Domänenübergreifende Anwendungskommunikation im IP-basierten Fahrzeugbordnetz

Kay Weckemann

Dissertation
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig–Maximilians–Universität München

vorgelegt von
Kay Weckemann

eingereicht am 01. April 2014

Domänenübergreifende Anwendungskommunikation im IP-basierten Fahrzeugbordnetz

Kay Weckemann

Dissertation
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig–Maximilians–Universität München

vorgelegt von
Kay Weckemann

1. Berichterstatter: Prof. Dr. Claudia Linnhoff-Popien
2. Berichterstatter: Prof. Dr. Jörg Hähner
Tag der Einreichung: 01. April 2014
Tag der Disputation: 01. August 2014

Eidesstattliche Versicherung

(siehe Promotionsordnung vom 12.07.11, § 8, Abs. 2 Pkt. 5)

Hiermit erkläre ich an Eidesstatt, dass die Dissertation von mir selbstständig, ohne unerlaubte Beihilfe angefertigt ist.

Kay Weckemann

Danksagung

Diese Dissertation ist unter der Betreuung von Frau Professor Dr. Claudia Linnhoff-Popien entstanden. In zahlreichen Lehrstuhl-Workshops begutachtet, ist die Arbeit unter dieser Obhut gereift, vielen Dank für diese Unterstützung. Die Zeit am Lehrstuhl – als Arbeitszeit, bei Diskussionen mit den anderen Doktoranden und zur Seminarbetreuung – war stets eine wertvolle Ergänzung zum Umfeld in der Automobilindustrie. Ebenso möchte ich mich herzlich bei meinem Zweitgutachter Herr Professor Dr. Jörg Hähner bedanken, besonders für ein außergewöhnlich langes, erstes Gespräch über meine Arbeit.

Dr.-Ing. Daniel Herrscher hat mich während der gesamten Promotionszeit begleitet. Meist beratend, immer offen für Herausforderungen und Diskussionen, manchmal anstachelnd, wenn die nächste Deadline nahte, oft beruhigend, wenn die unweigerlichen Zweifel kamen. Vielen Dank für deine fortwährende Unterstützung bei der Promotion, im Arbeitsalltag und oft auch einfach im Alltag.

Hyung-Taek Lim, Lothar Stolz, Alexandre Bouard und Holger Endt möchte ich dafür danken, dass sie zur selben Zeit und mit ähnlichen Aufgaben dieselben Leiden durchlebt haben. Zu viert ertragenes Leid ist gevierteiltes Leid. Die vielen Diskussionen und die gegenseitige Unterstützung vor der nächsten Deadline haben mich gestützt und bewegt. Dr. Eckhard Widenmann möchte ich dafür danken, dass er mich zum letzten Teil der Arbeit motiviert hat, der mir am schwersten gefallen ist. Danke für das Aufrütteln, die Schreibmotivation und die gemeinsamen Abende in der Bibliothek.

Die BMW Forschung und Technik GmbH hat mein Promotionsvorhaben in engem Fahrzeugbezug unterstützt. Ich habe dieses Umfeld immer offen und neugierig erlebt. Auch dem Bundesministerium für Bildung und Forschung möchte ich für die Förderung im Rahmen des Projekts *Sicherheit in eingebetteten IP-basierten Systemen (SEIS)* danken und gleichermaßen den SEIS Partnern für die gute Zusammenarbeit.

Zuletzt, aber am aller wichtigsten, möchte ich meinen Eltern danken. Ohne eure bedingungslose Unterstützung hätte ich diesen Berg wohl nie erklommen. Vielen Dank für Alles.

Kay Weckemann

Zusammenfassung

In heutigen Premiumfahrzeugen kommunizieren bis zu 80 Steuergeräte über bis zu sechs verschiedene Vernetzungstechnologien. Dabei öffnet sich die Fahrzeugkommunikation nach außen: Das Fahrzeug kommuniziert auch mit dem Smartphone des Fahrers und dem Internet. Für die Kommunikation über verschiedene Anwendungsdomänen im Fahrzeug müssen heute Gateways eingesetzt werden, die zwischen den nicht-kompatiblen Protokollen übersetzen. Deswegen geht der Trend auch in der Fahrzeugkommunikation zum Internet Protocol (IP), das für technologie- und domänenübergreifende Kommunikation entwickelt wurde. Neben dem durchgängigen Protokoll auf der Vermittlungsschicht ist für die effiziente Entwicklung eines komplexen, verteilten Systems wie einem Fahrzeug auch eine entsprechende Kommunikationsmiddleware notwendig.

Die Kommunikation in einem Fahrzeug stellt spezielle Anforderungen an die Kommunikationsmiddleware. Zum einen werden in Fahrzeugen unterschiedliche Kommunikationsparadigmen genutzt, beispielsweise signalbasierte und funktionsbasierte Kommunikation. Zum anderen können sich die Kommunikationspartner in einem Fahrzeug hinsichtlich ihrer Ressourcen und ihrer Komplexität erheblich unterscheiden. Keine existierende IP-basierte Kommunikationsmiddleware erfüllt die in der vorliegenden Arbeit identifizierten Anforderungen für den Einsatz im Fahrzeug.

Ziel dieser Arbeit ist es daher, eine Kommunikationsmiddleware zu konzipieren, die für den Einsatz im Fahrzeug geeignet ist. Die vorgestellte Lösung sieht mehrere interoperable Ausprägungen der Middleware vor, die den Konflikt zwischen unterschiedlichen funktionalen Anforderungen einerseits und den sehr heterogenen Kommunikationspartnern andererseits auflösen.

Ein weiterer elementarer Teil der Lösung ist die Umsetzung der im Fahrzeug erforderlichen Kommunikationsparadigmen. Das funktionsbasierte Paradigma wird durch einfache Remote Procedure Calls implementiert. Das signalbasierte Paradigma wird durch ein darauf aufbauendes Notification-Konzept implementiert. Somit wird eine stärker am aktuellen Informationsbedarf orientierte Umsetzung ermöglicht, als dies im heutigen Fahrzeugbordnetz durch das einfache Verteilen von Daten der Fall ist. Es wird gezeigt, dass sich prinzipiell beide Kommunikationsparadigmen durch einen einzigen Mechanismus abbilden lassen, der abhängig von den beteiligten Ausprägungen mit

dynamischen oder nur statischen Daten operiert. Ein skalierbares Marshalling berücksichtigt darüber hinaus die unterschiedlichen Anforderungen der Anwendungen und die unterschiedliche Leistungsfähigkeit der beteiligten Steuergeräte. Hiermit wird die Kommunikation zwischen allen Anwendungen im IP-basierten Fahrzeugbordnetz durchgängig ermöglicht.

Auf dieser Basis wird die Lösung um wichtige Systemdienste erweitert. Diese Dienste implementieren Funktionen, die nur in der Kooperation mehrerer Komponenten erbracht werden können oder kapseln allgemeine Kommunikationsfunktionalität zur einfachen Wiederverwendung. Zwei für die Anwendung im Fahrzeug wichtige Systemdienste werden prototypisch dargestellt: Ein Service-Management ermöglicht die Verwaltung von Diensten in unterschiedlichen Zuständen, ein Security-Management bildet Security-Ziele auf die bestmögliche Kombination von implementierten Security-Protokollen der beteiligten Kommunikationspartner ab. Diese Systemdienste sind selbst skalierbar und lassen sich damit an das Konzept unterschiedlicher Ausprägungen der Kommunikationsmiddleware anpassen.

Durch Leistungsmessungen an den im Rahmen dieser Arbeit entstandenen Prototypen wird gezeigt, dass die konzipierte Kommunikationsmiddleware für den Einsatz auf eingebetteten Systemen im Fahrzeug geeignet ist. Der Versuchsaufbau orientiert sich an typischen Anwendungsfällen für die Fahrzeugkommunikation und verwendet Automotive-qualifizierte, eingebettete Rechenplattformen. Insbesondere wird nachgewiesen, dass mit dem beschriebenen Konzept auch leistungsschwache Steuergeräte ins System eingebunden werden können. Die IP-basierte Kommunikationsmiddleware ist damit auf allen relevanten Steuergeräten im Fahrzeug durchgängig einsetzbar.

Abstract

In today's premium cars, up to 80 electronic control units communicate over up to six networking technologies. Additionally, vehicle communication opens to off-board: the car connects to the driver's smartphone and the Internet. The communication between different application domains within the vehicle builds on additional hardware components as application layer gateways to translate between the incompatible protocols. Thus, also for in-car communication, the trend goes towards networking over the Internet Protocol (IP) that has been developed for being independent of technologies and application domains. Besides the universal protocol at the network layer, an efficient development of a complex distributed system requires communication middleware.

In-car communication makes special demands on the communication middleware. On the one hand, a variety of communication paradigms are used for in-car communication, such as signal-based and function-based communication. On the other hand, the communication partners differ considerably in terms of computing resources and complexity of the hosted applications. No existing IP-based middleware fulfils the identified requirements for in-car communication.

The objective of this research is to design a middleware that is suitable for IP-based in-car communication. The presented solution provides multiple interoperable specifications of the middleware which resolves the conflict between different functional requirements on the one hand and the very heterogeneous communication partners on the other hand.

Another fundamental part of the solution is the implementation of required communication paradigms. The function-based paradigm is implemented by simple remote procedure calls. The signal-based paradigm is implemented by a notification concept that allows for a more demand-oriented communication compared to today's practice. It is shown, how both communication paradigms can be implemented through a single mechanism that operates on dynamic or static data – depending on the involved middleware specifications. A scalable marshalling considers the different requirements and performance levels of the participating electronic control units. Scalable specifications of the communication middleware enable seamless operations on restricted embedded and more powerful platforms.

On this basis, the solution is enhanced with important system services. Such services implement functionality that can only be provided in cooperation of multiple components or that encapsulate general communication functionality for easy reuse. Two essential services are prototyped: a service management allows the management of services in different operational states. A security management matches security objectives in the best possible combination of implemented security protocols that two given communication partners have in common. These system services are designed to be scalable and can therefore be adapted to the concept of different specifications of the communication middleware.

