senden die SACK-Chunks über Pfade mit unterschiedlichen Laufzeiten zurück. Dies führt zu einer Vertauschung der SACKs. Bei dieser Variante kommen zwar die Daten möglicherweise vertauscht beim Empfänger an, dieser sendet jedoch alle SACK-Chunks über denselben Pfad zurück. Mithin werden diese nicht auch noch vertauscht. Das führt insgesamt zu einem homogenen Wachstum der $cwnd$-Werte aller Pfade und optimiert den Prozess des *Sliding Window*. Der Sender kann so schnell Daten aus dem Retransmission Buffer löschen und neue Daten senden. Allerdings verhält sich diese Protokollvariante nicht so robust beim Ausfall dieses schnellsten Pfades, was zunächst zu einem Ausbleiben aller Bestätigungen führen würde. Die Ergebnisse entsprechender Simulationen

mit dem hier diskutierten Algorithmus werden in Abschnitt 6.5.3 diskutiert.

6.5 Simulative Bewertung von Lastverteilungsalgorithmen

Die in den Abschnitten 6.3 und 6.4 diskutierten Lastverteilungsalgorithmen wurden mit Hilfe geeigneter OPNET-Modelle in umfangreichen Simulationen untersucht. Beispielhaft werden im Folgenden die wichtigsten Ergebnisse aus diesen Simulationen anhand von drei relevanten Szenarien vorgestellt. Abschließend folgt eine Bewertung der vorgestellten Protokollvarianten und eine Empfehlung hinsichtlich deren Eignung für den Einsatz zur Lastverteilung in IP-basierten Signalisierungsnetzen.

6.5.1 Die Simulationsumgebung

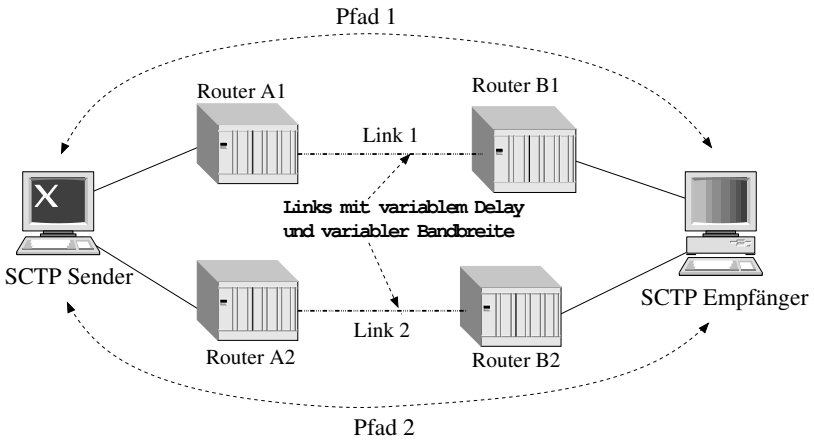

Abbildung 6.1: Simulations-Szenario für die LS-SCTP Untersuchung

Zur Bewertung der verschiedenen Algorithmen wurde das in Abbildung 6.1 dargestellte Referenzszenario entworfen, das einen unidirektionalen Datentransfer zwischen zwei Endgeräten modelliert. Diese Endgeräte sind dual-homed, d.h. über zwei IP-Adressen und entsprechend über zwei Interfaces (Gigabit Ethernet) erreichbar. Die verwendeten Modelle der SCTP-Endgeräte setzen auf das in Ab-

schnitt 5.3 erläuterte Modell auf. Dieses wurde um die drei folgenden LS-SCTP-Varianten erweitert:

- Zum einen wurden der von Iyengar et al. in [41] diskutierte Algorithmus implementiert.
- Darüber hinaus wurde ein Lastverteilungsalgorithmus implementiert, der das Aussenden pfadbasierter SACKs erlaubt (vgl. Abschnitt 6.4.1).
- Drittens wurde der Algorithmus implementiert, bei dem der Empfänger die SACKs auf dem Pfad mit der jeweilig kürzesten Rundlaufzeit zurücksendet (vgl. Abschnitt 6.4.2).

Prinzipiell könnten die nachfolgend getroffenen Aussagen über das Protokollverhalten auch analytisch ermittelt werden, da die Ergebnisse nicht von Zufallsprozessen abhängen und somit deterministisch sind. Aufgrund der hohen Protokoll-Komplexität wurden die vorliegenden Ergebnisse jedoch unter Zuhilfenahme des OPNET-Modelers in der oben erwähnten Umgebung ermittelt.

6.5.2 Beschreibung der Parameter

Bei den untersuchten Szenarien wurde angenommen, dass das SCTP zur Übertragung von langen Signalisierungsmeldungen eingesetzt wird, wie sie etwa typisch beim Session Initiation Protocol (SIP) [72, 55, 73] oder auch bestimmten SS7-Nachrichten (Breitband-MTP [33]) vorkommen können. Die Länge der Nachrichten wurde konstant auf 1000 Bytes festgelegt. Die Parametrisierung der zur Verfügung stehenden Links wurde so gewählt, dass sie typische Breitband-WAN Übertragungssysteme auf PDH/SDH-Basis mit Bandbreiten von 34,468 MBit/s (E3) oder 155,58 MBit/s (STM-1) modelliert.
Die Linkverzögerungen wurden im Bereich von 10-50 ms für den Pfad 1, bzw. 10-200 ms für den Pfad 2 konfiguriert. Dies deckt die typischen Übertragungsverzögerungen kontinentaler und interkontinentaler WAN-Links ab [94] (die reine Laufzeitverzögerung für 2000 km entspricht ca. 10 ms bei Glasfasern).
In den Szenarien wurde von einem unidirektionalen Verkehrsmuster ausgegangen, bei dem das Quellverhalten einer typischen gesättigten Quelle entspricht, die nur durch den Fluss- und Überlastkontrollalgorithmus der jeweiligen LS-SCTP Protokollvariante beschränkt ist. Dadurch kann der maximal zu erreichende Durchsatz sowie die maximale Nachrichtenverzögerung zwischen dem Sendezeitpunkt der

Quelle und dem Zeitpunkt des Empfangs durch die Empfängeranwendung ermittelt werden.

6.5.3 Erläuterung der Simulationsszenarien

Im Folgenden wird das in der Simulation ermittelte Protokollverhalten von drei Protokollvarianten anhand von drei verschiedenen Szenarien vorgestellt. Dabei wurden zwei Szenarien mit symmetrischen Link-Bandbreiten (je zweimal E3 und zweimal STM-1) sowie ein Szenario mit asymmetrischen Link-Bandbreiten (je einmal STM-1 und einmal E3) untersucht. Die betrachteten Protokollvarianten sind CMT (vgl. Abschnitt 6.3.2), LS-SCTP mit pfadbezogenen Bestätigungen (PB-SACK, vgl. Abschnitt 6.4.1) und LS-SCTP mit Bestätigungen, die über den Pfad mit der kürzesten RTT gesendet werden (S-SACK, vgl. Abschnitt 6.4.2).
Der Nutzdurchsatz wird in den jeweiligen Szenarien zum einen durch die physikalisch zur Verfügung stehende maximale Bandbreite auf den Links beschränkt, zum anderen durch die Fluss- und Überlastkontrollmechanismen der verschiedenen SCTP-Varianten. Dabei gilt insbesondere für große Rundlaufzeiten, dass der Durchsatz durch das Sendefenster – also durch die Flusskontrolle – und nicht durch die Überlastkontrolle beschränkt ist (pro Rundlaufzeit kann maximal eine der Größe des $rwnd$ entsprechende Menge an Nachrichten gesendet werden). Damit die Einschränkung durch das Sende- bzw. Empfangsfenster $rwnd$ in den jeweiligen Szenarien nicht zu stark ausfällt und die Qualität der Überlastkontrollalgorithmen der verschiedenen LS-SCTP Varianten bewerten zu können, wurde es den verschiedenen Szenarien angepasst. Exemplarisch werden im Folgenden nur die Ergebnisse für eine konstante Linkverzögerung des Links 1 von $d_1 = 10$ ms diskutiert.

6.5.4 Szenario 1: LS-SCTP mit zwei E3 Links

In diesem Szenario stehen zwei E3 Links zur Verfügung, von denen Link 1 die konstante Link-Verzögerung $d_1 = 10$ ms aufweist, während das Protokollverhalten in Abhängigkeit der Verzögerung des zweiten Links d_2 betrachtet wird. Diese variiert zwischen 10 ms und 200 ms. Das $rwnd$ wurde auf 400.000 Bytes gesetzt.

6.5.4.1 Untersuchung des Protokolldurchsatzes im Szenario 1

Abbildung 6.2 zeigt den von der Anwendung erfahrenen Nutzdurchsatz in diesem Szenario für die verschiedenen LS-SCTP Varianten. Die untere Kurve zeigt

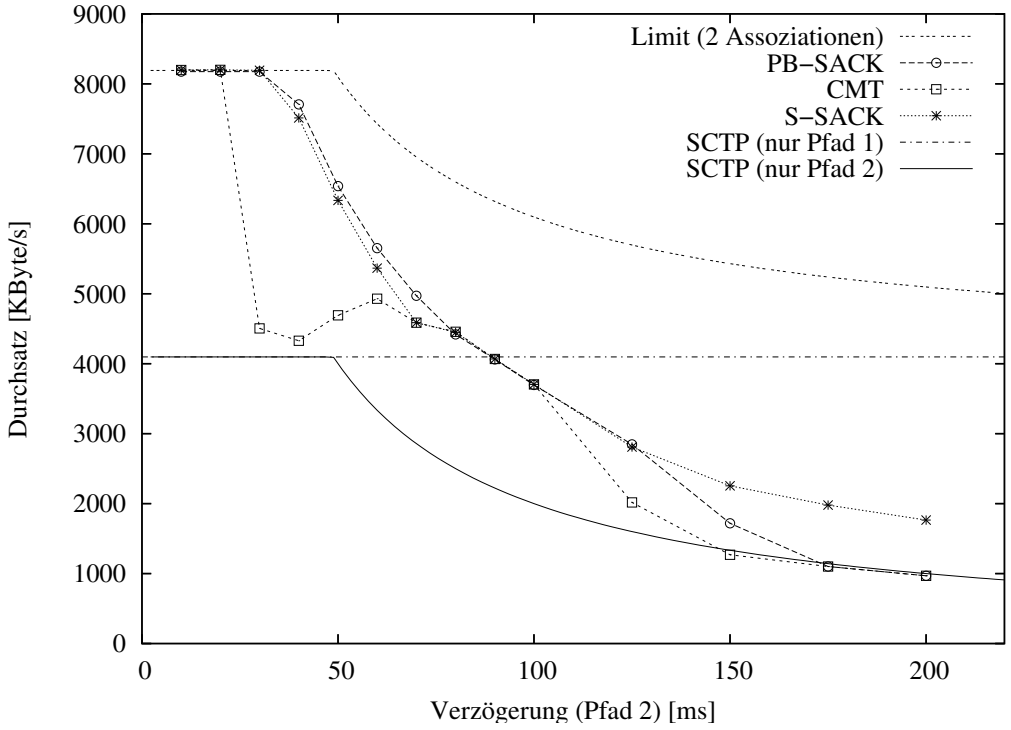

Abbildung 6.2: Durchsatz verschiedener LS-SCTP Varianten (Szenario 1)

den maximalen Durchsatz einer herkömmlichen SCTP-Assoziation, die nur den Pfad 2 benutzt. Dieser wird für $d_2 > 50$ ms mit steigendem d_2 durch das Sendefenster zunehmend beschränkt. Nutzt das Standard-SCTP nur den Pfad 1, so ist der Durchsatz konstant etwa 4096 KByte/s, da er nicht von der Linkverzögerung des zweiten Pfads abhängt. Dies entspricht dem maximale Durchsatz einer SCTP-Assoziation über einem E3-Link mit einer Bandbreite von 34,468 MBit/s unter den gegebenen Umständen. Addiert man den maximalen Durchsatz, der jeweils über die beiden Pfade möglich ist, so ergibt sich die obere Grenze für den Durchsatz (diese wird in Abbildung 6.2 mit *Limit* bezeichnet).

Für kleine Werte von d_2, in diesem Fall $d_2 \leq 20$ ms, erreichen alle 3 Varianten im Mittel dieses Limit, und damit den maximal möglichen doppelten Durchsatz von 8192 KByte/s, die Varianten PB-SACK und S-SACK erreichen diesen Durchsatz sogar für Werte von $d_2 \leq 30$ ms. Während der Durchsatz für die CMT-Variante ab einem Wert von $d_2 > 20$ ms stark abfällt, sinkt er für die beiden anderen Varianten

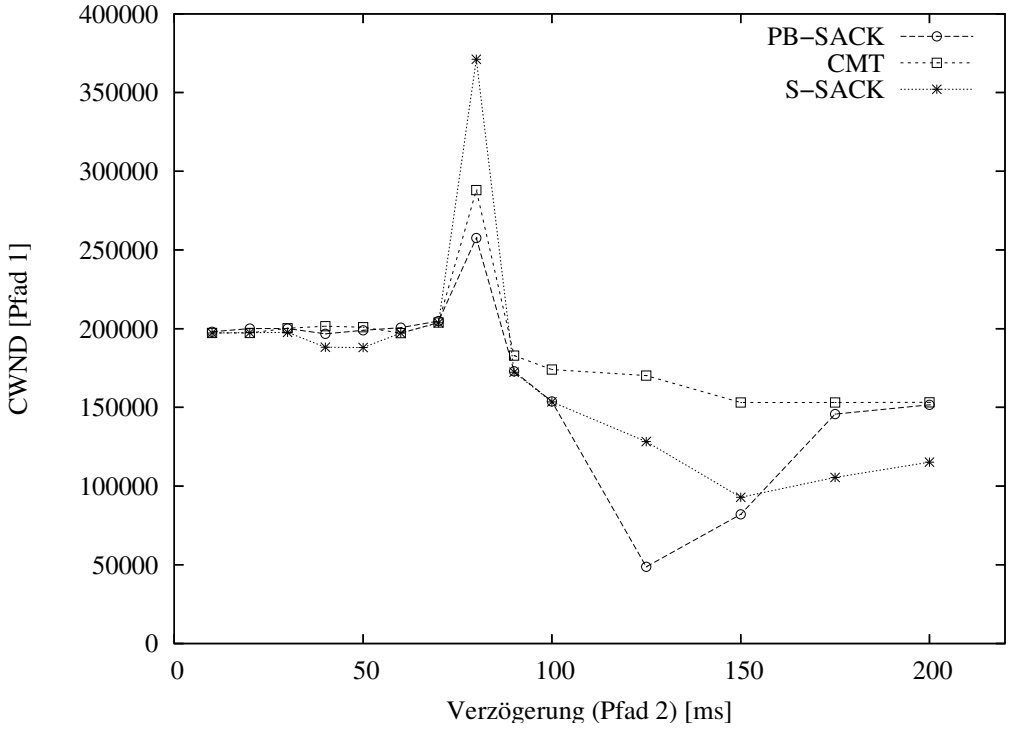

Abbildung 6.3: Congestion Window ($cwnd$) verschiedener LS-SCTP Varianten (Pfad 1 in Szenario 1, d_1=10 ms)

für $d_2 > 30$ ms wesentlich langsamer als bei der CMT-Variante. Der Durchsatz der PB-SACK-Variante liegt im Bereich 30 ms $< d_2 < 80$ ms über dem der beiden anderen Varianten, wobei sich die S-SACK-Variante ähnlich verhält, aber einen geringfügig kleineren Durchsatz aufweist. Zwischen 80 ms $\leq d_2 \leq 100$ ms erzielen alle Varianten einen nahezu gleichen Durchsatz, der sich mit zunehmender Verzögerung d_2 der Funktion $b_{2w}(d_2) = 2 \cdot b_w(d_2) = 2 \cdot (rwnd/d_2)$ annähert.

Die CMT-Variante nähert sich für $d_2 \geq 100$ ms und die PB-SACK-Variante für $d_2 \geq 125$ ms der Funktion $b_w(d_2) = rwnd/d_2$ an, die sich durch das konstante Produkt aus Bandbreite und Verzögerung ergibt (vgl. auch [49]). Damit erzielt die S-SACK Variante im Bereich von $d_2 \geq 125$ ms den höchsten Durchsatz aller Varianten. Für $d_2 > 100$ ms kann dieses Verhalten nicht mit den in den Abbildungen 6.3 und 6.4 dargestellten mittleren $cwnd$-Werten erklärt werden, da alle Varianten durch das Sendefenster und die Rundlaufzeit beschränkt sind. Vielmehr

spielen die Protokollmechanismen der einzelnen Varianten in diesem Fall eine Rolle. Es ist ferner festzustellen, dass für $d_2 \geq 90$ ms der Durchsatz aller Varianten – trotz Nutzung beider Pfade – unter die Grenze von 4096 KByte/s, die das Standard-SCTP bei ausschließlicher Nutzung des ersten Pfades (nur Link 1 mit konstanter Verzögerung $d_1 = 10$ ms) erzielen würde, sinkt. Daraus resultiert, dass nicht empfohlen werden kann, die LS-SCTP Varianten im gesamten Parameterraum zu verwenden. Um es einer SCTP-Implementierung zu erlauben, festzustellen, wann die Nutzung von LS-SCTP sinnvoll ist und wann nicht, soll aus diesem Grund ein geeignetes Kriterium für diese Einschätzung abgeleitet werden (vgl. auch Abschnitt 6.5.4.2).