Performance measurements using the implemented prototypes show that the designed communication middleware is suitable for the application on embedded systems in the vehicle. The experimental set-up is based on typical use cases for in-car communication and uses automotive-qualified, embedded computing platforms. In particular, the set-up practically demonstrates that the concept also incorporates low-performance electronic control units into the system. The IP-based communication middleware enables communication between all applications in the IP-based in-car communication system.

Inhaltsverzeichnis

1. Einleitung

„Middleware ist die Schicht Software oberhalb des Betriebssystems, die ein Modell oder Paradigma implementiert, so dass ein verteiltes System gegenüber einer Anwendung wie ein einziges, kohärentes System erscheint.“ [Tane 06]

1.1. Verteilte Systeme im Fahrzeug

In heutigen Premiumfahrzeugen kommunizieren bis zu 80 Steuergeräte über bis zu sechs verschiedene Vernetzungstechnologien. Ein Steuergerät (Electronic Control Unit) (ECU) ist ein eingebettetes System, das in der Regel für eine Menge von Anwendungen optimiert entwickelt wurde. Dieses Kommunikationsbordnetz ist historisch gewachsen. Neue Kommunikationsanforderungen haben jeweils zu neuen Vernetzungstechnologien geführt, die speziell für den automotiven Einsatz entwickelt und ergänzend zu den bereits verwendeten Technologien eingesetzt wurden. Das Kommunikationsbordnetz vernetzt eingebettete Systeme, Aktoren und Sensoren, die unter hohem Aufwand optimiert wurden. Spätere Anpassungen sind folglich nur unter erheblichen Kosten möglich. So ist über die letzten 20 Jahre ein heterogenes Gesamtsystem entstanden.

Dies führt zu mehreren Herausforderungen für die Evolution des bestehenden Fahrzeugbordnetzes in einer Umgebung, die stets ein Mehr an Kommunikation unter anspruchsvolleren Rahmenbedingungen, wie beispielsweise höherer Bandbreite, fordert. Das bestehende physikalische, heterogene Rechnernetz lässt sich nicht trivial auf eine moderne Kommunikationsarchitektur migrieren. Die etablierten Vernetzungstechnologien unterstützen keine Schichtentrennung, wie sie in anderen Bereichen der Rechnernetzgestaltung üblich ist. Änderungen an den Vernetzungstechnologien fordern so stets Änderungen an Anwendungsschnittstellen. Die existierenden Vernetzungstechnologien bieten genauer ausgedrückt keine untereinander kompatiblen Protokolle. Damit entsteht eine Abhängigkeit einer Anwendung zu einer bestimmten Vernetzungstechnologie, die unter anderem erschwert, dass eine Anwendung bei sich ändernden Anforderungen an die Kommunikationsschnittstelle auf eine andere verfügbare Vernetzungstechnologie umgezogen werden kann.

Diese Inkompatibilität aus Anwendungssicht führt zu einer weiteren Herausforderung. Das Kommunikationsbordnetz im Fahrzeug ist aus Anwendungssicht logisch in verschiedene Domänen unterteilt. Dies erlaubt die isolierte

Behandlung der Kommunikation zwischen Anwendungen, die ähnliche Aufgaben erbringen. Beispielsweise herrschen im Bereich der Motorsteuerung andere Anforderungen und Rahmenbedingungen als für Komfortfunktionen in der Fahrgastzelle. Diese logische Aufteilung geht praktisch mit der Zuordnung von Domänen auf bestimmte Vernetzungstechnologien einher, was historisch und organisatorisch bedingt ist.

Die Prämisse der isolierten Behandlung von Anwendungen einer Domäne ist in modernen Fahrzeugen allerdings nicht länger uneingeschränkt gegeben. Innovative Komfortfunktionen nutzen beispielsweise heute Informationen der Motorsteuerung, um eine Adaption an das momentane Fahrverhalten zu bieten. Durch den steigenden Vernetzungsgrad zwischen Funktionen nimmt insbesondere die Kommunikation zwischen unterschiedlichen Anwendungsdomänen zu. Dies wird als Interdomänenkommunikation bezeichnet. Da die Interdomänenkommunikation in der Praxis meist über verschiedene Vernetzungstechnologien erfolgt, muss die Kommunikation durch eine dritte Komponente vermittelt werden. Diese Komponente wird als Gateway bezeichnet. Das Gateway muss die Protokolle beider beteiligten Vernetzungstechnologien implementieren, die Anwendungsschnittstellen kennen und folglich zwischen den Anwendungen übersetzen.

Dieselben Herausforderungen wurden für das Internet-Ökosystem besser gelöst. Das Internet ist als Zusammenschluss von Rechnernetzen in der Anzahl von Knoten und Verbindungen nicht mit einem Fahrzeugbordnetz vergleichbar. Es weist aber Eigenschaften bezüglich der Heterogenität von Vernetzungstechnologien und Anwendungen auf, die prinzipiell den Gegebenheiten im Fahrzeug ähneln. Die Vernetzungstechnologien, sowohl für den Zugang eines Nutzers, als auch für die Infrastruktur eines Rechnernetzes, werden beständig durch neuere, leistungsfähigere Technologien ersetzt, die bessere Übertragungseigenschaften insbesondere hinsichtlich der verfügbaren Bandbreite aufweisen. Außerdem werden im Allgemeinen für die Ende-zu-Ende-Übertragung unterschiedliche Vernetzungstechnologien zwischen je zwei Knoten auf der Übertragungsstrecke genutzt, was zwar die Effizienz der Kommunikation, nicht aber die Schnittstellen ändert. Auf Basis etablierter Standards wächst die Anzahl der über das Internet möglichen Anwendungen stark. Für die Entwicklung einer Anwendung ist kein Wissen über die zugrundeliegende Infrastruktur notwendig. Wesentliche Vernetzungsdienste wie Adressierung und Dienstsuche sind standardisiert und etabliert. Das Internet ist ein Ökosystem, das eine einfache Grundlage für innovative neue Anwendungen bietet.

Dieser Vergleich führt zur Fragestellung, ob relevante Prinzipien des Internets auf das Kommunikationsbordnetz im Fahrzeug angewendet werden können. Darauf aufbauend wird untersucht, inwieweit sich die etablierten Protokolle des Internets in diese Überlegungen einbeziehen lassen. Das Internet Protocol (IP) ist ein erprobter Standard mit sehr hohem Reifegrad in zahlreichen Implementierungen für unterschiedlichste Plattformen. Darauf aufsetzend gibt es viele Anwendungsprotokolle, die auch für die Nutzung im Fahrzeug

geeignet erscheinen. Dies geht einher mit aktuellen Trends der Fahrzeugkommunikation, die sich salopp als eine Öffnung der Fahrzeugkommunikation nach außen beschreiben lassen. Ein Consumer Electronics-Gerät (CE-Gerät) wie ein Smartphone oder ein Tablet wird mit dem Fahrzeug über IEEE802.11 WLAN [IEEE 12] verbunden, um die dort installierten Anwendungen und Daten auch während der Fahrt nutzen zu können. Das Fahrzeug wird – über für den mobilen Empfang während der Fahrt optimierte Antennensysteme – den Reisenden eine *Always-Online* Internet-Konnektivität über LTE [Dahl 11] bieten. Car2X-Kommunikation, also sowohl Kommunikation zwischen Fahrzeugen untereinander, als auch mit der nahen Verkehrsinfrastruktur, wird über IEEE802.11p [IEEE 10c] oder LTE den Straßenverkehr sicherer und effizienter machen. Mindestens an einem Kommunikationsendpunkt des Fahrzeugs für die Kommunikation nach außen findet eine durchgängige IP-Kommunikation statt. Das Fahrzeug wird zum IP-Knoten [Herr 11].

Elmar Frickenstein, BMW Group: „Das IP-basierte Kommunikationsbordnetz kommt!“ [Fric 11]

Das Internet Protocol wurde erfunden, um heterogene Vernetzungstechnologien zu koppeln. Es kann dazu genutzt werden, die gewachsene Infrastruktur im Fahrzeug zu vereinfachen. Das Gateway im Fahrzeug, das die zunehmende Kommunikation zwischen Anwendungen an verschiedenen Vernetzungstechnologien übersetzt, wird zu einem Flaschenhals der Kommunikation und erschwert damit neue Innovationen. Mit der Einführung von IP als einheitliches Vermittlungsprotokoll für die Interdomänenkommunikation für innovative Anwendungsfälle im Fahrzeug kann die bestehende Komplexität reduziert werden [Glas 10].

Wenn Anwendungsentwickler direkt auf Betriebssystemprimitive zur Kommunikation im Rechnernetz zurückgreifen, ist dies zeitaufwendig und fehleranfällig. Für ein kohärentes, verteiltes System bietet Middleware eine Abstraktionsschicht, die ein vereinfachtes Modell der Kommunikation implementiert [Tane 06].

Die existierenden Vernetzungstechnologien im Fahrzeug unterscheiden sich in diesem Punkt stark. Das Spektrum reicht von der festen Codierung von Signalen bis hin zur Middleware, die ein Modell von Funktionsbausteinen und zugehörigen Kommunikationsprimitiven umsetzt. Die Lösungen haben gemeinsam, dass sie jeweils speziell an ihre Vernetzungstechnologie angepasst sind.

Ein zukünftiges IP-basiertes Fahrzeugbordnetz benötigt auch einen definierten Umfang für die Anwendungsinteraktion und muss dabei mindestens die entsprechende Funktionalität der etablierten Vernetzungstechnologien bereitstellen. Dieser Funktionsumfang muss zudem flexibel und zukunftssicher für kommende Anwendungen sein. Dabei müssen einige Herausforderungen der Kommunikation im Fahrzeug berücksichtigt werden. Die interne Kommuni-

kation im Fahrzeug ist von eingebetteten, optimierten Systemen mit strengen Einschränkungen bezüglich der verfügbaren Ressourcen gekennzeichnet. Hieraus ergeben sich die beiden wesentlichen Richtungen für die wissenschaftliche Untersuchung: Zum einen müssen die Anforderungen für eine Kommunikationsabstraktion im Fahrzeug auf Basis des Internet Protocols detailliert herausgestellt werden, zum anderen müssen die heutigen Gegebenheiten im gewachsenen, optimierten Kommunikationsbordnetz berücksichtigt werden.

Die vorliegende Arbeit ist ein praktischer Schritt, zwei unterschiedliche Kommunikationswelten zusammenzuführen. Dafür evaluiert sie mögliche Konzepte, Technologien und Lösungen. Die Relevanz der zugrundeliegenden wissenschaftlichen Fragestellungen wurde dennoch bereits diskutiert, wie der folgende Abschnitt kurz zusammenfasst.

1.2. Herausforderungen des Automotive Software Engineerings

Bereits 2007 stellten Pretschner und Broy in „Software Engineering for Automotive Systems: A Roadmap“ [Pret 07] die enorme Zunahme von Software im Fahrzeug fest und leiteten daraus die Folgen für deren Verteilung und Kommunikation innerhalb des Fahrzeugs sowie daraus resultierende Forschungsfragen ab. Sie sagten voraus, dass der Umfang der in einem Fahrzeug installierten Software 2010 die 1 GB Marke überschreiten wird, reine Datensammlungen wie Kartenmaterial für das Navigationssystem ausgenommen. Aus dieser Arbeit lassen sich einige Schlüsse für Kommunikationssoftware im Fahrzeug ziehen.

Die Heterogenität von Software im Fahrzeug – in eingebetteten Systemen und Infotainment-Systemen – nimmt stark zu und resultiert vor allem in einer entsprechend komplexer werdenden Systemintegration. Software für das Fahrzeug ist sehr vielfältig und reicht von Unterhaltungs- und Büroanwendungs-Software bis hin zu sicherheitskritischer Echtzeitsteuerung.

Eine Einteilung kann laut Pretschner und Broy [Pret 07] gemäß den Anwendungsdomänen und den damit verbundenen nicht-funktionalen Anforderungen erfolgen: *Infotainment*, *Telematik* und die *Mensch-Maschine-Schnittstelle (Human Machine Interface) (HMI)* erfordern weiche Echtzeit, dominiert vom Verteilen diskreter Ereignisse und Datenströme. Als Besonderheit müssen hierbei auch sogenannte Off-Board-Schnittstellen gepflegt werden, sei es für Datenaktualisierung oder das Einbinden von Endgeräten des Nutzers. In der *Body/Comfort*-Domäne wird ebenfalls weiche Echtzeit gefordert. Der Großteil der Kommunikation besteht hier im Verteilen der Informationen zu diskreten Ereignissen an andere Steuergeräte. Für die *Fahrerassistenz* steht die verlässliche Übertragung an erster Stelle. Hier wird harte Echtzeit und eine sichere Semantik für die Kommunikation verlangt. Für die Domänen *Power Train* und *Chassis* wird ebenfalls harte Echtzeit gefordert. Kommunikation zur Steuerung dominiert gegenüber dem Verteilen von diskreten Ereignissen.

Die Verfügbarkeit der Kommunikation ist die wichtigste Forderung.

Für Software-Updates (das sogenannte Flashen einer ECU) und Diagnose vermischen sich die genannten Anforderungen, weil die Kommunikation in die anderen Domänen spielt. Folglich bestehen die verschiedenen SW/HW-Systeme aus einer hohen Anzahl von getrennten Funktionen und Prozessen, die Informationen über diverse Kommunikationsverbindungen austauschen und dabei unterschiedliche Eigenschaften fordern sowie unterschiedlichen Paradigmen folgen. So zeigen diese relativ kleinen, verteilten Rechnernetze eine hohe Komplexität.

Neben der Heterogenität arbeiten Pretschner und Broy auch eine weitere Spezialität der Softwareentwicklung für Fahrzeuge heraus: Entwicklungsprozesse laufen entlang einer hierarchischen Wertschöpfungskette von Zulieferern bis hin zum Hersteller und damit Systemintegrator des Fahrzeugs. Diese Arbeitsteilung verschärft die Aufteilung von Funktionen auf mehrere ECUs noch darüber hinaus, was logisch für die Partitionierung des Systems zweckmäßig erscheint.

Spätestens solch verteilte Entwicklungsprozesse fordern klare Schnittstellen, die nur über eine adäquate Kommunikationsinfrastruktur erreicht werden können. Dies beinhaltet standardisierte Betriebssysteme und eine Kommunikationsmiddleware, die Eigenschaften wie Verlässlichkeit, Robustheit und weitere spezifische Anforderungen einheitlich abbilden. Daraus leiten die Autoren auch eine ihrer 13 konkreten Forschungsthemen ab. „Middleware at different levels of abstraction that enables the communication between heterogeneous subsystems. [Pret 07]“. Die zukünftige Kommunikationsinfrastruktur im Fahrzeug benötigt Middleware in unterschiedlichen Abstraktionsstufen, die Kommunikation zwischen heterogenen Subsystemen ermöglicht.

Bereits hieraus lässt sich die grundlegende Themenstellung der vorliegenden Ausarbeitung ableiten: Domänenübergreifende Anwendungskommunikation im IP-basierten Fahrzeugbordnetz kann durch geeignete Kommunikationsmiddleware erreicht werden. Middleware und Kommunikationsdienste stellen dabei einen klassischen *Separation of Concerns*-Ansatz dar. Die Anwendungslogik wird von der darunter liegenden Kommunikationsinfrastruktur getrennt. Alle Anwendungen reagieren gleichermaßen auf Kommunikation.

1.3. Beitrag und Struktur der Dissertation

Diese Arbeit behandelt Systemsoftware, welche die Kommunikation in einem IP-basierten Fahrzeugbordnetz vereinfacht. Das Konzept einer skalierbaren Middleware ermöglicht Kommunikationsabstraktion im Fahrzeug auf Basis des Internet Protocols.

Dies berücksichtigt sowohl die Migration heutiger, als auch Anforderungen zukünftiger Anwendungen. Ziel ist die Beschreibung und Validierung eines geeigneten Konzepts, durch das sich das IP-basierte Fahrzeugbordnetz als kohärentes, verteiltes System gestalten lässt. Dieses Konzept wird daran

gemessen, innerhalb der eingeschränkten Möglichkeiten in einem gewachsenen, optimierten Kommunikationsbordnetz umsetzbar zu sein. So entsteht schrittweise das Wissen um alle notwendigen Bausteine, die zusammengenommen eine domänenübergreifende Anwendungskommunikation im IP-basierten Fahrzeugbordnetz ermöglichen.

In Kapitel 2 wird ausführlich in die Thematik *Kommunikationsmiddleware* eingeführt. Eine Klassifizierung erlaubt eine klare Definition dieses weitläufigen Begriffs für die vorliegende Arbeit.

In Kapitel 3 wird die verwandte Literatur über Kommunikationsnetze im Fahrzeug und im Internet diskutiert. Dieser Vergleich zeigt, wie die Architekten eines zukünftigen Fahrzeugbordnetzes vom Internet-Ökosystem lernen können. Damit werden die für die Fahrzeugkommunikation adaptierbaren Eigenschaften aufgezeigt. Das Kapitel bietet außerdem eine Übersicht über existierende Kommunikationsmiddleware-Lösungen.

Ausgehend von heutigen Anwendungen, die in ein zukünftiges Fahrzeugbordnetz übernommen werden, und potentiellen neuen Anwendungen, werden in Kapitel 4 die Anforderungen erfasst, welche die Kommunikationsmiddleware im Fahrzeug erfüllen muss. Diese werden schrittweise untersucht und verfeinert. In diesem Zusammenhang wird auch beschrieben, wie sich das heutige Fahrzeugbordnetz in ein IP-basiertes migrieren lässt. Aus der Analyse wird gefolgert, welche Eigenschaften die Middleware aufweisen muss, um eine Migration des Fahrzeugbordnetzes aus Anwendungssicht vollumfänglich zu erlauben. Diese Anforderungen bilden die Basis für den Entwurf und die Validierung des eigenen Konzepts.

In Kapitel 5 werden das Konzept und die Architektur einer skalierbaren, automotiven Kommunikationsmiddleware detailliert beschrieben. Das Konzept der binärkompatiblen Interoperabilität ermöglicht den Einsatz einer einzigen Middleware für die Kommunikation im IP-basierten Fahrzeugbordnetz, indem es die Heterogenität der Steuergeräte auf einfache Weise abstrahiert. Der Entwurf interoperabler Ausprägungen zeigt, welche funktionalen Umfänge die Middleware haben muss und wie Interoperabilität zwischen den Ausprägungen hergestellt wird. Ein besonderes Augenmerk liegt dabei auf der kleinen Ausprägung, da leistungsschwache Steuergeräte eine kritische Betrachtung hinsichtlich Ausführungsverhalten und Ressourcenbedarf erforderlich machen. Implementierungen und Messungen validieren das vorgestellte Konzept aus interoperablen Ausprägungen der Kommunikationsmiddleware.

In Kapitel 6 werden Systemdienste für ein Service-, Teilnetzbetrieb-, Dienstgüte- und Security-Management beschrieben und wie jene das Konzept der interoperablen Ausprägungen erweitern. Dazu wird die Architektur der

Dienste anhand von Untersuchungen und Implementierungen beschrieben.

In Kapitel 7 werden prototypische Anwendungen beschrieben, um abschließend eine praktische Demonstration des Konzepts zu geben. Die entworfene Kommunikationsmiddleware dient als wissenschaftlich fundierte Basis für die Entwicklung und Standardisierung einer zukünftigen Kommunikationsmiddleware-Lösung für den Einsatz im Fahrzeug.