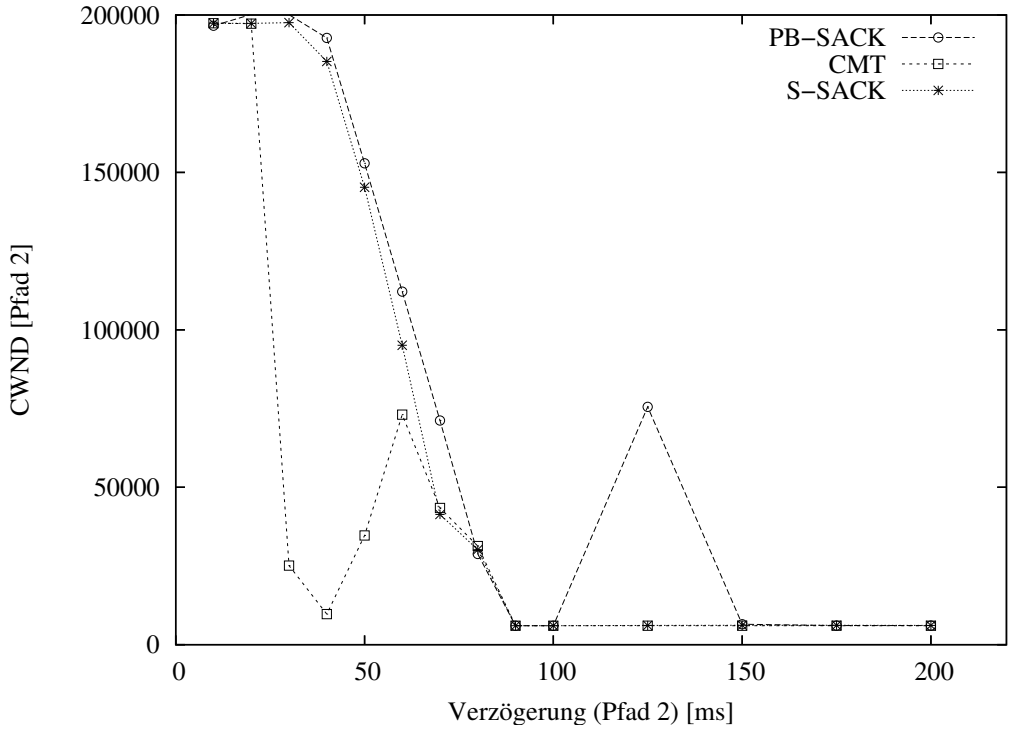

Abbildung 6.4: Congestion Window ($cwnd$) verschiedener LS-SCTP Varianten (Pfad 2 in Szenario 1, d_1=10 ms)

Für Werte von 40 ms $< d_2 <$ 60 ms steigt der Durchsatz der CMT Variante wieder leicht an, was auf die Tatsache zurückzuführen ist, dass das Congestion Window des Pfades 2 ($cwnd_2$) für diesen Parameterbereich im Mittel ebenfalls ansteigt und

mehr Daten über den zweiten Pfad übertragen werden. Aus der Detailbetrachtung der einzelnen Simulationen in diesem Bereich geht hervor, dass die starke Reduktion des mittleren $cwnd_2$ Wertes bei der CMT-Variante zunächst durch die Beschränkung des Congestion Window beim Einschaltvorgang der Quelle hervorgerufen wird. Diese Beschränkung wird durch den im Abschnitt 2.14 des "SCTP Implementer's Guide" [80] beschriebenen Algorithmus erzwungen. Dieser definiert einen neuen Protokollparameter *MaxBurstSize*, der angibt, wie viel Daten auf einmal (als *Burst*) gesendet werden können. Wenn mit *osb(2)* die Anzahl der auf dem Pfad 2 gesendeten, aber noch unbestätigten Bytes bezeichnet wird und gilt: $cwnd_2 > osb(2) + MaxBurstSize$, dann wird durch die Anweisung $cwnd_2 = osb(2) + MaxBurstSize$ der Wert des $cwnd_2$ zurückgesetzt. Dieser Algorithmus ist in der in [80] beschriebenen Weise nicht für den Einsatz beim LS-SCTP geeignet und sollte daher modifiziert werden. Geeignete Modifikationen konnten im Rahmen dieser Arbeit nicht näher untersucht werden, aber es ist z.B. denkbar, dass der Parameter $MaxBurstSize$ proportional zur Anzahl der genutzten Pfade erhöht wird, oder in Abhängigkeit von den Pfadcharakteristika unterschiedliche Werte für jeden Pfad annimmt.
Betrachtet man die Änderungen der mittleren $cwnd$-Werte des Pfades 1 in Abbildung 6.3 und des Pfades 2 in Abbildung 6.4, so werden weitere Unterschiede der drei LS-SCTP-Varianten deutlich. Für $d_2 \leq 70$ ms weisen alle Varianten einen annähernd konstanten mittleren $cwnd_1$-Wert auf. Dieser macht mit ca. 200.000 etwa die Hälfte des $rwnd$-Wertes von 400.000 aus und entspricht dem $ssthresh$-Wert. d.h. alle Varianten bleiben an der Grenze zwischen Slow Start und Congestion Avoidance.
Auf dem zweiten Pfad jedoch sinkt der mittlere $cwnd_2$-Wert der CMT-Variante für 20 ms $< d_2 < 40$ ms zunächst schnell ab, um dann wieder zu steigen (dieser Effekt entsteht durch die Beschränkung des $cwnd$ mit dem *MaxBurstSize*-Parameter, s.o.). Eine entsprechende Reduktion des $cwnd_2$-Wertes erfolgt bei den beiden anderen Varianten langsamer, und diese erreichen ein Minimum erstmalig bei $d_2 = 90$ ms, wo alle Varianten den minimalen mittleren $cwnd_2$-Wert aufweisen. Dagegen haben alle Varianten für $d_2 = 80$ ms ein Maximum beim zugehörigen mittleren $cwnd_1$-Wert. Dieser entsteht, da der $cwnd_2$-Wert bei dieser Linkverzögerung nahezu konstant und niedrig bleibt (bei allen Varianten liegt $cwnd_2$ zwischen 29.000 und 31.000), und damit ein deutliches Wachstum des $cwnd_1$-Wertes im Congestion Avoidance Modus über den $ssthresh$-Wert erlaubt. Bei geringeren Linkverzögerungen ändert sich der $cwnd_2$-Wert häufig, so dass das $cwnd_1$ im Congestion-Avoidance Modus nur geringfügig über den Wert der $ssthresh$ ansteigen kann. Bei größeren Linkverzögerungen d_2 sinkt nicht nur der

mittlere $cwnd_2$-Wert, sondern durch die Beschränkung durch das Sendefenster $rwnd$ auch der jeweilige mittlere $cwnd_1$-Wert und der Gesamtdurchsatz.

6.5.4.2 Kriterium für den Einsatz des LS-SCTP im Szenario 1

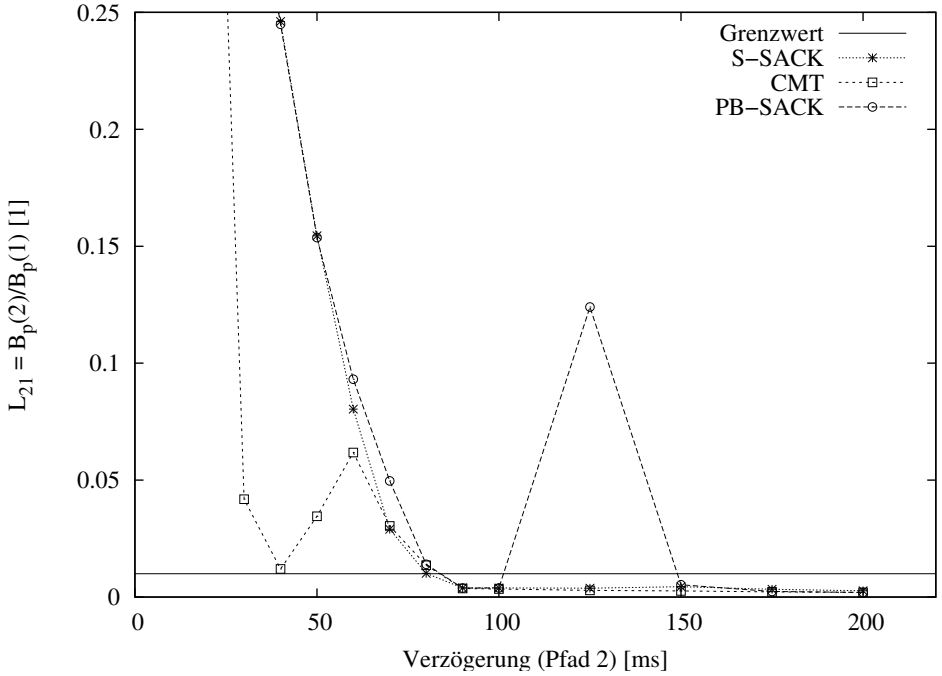

Abbildung 6.5: Betrachtung des Verhältnisses der relativen Bandbreite L_{21} im Szenario 1

Da die einzelnen LS-SCTP Varianten selbst unter günstigen Bedingungen (zwei Links mit symmetrischer Bandbreite) nur für geringe Link-Verzögerungen gute Ergebnisse erzielen und den Durchsatz signifikant erhöhen können, wird im Folgenden ein Kriterium definiert, nach dem eine Implementierung dynamisch entscheiden kann, ob die Anwendung des Load Sharing vorteilhaft ist oder nicht. Als Kenngröße eines jeden Pfades wird der Quotient aus dem mittleren $cwnd$-Wert $cwnda$ des Pfades[2] und der gemessenen mittleren Rundlaufzeit $rtta$ vorgeschlagen. Für den i-ten Pfad einer Assoziation wird dieser Quotient $B_p(i) =$

[2]Dieser kann von einer Implementierung einfach näherungsweise bestimmt werden

$cwnda_i/rtta_i$ nachfolgend als Pfad-Bandbreite bezeichnet. Als Kriterium für oder gegen die Nutzung von Load Sharing wird der Quotient $L_{jk} = B_p(j)/B_p(k)$ zweier Pfade j und k betrachtet. Nimmt dieser Quotient einen Wert $L_{jk} > 100$ oder $L_{jk} < 0,01$ an, so wird der Pfad mit der geringeren Pfadbandbreite *nicht* für das Load-Sharing genutzt. Die entsprechenden Werte für das Load-Sharing Kriterium L_{21} aus Szenario 1 zeigt Abbildung 6.5.
Bei einem System mit Dualhoming führt die Anwendung dieses Kriteriums dazu, dass bei Überschreiten des Schwellwertes ausschließlich der günstigere Pfad für die Übertragung genutzt wird und somit der Durchsatz nicht unter den auf diesem Pfad zu erzielenden Durchsatz fällt. Bezogen auf das Szenario 1 würde dies bedeuten, dass die LS-SCTP Varianten mindestens den maximalen Durchsatz auf dem ersten Pfad erzielen könnten, mithin also 4096 KByte/s. Demnach würden alle Varianten für Werte von $d_2 < 90$ ms Load Sharing nutzen. Für größere Werte von d_2 wäre Load Sharing in der Regel deaktiviert. Eine Ausnahme bildet hier nur die PB-SACK Variante, die bei $d_2 \approx 125$ ms einen sehr hohen mittleren $cwnd$-Wert für den zweiten Pfad aufweist (und damit eine hohe Pfad-Bandbreite). In diesem Fall ist eine Erweiterung des vorgeschlagenen Kriteriums notwendig. Sie wird in Abschnitt 6.5.5.2 erläutert. In erster Näherung erlaubt die Anwendung des hier vorgeschlagenen Kriteriums zwar nicht in jedem Fall, den *optimalen* Durchsatz zu erzielen, trifft aber eine sinnvolle erste Auswahl.

6.5.4.3 Untersuchung der Nachrichtenverzögerung im Szenario 1

Abbildung 6.6 stellt die maximale Nachrichtenverzögerung $d_{max}(d_2)$ (Ende zu Ende) zwischen dem Zeitpunkt des ersten Aussendens einer Nachricht durch die Quelle und dem Zeitpunkt der Zustellung an die Empfängerapplikation dar. Die maximale Nachrichtenverzögerung ergibt sich in diesem Fall aus der Wartezeit in der Sendewarteschlange, der bandbreitenabhängigen Emissionszeit auf dem jeweiligen WAN-Link, der Übertragungsverzögerung auf dem WAN-Link (die Verzögerungen durch die Anschlussnetze werden hier vernachlässigt) und der Wartezeit auf die Zustellung beim Empfänger, die durch die Wiederherstellung der Nachrichtenreihenfolge (engl. Reordering) zustande kommt. Wie aus der Abbildung 6.6 ersichtlich ist, liegt $d_{max}(d_2)$ bei allen LS-SCTP-Varianten für Linkverzögerungen von $d_2 > 20$ ms innerhalb der Region, die von den Funktionen $f(d_2) = 3d_2$ und $f(d_2) = d_2$ eingegrenzt wird.
Für kleine Werte von $d_2 \leq 40$ ms liegt die maximale Nachrichtenverzögerung aller Varianten nahezu konstant zwischen 50 ms und 70 ms. Im Bereich von 50 ms $< d_2 \leq 100$ ms wächst sie für alle Varianten leicht an, liegt aber immer unter

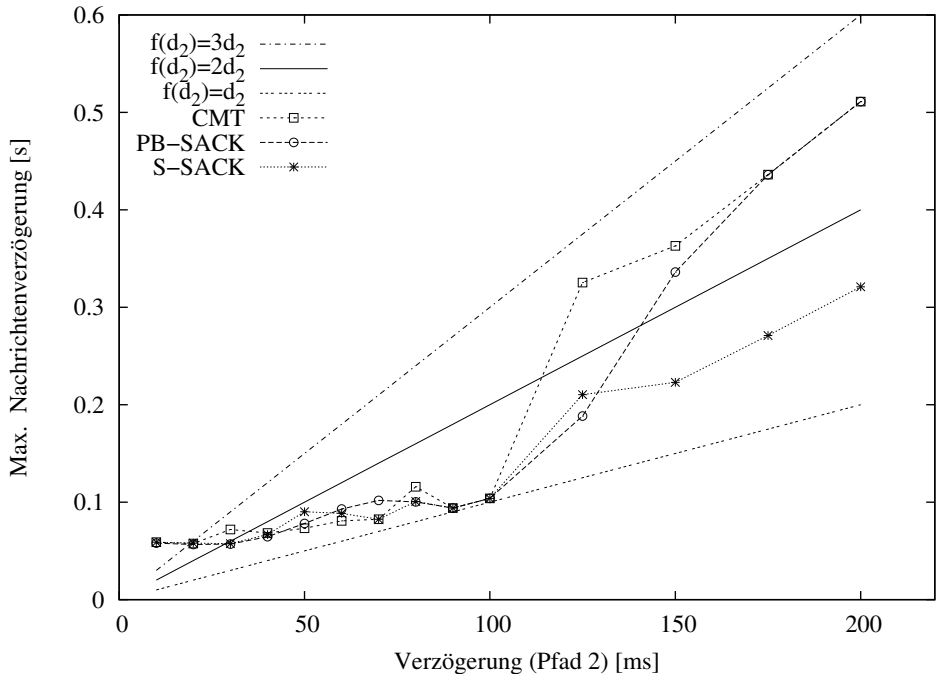

Abbildung 6.6: Maximale Nachrichtenverzögerung der LS-SCTP-Varianten in Szenario 1 (d_1=10 ms)

120 ms. Erst für $d_2 > 100$ ms wachsen die maximalen Pfadverzögerungen deutlich an. Dies wird durch Nachrichten verursacht, die über den Link 2 übertragen werden. Wenn der Sender bei den Varianten CMT oder PB-SACK blockiert ist (für $d_2 \geq 125$ ms), muss er auf Bestätigungen warten, die gegebenenfalls über den langsameren Link 2 zurückgesendet werden. Daher überschreitet bei diesen Varianten die maximale Nachrichtenverzögerung für $d_2 \geq 125$ ms die Funktion $f(d_2) = 2d_2$. Da bei der S-SACK-Variante alle Bestätigungen über den Link 1 mit der kürzeren Linkverzögerung zurückgesendet werden, liegt in dem Fall $d_{max}(d_2)$ unterhalb der Funktion $f(d_2) = 2d_2$.