Die Arbeit wird in Kapitel 8 mit einer Zusammenfassung des wissenschaftlichen Beitrags und der wichtigsten Ergebnisse sowie offenen Fragestellungen abgeschlossen.

Teile dieser Dissertation wurden bereits in [Grot 10], [Weck 10], [Lim 11c], [Weck 11], [Endt 12], [Weck 12] und [Stol 12b] veröffentlicht. Diese Arbeiten wurden teilweise im Rahmen des Forschungsprojekts SEIS vom Bundesministerium für Bildung und Forschung (BMBF) unter den Kennzeichen 01BV0900-01BV0917 gefördert.

2. Begriffsklärungen zu Kommunikationssoftware

Dieses Kapitel beginnt mit einer kurzen Beschreibung des OSI-Modells, da sowohl bei der Einführung des heutigen Kommunikationsbordnetzes im Fahrzeug, als auch in der Hinführung zur vorgeschlagenen Systemsoftware, stets auf dieses Schichtenmodell der Kommunikation Bezug genommen wird. Dieses Kapitel führt detailliert in die Thematik Kommunikationsmiddleware ein, um diesen in der Literatur sehr weit gefassten Begriff verbindlich für die vorliegende Arbeit zu definieren. Dafür werden sowohl Kommunikationsmiddleware im Allgemeinen als auch einzelne Bestandteile im Speziellen behandelt.

2.1. Schichtenmodelle der Kommunikation

Das ISO/OSI-Schichtenmodell [Inte 94] (im Folgenden kurz OSI-Modell) formalisiert die Idee einer Organisation von aufeinander aufbauenden Schichten für die Kommunikation in einem verteilten System. Jede Schicht bietet an einer definierten Schnittstelle die Erbringung der vorgesehenen Funktionalität [Tane 06]. Diese Modularisierung reduziert die Komplexität und ermöglicht die Austauschbarkeit von Implementierungen auf Schichtebene. Während die tatsächliche Kommunikation zwischen benachbarten Schichten verläuft, findet die logische Kommunikation mit einem entfernten Kommunikationspartner – einem sogenannten *Peer* – auf derselben Schicht statt. Die Vereinbarung über den Ablauf, wie auf einer Schicht zwischen Peers kommuniziert wird, wird als Protokoll bezeichnet. Der physikalische Datentransfer findet erst auf der untersten Schicht über ein physikalisches Medium statt. Eine Liste von aufeinander aufbauenden Protokollen wird als Protokoll-Stack bezeichnet. Das OSI-Modell ist ein theoretisches Referenzmodell, das von der International Standards Organization (ISO) vorgeschlagen wurde. Es besteht aus sieben Schichten. Die ursprünglich entwickelten Protokolle haben sich allerdings nicht durchgesetzt. Das Modell stellt aber die Merkmale, die auf den Schichten erbracht werden, in einer allgemeinen Form dar. Daher eignet es sich dazu, das IP-basierte Fahrzeugbordnetz und eine IP-basierte Kommunikationsmiddleware einzuordnen.

Abbildung 2.1 zeigt die sieben Schichten des OSI-Modells. Die unteren beiden Schichten sind die Bitübertragungs- und die Sicherungsschicht. Die Bitübertragungsschicht definiert, wie die Datenübertragung auf dem Übertragungsmedium physikalisch stattfindet. Die Sicherungsschicht spezifiziert die

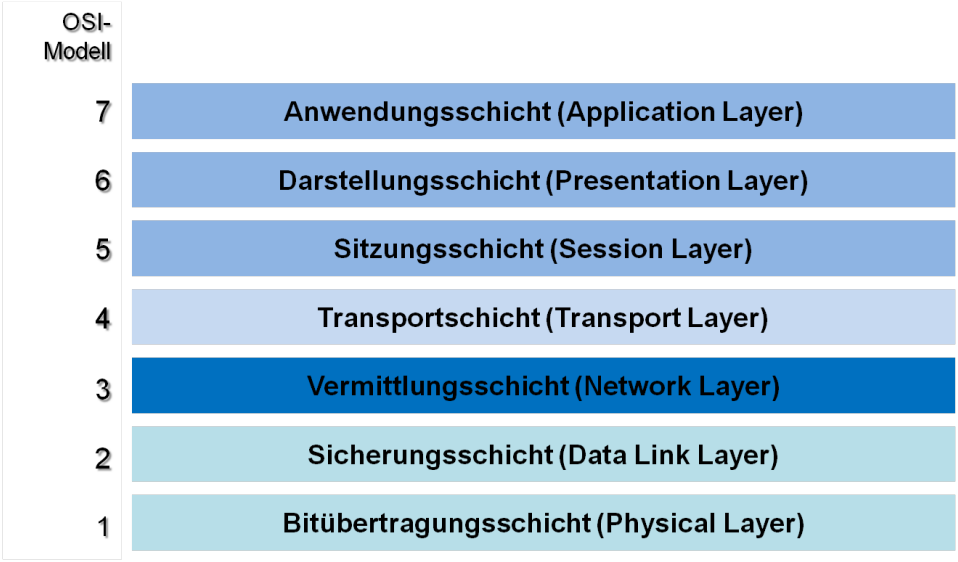

Abbildung 2.1.: Die sieben Schichten des OSI-Modells.

Adressierung, wie die Datenübertragung zwischen zwei direkten Kommunikationsknoten geprüft wird und weitere Aspekte, die ein einzelnes Rechnernetz betreffen. Wie viele Vernetzungstechnologien spezifizieren die meisten speziellen Fahrzeugvernetzungstechnologien nur diese beiden Schichten.

Die Vermittlungsschicht dient der Vermittlung von Daten zwischen unterschiedlichen Rechnernetzen. Protokolle müssen demnach rechnernetzübergreifende Adressen definieren. Vermittelte Daten gelangen insbesondere nicht an höhere Schichten. Sie werden stattdessen über Zwischenziele *geroutet*. Die vierte Schicht des OSI-Modells beschreibt Transportprotokolle, welche die transparente Datenübertragung zwischen zwei entfernten Anwendungen ermöglichen. Dies kann neben der Adressierung einer Anwendung auch eine Flusskontrolle und Fehlererkennung beinhalten. Mit Flusskontrolle bezeichnet man den Prozess zur Steuerung der Daten-Übertragungsrate zwischen zwei Endsystemen, um Datenverlust am Empfänger durch Puffer-Überlauf zu vermeiden.

Die Sitzungsschicht sichert die Prozesskommunikation zwischen zwei Systemen durch einen strukturierten Datenaustausch, während die Darstellungsschicht die systemabhängige Darstellung der Daten in eine unabhängige Form umsetzt. In der Entwicklung des Internets haben sich dafür keine Protokolle etabliert. Die Anforderung an strukturierte und systemunabhängige Kommunikation ist aber im heterogenen Fahrzeugbordnetz eine wichtige Anforderung. Dies wird in Kapitel 4 näher ausgeführt. Damit sind diese beiden Schichten für das Konzept einer Kommunikationsmiddleware relevant. Die Anwendungsschicht stellt schließlich ihrem Namen entsprechend Funktionen für klar umrissene Anwendungsbereiche zur Verfügung.

Daneben hat sich aus der Praxis des Internets das TCP/IP Referenzmodell entwickelt [Tane 06]. Da sich dieses Modell aus der Umsetzung der Protokol-

le für das Internet ergibt und bei deren Entwicklung unzureichend zwischen Spezifikation und Implementierung unterschieden wurde, ist dieses Modell als theoretische Referenz kaum anwendbar. Die Protokolle bieten allerdings das Gerüst des Internets und damit auch für ein IP-basiertes Fahrzeugbordnetz. Die für die vorliegende Arbeit relevanten Protokolle werden im folgenden Kapitel in Abschnitt 3.2 behandelt.

2.2. Einordnung der Kommunikationsmiddleware in das Schichtenmodell

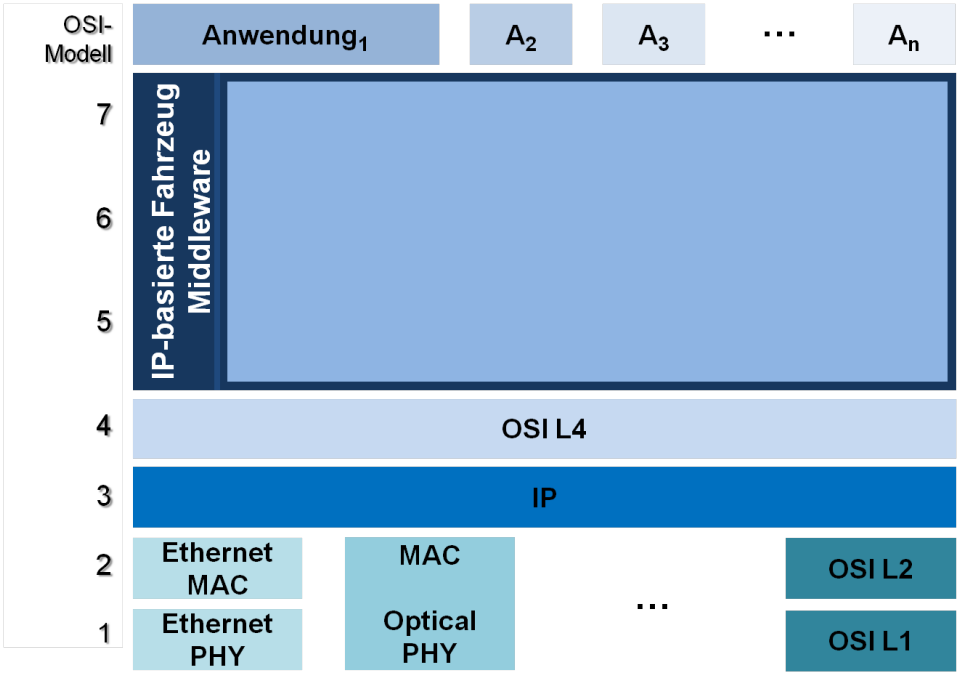

Abbildung 2.2.: Kommunikationsmiddleware abstrahiert die Vernetzungstechnologien von den Anwendungen.