6.5.5 Szenario 2: LS-SCTP mit zwei STM-1 Links

Dieses Szenario ist analog zum Szenario 1, allerdings stehen zwei STM-1 Links zur Verfügung. Das $rwnd$ wurde bei diesem Szenario auf 1.600.000 Bytes erhöht, um auch bei größeren Rundlaufzeiten einen sinnvollen Mindestdurchsatz zu erzie-

len und nicht schon bei kurzen Rundlaufzeiten den Durchsatz durch das Sende- bzw. Empfangsfenster $rwnd$ zu beschränken.

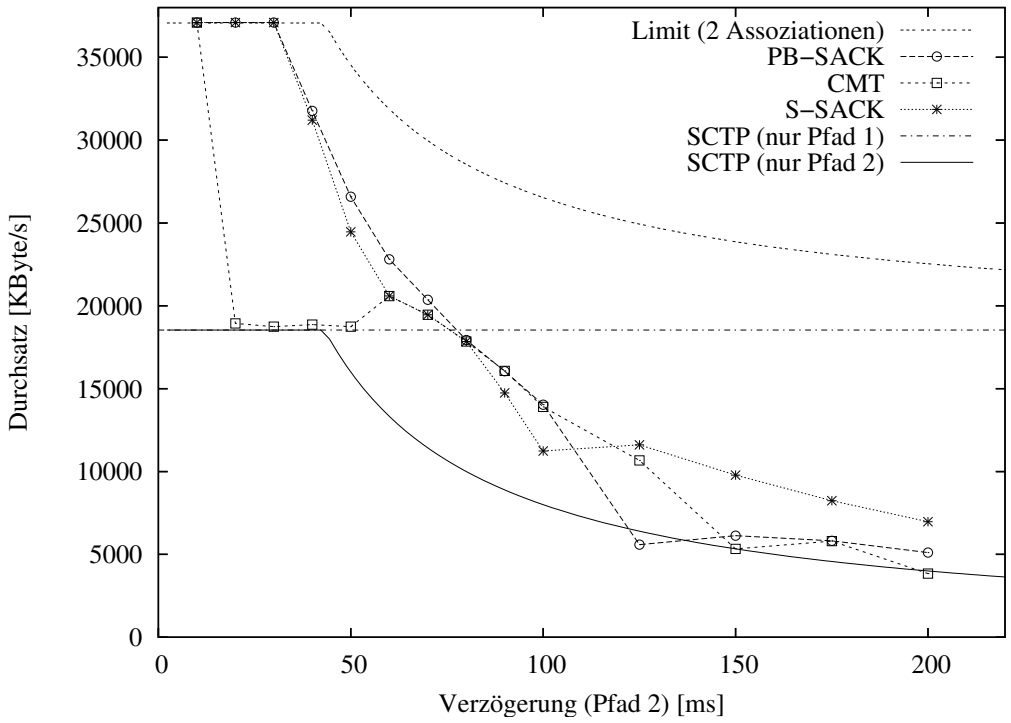

Abbildung 6.7: Durchsatz verschiedener LS-SCTP Varianten (Szenario 2)

6.5.5.1 Untersuchung des Protokolldurchsatzes im Szenario 2

Für die insbesondere in Weitverkehrsnetzen typischen Verzögerungen von $d_2 \leq 70$ ms erzielen die PB-SACK- und die S-SACK-Variante im Szenario 2 einen in der Regel signifikant höheren, mindestens aber gleichen Durchsatz wie die CMT-Variante (vgl. Abbildung 6.7). Während die CMT-Variante nur für $d_2 = 10$ ms den optimalen Durchsatz, mit dem beide STM-1 Links voll ausgelastet sind, erreicht, gelingt dies der PB-SACK- und der S-SACK-Variante – ähnlich wie im Szenario 1 – für $d_2 \leq 30$ ms.
Bei einer Linkverzögerung von 80 ms $< d_2 \leq 100$ ms sinkt der Durchsatz der S-SACK-Variante unter den der beiden anderen Varianten, liegt aber für

$d_2 \geq 125$ ms wieder über deren Durchsatz. Für $d_2 \geq 80$ ms sinkt der Durchsatz aller Varianten unter die Grenze von 18.535 KByte/s, die eine Standard-SCTP Assoziation bei ausschließlicher Nutzung des Pfades 1 erreichen würde. Dies bestätigt die Ergebnisse aus Abschnitt 6.5.4.1. Aus diesem Grund soll auch für das Szenario 2 das bereits in Abschnitt 6.5.4.2 diskutierte Kriterium zur Abschätzung der Anwendbarkeit des Load Sharing untersucht werden.

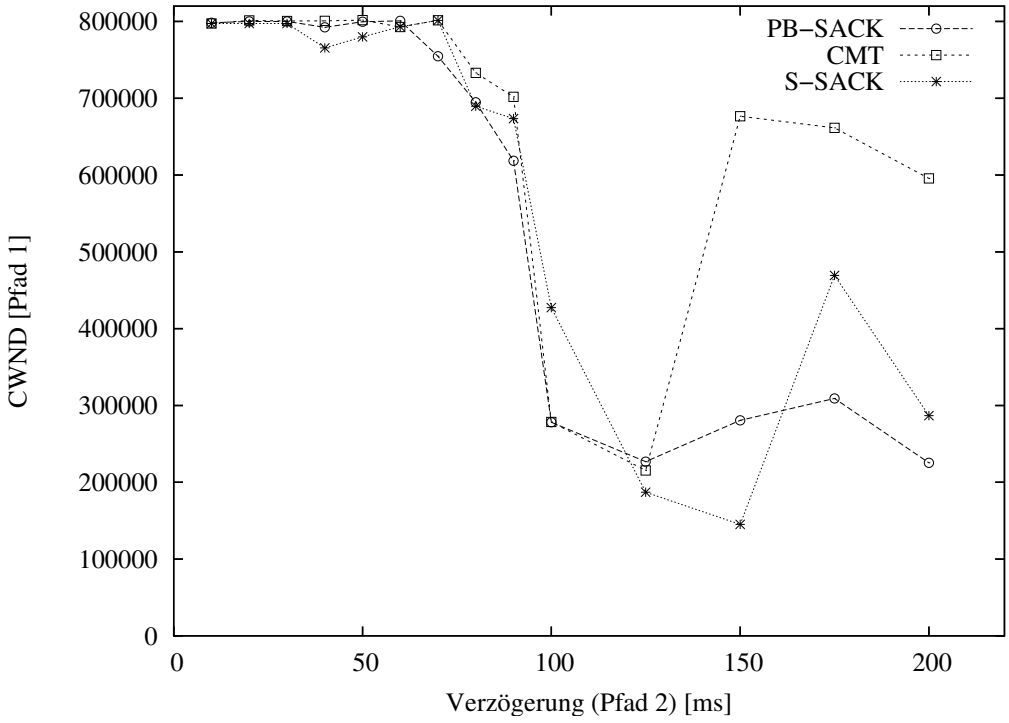

Abbildung 6.8: Congestion Window ($cwnd_1$) verschiedener LS-SCTP Varianten (Pfad 1 in Szenario 2, d_1=10 ms)

Wie aus den Abbildungen 6.8 und 6.9 hervorgeht, sind bei allen Varianten für $d_2 < 80$ ms die Werte des $cwnd_1$ nahezu konstant hoch zwischen 750.000 und 800.000, und betragen somit etwa die Hälfte des maximalen *arwnd*-Wertes (vgl. Abschnitt 2.3.8). Für $d_2 \leq 10$ ms bei der CMT-Variante und für $d_2 \leq 30$ ms bei den beiden anderen Varianten liegt der $cwnd_2$-Wert ebenfalls bei etwa 800.000 und somit wird auch im Szenario 2 das $rwnd$ fair auf beide Pfade verteilt. In diesem Bereich erzielen die Varianten auch den entsprechenden maximalen Durch-

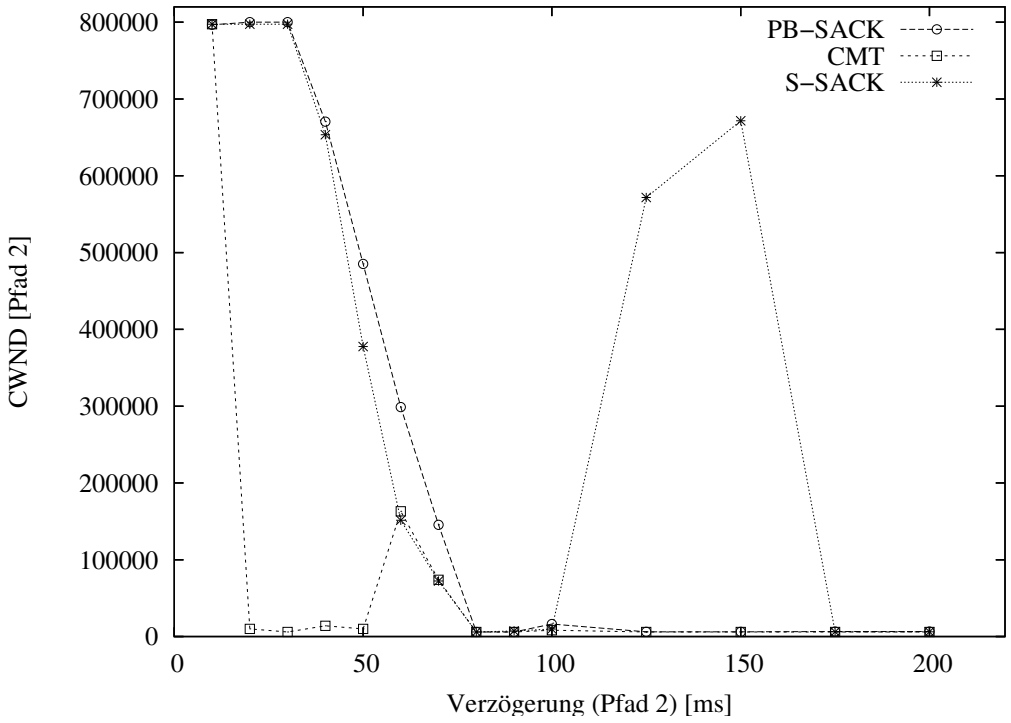

Abbildung 6.9: Congestion Window ($cwnd_2$) verschiedener LS-SCTP Varianten (Pfad 2 in Szenario 2, d_1=10 ms)

satz von je etwa 37.100 KByte/s.

Bei der CMT-Variante sinkt der mittlere $cwnd_2$-Wert für 10 ms $< d_2 <$ 50 ms wieder aufgrund der Beschränkung des Congestion Windows auf dem zweiten Pfad, die erst bei Pfadverzögerungen von 50 ms $< d_2 <$ 80 ms weniger stark ausfällt. Bei der PB-SACK- und der CMT-Variante wird für $d_2 \geq 80$ ms der zweite Pfad nur noch wenig genutzt. Für Werte von $d_2 > 100$ ms nähert sich der Durchsatz dieser Varianten daher für steigende Werte von d_2 dem Durchsatz einer Assoziation an, die nur den zweiten Pfad benutzt (aufgrund der Zustellverzögerung von Nachrichten, die über den Pfad 1 gesendet werden und nachfolgend auf Nachrichten warten, die über den Pfad 2 gesendet wurden). In dem Bereich von $d_2 > 100$ ms liegt der Durchsatz der S-SACK-Variante aufgrund der geringeren Nachrichtenverzögerungen (vgl. auch Abschnitt 6.5.5.3) über dem der beiden anderen Varianten. Die mittleren $cwnd$-Werte der Pfade spielen dabei nur noch eine untergeord-

nete Rolle, da das Sendeverhalten durch das Empfänger- bzw. Sendefenster $rwnd$ bestimmt wird.

6.5.5.2 Kriterium für den Einsatz des LS-SCTP im Szenario 2

Auch in Szenario 2 greift im Wesentlichen das in Abschnitt 6.5.4.2 erwähnte Kriterium. Betrachtet man wieder als Load Sharing Kriterium L_{21} den Quotienten $L_{21} = B_p(2)/B_p(1)$ der beiden Pfade 2 und 1, wobei $B_p(n)$ die Pfad-Bandbreite des Pfades n bezeichnet, so ergeben sich die in Abbildung 6.10 dargestellten Werte. Auch in Szenario 2 erscheint es sinnvoll, dass die jeweilige Implementierung

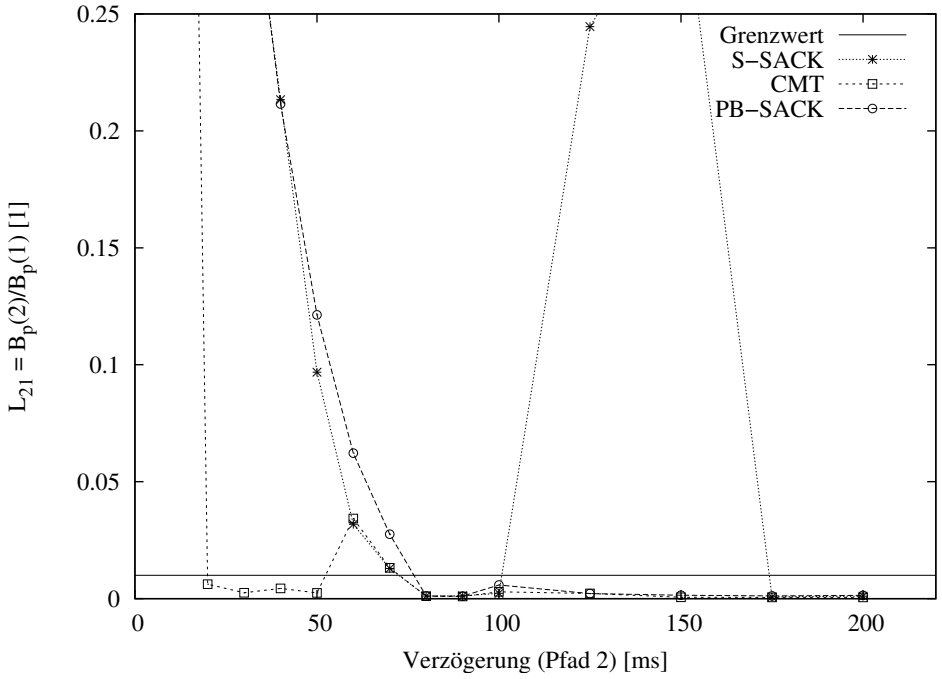

Abbildung 6.10: Betrachtung des Verhältnisses der relativen Bandbreiten L_{21}

kein Load Sharing einsetzt, wenn für den Quotienten gilt:

$$L_{12} > 100 \text{ bzw. } L_{21} < 0,01$$

und in diesem Fall ausschließlich den besseren Pfad mit der größeren Pfad-Bandbreite nutzt. Für alle anderen Werte von L_{21} wäre das Load Sharing in der

Regel deaktiviert. Demnach würde die CMT-Variante für Werte von $d_2 < 20$ ms, und für Werte von 50 ms $< d_2 \leq 70$ ms Load Sharing einsetzen. Die PB-SACK-Variante würde entsprechend für Werte von $d_2 < 80$ ms Load Sharing einsetzen, ebenso wie die S-SACK-Variante. L_{21} liegt bei letzterer allerdings auch für Werte von 100 ms $< d_2 < 175$ ms über dem Schwellwert von $0,01$, da diese dort einen großen mittleren $cwnd_2$-Wert aufweist. Daher kann die oben erwähnte Bedingung nicht alleine als Kriterium für die Nutzung des Load Sharing dienen. Ein sinnvolles zusätzliches Kriterium ist die Anzahl der mit den SACK-Chunks übertragenen Gap Reports (vgl. Abschnitt 2.3.5). Für 100 ms $< d_2 < 175$ ms liegt diese Anzahl bei der S-SACK-Variante mehrfach über 100. Eine SCTP-Implementierung kann daher eine Schwellwertentscheidung treffen, und Load Sharing ebenfalls deaktivieren, wenn die Anzahl der empfangenen Gap Reports in einem SACK-Chunk mehrfach größer als 100 ist. Die Kombination der erwähnten Kriterien erlaubt es einer Implementierung dann mit hinreichender Genauigkeit zu entscheiden, wann der Einsatz des Load Sharing sinnvoll ist.

6.5.5.3 Untersuchung der Nachrichtenverzögerung im Szenario 2

Abbildung 6.11 zeigt, dass die maximalen Nachrichtenverzögerungen $d_{max}(d_2)$ der drei LS-SCTP Varianten im Szenario 2 für $d_2 > 10$ ms in der Region liegen, die durch die Funktionen $f(d_2) = d_2$ und $f(d_2) = 3d_2$ eingegrenzt wird. Für Werte von $d_2 \leq 30$ ms sind die Werte von $d_{max}(d_2)$ in etwa konstant, und betragen zwischen 37 ms und 40 ms. Für 40 ms $\leq d_2 \leq 80$ ms entwickeln sich die maximalen Nachrichtenverzögerungen *aller* Varianten wie die Funktion $f(d_2) = d_2$ und liegen nur wenige Millisekunden darüber. Während sich $d_{max}(d_2)$ bei der PB-SACK-Variante für 40 ms $\leq d_2 \leq 100$ ms wie $f(d_2) = d_2$ entwickelt, ist dies bei der CMT-Variante für 40 ms $\leq d_2 \leq 125$ ms der Fall. Hingegen steigt die maximale Nachrichtenverzögerung der S-SACK-Variante bereits für 80 ms$< d_2 \leq 100$ ms stärker an, als die der beiden anderen Varianten, da bei dieser Variante die Anzahl der wartenden Nachrichten in der Empfängerwarteschlange stärker zunimmt.
Für große Linkverzögerungen ($d_2 > 100$ ms) sind alle LS-SCTP-Varianten wieder durch das Sendefenster und die hohen Rundlaufzeiten auf dem zweiten Link beschränkt, und die $d_{max}(d_2)$-Werte der PB-SACK- und CMT-Varianten verlaufen oberhalb der Funktion $f(d_2) = 2d_2$ (bei der S-SACK-Variante erst etwa für $d_2 > 190$ ms). Die PB-SACK-Variante entwickelt sich erst für $d_2 \geq 125$ ms wie $f(d_2) = 2d_2 + C$, mit $C = 31$ ms. Dieser Offset wird durch die Verzögerung in der Empfängerwarteschlange bzw. die Wartezeit auf die Wiederherstellung der korrekten Nachrichtenreihenfolge bewirkt.

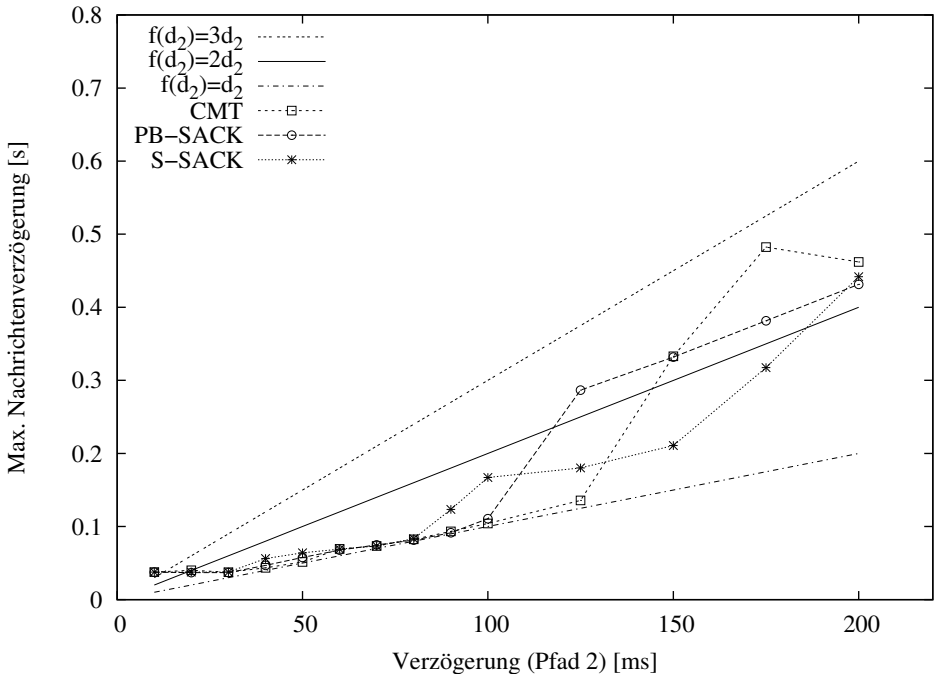

Abbildung 6.11: Maximale Nachrichtenverzögerung der LS-SCTP Varianten (d_1=10 ms)

Abbildung 6.11 macht auch deutlich, dass die PB-SACK-Variante Werte für $d_{max}(d_2)$ erzielt, die (bis auf eine Ausnahme bei $d_2 = 125$ ms) absolut vergleichbar mit den entsprechenden Werten der CMT-Variante sind.

6.5.6 Szenario 3: LS-SCTP mit einem STM-1 und einem E3 Link

In diesem Szenario stehen asymmetrische Links zur Verfügung: Link 1 ist ein STM-1-Link und Link 2 ein E3-Link. Das Protokollverhalten wird für eine konstante Link-Verzögerung $d_1 = 10$ ms des Link 1 betrachtet, während die Verzögerung d_2 des zweiten Links zwischen 10 ms und 200 ms variiert (analog zu den vorigen Szenarien). Das $rwnd$ wurde hier auf 1.000.000 Bytes dimensioniert.

6.5.6.1 Untersuchung des Protokolldurchsatzes im Szenario 3

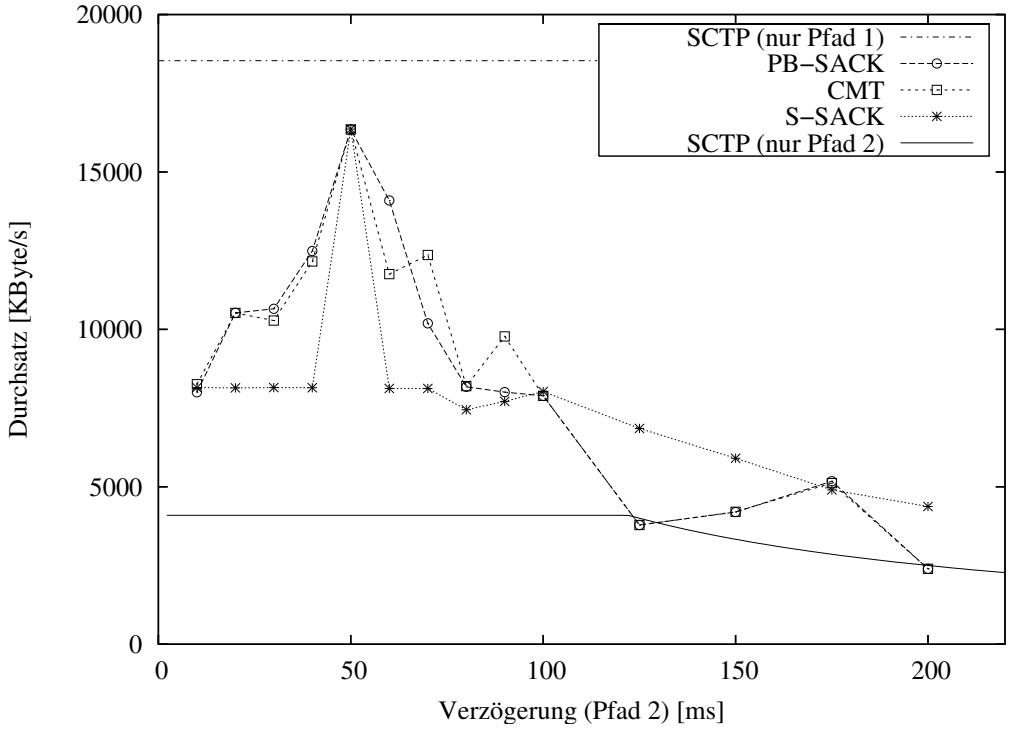

Abbildung 6.12: Durchsatz verschiedener LS-SCTP Varianten (Szenario 3)

Das Ergebnis der Simulationen in diesem Szenario mit asymmetrisch verteilten Link-Bandbreiten (vgl. Abbildung 6.12) verdeutlicht, wie schwierig es ist, einen unter allen Randbedingungen optimalen Algorithmus für die SCTP-basierte Lastverteilung zu entwerfen. Keiner der drei Algorithmen erreicht in diesem Szenario mit einem STM-1 und einem E3 Link die erhoffte Optimierung des Durchsatzes[3]. Zwar übertreffen alle Varianten für $d_2 < 125$ ms den Durchsatz, der bei Nutzung eines einzelnen E3 Links erzielt werden könnte, jedoch erreicht keine der Varianten einen Durchsatz, der den STM-1 Link auslastet.
Die S-SACK-Variante erzielt für $d_2 < 50$ ms sowie für $d_2 \geq 100$ ms den doppelten Durchsatz im Vergleich zu einer Standard-SCTP Assoziation, die nur den Pfad 2

[3]Bei gleichzeitiger Nutzung beider Links wäre maximal ein Durchsatz von etwa 22 MByte/s möglich.

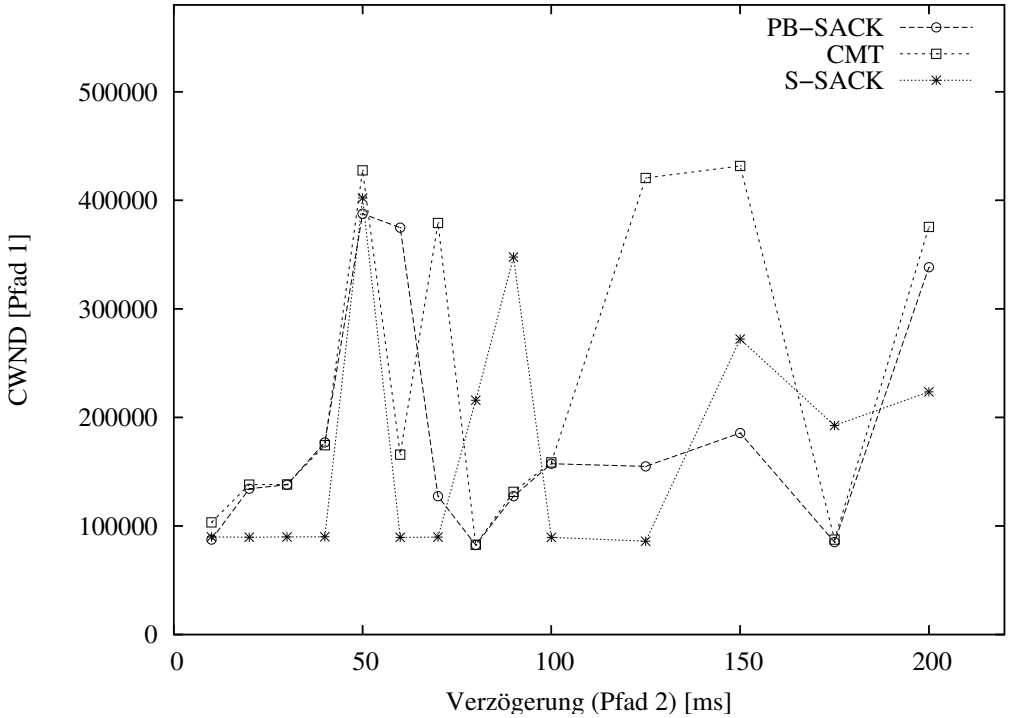

Abbildung 6.13: Congestion Window ($cwnd$) verschiedener LS-SCTP Varianten (Pfad 1 in Szenario 3, d_1=10 ms)

nutzt. Alle Varianten erreichen den Spitzendurchsatz bei $d_1 = 10$ ms und $d_2 = 50$ ms, wenn die mittleren $cwnd$-Werte für den Pfad 1 lokale Maxima erreichen (ca. 387.000 für PB-SACK, bzw. 437.000 für CMT und 402.000 für S-SACK), und die entsprechenden Werte für den Pfad 2 lokale Minima aufweisen (ca. 38.000 für PB-SACK, bzw. 22.000 für CMT und S-SACK).

Der verhältnismäßig geringe Durchsatz aller Varianten ist auf die bereits in Abschnitt 6.5.4.1 erwähnte Beschränkung des Congestion Windows, in diesem Fall für den $cwnd_1$ Wert, zurückzuführen. Dabei wächst das $cwnd_1$ beim Einschaltvorgang schnell, und beim Empfänger treten, bedingt durch die stark unterschiedlichen Bandbreiten der beiden Links, vermehrt Lücken in der Ankunftsreihenfolge der Nachrichten auf (bis zu 100 Gap Reports pro SACK). Dies führt nachfolgend dazu, dass der Sender den Pfad 1 so lange blockiert, bis die auf dem Pfad 2 gesendeten Nachrichten beim Empfänger eintreffen, und dieser entsprechende

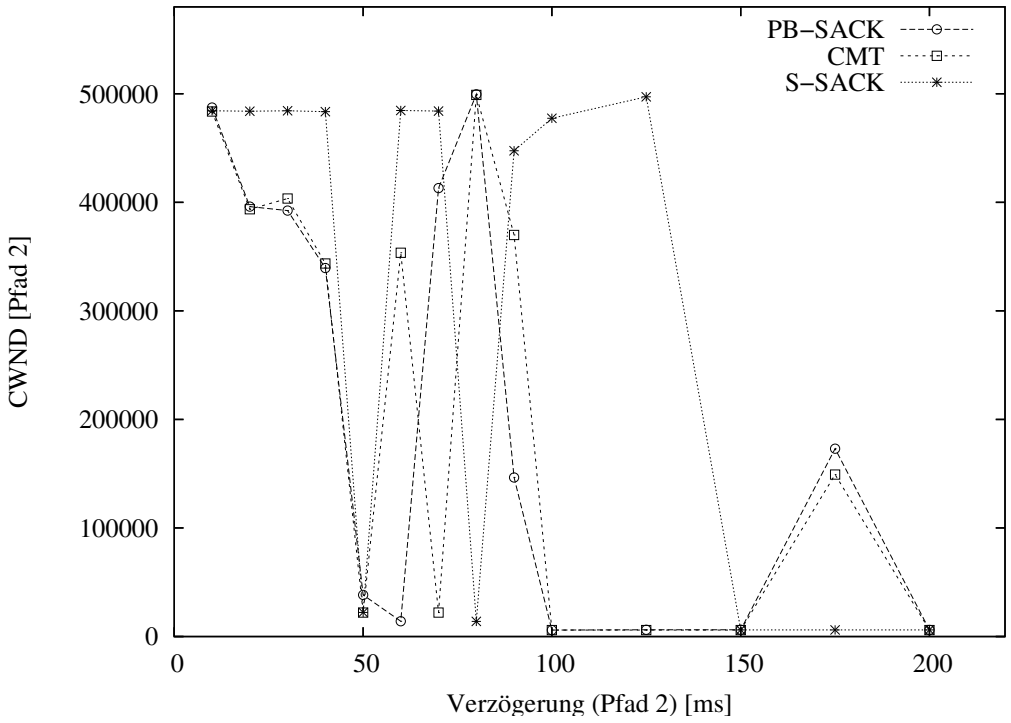

Abbildung 6.14: Congestion Window ($cwnd$) verschiedener LS-SCTP Varianten (Pfad 2 in Szenario 3, d_1=10 ms)

Bestätigungen zurücksendet. Während der Sender auf dem Pfad 1 blockiert ist, wird sein Congestion Window durch die Anwendung der im Abschnitt 2.14 des “SCTP Implementer’s Guide” [80] festgelegte Regel zurückgesetzt. Abbildung 6.13 verdeutlicht, dass der $cwnd_1$ Wert aller Varianten erst für $d_2 = 50$ ms ein Maximum erreicht. Für $d_2 = 50$ ms sinkt auch bei allen Varianten die Anzahl der von den SACK-Chunks transportierten Gap Reports (auf maximal 5 Gaps) und führt so insgesamt zu einem günstigeren Protokollverlauf. Mit zunehmendem d_2 steigt die Anzahl der vom Empfänger zurückgesendeten Gap Reports wieder deutlich an. So erhält z.B. bei der PB-SACK-Variante der Sender für $d_2 = 80$ ms SACKs mit bis zu 95 Gap Reports.

Für Werte von $d_2 \geq 100$ ms beschränkt wieder das Sende-Fenster $rwnd$ bei allen Varianten den Sender insgesamt. Die auf den jeweiligen Pfaden erzielten $cwnd$ Werte sind daher für größere Werte von $d_2 > 50$ ms wenig aussagekräftig und

schwanken stark.
In diesem Szenario mit asymmetrischen Link-Bandbreiten erzielt die CMT-Variante nur in dem Parameterbereich von 70 ms $\leq d_2 \leq 90$ ms einen höheren (oder mindestens gleichen) Durchsatz wie die PB-SACK-Variante. Für den restlichen Parameterbereich von d_2 erzielt die PB-SACK-Variante einen mindestens gleichen oder sogar höheren Durchsatz als die CMT-Variante. Die S-SACK-Variante erreicht nur für 100 ms $\leq d_2 < 170$ ms und $d_2 > 180$ ms einen höheren Durchsatz als die beiden anderen Varianten.