In einem IP-basierten Fahrzeugbordnetz kann die Middleware auf einen implementierten TCP/IP-Stack zurückgreifen. Es kann also vorausgesetzt werden, dass die zugrundeliegenden Vernetzungstechnologien IP-Pakete transportieren können. Außerdem können erprobte Transportschichtprotokolle wie Transmission Control Protocol (TCP) und User Datagram Protocol (UDP) verwendet werden. Gesucht ist also Systemsoftware für die Kommunikation, die oberhalb der Transportschicht abstrakte Kommunikationsaufgaben übernimmt, welche die Anwendungsentwicklung erleichtern. Der Lösungsraum befindet sich demnach im OSI-Schichtenmodell im Bereich von Sitzungs- und

Darstellungsschicht. Diese sichern die Prozesskommunikation zwischen zwei Systemen durch einen organisierten Datenaustausch bzw. sorgen für die Umsetzung der systemabhängigen Repräsentation der Daten in eine unabhängige Form. Abbildung 2.2 zeigt schematisch, wie die Kommunikationsmiddleware auf das IP-basierte Fahrzeugbordnetz aufsetzt. Dieses schematische Bild wird im Laufe der Arbeit schrittweise verfeinert.

2.3. Allgemeine Funktionalität einer Kommunikationsmiddleware

Bakken definiert in seinem Enzyklopädie-Beitrag „Middleware“ [Bakk 01] selbige als eine Klasse von Software-Technologien, welche die Verwaltung von Komplexität und Heterogenität unterstützen, die inhärent für verteilte Systeme sind. Die Middleware liegt demnach zwischen dem Betriebssystem und den Anwendungsprogrammen. Sie bietet dadurch Abstraktion für die Entwicklung von Programmen im verteilten System. Dafür bietet sie höherschichtige Bausteine als eine reine Programmierschnittstelle, wie sie beispielsweise durch Sockets gegeben sind, die vom Betriebssystem bereitgestellt werden.

Der Begriff Middleware bezeichnet dem Wortursprung nach eine Komponente, die zwischen anderen Komponenten liegt, oftmals so interpretiert, dass Middleware zwischen den anderen Komponenten vermittelt. Dies kann sowohl die Kommunikation in einem verteilten System als auch die Integration mehrerer Komponenten einer einzelnen Anwendung betreffen. Im ersten Fall lassen sich schließlich noch verschiedene Schwerpunkte unterscheiden, abhängig von der angestrebten Abstraktionstiefe. Der Begriff ist damit kontextabhängig für die Anwendungsdomäne, in der die Middleware eingesetzt wird. Dieser Abschnitt definiert Middleware als Systemsoftware zur Kommunikation in einem verteilten Rechnernetz, wie sie in der vorliegenden Arbeit behandelt wird. Daher wird meist synonym der genauere Begriff *Kommunikationsmiddleware* verwendet.

Navet et al. [Nave 05] beschreiben die Motivation für den Einsatz von Kommunikationsmiddleware unter vier Aspekten:

- Abstraktion der Verteilung
- Abstraktion der Heterogenität von Plattformen
- Angebot höherschichtiger Dienste
- Sichern der Dienstgüte

Abstraktion von Verteilung meint, dass die Kommunikation unabhängig von der Anbindung und der Zugehörigkeit der Kommunikationspartner zu einer Anwendungsdomäne erfolgen soll. Abstraktion von Heterogenität bezieht sich auf die Unterschiede zwischen den beteiligten Systemen in Hardware und

in Betriebssystemen. Die Kommunikationsmiddleware muss Unabhängigkeit von Anbindung und Systemunterschieden ermöglichen. Mit höherschichtigen Diensten bezeichnen die Autoren allgemein Aufgaben, die viele Anwendungen zur Erfüllung ihres Kommunikationsbedarfs selbst implementieren müssten. Die Auslagerung dieser Aufgaben in eine Kommunikationsmiddleware senkt dagegen die Entwicklungszeit und erhöht die Qualität durch Wiederverwendung erprobter Funktionen. Das Sichern der Dienstgüte kann durch eine Kommunikationsmiddleware unterstützt werden, falls die Dienstgüte der unteren Kommunikationsprotokolle unzureichend ist.

Kommunikationsmiddleware abstrahiert also die Kommunikation von der verteilten Anwendung. Nach der Definition von Issarny et al. [Issa 07] definiert Middleware folgende Bestandteile:

- Eine Schnittstellenbeschreibungssprache (Interface Definition Language) (IDL).
- Ein Koordinationsmodell, das auf Kommunikationsparadigmen mit festgelegter Semantik basiert.
- Ein Naming- und Dienstvermittlungsprotokoll inklusive Konventionen zum Veröffentlichen und Finden von Ressourcen.
- Ein Transportprotokoll.
- Ein höherschichtiges Adressierungsschema.

Aus der im vorigen Kapitel behandelten Betrachtung des historisch gewachsenen Fahrzeugbordnetzes mit seinen heterogenen Vernetzungstechnologien muss diese Aufzählung noch um einen weiteren Bestandteil ergänzt werden:

- Marshalling-Vorschriften, die eine systemunabhängige, serialisierte Aufbereitung der Anwendungsdaten für die Kommunikation ermöglichen.

Abbildung 2.3 zeigt die prinzipiellen funktionalen Bausteine der Kommunikationsmiddleware, wobei das Transportprotokoll und damit auch ein höherschichtiges Adressierungsschema zwischen Prozessen als grundlegend betrachtet wird. Geeignete Transportprotokolle werden in Abschnitt 3.2 eingeführt und aufgrund ihrer Verfügbarkeit als erprobter Standard als allgemein verfügbar angenommen.

Schnittstellenbeschreibungssprache

Eine Schnittstellenbeschreibungssprache (Interface Definition Language) ist eine deklarative, formale Sprache, die eine Syntax zur Beschreibung von Schnittstellen einer Software-Komponente bereitstellt. Eine IDL ermöglicht in einem verteilten System, Schnittstellen unabhängig von einer konkreten Plattform

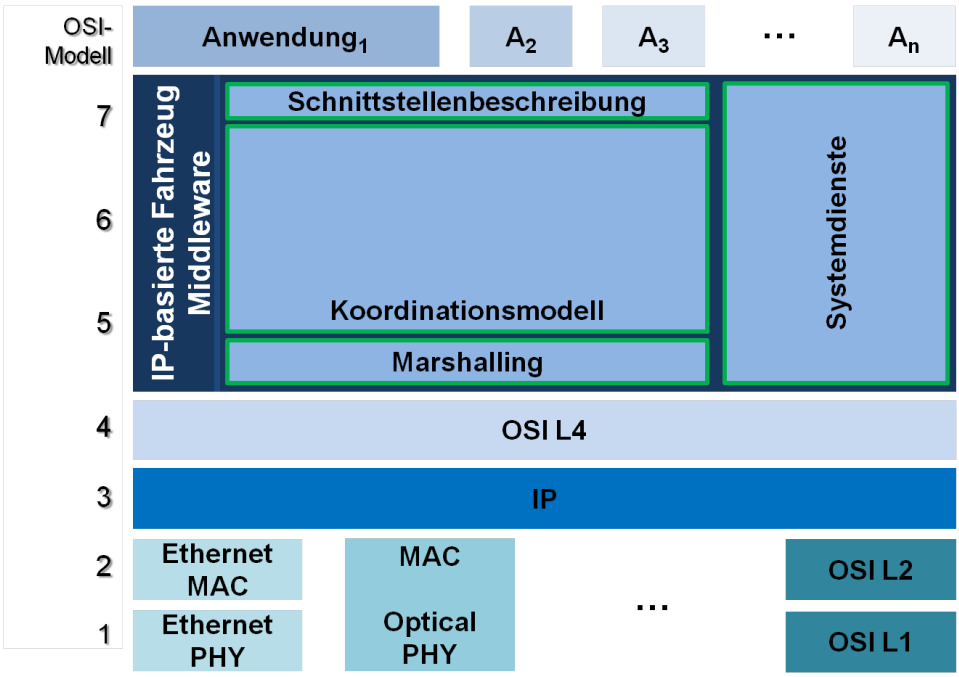

Abbildung 2.3.: Übersicht über die IP-basierte Kommunikationsmiddleware und ihre funktionalen Bestandteile.

und Programmiersprache zu beschreiben, so dass sich ein oder mehrere Kommunikationsparadigmen realisieren lassen. Die IDL dient nicht der Formulierung von Algorithmen. Diese werden erst später für eine bestimmte Programmiersprache und Plattformarchitektur implementiert und mittels Compiler an die Schnittstelle gebunden.

Koordinationsmodell

Das Koordinationsmodell beschreibt das dynamische Verhalten der Middleware für die vermittelte Kommunikation. Es kommt dabei mindestens ein Kommunikationsparadigma zur Anwendung. Diese Kommunikationsparadigmen folgen einer definierten Semantik und stellen somit das eigentliche Protokoll dar, nach dem die Middleware Kommunikation herstellt. Die unterschiedlichen Paradigmen dienen meist zur Unterscheidung verschiedener Arten von Kommunikationsmiddleware, sie werden daher in diesem Zusammenhang im folgenden Abschnitt 2.4 ausführlicher betrachtet.