6.5.6.2 Untersuchung der Nachrichtenverzögerung im Szenario 3

Abbildung 6.15 zeigt die maximale Nachrichtenverzögerung $d_{max}(d_2)$ der verschiedenen LS-SCTP-Varianten im Szenario 3.

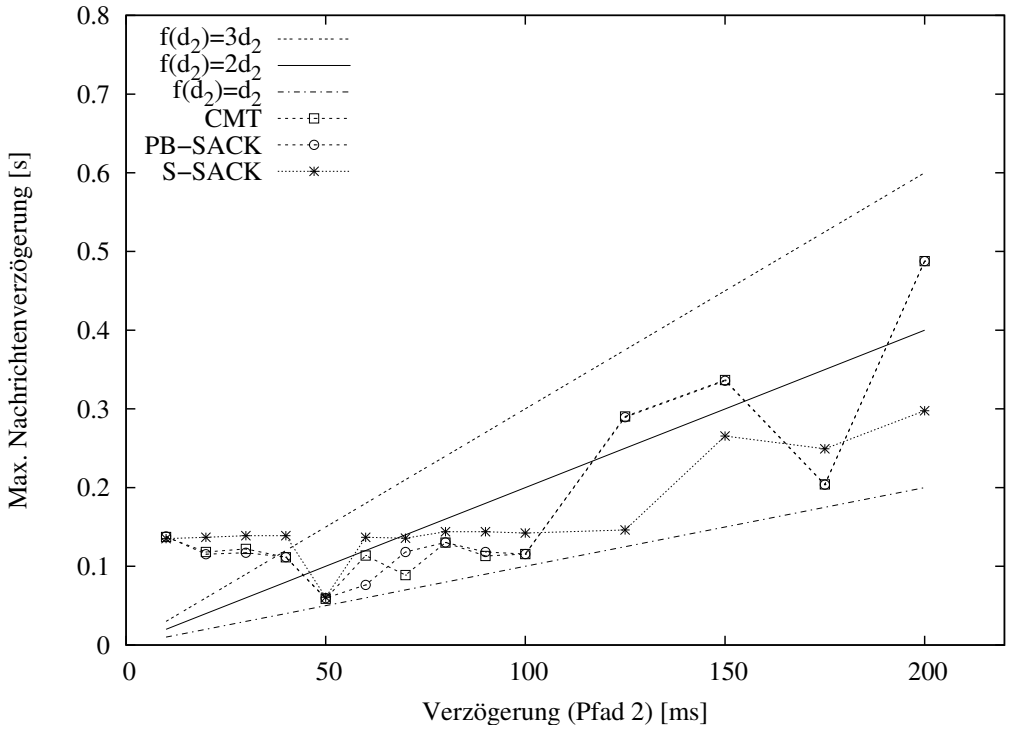

Abbildung 6.15: Durchsatz verschiedener LS-SCTP Varianten (d_1=10 ms)

Für $d_2 \geq 50$ ms bleibt die maximale Nachrichtenverzögerung aller LS-SCTP-

Varianten in dem Bereich zwischen den Funktionen $f(d_2) = d_2$ und $f(d_2) = 3d_2$. Bei allen Varianten ist $d_{max}(d_2)$ für $d_2 < 50$ ms annähernd konstant und liegt zwischen 111 ms und 138 ms. Insbesondere bei der S-SACK-Variante ist für $d_2 \leq 125$ ms und $d_2 \neq 50$ ms die maximale Nachrichtenverzögerung konstant zwischen 138 ms und 145 ms und nahezu unabhängig von der Linkverzögerung d_2. Wie bereits in Abschnitt 6.5.4.3 erwähnt, kommt die Nachrichtenverzögerung in der Hauptsache durch die Wartezeit in der Sendewarteschlange, die Emissionszeit auf dem jeweiligen WAN-Link, die Übertragungsverzögerung auf dem WAN-Link sowie durch die Wartezeit auf die Zustellung beim Empfänger zustande.

Die Verzögerungen bei allen Varianten werden in dem vorliegenden Szenario (insbesondere für $d_2 \leq 100$ ms durch die notwendige Wiederherstellung der Nachrichtenreihenfolge beim Empfänger verursacht. Die Detailbetrachtung einzelner Simulationen zeigt hier, dass die Anzahl der über den Pfad 2 gesendeten, aber noch unbestätigten, Bytes $osb(2)$ (vgl. auch Abschnitt 6.2.1) hier bei allen Varianten große Werte annimmt (bis zu 500.000, entsprechend 500 Nachrichten). Bei einer Linkbandbreite des Pfades 2 von 34,468 MBit/s dauert die Übertragung einer Menge von 500 Nachrichten über den Link 2 etwa 120 ms. Die Nachrichten werden zwischenzeitlich nach dem Aussenden im Router A2 (vgl. Abbildung 6.1) gepuffert, und es kommt nachfolgend zu der erwähnten Verzögerung der Zustellung beim Empfänger (d.h. der Empfängerapplikation können Nachrichten, die über den Pfad 1 gesendet wurden, erst dann zugestellt werden, wenn auch alle vorher über den Pfad 2 gesendeten Nachrichten beim Empfänger angekommen sind).

Für Werte von $d_2 > 100$ ms steigt die maximale Nachrichtenverzögerung der PB-SACK- und der CMT-Variante wieder stärker an, für $d_2 > 125$ ms auch die der S-SACK-Variante. Die für $d_2 = 175$ ms zu beobachtende vorübergehend reduzierte maximale Nachrichtenverzögerung ergibt sich durch die Tatsache, dass die beiden Varianten bei diesem Wert von d_2 einen Pfaddurchsatz erreichen, der in etwa dem Verhältnis der Bandbreiten der Links 1 und 2 entspricht (aus der Detailsimulation ist zu entnehmen, dass in diesem Fall etwa 4100 KByte/s über den Link 1 und 1100 KByte/s über den Link 2 gesendet werden). In der Folge warten nur wenige der über den ersten Pfad gesendeten Nachrichten in der Empfängerwarteschlange bis die entsprechenden Nachrichten über den zweiten Pfad ankommen, und eine Wiederherstellung der Reihenfolge beim Empfänger erlauben. Die maximale Verzögerung liegt mit 204 ms daher in diesem Fall nur unwesentlich über der Linkverzögerung von $d_2 = 175$ ms.

6.5.6.3 Kriterium für den Einsatz des LS-SCTP im Szenario 3

Das in den Abschnitten 6.5.4.2 und 6.5.5.2 beschriebene Kriterium zur Beschränkung der Lastverteilung ergibt in dem hier beschriebenen Szenario 3 aufgrund der starken Schwankungen der $cwnd$-Werte keine sinnvollen Ergebnisse und ist daher im Falle der vorliegenden asymmetrischen Verteilung der Link-Bandbreiten so nicht anwendbar.

6.6 Abschätzende Bewertung der Algorithmen

In Abschnitt 6.3 wurde eine von Iyengar et al. dargelegte LS-SCTP-Variante beschrieben. Im folgenden Abschnitt 6.4 wurden zwei weitere im Laufe dieser Arbeit entstandene Modifikationen dieses Lastverteilungsalgorithmus diskutiert. Alle drei Algorithmen wurden mit Hilfe einer ereignisorientierten Simulation in der OPNET-Umgebung, basierend auf einem zuvor entwickelten SCTP-Modell, in verschiedenen Szenarien untersucht.
Hinsichtlich der wichtigsten Anwendung der SCTP-basierten Lastverteilung, dem Transport von Signalisierungsnachrichten über Breitbandnetze, können einige wichtige Schlüsse in Bezug auf die Dimensionierung von Transportnetzen für den Signalisierungstransport gezogen werden: in dem für Signalisierungsnetze relevanten Bereich der Linkverzögerungen $d < 50$ ms kann durch die Anwendung von Lastverteilungsalgorithmen eine erhebliche Steigerung des Nutzdurchsatzes erzielt werden. In Dualhoming-Szenarien ist eine Verdoppelung des Durchsatzes möglich, wenn gewährleistet ist, dass sich die Charakteristika der genutzten SCTP-Pfade in Bezug auf die zur Verfügung stehende Bandbreite und Verzögerungen nicht zu stark voneinander unterscheiden. In diesem Parameterbereich erzielt die im Laufe dieser Arbeit entwickelte PB-SACK-Variante (vgl. Abschnitt 6.4) im Vergleich zu den anderen Varianten – ohne eine signifikant höhere maximale Nachrichtenverzögerung – einen höheren oder mindestens gleichwertigen maximalen Durchsatz.
Ferner wurden Kriterien vorgeschlagen, die sich in Szenarien mit symmetrischen Link-Bandbreiten bei ungünstigen Parametern (hohe Pfadverzögerungen) anwenden lassen, um die Lastverteilung zu deaktivieren und somit auch unter diesen Umständen den maximalen Durchsatz zu erreichen.
Die Fluss- und Überlastkontroll-Algorithmen aller Varianten beruhen nach wie vor auf denen des Standard-SCTP; somit ist davon auszugehen, dass sie auch in Konkurrenz zu etablierten Protokollen (wie z.B. TCP) eingesetzt werden können,

ohne diese zu verdrängen. Es bedarf jedoch weiterer Untersuchungen des Zusammenspiels der erwähnten Algorithmen mit anderen Protokollen, die gemeinsame Links gleichzeitig nutzen. Da die in diesem Kapitel diskutierten Algorithmen in Kombination mit asymmetrischen Link-Verhältnissen insgesamt unzureichende Ergebnisse liefern, ist hier noch Optimierungspotenzial zu erwarten.

Kapitel 7

Zusammenfassung und Ausblick

In der vorliegenden Arbeit wurden Eigenschaften und Anwendungsbereiche des neuen IETF-Transportprotokolls SCTP vorgestellt. Das verbindungsorientierte SCTP zeichnet sich durch zuverlässigen Transport von Nachrichten, die Unterstützung des Multihoming, vielfältige Möglichkeiten der Nachrichtenzustellung, eine zuverlässige und schnelle Erkennung von Netzwerk- und Hostausfällen und aufgrund der flexiblen Erweiterbarkeit nicht zuletzt durch Zukunftssicherheit aus. Daher empfiehlt es sich als Transportprotokoll für eine Vielzahl verschiedener Anwendungen, von denen einige der wichtigsten im Kapitel 2 diskutiert wurden.
Um die Möglichkeit zu haben, das Protokoll in einem realen System zu testen und zu bewerten, wurde mit der `sctplib` eine umfangreiche und portable Implementierung erstellt. Bis zum gegenwärtigen Zeitpunkt ist dies die einzige auch für Windows-Betriebssysteme frei verfügbare Implementierung, was eine entscheidende Voraussetzung für die weitere Verbreitung des SCTP ist. Die `sctplib` bildete die Grundlage für weitreichende Untersuchungen, von der Bewertung des Zusammenspiels des neuen SCTP mit etablierten Transportprotokollen wie TCP bis hin zu Protokollmodifikationen. Dabei wurde u.a. auch die Eignung für den Einsatz als Transportprotokoll für Signalisierungsmeldungen untersucht.
Für noch weiter führende Untersuchungen des SCTP war der Einsatz der ereignisorientierten Simulation unabdingbar. Die Randbedingungen für die Erstellung eines Simulationsmodells wurden diskutiert; ebenso die Validierung des Modells durch den Vergleich mit den Ergebnissen aus dem Einsatz der `sctplib`-Implementierung.
Abschließend wurde der Aspekt der simultanen Pfadnutzung, der im SCTP-Standard nicht explizit vorgesehen ist, detaillierter untersucht. Dazu wurden – basierend auf dem Simulationswerkzeug OPNET Modeler – geeignete Werkzeuge

entwickelt, um diesen Aspekt in unterschiedlichen Szenarien zu bewerten.
Anwendungen der entsprechenden Algorithmen finden sich insbesondere beim Einsatz des SCTP für den Signalisierungstransport. Dabei kann es durchaus notwendig sein, das Merkmal der Multihoming-Unterstützung nicht allein für die Redundanz der Netzanbindung einzusetzen, sondern auch um hohe Belastungen abzufangen und die Netzauslastung optimaler zu verteilen. Neben den von Iyengar et al. in [41] untersuchten Algorithmen wurden auch noch eigene Optimierungen der Lastverteilungsalgorithmen vorgeschlagen und durch Simulationen gezeigt, dass sie in einem weiten Parameterraum bessere Ergebnisse liefern.
SCTP ist ein viel versprechendes, komplexes neues Transportprotokoll, welches mittlerweile für alle relevanten Betriebssysteme verfügbar ist. Die ersten Schritte, *neue* Anwendungsprotokolle an die Dienste des SCTP anzupassen, sind in der Standardisierung gegangen. In naher Zukunft werden auch etablierte Anwendungsprotokolle (z.B. FTP, HTTP, SSH) für die Nutzung mit SCTP optimiert werden.
Neue Erweiterungen des SCTP werden es erlauben, die Authentizität von SCTP-Kontrollnachrichten zu verifizieren [90] und somit die bereits in Abschnitt 2.4.2 erwähnte Funktionalität des SCTP-basierten Mobilitätsmanagements gesichert umzusetzen.
Darüber hinaus kann die Flexibilität des SCTP genutzt werden, um das Protokoll an die immer wichtigere Übertragung in drahtlosen Netzen so anzupassen, dass die in diesen Netzen vorherrschenden Beschränkungen hinsichtlich Paketverlusten und Latenzzeiten nicht – wie beispielsweise beim TCP – dazu führen, dass vorhandene Ressourcen nur unzureichend ausgeschöpft werden können. Entsprechende Vorarbeiten dazu sind mit dem Draft “SCTP Packet Drop Reporting” [81] bereits geleistet worden.

Literaturverzeichnis

[1] 3RD GENERATION PARTNERSHIP PROJECT (3GPP): *UTRAN Iu interface Radio Access Network Application Part (RANAP) signaling*, September 2003. 3GPP TS 25.413 V3.14.0 (2003-09).

[2] 3RD GENERATION PARTNERSHIP PROJECT (3GPP): *IP Multimedia Subsystem (IMS); Stage 2*, Januar 2005. 3GPP TS 23.228 V5.13.0 (2005-01).

[3] ALLMAN, M., V. PAXSON und W. STEVENS: *TCP Congestion Control*. IETF, Network Working Group, April 1999. RFC 2581.

[4] ANAGNOSTAKIS, K.G., M.B. GREENWALD und R.S. RYGER: *On the Sensitivity of Network Simulation to Topology*. In: *Proceedings of the Tenth IEEE/ACM Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS 2002)*, Seiten 117–128, 2002.

[5] BALK, A., M. SIGLER, M. GERLA und M. SANADIDI: *Investigation on MPEG-4 Video Streaming over SCTP*. In: *Proceedings of the SCI 2002*, Band 10, Seiten 337–340, Orlando, Juli 2002. IIIS. Mobile/Wireless Computing and Communication Systems II.

[6] BANSAL, D., H. BALAKRISHNAN, S. FLOYD und S. SHENKER: *Dynamic Behavior of Slowly-Responsive Congestion Control Algorithms*. In: *SIGCOMM 2001*, Seiten 263–274, San Diego, CA, August 2001.

[7] BLANTON, E. und M. ALLMAN: *On Making TCP More Robust to Packet Reordering*. ACM Computer Communication Review, 32(1), Januar 2002.

[8] BOEHM, B.W.: *Software Engineering Economics*. Englewood Cliffs, 1981.

[9] BOSSE, J.G. VAN: *Signaling in Telecommunication Networks*. Wiley Series in Telecommunications and Signal Processing. John Wiley and Sons, Inc., 1998.

[10] BRADEN, R.: *Requirements for Internet Hosts - Communication Layers*. IETF, Network Working Group, Oktober 1989. RFC 1122.

[11] BRENNAN, R. und T. CURRAN: *SCTP congestion control: Initial simulation studies*. In: SOUZA, J. MOREIRA DE, N.I.S. FONSECA und E.A. DE SOUZA E SILVA (Herausgeber): *Proceedings of the International Teletraffic Congress (ITC 17)*, Brazil, 2001. Elsevier Science.

[12] BRONSTEIN, I.N. und K.A. SEMENDJAJEW: *Taschenbuch der Mathematik*, Band 25. Auflage. B.G. Teubner Verlagsgesellschaft, Stuttgart, Leipzig, 1993.

[13] CALHOUN, P., J. LOUGHNEY, E. GUTTMAN, G. ZORN und J. ARKKO: *Diameter Base Protocol*. IETF, Network Working Group, September 2003. RFC 3588.

[14] CAMARILLO, G., R. KANTOLA und H. SCHULZRINNE: *Evaluation of Transport Protocols for the Session Initiation Protocol*. IEEE Network, Vol. 17(No. 5):40–46, September/Oktober 2003.

[15] CARO, A. und J. IYENGAR: *NS-2 SCTP module release 3.4*, August 2003. Siehe http://pel.cis.udel.edu.

[16] CARSON: *NIST Network Emulation Tool*. März 2001. Siehe http://www-x.antd.nist.gov/nistnet.

[17] CASE, J.D., M. FEDOR, M.L. SCHOFFSTALL und J. DAVIN: *Simple Network Management Protocol (SNMP)*. IETF, Network Working Group, Mai 1990. RFC 1157.

[18] DEERING, S. und R. HINDEN: *Internet Protocol, Version 6 (IPv6) Specification*. IETF, Network Working Group, Dezember 1998. RFC 2460.

[19] DIERKS, T. und C. ALLEN: *The TLS Protocol Version 1.0*. IETF, Network Working Group, Januar 1999. RFC 2246.

[20] DREIBHOLZ, T., A. JUNGMAIER und M. TÜXEN: *A New Scheme for IP-based Internet-Mobility*. In: *Proceedings of the LCN 2003*, Seiten 99–108, Bonn, Oktober 2003. IEEE Conference on Local Computer Networks.

[21] EASTLAKE, D. und P. JONES: *US Secure Hash Algorithm 1 (SHA1)*. IETF, Network Working Group, September 2001. RFC 3174.

[22] FLOYD, S.: *Congestion Control Principles*. IETF, Network Working Group, September 2000. RFC 2914.

[23] FLOYD, S.: *HighSpeed TCP for Large Congestion Windows*. IETF, Transport Area Working Group, Dezember 2003. RFC 3649.

[24] FLOYD, S., M. HANDLEY, J. PADHYE und J. WIDMER: *Equation-based congestion control for unicast applications*. In: *SIGCOMM 2000*, Seiten 43–56, Stockholm, Schweden, August 2000.

[25] FLOYD, S., J. MAHDAVI, M. MATHIS und M. PODOLSKY: *An Extension to the Selective Acknowledgement (SACK) Option for TCP*. IETF, Network Working Group, Juli 2000. RFC 2883.

[26] FLOYD, S., M. MATHIS, J. MAHDAVI und A. ROMANOW: *TCP Selective Acknowledgment Options*. IETF, Network Working Group, Oktober 1996. RFC 2018.

[27] FREE SOFTWARE FOUNDATION: *GNU Lesser General Public License*, Februar 1999. Version 2.1.

[28] GEORGE, T., B. BIDULOCK, R. DANTU, H.J. SCHWARZBAUER und K. MORNEAULT: *SS7 MTP2-User Peer-to-Peer Adaptation Layer*. IETF, Network Working Group, Juni 2004. draft-ietf-sigtran-m2pa-12.txt, work in progress.

[29] GILLIGAN, R. und E. NORDMARK: *Transition Mechanisms for IPv6 Hosts and Routers*. IETF, Network Working Group, August 2000. RFC 2893.

[30] INTERNATIONAL TELECOMMUNICATION UNION: *Introduction to CCITT Signaling System No. 7*, März 1993. ITU-T Recommendation Q.700.

[31] INTERNATIONAL TELECOMMUNICATION UNION: *Signaling System No. 7 – Message Transfer Part Signaling Performance*, März 1993. ITU-T Recommendation Q.706.

[32] INTERNATIONAL TELECOMMUNICATION UNION: *Signaling System No. 7 (SS7) - Message Transfer Part (MTP)*, März 1993. ITU-T Recommendations Q.701 - Q.705.

[33] INTERNATIONAL TELECOMMUNICATION UNION: *Message Transfer Part Level 3 functions and messages using the services of ITU Recommendation Q.2140*, Juli 1996. ITU-T Recommendation Q.2210 (07/96).

[34] INTERNATIONAL TELECOMMUNICATION UNION: *Signaling System No. 7 (SS7) - ISDN User Part (ISUP)*, 1996. ITU-T Recommendations Q.761 - Q.767.

[35] INTERNATIONAL TELECOMMUNICATION UNION: *Signaling System No. 7 (SS7) - Signaling Connection Control Part (SCCP)*, Juli 1996. ITU-T Recommendations Q.711 - Q.715.

[36] INTERNATIONAL TELECOMMUNICATION UNION: *Digital Subscriber Signaling System No. 1 (DSS 1) – ISDN user-network interface layer 3 – General aspects*, Mai 1998. ITU-T Recommendation Q.931.

[37] INTERNATIONAL TELECOMMUNICATION UNION: *Recommendation Z.100 - Specification and description language (SDL)*, November 1999. ITU-T Recommendation Z.100 (11/99).

[38] ISO/IEC: *International Standard MPEG-4 Systems*, Oktober 1998. Final Draft 14496-1.

[39] IYENGAR, J., P. AMER, A. CARO und R. STEWART: *Transport Layer Load Balancing in FCS Networks*. In: *Military Communications Conference (MILCOM) 2003*, Oktober 2003.

[40] IYENGAR, J., A. CARO, P. AMER und R. STEWART: *Making SCTP More Robust to Changeover*. In: *2003 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'03)*, Band 34, Seiten 123–135, Montreal, Kanada, Juli 2003.

[41] IYENGAR, J., K. SHAH, P. AMER und R. STEWART: *Concurrent Multipath Transfer Using SCTP Multihoming*. In: *2004 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'04)*, San Jose, CA, USA, Juli 2004.

[42] IYENGAR, J. R., P. D. AMER, R. STEWART und I. ARIAS-RODRIGUEZ: *Preventing SCTP Congestion Window Overgrowth During Changeover*. IETF, Transport Area Working Group, Februar 2004. draft-iyengar-sctp-cacc-02.txt, work in progress.

[43] JACOBSON, V., R. BRADEN und D. BORMAN: *TCP Extensions for High Performance*. IETF, Network Working Group, Mai 1992. RFC 1323.

[44] JOHNSON, D., C. PERKINS und J. ARKKO: *Mobility Support in IPv6*. IETF, Network Working Group, Juni 2004. RFC 3775.

[45] JUNGMAIER, A.: *SCTP for Beginners*, 2001-2005. Siehe http://tdrwww.exp-math.uni-essen.de/inhalt/forschung/sctp_fb/.

[46] JUNGMAIER, A., E. P. RATHGEB, M. SCHOPP und M. TÜXEN: *SCTP - A Multi-link End-to-end Protocol for IP-based Networks*. AEÜ Archiv für Elektronik und Übertragungstechnik, Vol. 55(1):46–54, 2001.

[47] JUNGMAIER, A., E.P RATHGEB und M. TÜXEN: *On the Use of SCTP in Failover-Scenarios*. In: *Proceedings of the SCI 2002*, Band 10, Seiten 363–368, Orlando, Juli 2002. Mobile/Wireless Computing and Communication Systems II.

[48] JUNGMAIER, A., E. RESCORLA und M. TÜXEN: *Transport Layer Security over Stream Control Transmission Protocol*. IETF, Network Working Group, Dezember 2002. RFC 3436.

[49] JUNGMAIER, A., M. SCHOPP und M. TÜXEN: *Performance Evaluation of the Stream Control Transmission Protocol*. In: *ATM 2000 - Proceedings of the IEEE Conference on High Performance Switching and Routing*, Seiten 141–148, 2000.

[50] KARN, P. und C. PARTRIDGE: *Improving round-trip time estimates in reliable transport protocols*. In: *Proceedings of the ACM workshop on Frontiers in computer communications technology*, Band 10, Seiten 2–7, Stowe, Vermont, USA, August 1987. ACM.

[51] KENT, S. und R. ATKINSON: *Security Architecture for the Internet Protocol*. IETF, Network Working Group, November 1998. RFC 2401.

[52] KOH, S.J. und Q. XIE: *Mobile SCTP with Mobile IP for Transport Layer Mobility*. IETF, Network Working Group, Juni 2004. draft-sjkoh-mobile-sctp-mobileip-04.txt, work in progress.

[53] KRAWCZYK, H., M. BELLARE und R. CANETTI: *HMAC: Keyed-Hashing for Message Authentication*. IETF, Network Working Group, Februar 1997. RFC 2104.

[54] MACFADEN, M., D. PARTAIN, J. SAPERIA und W. TACKABURY: *Configuring Networks and Devices with Simple Network Management Protocol (SNMP)*. IETF, Network Working Group, April 2003. RFC 3512.

[55] MARCO, G. DE, D. DE VITO, M. LONGO und S. LORETO: *SCTP as a transport for SIP: a case study*. In: *Proceedings of the SCI 2003*, Orlando, Juli 2003.

[56] MOGUL, J.C. und S.E. DEERING: *Path MTU discovery*. IETF, Network Working Group, November 1990. RFC 1191.

[57] MOLTENI, M. und M. VILLARI: *Using SCTP with Partial Reliability for MPEG-4 Multimedia Streaming*. In: *Proceedings of BSDCon Europe 2002*, Band 10, Seiten 337–340, Amsterdam, Niederlande, November 2002.

[58] MONTRESOR, ALBERTO, GIANNI DI CARO und POUL E. HEEGAARD: *BISON - Architecture of the Simulation Environment*. Technischer Bericht, Universität von Bologna, Januar 2004. Biology-Inspired Techniques for Self Organization in Dynamic Networks (BISON), IST-2001-38923.

[59] MÜLLER-CLOSTERMANN, B. und J. HINTELMANN: *Diskrete Simulation*. Fachgebiet Systemmodellierung, Universität Duisburg-Essen, 2004. Begleitmaterial zur Veranstaltung ”Diskrete Simulation”.

[60] ONG, L., I. RYTINA, M. GARCIA, H. SCHWARZBAUER, L. COENE, H. LIN, I. JUHASZ, M. HOLDREGE und C. SHARP: *Framework Architecture for Signaling Transport*. IETF, Network Working Group, Oktober 1999. RFC 2719.

[61] PARTRIDGE, CRAIG, JON C. R. BENNETT und NICHOLAS SHECTMAN: *Packet Reordering is Not Pathological Network Behavior*. IEEE/ACM Transactions on Networking, 7(6), Dezember 1999.

[62] PASTOR, J. und M. BELINCHON: *Stream Control Transmission Protocol Management Information Base*. IETF, Transport Area Working Group, September 2004. RFC 3873.

[63] PERKINS, C.: *IP Mobility Support for IPv4*. IETF, Network Working Group, August 2002. RFC 3344.

[64] POSTEL, J.: *User Datagram Protocol*. IETF, Network Working Group, August 1980. RFC 768.

[65] POSTEL, J.: *Internet Protocol*. IETF, Network Working Group, September 1981. RFC 791.

[66] POSTEL, J.: *Transmission Control Protocol*. IETF, Network Working Group, September 1981. RFC 793.

[67] RAMAKRISHNAN, K., S. FLOYD und D. BLACK: *The Addition of Explicit Congestion Notification (ECN) to IP*. IETF, Network Working Group, September 2001. RFC 3168.

[68] REJAIE, R., M. HANDLEY und D. ESTRIN: *RAP: An End-to-End Rate-Based Congestion Control Mechanism for Realtime Streams in the Internet*. In: *INFOCOM (3)*, Seiten 1337–1345, 1999.

[69] RIEGEL, M. und M. TÜXEN: *Mobile SCTP*. IETF, Network Working Group, Oktober 2004. draft-riegel-tuexen-mobile-sctp-04.txt, work in progress.

[70] RIGNEY, C., A. RUBENS, W. SIMPSON und S. WILLENS: *Remote Authentication Dial In User Service (RADIUS)*. IETF, Network Working Group, April 1997. RFC 2138.

[71] RIZZO, L.: *Dummynet: a simple approach to the evaluation of network protocols*. ACM Computer Communication Review, 27:31–41, Jan. 1997.

[72] ROSENBERG, J., H. SCHULZRINNE und G. CAMARILLO: *The Stream Control Transmission Protocol as a Transport for the Session Initiation Protocol*. IETF, Transport Area Working Group, November 2004. draft-ietf-sip-sctp-05.txt, work in progress.

[73] ROSENBERG, J., H. SCHULZRINNE, G. CAMARILLO, A. JOHNSTON, J. PETERSON, R. SPARKS, M. HANDLEY und E. SCHOOLER: *SIP: Session Initiation Protocol*. IETF, Network Working Group, Juni 2002. RFC 3261.

[74] *SCTPLIB Discussion Mailing Liste*, Januar 2005. Siehe http://www.sctp.de/sctp.html.

[75] SEWARD, J. und N. NETHERCOTE: *Valgrind: A Program Supervision Framework*. In: SOKOLSKY, OLEG und MAHESH VISWANATHAN (Herausgeber): *Electronic Notes in Theoretical Computer Science*, Band 89. Elsevier, 2003. Siehe auch http://valgrind.kde.org/.

[76] SIDEBOTTOM, G., K. MORNEAULT und J. PASTOR-BALBAS: *Signaling System 7 (SS7) Message Transfer Part 3 (MTP3) - User Adaptation Layer (M3UA)*. IETF, Network Working Group, September 2002. RFC 3332.

[77] SLATTERY, T.C., M. MUUSS und T. SLATTERY: *TTCP: a Test of TCP and UDP Performance*, Dezember 1984. http://ftp.arl.mil/ mike/ttcp.html.

[78] SPRING, N., D. WETHERALL und D. ELY: *Robust Explicit Congestion Notification (ECN) Signaling with Nonces*. IETF, Network Working Group, Juni 2003. RFC 3540.

[79] STEWART, R.: *On the Use of Stream Control Transmission Protocol (SCTP) with IPsec*. IETF, Network Working Group, Juni 2003. RFC 3554.

[80] STEWART, R., I. ARIAS-RODRIGUEZ, K. POON, A. CARO und M. TÜXEN: *Stream Control Transmission Protocol (SCTP) Implementer's Guide*. IETF, Network Working Group, Oktober 2004. draft-ietf-tsvwg-sctpimpguide-12.txt, work in progress.

[81] STEWART, R., P. LEI und M. TÜXEN: *Stream Control Transmission Protocol (SCTP) Packet Drop Reporting*. IETF, Transport Area Working Group, Juni 2004. draft-stewart-sctp-pktdrprep-01.txt, work in progress.

[82] STEWART, R., M. RAMALHO, Q. XIE, M. TÜXEN und P. CONRAD: *Stream Control Transmission Protocol (SCTP) Partial Reliability Extension*. IETF, Network Working Group, Mai 2004. RFC 3758.

[83] STEWART, R., M. RAMALHO, Q. XIE, M. TÜXEN und P. CONRAD: *Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration*. IETF, Network Working Group, Januar 2005. draft-ietf-tsvwg-addip-sctp-10.txt, work in progress.

[84] STEWART, R. und Q. XIE: *Multi-Network Datagram Transmission Protocol (MDTP)*. IETF, Network Working Group, Juni 2000. draft-ietf-sigtran-mdtp-0X.txt, expired December 2000.

[85] STEWART, R. und Q. XIE: *Stream Control Transmission Protocol (SCTP)*. Addison Wesley, 2002. A Reference Guide.

[86] STEWART, R., Q. XIE, K. MORNEAULT, C. SHARP, H. SCHWARZBAUER, T. TAYLOR, I. RYTINA, M. KALLA, L. ZHANG und V. PAXSON: *Stream Control Transmission Protocol*. IETF, Network Working Group, Oktober 2000. RFC 2960.

[87] STEWART, R., Q. XIE, M. STILLMAN und M. TÜXEN: *Aggregate Server Access Protocol (ASAP)*. IETF, Network Working Group, Oktober 2004. draft-ietf-rserpool-asap-10.txt, work in progress.

[88] STEWART, R., Q. XIE, L. YARROLL, J. WOOD, K. POON, K. FUJITA und M. TÜXEN: *Sockets API Extensions for Stream Control Transmission Protocol*. IETF, Network Working Group, September 2004. draft-ietf-tsvwg-sctpsocket-09.txt, work in progress.

[89] STONE, J., R. STEWART und D. OTIS: *Stream Control Transmission Protocol (SCTP) Checksum Change*. IETF, Network Working Group, September 2002. RFC 3309.

[90] TÜXEN, M., R. STEWART und P. LEI: *Authenticated Chunks for Stream Control Transmission Protocol (SCTP)*. IETF, Transport Area Working Group, Juli 2004. draft-tuexen-sctp-auth-chunk-01.txt, work in progress.

[91] TÜXEN, M., Q. XIE, R. STEWART, M. SHORE und J. LOUGHNEY: *Architecture for Reliable Server Pooling*. IETF, Network Working Group, Oktober 2004. draft-ietf-rserpool-arch-08.txt, work in progress.

[92] WETHERALL, D. und C. LINDBLAD: *Extending Tcl for dynamic object-oriented programming*. In: *Proceedings of the Tcl/Tk Workshop*, Ontario, Canada, Juli 1995.

[93] XIE, Q., R. STEWART, M. STILLMAN und M. TÜXEN: *Endpoint Name Resolution Protocol (ENRP)*. IETF, Network Working Group, Oktober 2004. draft-ietf-rserpool-enrp-10.txt, work in progress.