Marshalling

Marshalling dient einer systemunabhängigen Aufbereitung der Anwendungsdaten für die Kommunikation. Auf einem System liegen Daten gegebenenfalls in strukturierte Form vor, numerische Daten sind anhand einer systemabhängigen

Endianess gespeichert, andere Daten anhand eines vorgegebenen Zeichensatzes. Die Übertragung kann aber nur in einer einheitlichen, serialisierten Form erfolgen. Beim Marshalling der Parameter werden die Hardware-spezifischen Anwendungsdaten in ein verbindliches Format für die Übertragung serialisiert. Der umgekehrte Prozess am Empfänger wird als De-Marshalling bezeichnet. Diese Übersetzung der serialisierten Daten bringt diese in eine vom empfangenden System abhängige Form. Neben der Definition der einheitlichen Serialisierung der Daten für die Kommunikation muss also für jedes an der Kommunikation beteiligte System, die entsprechende Funktionalität für Marshalling und De-Marshalling entwickelt werden.

Systemdienste

Die Frage nach Naming- und Dienstvermittlungsprotokollen stößt in einem Fahrzeugbordnetz auf einige Besonderheiten. Das Thema wird daher als Service-Management behandelt und zunächst unter dem Begriff Systemdienste allgemeiner zusammengefasst. Kapitel 6 befasst sich mit dem Service-Management und weiteren Systemdiensten, die über das Koordinationsmodell hinaus kommunikationsrelevante Aufgaben übernehmen.

All diese Funktionen können unterschiedlich erbracht werden. Eine typische Unterscheidung verschiedener Arten von Kommunikationsmiddleware basiert auf dem verwendeten Koordinationsmodell. Dazu wird eine Taxonomie aus der Literatur eingeführt, die eine Klassifizierung verschiedener Konzepte und eine erste Beurteilung ihrer Eignung für das IP-basierte Fahrzeugbordnetz erlaubt.

2.4. Klassifizierung anhand des Koordinationsmodells

Im Folgenden werden verschiedene Arten von Kommunikationsmiddleware gemäß der Taxonomie von Issarny et al. [Issa 07] klassifiziert. Die Taxonomie kategorisiert Middleware anhand des implementierten Koordinationsmodells. Die Taxonomie basiert auf der von Emmerich [Emme 00] vorgestellten, wo sich die Kriterien finden, die der Identifikation einer adäquaten Middleware für das Fahrzeugbordnetz dienen. Die Taxonomie listet folgende Klassen:

- Transaktionale Middleware
- Tuplespace-basierte Middleware
- Nachrichtenorientierte Middleware
- RPC-Framework
- Objekt- und komponentenorientierte Middleware

- Service-orientierte Middleware

Transaktionale Middleware nutzt ein Protokoll zur Bestätigung in zwei Phasen (two-phase commit protocol) [Bern 87], um verteilte Transaktionen zu implementieren. Damit gewährleistet die Middleware ein hohes Maß an Konsistenz. Jene wird allerdings durch einen enormen Overhead in der Kommunikation und der Speicherung von versendeten Identifikatoren erkauft, die eine eindeutige Identifizierung einzelner Nachrichten erlauben.

Tuplespace-basierte Middleware bietet eine große Abstraktionstiefe, indem sie virtuell einen verteilten gemeinsamen Speicher bietet. Seine Vorteile bietet dieses Kommunikationsparadigma bei der Verteilung von Informationen – gegebenenfalls auch an eine unbestimmte Empfängergruppe. Andererseits ist es aufgrund dieser Abstraktion schwierig, Garantien für die Übertragung zu gewährleisten. Die Virtualisierung gegenüber der Anwendung wird durch eine darunterliegende, feste Zuordnung von Kommunikationspartnern transparent erbracht. Dies erzeugt wiederum Overhead. Außerdem erschwert die Unterscheidung der Schichten die Diagnose hinsichtlich des Systemstatus.

Bei nachrichtenorientierter Middleware basiert die Kommunikation auf einfachem Nachrichtenaustausch. Die Nachricht entspricht einem ereignisbasierten Signal oder einer Anfrage an eine Server-Komponente, allerdings ohne eine höherwertige Aufrufsemantik zu ermöglichen. Das nachrichtenorientierte Kommunikationsparadigma ist asynchron. Falls eine Synchronisation gefordert ist, muss dies in der Client-Anwendung gesichert werden. Meist ist in einem solch einfachen Paradigma auch keine Behandlung heterogener Daten vorgesehen. Falls dies benötigt wird, muss sich der Anwendungsentwickler um das Marshalling kümmern. Damit bietet einfache nachrichtenorientierte Middleware nur geringe funktionale Umfänge.

Ein Remote Procedure Call (RPC) ist ein entfernter Funktionsaufruf und wird in Abschnitt 5.1.1 noch detailliert beschrieben. Prinzipiell unterstützt ein RPC-Framework die Definition von Server-Komponenten als RPC-Programme. Ein RPC-Programm exportiert eine Anzahl parametrisierter Prozeduren und assoziierter Parameter. Client-Komponenten können diese Prozeduren über das Netzwerk aufrufen. Ein RPC-Framework oder auch Procedural Middleware implementiert diese Aufrufe, indem die Parameter per Marshalling in eine Nachricht übersetzt werden, die an eine Server-Adresse gesendet wird. Auf diese Weise lässt sich ein entfernter Aufruf für die Client-Anwendung transparent wie ein lokaler Funktionsaufruf umsetzen.

Objekt- und komponentenorientierte Middleware erweitert das RPC-Framework um objektorientierte Prinzipien wie Objektidentifikation durch Referenzieren und Vererbung. Sie bietet mächtige Komponentenmodelle, welche die Funktionalität von transaktionaler, nachrichtenorientierter und RPC Middleware integrieren. Obwohl Objektorientierung für die Infotainmentdomäne geeignet wäre, ist die damit eingeführte Komplexität zu groß für die meisten ECUs, die prozedurale Programme beherbergen. Jene werden meist noch in einer prozeduralen Programmiersprache wie beispielsweise C entwickelt.

Service-orientierte Middleware ist der nächste Schritt auf dem Evolutionspfad zur komponentenbasierten Programmierung. Hier besteht verteilte Software aus lose gekoppelten Netzdiensten. Ressourcen im Netzwerk sind als autonome Dienste verfügbar, die vollkommen abstrahiert von ihrer zugrundeliegenden Technologie verwendet werden können. Diese unabhängigen Entitäten werden über definierte Schnittstellen auf eine standardisierte Zugriffsart aufgerufen.

Die tatsächlichen Bestandteile einer geeigneten Kommunikationsmiddleware für das IP-basierte Fahrzeugbordnetz werden in den folgenden Kapiteln schrittweise aus Anforderungsanalyse und Evaluation herausgearbeitet.

2.5. Zusammenfassung

Das OSI-Modell bietet den strukturellen Rahmen, um den funktionalen Umfang einer IP-basierten Kommunikationsmiddleware einzuordnen. Kommunikationsmiddleware ist Systemsoftware für die Kommunikation, die oberhalb der Transportschicht generelle Kommunikationsaufgaben übernimmt und somit die Anwendungsentwicklung und -ausführung erleichtert. Kommunikationsmiddleware abstrahiert die Verteilung von Komponenten und die Plattformheterogenität der Kommunikationspartner. Sie besteht erstens aus einer Schnittstellenbeschreibungssprache zur einheitlichen Definition der Kommunikationsschnittstellen zwischen Anwendungen. Ein Koordinationsmodell ermöglicht zweitens die operative Ausführung der Kommunikation durch ein oder mehrere Kommunikationsparadigmen mit festgelegter Semantik. Marshalling sorgt dabei drittens für eine systemunabhängige, serialisierte Aufbereitung der Anwendungsdaten für die Kommunikation. Systemdienste erweitern viertens die Basisfunktionalität um weitere allgemeine Kommunikationsaufgaben.

In der Literatur wird Kommunikationsmiddleware oftmals anhand des Koordinationsmodells klassifiziert. Unter den verschiedenen Paradigmen bieten sich nachrichtenorientierte Middleware, ein Remote Procedure Call-Framework und objektorientierte Middleware für die weitere Untersuchung an. Davor wird aber im folgenden Kapitel zunächst das heutige Fahrzeugbordnetz betrachtet.

3. Kommunikation im Fahrzeug und verwandte Arbeiten

Um die zukünftigen Herausforderungen für die IP-basierte Kommunikation im Fahrzeug darlegen zu können, werden in diesem Kapitel zunächst die heute relevanten Vernetzungstechnologien behandelt. Dabei geht es um die Eigenschaften der Technologien und die Anwendungen, die damit bedient werden. Nicht zuletzt geht es aber auch um Formen der Kommunikationsabstraktion. Außerdem werden zwei für diese Arbeit wesentlichen Unterscheidungsmerkmale des Internets gegenüber dem Fahrzeugbordnetz untersucht: Trennung zwischen Anwendung und Netzzugang sowie standardisierte Protokolle. Diese Betrachtung zeigt, wie Architekten ein IP-basiertes Kommunikationsbordnetz aufbauen können und welches die adaptierbaren Eigenschaften sind. Daraus werden die Grundzüge eines zukünftigen IP-basierten Kommunikationsbordnetzes gezeichnet, auf dessen Basis eine geeignete Kommunikationsmiddleware konzipiert werden kann.

3.1. Das heutige Fahrzeugbordnetz

Das heutige Kommunikationsbordnetz im Fahrzeug ist ein heterogenes Rechnernetz bestehend aus mehreren Controller Area Networks (CANs) [Robe 91] für allgemeine signalbasierte Kommunikation, FlexRay [Flex 05] für Echtzeitkommunikation, Media Oriented Systems Transport (MOST) [Grze 08] für Bandbreiten-intensive Multimedia-Anwendungen und weiteren Vernetzungstechnologien [Nolt 05]. Da diese Technologien nicht kompatibel sind, muss die Kommunikation durch ein Anwendungsschicht-Gateway übersetzt werden.

Ugur Keskin bietet eine gute Literaturübersicht zu Vernetzungstechnologien im Fahrzeug [Kesk 09]. Ausführliche Erklärungen der verschiedenen Technologien finden sich in den Lehrbüchern von Reif „Automobilelektronik“ [Reif 06] und Zimmermann „Bussysteme in der Fahrzeugtechnik“ [Zimm 07].