[94] ZEITOUN, AMGAD, CHEN-NEE CHUAH, SUPRATIK BHATTACHARYYA und CHRISTOPHE DIOT: *An AS-Level Study of Internet Path Delay Characteristics*. In: *Global Internet and Next Generation Network (GINGN) Workshop*, Dallas, TX, November 2004.

[95] ZUIDWEG, J.: *Next Generation Intelligent Networks*. Artech House Telecommunications Library. Artech House, Inc., 2002.

Anhang A

SCTP Dienstzugangspunkte

A.1 Die Schnittstelle der SCTPLIB

Abschnitt 10 von [86] stellt beispielhaft einen Dienst-Zugangspunkt für das SCTP vor. Dieser wurde als Grundlage für die Entwicklung der `sctplib` genommen. Dabei existieren Funktionen für den Auf- und Abbau von Verbindungen, zum Senden und Empfangen von Nachrichten, und für die Überwachung des Status.
Folgende Dienstprimitive dienen dabei als Eintrittspunkte in die SCTPLIB:

1. Die Initialisierung der `sctplib` wird durch die Funktion `sctp_initLibrary()` durchgeführt. Diese Funktion muss vor der Ausführung aller weiteren Funktionen einmal aufgerufen werden, und initialisiert die für die Netzwerk Ein- und Ausgabe nötigen Sockets, sowie alle weiteren Strukturen der `sctplib`.

2. INITIALIZE. Dieses Nutzerprimitiv wird durch die Funktion `sctp_registerInstance()` implementiert, mit der eine SCTP-Instanz angelegt wird, aus der nachfolgend SCTP-Assoziationen erzeugt werden können. Dieser Funktion werden entsprechende Parameter übergeben, wie z.B. lokaler Port (für Server), maximale Anzahl der Nachrichtenströme, und die lokal gebundenen Adressen. Wird `INADDR_ANY` angegeben, so werden alle lokalen Adressen gebunden. Der Rückgabewert identifiziert die SCTP-Instanz eindeutig. Eine Instanz kann einem Server oder einem Client zugeordnet werden, je nach Art des übergebenen Ports. Ist der Port ungleich Null (Server), so können ankommende gültige COOKIE-ECHO Chunks eine neue Assoziation erzeugen. Ist der Port

gleich Null (Client), so kann die Anwendung durch den Aufruf des Nutzerprimitivs *ASSOCIATE* den Aufbau einer neuen Assoziation beginnen, und den zugehörigen Port zufällig auswählen lassen.

3. ASSOCIATE. Ermöglicht der Anwendung den Aufbau einer neuen Assoziation. Die `sctplib` erzeugt eine Assoziations-Struktur, sendet einen INIT-Chunk, und wechselt in den Zustand COOKIE-WAIT (vgl. Abbildung 2.9). Die zugehörige Funktion heißt `sctp_associate()`. Sie erhält als Parameter die Zieladresse und den Zielport des jeweiligen Servers, sowie die zugehörige SCTP Instanz und die maximale Anzahl von Nachrichtenströmen zum Server.

4. SHUTDOWN. Beendet die Assoziation, nachdem zuvor alle ausstehenden Daten zuverlässig zugestellt wurden. Wird durch die Funktion `sctp_shutdown()` implementiert.

5. ABORT. Beendet die Assoziation sofort, d.h. ohne dass der zuverlässige Datentransport gewährleistet ist. Wird durch die Funktion `sctp_abort()` implementiert.

6. SEND. Übergibt Daten an die `sctplib`, nachdem eine Assoziation etabliert wurde. Diese werden durch das Protokoll an den assoziierten Endpunkt gesendet. Die entsprechende Funktion `sctp_send()` erhält neben einem Bezeichner für die entsprechende Assoziation einen Zeiger auf die Nutzerdaten, deren Länge, die Stream-ID, den Pfad, über den die Daten gesendet werden sollen (s.u.), Flags für die reihenfolgegesicherte Zustellung, sowie die Lebensdauer in Millisekunden.

7. SET PRIMARY. Setzt einen (neuen) Primärpfad. Über diesen Pfad werden fortan alle erstmals übertragenen Daten-Chunks gesendet. Die zugehörige Funktion der `sctplib` heißt `sctp_setPrimary()`.

8. RECEIVE. Sind Daten für den Benutzerprozess verfügbar, so können sie mit dem Receive-Primitiv abgeholt werden. Die entsprechende Funktion der `sctplib` heißt `sctp_receive()`. Ihr wird ein Zeiger auf einen Pufferbereich und die Größe des Puffers übergeben, sowie die Stream ID, auf der Daten empfangen werden sollen. Als Rückgabewerte liefert diese Funktion die tatsächliche Länge der in den Pufferspeicher kopierten Daten, die Strom-Sequenznummer und die TSN der empfangenen Nachricht (bzw. eine TSN bei segmentierten Daten).

9. STATUS. Der RFC 2960 sieht eine Funktion zur Abfrage des Assoziationszustands vor. Aufgrund der komplexen Strukturen wurde diese Funktion durch drei Schnittstellenfunktionen der `sctplib` implementiert. `sctp_getAssocDefaults()` liest Parameter, die sich auf eine SCTP-Instanz beziehen und nachfolgend von neu erzeugten Assoziationen übernommen werden (wie z.B. `path.rto.min`, `path.rto.mx`), oder `assoc.max.retrans`). `sctp_getAssocStatus()` liest die entsprechenden Parameter einer aktuell etablierten Assoziation. Schließlich liefert `sctp_getPathStatus()` die zu einem Pfad gehörigen Parameter aus, darunter z.B. den Pfadzustand, den aktuellen $cwnd$-Wert, oder den aktuellen Schätzwert für die Rundlaufzeit (engl. smoothed round trip time, SRTT).

10. CHANGE HEARTBEAT. Schaltet den Heartbeat für einen Pfad an oder aus, oder setzt ein neues Intervall für das Aussenden von Heartbeats. Die Funktion der `sctplib` heißt `sctp_changeHeartBeat()`.

11. REQUEST HEARTBEAT. Sendet einen HEARTBEAT-REQUEST Chunk über einen Pfad. Die Funktion dazu heißt `sctp_requestHeartbeat()`.

12. GET SRTT REPORT. Liefert den aktuellen Schätzwert für die Rundlaufzeit eines Pfades. Diese Funktion implementiert eine Untermenge der Funktion `sctp_getPathStatus()`, steht der `sctplib` aber als separate Funktion mit Namen `sctp_getSrttReport()` zur Verfügung.

13. SET PROTOCOL PARAMETERS. Erlaubt das Setzen von Protokollparametern. Entsprechend dem Primitiv “STATUS” können hierbei mit drei Schnittstellenfunktionen die Parameter für Instanzen, Assoziationen und Pfade gesetzt werden. Die Funktionen der `sctplib` heissen `sctp_setAssocDefaults()`, `sctp_setAssocStatus()` und `sctp_setPathStatus()`.

14. SET FAILURE THRESHOLD. Setzt den SCTP Protokollparameter `path.max.retrans` für einen Pfad einer etablierten Assoziation. Die Funktion dazu heißt `sctp_setFailureThreshold()` und implementiert eine Untermenge der Funktion `sctp_setPathStatus()`.

15. RECEIVE UNSENT. Eine Assoziation kann beendet werden, bevor alle Daten übertragen wurden. Diese Schnittstellenfunktion erlaubt in

diesem Fall, alle noch nicht gesendeten Nachrichten aus der Sendewarteschlange auszulesen. Diese können dann gegebenenfalls über eine andere Assoziation gesendet werden. Die Funktion dazu heißt `sctp_receiveUnsent()`.

16. RECEIVE UNACKED. Ebenso kann es vorkommen, dass eine Assoziation beendet wurde, und Daten zwar gesendet, aber nie von der Gegenseite bestätigt wurden. Die Funktion `sctp_receiveUnacked()` liest diese Daten-Chunks aus der entsprechenden Warteschlange aus.

17. DESTROY. Wurde eine Assoziation beendet, und können ihre Ressourcen freigegeben werden, so kann die Anwendung die Funktion `sctp_deleteAssociation()` aufrufen. Es gibt auch eine solche Funktion für registrierte Instanzen: Der von ihnen belegte Speicher kann mit `sctp_unregisterInstance()` freigegeben werden.

Bei der Registrierung von Instanzen kann eine Anwendung mehrere Rückruffunktionen an die `sctplib` übergeben. Tritt das zu einer solchen Funktion gehörige Ereignis ein, wird die Funktion aufgerufen, und ihr werden alle notwendigen Parameter übergeben, um auf das Ereignis entsprechend zu reagieren. Hierbei wurde vorgesehen, dass pro Instanz genau eine Struktur mit solchen Rückruffunktionen übergeben werden kann. Eine Serverinstanz, die mehrere aktive Assoziationen verwaltet, muss mit diesen Rückruffunktionen auf die möglichen Ereignisse aller Assoziationen reagieren können. Während die oben beschriebenen Funktionen den Request-Methoden des Dienst-Zugangspunkts entsprechen, folgt nun die Beschreibung der Funktionen 1 bis 8 für die Indication-Methoden. Die letzten beiden dieser Funktionen wurden nicht in [86] vorgesehen, aber als Hilfestellung zur effizienten Anwendungsentwicklung in die `sctplib` integriert.

1. DATA ARRIVE. Wird für jede vollständige, angekommene Nachricht aufgerufen, die an den SCTP-Nutzer übergeben werden kann. Sie teilt der Anwendung Informationen wie die Länge der empfangenen Nachricht, Bezeichner des Nachrichtenstroms, Strom-Sequenznummer und Flags mit.

2. SEND FAILURE. Zeigt der Anwendung an, dass das Senden einer Nachricht fehlgeschlagen ist.

3. NETWORK STATUS CHANGE. Zeigt die Änderung des Zustands eines Pfades an. Übergibt der Anwendung einen Bezeichner (numerischen Index) für den Pfad und den neuen Pfadzustand.

4. COMMUNICATION UP. Teilt der Anwendung mit, dass eine neue Verbindung aufgebaut wurde. Diese Funktion übergibt alle wichtigen Daten der neuen Assoziation an die Anwendung, wie z.B. die Anzahl der ankommenden und ausgehenden Streams, Anzahl der Pfade zum assoziierten Endpunkt und den Grund für die Benachrichtigung (entweder Ankunft eines gültigen Cookie, oder Ankunft eines COOKIE-ACK Chunks).

5. COMMUNICATION LOST. Wird eine Assoziation beendet, weil entweder die Fehlerschwelle `assoc.max.retrans` überschritten wird, alle Pfade inaktiv werden, oder ein ABORT-Chunk vom assoziierten Endpunkt kommt, dann wird die Anwendung mit dieser Rückruffunktion benachrichtigt.

6. COMMUNICATION ERROR. Dies ist eine optionale Benachrichtigung im Fehlerfall (z.B. beim Empfang eines ERROR-Chunks). Wird gegenwärtig nicht durch die `sctplib` implementiert.

7. RESTART. Wird bei einer etablierten Assoziation ein erneuter Verbindungsaufbau durchgeführt, so nennt man dies Assoziations-Neustart. Dabei können einige Parameter (z.B. die Anzahl der Nachrichtenströme) neu ausgehandelt werden. Eine Änderung der zu einer Assoziation gehörigen Adressen könnte aber Konsequenzen hinsichtlich der Sicherheit einer Assoziation haben, und ist daher verboten.

8. SHUTDOWN COMPLETE. Wird die Verbindung ordnungsgemäß beendet, so wird die Anwendung darüber mit Hilfe dieser Funktion unterrichtet.

9. PEER SHUTDOWN RECEIVED. Mit dieser Funktion wird die Anwendung davon unterrichtet, dass der assoziierte Endpunkt damit begonnen hat, die Assoziation zu beenden (d.h. es wurde ein SHUTDOWN-Chunk empfangen).

10. QUEUE STATUS CHANGE. Die `sctplib` verfügt über eine Sendewarteschlange. Diese kann auf eine maximale Länge (z.B. 100 Nachrichten) beschränkt werden. Wird die Länge beschränkt und überschritten, so wird die Anwendung benachrichtigt. Ein anschließender SEND-Aufruf wird dann mit einem Fehler abgelehnt.

A.2 Die SCTP Socket-Schnittstelle

A.2.1 Die TCP-ähnliche Schnittstelle

Dieser Socket-Typ bietet eine größtmögliche Kompatibilität mit dem verbindungsorientierten TCP. Dadurch können Programme, die bisher nur TCP nutzten, mit geringem Aufwand auch an SCTP angepasst werden. Ähnlich wie beim TCP wird eine typische Serveranwendung durch eine bestimmte Abfolge von Betriebssystem-Funktionen aufgebaut:

- `socket()`. Erzeugt einen Assoziationsendpunkt und gibt einen Datei-Deskriptor (oder: Socket-Deskriptor) zurück. Die Syntax ist wie folgt:

  ```
  sd = socket(PF_INET, SOCK_STREAM, IPPROTO_SCTP);
  ```

  Der einzige Unterschied zum TCP ist mithin die Verwendung des Protokolltyps `IPPROTO_SCTP` statt `IPPROTO_TCP`.

- `bind()`. Diese Funktion weist einem lokalen Endpunkt eine Transportadresse (Kombination aus IP-Adresse und Port) zu. Die Anwendung kann auch die Wildcard-Adresse `INADDR_ANY` angeben, um alle lokalen Adressen des Endgeräts zu binden. Sollen selektiv spezielle lokale Adressen gebunden werden, so steht neben `bind()` auch noch die SCTP-spezifische Funktion `sctp_bindx()` zur Verfügung (siehe unten).

- `listen()`. Erst wenn diese Funktion aufgerufen wird, akzeptiert ein Server auch ankommende INIT-Chunks und ist bereit, eine Verbindung aufzubauen.

- `accept()`. Diese blockierende Funktion liefert einen *neuen* Socket-Deskriptor für eine etablierte Assoziation zurück. Wurden mehrere Assoziationen aufgebaut, bevor `accept()` aufgerufen wurde, so werden diese in einer Liste verwaltet, und der Socket-Deskriptor für die erste Assoziation dieser Liste zurückgegeben. In diesem Fall blockiert `accept()` natürlich nicht.

- `recv()`. Empfängt Daten über den neu erzeugten Socket-Deskriptor. Alternativ kann auch die Funktion `recvmsg()` aufgerufen werden. Diese liefert dem Empfänger zusätzliche Daten (wie z.B. die Absenderadresse).

- `send()`. Sendet Daten über den neu erzeugten Socket-Deskriptor an den assoziierten Endpunkt. Alternativ kann auch die Funktion `sendmsg()` benutzt werden. Dieser können zusätzliche Daten übergeben werden (z.B. bevorzugte Zieladresse, ein Feld mit Zeigern auf verschiedene Pufferspeicher, die alle nacheinander zu einer Nachricht zusammengefügt werden).
- `close()`. Beendet die Assoziation. Dazu wird die SCTP-Shutdown Prozedur eingeleitet (vgl. “ordentliche Beendigung” im Abschnitt 2.3.4).

Eine typische Clientanwendung hingegen baut eine Verbindung zum entsprechenden Server auf, sendet oder empfängt Daten, und beendet die Verbindung (vereinfachte Darstellung).

- `socket()`. Siehe oben.
- `connect()`. Startet den Aufbau einer Assoziation mit der in dieser Funktion übergebenen Zieladresse. Dies entspricht dem oben erwähnten ASSOCIATE-Primitiv. Blockiert gegebenenfalls bis die Assoziation etabliert ist.
- `send()`. Siehe oben. Sendet z.B. einen Request an den Server.
- `recv()`. Siehe oben. Empfängt z.B. die Antwort auf den Request.
- `close()`. Siehe oben.

A.2.2 Die UDP-ähnliche Schnittstelle

Die UDP-ähnliche Schnittstelle des SCTP zeichnet sich dadurch aus, dass der Aufbau von ausgehenden Verbindungen automatisch durchgeführt wird (d.h. die Anwendung ruft nicht explizit `connect()` auf, sondern direkt `sendmsg()`. Nachrichten werden vollständig zugestellt (zumindest bis zu einer bestimmten Länge; sind sie länger, so kann die Nachricht gegebenenfalls in Teilen gelesen werden). Ein Socket-Deskriptor hat eine 1-zu-N Beziehung zu N zugeordneten Assoziationen.
Damit ist die typische Reihenfolge von System-Aufrufen für einen UDP-ähnlichen SCTP-Server die folgende:

1. `socket()`. Erzeugt einen Assoziationsendpunkt und gibt einen Datei-Deskriptor (oder: Socket-Deskriptor) zurück. Die Syntax ist wie folgt:

   ```
   sd = socket(PF_INET, SOCK_SEQPACKET, IPPROTO_SCTP);
   ```

Unterschiede zum UDP sind die Verwendung des Protokolls `IPPROTO_SCTP` statt `IPPROTO_UDP`, sowie des Typs `SOCK_SEQPACKET` statt `SOCK_DGRAM`.