Das Ziel dieses Abschnitts ist es, in die Grundzüge der Fahrzeugkommunikation einzuführen und einige Herausforderungen herauszustellen, die im Laufe der Arbeit adressiert werden. Die vorgestellten Vernetzungstechnologien sind nicht vollständig. Ausgelassen werden Technologien, die nicht für die Interdomänenkommunikation geeignet sind, die das IP-basierte Fahrzeugbordnetz erst motiviert, wie beispielsweise Local Interconnect Network (LIN). Technologien wie Time-Triggered Protocol Class C (TTP/C) weisen ähnliche Eigenschaften auf wie CAN und werden daher nicht gesondert betrachtet. Da es im Kern der

Untersuchung nicht um die Technologie, sondern um die Anwendungsgebiete und Kommunikationsprinzipien dahinter geht, wird in solchen Fällen nur die verbreitetere Technologie untersucht, im konkreten Beispiel damit CAN.

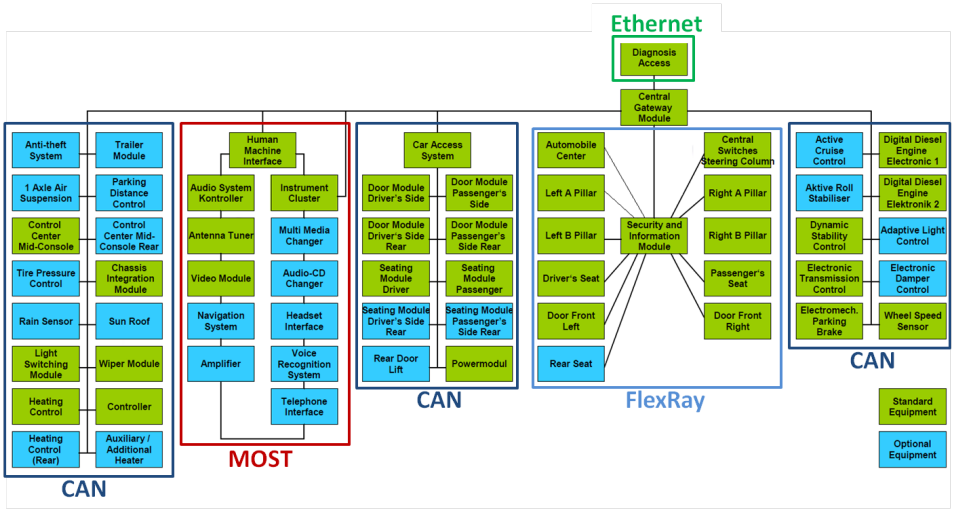

Abbildung 3.1.: Eine vereinfachte Darstellung des heutigen Fahrzeugbordnetzes in Anlehnung an [Frey 07].

Abbildung 3.1 zeigt eine vereinfachte Darstellung eines modernen Kommunikationsbordnetzes im Fahrzeug. Es werden bis zu 80 Steuergeräte in einem Fahrzeug verbaut, die tatsächliche Anzahl in einem konkreten Fahrzeug hängt von der durch den Kunden bestellten Ausstattung ab. Die Steuergeräte werden nach Anwendungsbereichen klassifiziert und damit in unterschiedliche Domänen unterteilt: Infotainment, Telematik, HMI, Body/Comfort, Fahrerassistenz, Power Train und Chassis sind übliche Anwendungsdomänen [Nave 05, Nave 13]. Die Steuergeräte werden gemäß der Anforderungen ihrer Domäne an eine oder mehrere Vernetzungstechnologien angebunden. Diese Bussysteme stellen jeweils ein Subnetz des Kommunikationsbordnetzes dar. Alle Subnetze werden an einem zentralen Gateway angeschlossen. Über dieses Gateway ist eine Kommunikation zwischen den Subnetzen möglich. Die Einteilung der Fahrzeugkommunikation in Domänen erlaubt die isolierte Betrachtung von Kommunikationsanforderungen für eine Klasse ähnlicher Anwendungen. Diese Anforderungen werden meist durch eine passende Vernetzungstechnologie hinreichend erfüllt, wodurch die Einteilung in Anwendungsdomänen sich auch durch den Einsatz getrennter Vernetzungstechnologien manifestiert.

Die heute verwendeten Vernetzungstechnologien haben eines gemeinsam: Sie sind proprietäre Technologien, die aus dem Bedarf einer Anwendung bzw. einer hinlänglich umrissenen Domäne von Anwendungen entwickelt wurden. Navet et al. zeichnen in ihrem Überblick „Trends in Automotive Communication Systems“ [Nave 05] die historische Entwicklung der Vernetzungstechnologien im Fahrzeug nach. Die Autoren beschreiben mit CAN, FlexRay und MOST die

wesentlichen heute verwendeten Technologien im Überblick. Aber auch Ethernet wird bereits als isolierte Lösung für die Übertragung von Massendaten, beispielsweise für das Aktualisieren des Kartenmaterials für die Navigation, beschrieben.

Andreas Schmeiser [Schm 07] beschreibt die heutige Verwendung unterschiedlicher Vernetzungstechnologien als durch Optimierungsanforderungen getrieben. Neue Anwendungen divergieren und fordern daher eine steigende Anzahl spezialisierter Netze oder Bussysteme. „Getrieben werden diese Entwicklungen durch unterschiedliche Anforderungen an Signallaufzeit, Bandbreite, Störsicherheit, Ausfallsicherheit und Topologie, durch anwendungsspezifische Sicherheitsaspekte, die Verfügbarkeit neuer Lösungen, aber auch durch positive Erfahrungen mit bereits eingeführten Systemen sowie dem gestiegenen Kostendruck und der damit einhergehenden möglichst optimalen Dimensionierung entsprechend dem Einsatzzweck. [Schm 07]“ Der Autor spricht in diesem Zusammenhang von einem ungewollten Manifestieren einer heterogenen Architektur.

Die Technologien, die für eine Migration in ein IP-basiertes Fahrzeugbordnetz in Frage kommen, sind CAN, FlexRay, MOST und Ethernet. Diese verbinden die Anwendungen mit hohem Potential für Interdomänenkommunikation. Im Folgenden werden diese Vernetzungstechnologien im Überblick vorgestellt und historisch eingeordnet. Sie werden in Kapitel 4 aufgegriffen, um die Anforderungen abzuleiten, die eine Kommunikationsmiddleware erfüllen muss.

Controller Area Network

Das Controller Area Network (CAN) ist ein standardisierter serieller Bus, der über ein geteiltes Medium kommuniziert. Der Standard wurde in den 1980er Jahren für die digitale Vernetzung von Steuergeräten im Fahrzeug von der Robert Bosch GmbH als robuste Vernetzungstechnologie für eine Umgebung mit potentiell hohem elektromagnetischen Rauschen entwickelt [Robe 91]. Heute wird die Technologie in weiteren industriellen Umgebungen zur Steuerung von eingebetteten Anwendungen mit reinen Steuerungsaufgaben eingesetzt. Im Fahrzeug ist das Hauptanwendungsgebiet die Vernetzung von Sensoren, Aktuatoren und einfachen Steuergeräten.

CAN spezifiziert mit der Bitübertragungsschicht und der Sicherungsschicht nur die beiden untersten Schichten des OSI-Modells (siehe Abschnitt 2.1). Höherschichtige Protokolle müssen bei Bedarf bei der Anwendungsentwicklung festgelegt werden, auch wenn es diverse Vorschläge für die Umsetzung gibt, beispielsweise CANopen. In der Literatur wird CAN oft als Echtzeit-fähige Vernetzungstechnologie bezeichnet. Allerdings beruht die Kommunikation auf dem geteilten Medium auf einfacher Priorisierung anhand von Identifikatoren (CAN-IDs) und ist daher nicht deterministisch. CAN eignet sich besonders für eine ereignisbasierte Übertragung, bei der ein Steuergerät bei definierten Ereignissen beginnt, Daten über den Bus zu senden, und für eine zyklische

Übertragung nach definiertem Zeitintervall. Dabei werden die Daten auf Basis von einfachen Signalen übertragen. Jeder Knoten an einem CAN-Bus liest den Identifikator jedes übertragenen Signals. Die Entscheidung, ob das Signal für einen Knoten relevant ist, wird anhand einer statischen Liste mit CAN-IDs getroffen. Ein Signal beinhaltet eine Dateneinheit von maximal 8 Byte. CAN ist damit gut geeignet für die Verbreitung relativ kleiner Dateneinheiten an mehrere Empfänger. Die Bandbreite variiert je nach Generation der CAN Spezifikation zwischen heute üblichen 125 kbit/s und 1 Mbit/s.

Eine typische Anwendung im Fahrzeug, die via CAN kommuniziert, ist ein Dynamic Stability Control (DSC) System (Beispiel übernommen aus [Joha 05]). Dieses System unterstützt den Fahrer im Falle eines Über- oder Untersteuerns. Wird ein solches Verhalten durch entsprechende Sensoren festgestellt, senden jene CAN-Signale an ein DSC-Steuergerät, welches wiederum über CAN-Signale an Motor- und Bremsaktuatoren die Leistung regelt. Die Übertragung von IP-Paketen über CAN wird in Ditze et al. [Ditz 03], in Lindgren et al. [Lind 08] und in Kern et al. [Kern 11] behandelt. Während die ersten beiden Arbeiten eine prinzipielle Umsetzbarkeit einer solchen Migration zeigen, werden bei Kern et al. Konzepte zur Optimierung des Protokoll-Overheads bei der Übertragung von CAN-Daten über Ethernet vorgestellt und anhand von realen Datensätzen aus einem Fahrzeug evaluiert.