2. `bind()`. Siehe oben.

3. `listen()`. Bei der UDP-ähnlichen Schnittstelle müssen Server-Anwendungen keinen `accept()`-Aufruf durchführen, um Assoziationen anzunehmen. Durch den Aufruf von `listen()` wird es dem Socket ermöglicht, den Assoziationsaufbau durchzuführen. Neue Assoziationen werden automatisch akzeptiert, und der Anwendung wird dies durch eine Benachrichtigung mitgeteilt, die durch `recvmsg()` empfangen werden kann.

4. `recvmsg()`. Wird zum Empfang von Nutzdaten, Zusatzdaten und Benachrichtigungen verwendet. Die Funktion füllt dazu eine Struktur vom Typ `struct msghdr` aus, die sowohl Felder mit Zeigern auf Pufferbereiche für Nutzdaten enthält, als auch Zeiger auf zusätzliche Datenstrukturen (engl. ancillary data), die sonstige Informationen enthalten. Ein wichtiges Beispiel für solche Zusatzdaten ist die Struktur vom Typ `struct sctp_sndrcvinfo`, die u.a. die Nummer des Nachrichtenstroms, die Strom-Sequenznummer und TSN einer empfangenen Nachricht enthält.

5. `sendmsg()`. Wird zum Senden von Nutzdaten eingesetzt. Die Anwendung übergibt einen Zeiger auf eine Struktur vom Typ `struct msghdr`, und kann über Zusatzdaten, ähnlich wie bei der `recvmsg()`-Funktion, die Inhalte der gesendeten Chunks steuern. So kann beispielsweise eine Struktur vom Typ `struct sctp_sndrcvinfo` übergeben werden, die die Nummer des Nachrichtenstroms festlegt, auf der die Nachricht gesendet werden soll.

6. `close()`. Beendet *alle* zum Socket-Deskriptor gehörigen Assoziationen.

Eine typische Client-Anwendung, die den UDP-ähnlichen Socket nutzt, wird mindestens die folgenden System-Funktionen einsetzen:

1. `socket()`. Siehe oben.

2. `sendmsg()`. Der Aufruf dieser Funktion baut automatisch eine Assoziation zu dem SCTP-Endpunkt auf, dessen Transportadresse dieser Funktion übergeben wurde.

3. `recvmsg()`. Siehe oben.
4. `close()`. Siehe oben. In der Regel wird dem Socket-Deskriptor des Clients aber nur eine einzige Assoziation zugeordnet sein.

Beim UDP-ähnlichen Socket werden also alle Assoziationen, die mit einem SCTP-Endpunkt verbunden sind, durch einen einzigen Socket-Deskriptor repräsentiert. Falls eine dieser Assoziationen getrennt behandelt werden soll (z.B. wenn ein Server nur diese eine Assoziation beenden möchte), dann kann die Anwendung eine neue, SCTP-spezifische Funktion aufrufen, und erhält einen neuen Socket-Deskriptor, der fortan nur diese spezielle Assoziation in einer 1-zu-1 Beziehung (also als TCP-ähnlichen Socket) repräsentiert. Diese Funktion heißt `sctp_peeloff()` und wird weiter unten näher erläutert.

A.2.3 Neue, SCTP-spezifische Socket-Optionen

Abschnitt 7 von [88] definiert eine Reihe von neuen Socket-Optionen, die pro Socket-Deskriptor gesetzt oder gelesen werden können. Diese entsprechen grundsätzlich den unter Abschnitt A.1 bereits erwähnten Primitiven STATUS und SET PROTOCOL PARAMETERS, da sie einen ähnlichen Zweck erfüllen. Jedoch werden die Optionen für einen Socket-Deskriptor gesetzt, und können dabei durch die mögliche 1-zu-N Beziehung bei UDP-ähnlichen Sockets mehrere Assoziationen betreffen. Der Standard lässt dabei offen, ob Implementierungen diese Parameter über die herkömmlichen System-Funktionen `setsockopt()`/`getsockopt()` setzen bzw. lesen, oder aber eine entsprechende Funktion `sctp_opt_info()` zur Verfügung stellen, der als Parameter u.a. eine Konstante, ein Zeiger und eine Adressstruktur übergeben werden können. Die Konstante dient dazu, anzuzeigen, welche Art von Informationen gesetzt oder abgefragt werden sollen, der Zeiger übernimmt die entsprechenden Datenstrukturen, und die Adresse bezeichnet gegebenenfalls einen bestimmten Pfad. Auch hier gilt, dass die genaue Syntax und die übergebenen Typen in [88] beschrieben sind. Grundsätzlich ist vorgesehen, dass bei der SCTP Socket-Schnittstelle die folgenden Datenstrukturen gesetzt oder abgefragt werden können:

- SCTP_RTOINFO. Liest oder setzt die Parameter `assoc.rto.max`, `assoc.rto.min` und `assoc.rto.initial`, also die Grenzwerte für den Ablauf von Neuübertragungs-Timern.
- SCTP_ASSOCINFO. Liest oder setzt (falls möglich) u.a. die Parameter `assoc.max.retrans` (für eine Assoziation), die Anzahl der Pfade zum

assoziierten Endpunkt, und die letzten $rwnd$-Werte, die gesendet oder empfangen wurden.

- SCTP_INITMSG. Übergibt die maximale Anzahl von ausgehenden und ankommenden Streams, die maximale Anzahl möglicher Neuübertragungen des INITs bis ein Fehler gemeldet wird, und die maximale Timerlaufzeit bevor eine Neuübertragung ausgelöst wird (normalerweise wird nach jeder Neuübertragung diese Laufzeit verdoppelt, bis die angegebene maximale Laufzeit erreicht ist).

- SCTP_DEFAULT_SEND_PARAM. Setzt Standardparameter für übertragene Daten-Chunks. Dazu gehören Stream-ID, Flags, Protokoll-ID für die Nutzlast und Lebensdauer.

- SCTP_GET_PEER_ADDR_INFO. Entspricht im Wesentlichen der oben erwähnten Funktion `sctp_getPathStatus()`. Es werden Daten über den Pfadzustand, $cwnd$, Schätzwert für die Rundlaufzeit, Pfad-MTU oder aktueller Timeout für den Pfad zurückgegeben. Diese Informationen werden nur gelesen.

- SCTP_PRIMARY_ADDR. Setzt einen neuen Primärpfad. Es wird eine Adresse angegeben, die zum assoziierten Endpunkt gehören muss.

- SCTP_SET_PEER_PRIMARY_ADDR. Fordert den assoziierten Endpunkt auf, die übergebene (lokale) Adresse zum neuen Primärpfad zu machen. Diese Option wird nur unterstützt, wenn die SCTP-Erweiterung gemäß Abschnitt 2.4.2 implementiert wurde.

- SCTP_PEER_ADDR_PARAMS. Setzt oder liest die aktuellen Werte für das Intervall, mit dem HEARTBEAT-Chunks gesendet werden, und die maximale Anzahl von Neuübertragungen, nach der ein Pfad in den Zustand *Inaktiv* wechselt.

- SCTP_STATUS. Entspricht im Wesentlichen dem Primitiv “STATUS” (s.o.), bezogen auf eine Assoziation. Es wird der Zustand der Assoziation (z.B. CLOSED), das $rwnd$ des assoziierten Endpunkts, die Zahl noch unbestätigter Chunks, die Zahl von Chunks, die empfangen werden können, und die Zahl ankommender und abgehender Streams zurückgegeben. Ferner werden noch Pfadinformationen über den Primärpfad übergeben, sowie die Länge, ab der Nachrichten durch das SCTP fragmentiert werden.

- SCTP_AUTOCLOSE. Setzt eine Zeit in Sekunden, nach der Assoziationen automatisch geschlossen werden können, wenn sie im UDP-ähnlichen Modus sind, und in dieser Zeit im etablierten Zustand keine Daten gesendet oder empfangen haben.

- SCTP_EVENTS. Eine Anwendung kann Ereignisse anfordern. Tritt ein angefordertes Ereignis auf, so kann die Funktion ereignisspezifische Daten genauso wie Nutzdaten mit der `recvmsg()`-Funktion lesen. Dabei zeigt ein gesetztes Flag an, dass keine Nutzdaten, sondern ereignisspezifische Daten gelesen werden. Bei bestimmten Ereignissen können neben Nutzdaten auch Zusatzdaten (engl. ancillary data) übergeben werden, wie z.B. die Informationen vom Typ `struct sctp_sndrcvinfo` (also Stream-ID, Sequenznummer, etc.) beim Empfang von Nachrichten. Typische Ereignisse sind das Senden und Empfangen von Nutzdaten, die Änderung des Assoziationsstatus (Auf- und Abbau), Fehler sowie Änderungen eines Pfadzustands.

A.2.4 SCTP-spezifische Schnittstellen-Funktionen

Neben den Socket-Optionen definiert der Standard [88] auch eine Reihe von neuen Funktionen, die noch nicht als Systemfunktionen zur Verfügung stehen, und daher zur effizienten Nutzung des SCTP entweder als neue System- oder Bibliotheksfunktionen implementiert werden müssen. Diese sind:

1. `sctp_bindx()`. Diese Funktion ist eine Erweiterung der `bind()`-Funktion, mit der entweder genau eine Adresse an einen Socket gebunden werden kann, oder über die Konstante `INADDR_ANY` alle Adressen eines Endpunktes. Die Funktion `sctp_bindx()` kann nun selektiv zusätzliche Adressen des lokalen Endpunktes an den Socket binden, oder Adressen, die bereits an einen Socket gebunden wurden, löschen. Unterstützt die SCTP-Implementierung die Erweiterung zur dynamischen Adress-Rekonfiguration (siehe Abschnitt. 2.4.2), so können auch durch das Hinzufügen oder Löschen von Adressen zu etablierten Assoziationen entsprechende Nachrichten an den jeweils assoziierten Endpunkt gesendet werden.

2. `sctp_peeloff()`. Diese Funktion kann aus einem UDP-ähnlichen (1-zu-N) Socket eine etablierte Assoziation lösen und gibt für diese einen neuen Socket-Deskriptor zurück. Dieser entspricht dann einem TCP-ähnlichen (1-

zu-1) Socket und kann (und muss) nachfolgend entsprechend benutzt werden.

3. `sctp_getpaddrs()`. Diese Funktion gibt alle Adressen des assoziierten Endpunkts in einer etablierten Assoziation zurück. Dabei werden Ressourcen reserviert, die mit der entsprechenden Funktion `sctp_freepaddrs()` wieder freigegeben werden müssen.

4. `sctp_getladdrs()`. Diese Funktion gibt alle Adressen zurück, die an einen lokalen Socket gebunden sind. Dabei werden Ressourcen reserviert, die durch den Aufruf einer entsprechenden Funktion `sctp_freeladdrs()` wieder freigegeben werden müssen.

5. `sctp_sendmsg()`. Der Draft-Standard [88] definiert auch Schnittstellenfunktionen, die zum direkten Senden und Empfang von SCTP-Nachrichten genutzt werden können, ohne den Umweg über die Strukturen, in denen Zusatzinformationen übergeben werden (siehe Funktion `sendmsg()` weiter oben). Diesen Schnittstellenfunktionen werden die entsprechenden Parameter direkt übergeben:

```
ssize_t sctp_sendmsg(int s, const void *msg,
                     size_t len,
                     struct sockaddr *to,
                     socklen_t tolen,
                     uint32_t ppid,
                     uint32_t flags,
                     uint16_t stream_no,
                     uint32_t timetolive,
                     uint32_t context);
```

6. `sctp_recvmsg()`. Mit dieser Funktion können SCTP-Nachrichten empfangen werden. Dabei werden der Struktur `struct sctp_sndrcvinfo` die entsprechenden Werte zugewiesen, wenn der Empfänger das Ereignis `sctp_data_io_events` mit der `SCTP_EVENTS`-Option aktiviert hat.

```
ssize_t sctp_recvmsg(int s, void *msg,
                     size_t *len,
                     struct sockaddr *from,
                     socklen_t *fromlen
                     struct sctp_sndrcvinfo *sinfo,
                     int *msg_flags);
```

7. `sctp_connectx()`. Diese Funktion erlaubt es, beim Verbindungsaufbau zu einem Endpunkt, der unter mehreren IP-Adressen zu erreichen ist, jede seiner Adressen zu benutzen. Die Syntax dieser Funktion ist:

```
int sctp_connectx(int s,
                  struct sockaddr *addrs,
                  int addrcnt);
```

   Je nach Implementierung können nach Aufruf dieser Funktion mehrere INIT-Chunks zu den übergebenen Adressen gesendet werden, und der Verbindungsaufbau über die Adresse, von der die erste Antwort kommt, fortgesetzt werden.

A.3 Eigenschaften der Socket-API Bibliothek

Die Socket-API Bibliothek ist eine in C++ programmierte Anpassungsschicht, die Anwendungen die SCTP-Funktionalität über die in [88] beschriebene Schnittstelle zur Verfügung stellt. Dazu binden Anwendungen eine Include-Datei namens `ext_socket.h` ein, und der Objektcode der Anwendung wird zusammen mit der `sctplib` und der Socket-API Bibliothek zu einem ausführbaren Programm gebunden.
Die Socket-API Bibliothek definiert eine globale Instanz vom Typ `sctpsocketmaster`, die die zentrale Schnittstelle zur `sctplib` bildet. In dieser Klasse wird ein Thread definiert und gestartet, der im Hintergrund einer Anwendung ausgeführt wird und ständig die Ereignis-Schleife der `sctplib` abfragt. Tritt ein für das SCTP relevantes Ereignis ein, so werden definierte Rückruf-Funktionen der `sctpsocketmaster`-Klasse ausgeführt, die die Ereignisse an die registrierten Sockets (die in eigenen Klassen definiert sind) weiterleiten.
In der Datei `ext_socket.h` sind sämtliche in [88] definierten Typen und Schnittstellenfunktionen deklariert und in der Socket-API Bibliothek implementiert, so dass die Kombination mit der SCTPLIB als erste weltweit die komplette Funktionalität sowohl UDP-ähnlicher als auch TCP-ähnlicher Sockets plattformübergreifend zur Verfügung stellt. Stellt ein Betriebssystem nativ SCTP zur Verfügung (vgl. auch Abschnitt 3.4), so erkennt die Socket-API Bibliothek dies bei der Installation und nutzt die nativen Systemfunktionen statt der Funktionalität der `sctplib`.

Aufgrund der Implementierung mit mehreren Threads (multi-threaded Applikation) sind Anwendungen, die die Socket-API Bibliothek nutzen, auf eine Untermenge aller Systemfunktionen beschränkt, die Betriebssysteme zur Verfügung stellen. Insbesondere kann die Funktion `fork()` nicht benutzt werden, die häufig in netzwerkbasierten Serveranwendungen zum Einsatz kommt. Diese Funktion erzeugt einen neuen (Kind-)Prozess, der identisch zum aufrufenden (Vater-)Prozess ist, sich von diesem aber durch die Prozess-ID unterscheidet. Durch diese Einschränkung können nicht alle Server-Programme einfach portiert werden, und ohne größere Modifikationen die Socket-API Bibliothek nutzen.
Jedoch können solche Programme multi-threaded programmiert werden, und statt der Erzeugung eines neuen Prozesses bei Aufbau einer neuen Verbindung durch die `fork()`-Funktion kann ein neuer Thread erzeugt werden, der die netzwerkbasierte Ein- und Ausgabe durchführt und gegebenenfalls die Verbindung wieder terminiert.

Lebenslauf

Name:	Andreas Jungmaier
12.06.1972	geboren in Oberhausen (Rheinland)
06/1992	Abitur am Heinrich Heine-Gymnasium, Oberhausen
10/1992 - 10/1997	Studium an der Gerhard-Mercator Universität, Duisburg in der Fachrichtung Elektrotechnik (Informationstechnik, Technische Informatik)
01/1998 - 02/1999	Wissenschaftliche Hilfskraft an der Gerhard-Mercator Universität, Duisburg Lehrstuhl Prof. Dr.-Ing. Geisselhardt, Institut für Datenverarbeitung
03/1999 - 08/2004	Wissenschaftlicher Mitarbeiter an der Universität Duisburg-Essen, Campus Essen Lehrstuhl Technik der Rechnernetze (Prof. Dr.-Ing. Rathgeb), Institut für Informatik und Wirtschaftsinformatik Institut für Experimentelle Mathematik
seit 09/2004	Mitarbeiter der Vodafone D2 GmbH Abteilung TNC (Network Technology and Concepts)

Selbständigkeitserklärung

Hiermit erkläre ich, die vorliegende Arbeit selbständig ohne fremde Hilfe verfasst zu haben und nur die angegebene Literatur und Hilfsmittel verwendet zu haben.

Andreas Jungmaier
8. Juni 2005