FlexRay

Auch bei FlexRay werden die Daten auf Basis von einfachen Signalen übertragen. FlexRay [Flex 05] ist eine deterministische und fehlertolerante Vernetzungstechnologie, die Datenraten von bis zu 10 Mbit/s unterstützt. Genau wie CAN spezifiziert der FlexRay-Standard die Bitübertragungsschicht und die Sicherungsschicht des OSI-Modells. Ein Signal beinhaltet eine Dateneinheit von maximal 64 Byte. Die Anwendungsgebiete von FlexRay und CAN sind prinzipiell identisch. Es sind die höheren Systemkosten mit einhergehender höherer Bandbreite und Verlässlichkeit von FlexRay, die über den Einsatz von CAN oder FlexRay entscheiden. Damit wird später bei der Ableitung der Anforderungen aus den verschiedenen Anwendungsdomänen weitestgehend nur CAN betrachtet, wo Anwendungen gleichermaßen via CAN oder FlexRay kommunizieren könnten.

Zwei Übertragungskanäle können verwendet werden, der zweite Kanal dient dann entweder der Verdopplung der Bandbreite oder der redundanten Übertragung für sicherheitskritische Anwendungen. FlexRay verwendet einen Netzzugang mit einer globalen Zeitsynchronisation und kann damit Latenzzeiten garantieren. Jeder Kommunikationszyklus besteht aus einem statischen und einem dynamischen Segment, einem „Symbol Window“ und einer „Idle Time“. Im statischen Segment wird Time Division Multiple Access (TDMA) mit konstanter Periode verwendet. Im dynamischen Segment kommt ein dynamisches Minislotting-basiertes System mit variablem Zeitfensterintervall zum Einsatz.

Im „Symbol Window“ können ausgezeichnete Symbole übertragen werden. Am Ende jedes Kommunikationszyklus gibt es stets eine Zeitspanne, in der das Rechnernetz kommunikationsfrei ist.

Durch das Angebot eines dynamischen und eines statischen Segments unterstützt FlexRay zwei Kommunikationsparadigmen. Das dynamische Segment eignet sich besonders für die ereignisbasierte Übertragung und bildet damit die Kommunikation in CAN ab. Das statische Segment eignet sich für die zyklische Übertragung von Daten, das heißt in festgelegten Zeitintervallen werden aktuelle Werte regelmäßig übertragen.

Media Oriented Systems Transport

Media Oriented Systems Transport (MOST) ist ein Standard für die Vernetzung von Multimedia-Komponenten, der seit 2001 eingesetzt wird [MOST]. MOST erlaubt verschiedene Topologien für die physikalische Vernetzung, wobei die Anordnung der Steuergeräte als Ring über ein optisches Medium die gebräuchlichste Variante ist [Grze 08].

Die MOST-Spezifikation [MOST 10] definiert alle sieben Schichten des OSI-Modells. MOST bietet eine hohe Bandbreite im Vergleich zu CAN und FlexRay, abhängig von der Generation der Spezifikation. Heute ist mit MOST 25 und MOST 50 eine Bandbreite von 25 Mbit/s bzw. 50 Mbit/s auf dem MOST-Ring möglich, diverse Fahrzeughersteller planen mit einer Ausbaustufe MOST 150 Bandbreiten von bis zu 150 Mbit/s.

MOST ist ein synchrones Rechnernetz mit einem Timing Master, mit dem sich alle anderen Steuergeräte synchronisieren. Dies erlaubt den Einsatz von nur kleinen Empfangspuffern, die lediglich dem Decoding von Multimedia-Daten dienen. Die Spezifikation unterscheidet drei unterschiedliche Kanäle:

- Ein Steuerungskanal für Steuerungsnachrichten und Verbindungsaufbau bzw. -abbau
- Ein synchroner Kanal mit fester Datenrate
- Ein asynchroner Kanal mit variabler Datenrate

Die Steuerung findet auf dem Steuerungskanal statt. Die Programmierschnittstelle (Application Programming Interface) (API) basiert auf sogenannten MOST-Funktionsblöcken (MOST Function Blocks), welche nach dem Prinzip eines RPC (siehe Abschnitt 5.1.1) arbeiten. Im Gegensatz zu CAN und FlexRay wird die Kommunikation bei MOST für den Anwendungsentwickler abstrahiert, so dass er sich auf die eigentliche Anwendungsfunktionalität konzentrieren kann. Die API beherrscht typische Funktionen von MOST-Steuergeräten wie Audio-Geräten, GPS-Empfängern, Telefonie- und Telematiksystemen und ist damit für die Multimedia-Anwendungsdomäne prädestiniert. Durch eine gemeinsame API wird die Kompatibilität zwischen den Steuergeräten sichergestellt. Der synchrone Kanal dient dem Streaming von

Audio- und Video-Inhalten, für die eine Anwendung eine zeitlich konstante Übertragungsrate benötigt. Der asynchrone Kanal dient der Übertragung von Massendaten, also großen Datenmengen wie beispielsweise Kartenmaterial für die Navigation, die keine garantierte Übertragungsdauer fordern.

Da erst MOST 150 die notwendige Bandbreite zur Übertragung von Video-Daten ermöglichen wird, werden heute dafür spezielle Punkt-zu-Punkt-Verbindungen eingesetzt (z.B. digitales Low Voltage Differential Signaling (LVDS) oder analoge Kabel).

Weiterhin ist MOST in der Lage, IP-Pakete zu übertragen, und kann somit für die IP-basierte Kommunikation genutzt werden.

Ethernet

Ethernet wird in der Literatur bereits als isolierte Lösung für Diagnose und das Erneuern des Navigationskartenmaterials aufgeführt [Bell 11]. Zukünftig wird Ethernet auch für die interne Kommunikation verwendet werden [Bruc 10] [Noba 11]. In den einschlägigen Lehrbüchern wird es allerdings bisher nicht als eine Vernetzungstechnologie aufgeführt, die für die interne Fahrzeugkommunikation geeignet ist. Ethernet wird daher in diesem Abschnitt ausführlicher behandelt.

Der Ethernet Standard IEEE 802.3 [IEEE 12] spezifiziert mit der Bitübertragungsschicht und der Sicherungsschicht die beiden untersten Schichten des OSI-Modells. In den meisten Anwendungsbereichen ist es heute üblich, dass der TCP/IP-Stack die höherschichtigen Protokolle bildet (siehe Abschnitt 2.1). Auch für Ethernet sind unterschiedliche Bandbreiten möglich. Wie in Local Area Networks (LANs) heute üblich wird in der vorliegenden Arbeit auch bei der internen Fahrzeugkommunikation von 100 Mbit/s ausgegangen. Die vorgestellten Arbeiten gehen dabei immer von einem Switched Ethernet aus, das heißt es handelt sich nicht um ein geteiltes Medium. Für die Bandbreite bedeutet dies, dass die 100 Mbit/s zwischen je zwei Links gelten, während beispielsweise die Bandbreite in FlexRay für den gesamten FlexRay-Verbund geteilt wird.

Vor allem der zunehmende Bedarf an Bandbreite für Advanced Driver Assistance Systems (ADAS) und der Wunsch nach einer Vernetzungstechnologie, die mehrere Anwendungsdomänen abdecken kann, macht Ethernet zu einem guten Kandidaten für das Fahrzeugbordnetz [Bell 11]. Die Einführung von Ethernet würde dabei eine Abkehr vom bisherigen Vorgehen der Automobilindustrie darstellen, für eine neue Klasse von Anwendungen mit veränderten Anforderungen stets auch eine neue proprietäre Vernetzungstechnologie zu entwickeln. Ethernet könnte im Gegenteil heute verwendete Vernetzungstechnologien ersetzen und damit zur Homogenisierung des Rechnernetzes beitragen. Erste Überlegungen CAN und MOST, und damit zwei Vernetzungstechnologien mit unterschiedlichen Anwendungsklassen und Kommunikationseigenschaften, gleichermaßen durch Ethernet zu ersetzen, werden in Daoud et al. beschrieben

und simulativ untersucht [Daou 06]. Die flexible Länge von Ethernet-Rahmen ermöglicht sowohl die häufige Übertragung von kurzen Kontrollnachrichten, also auch die Übertragung von Multimedia-Datenströmen. Sommer et al. schreiben dazu, dass die nächste Generation der internen Fahrzeugkommunikationssysteme robust gegenüber schnellen Evolutionsschritten sein muss und dass Ethernet bewiesen hat, eine zukunftsfähige Technologie für neuartige Anforderungen zu sein [Somm 10].

Ein kritischer Aspekt bezüglich der Eignung von Ethernet für die interne Fahrzeugkommunikation betrifft die mögliche Dienstgüte (Quality of Service) (QoS). Jene ist in den grundlegenden Spezifikationen nicht festgeschrieben. Gerade im Bereich der Multimedia-Anwendungen werden für kontinuierliche Audio- und Video-Ströme weiche Echtzeiteigenschaften gefordert. Diese können durch Pufferstrategien meist geeignet gewährleistet werden. Steuerungsnachrichten, wie die für CAN- und FlexRay typischen Signale, haben ebenfalls Anforderungen an harte Echtzeit und Verlässlichkeit. Die entsprechenden Herausforderungen bei der Einführung von Ethernet wurden bereits in Lim, Weckemann und Herrscher [Lim 11c] beschrieben. Die prinzipielle Idee ist es, Daten in verschiedene Klassen einzuteilen und diese unterschiedlich zu priorisieren, also ein gezieltes „Traffic Shaping“ durchzuführen [Rahm 09]. Die von Rahmani verwendeten Datenklassen sind *hard real-time control data*, *real-time audio and video data*, *multimedia data* und *best effort data*. So steht in einem guten Systemdesign für die höher priorisierten Daten immer ausreichend Bandbreite zur Verfügung. Die Priorisierung kann beispielsweise durch den in der Spezifikation IEEE 802.1Q beschriebenen Priorisierungsmechanismus erfolgen. Lim [Lim 11b] beschreibt einen entsprechenden Versuchsaufbau mit Switched Ethernet und verschiedenen Fahrzeug-spezifischen Datenklassen inklusive Steuerdaten. Die Ende-zu-Ende-Latenz liegt in der Simulation für die Zeit-kritischen Datenklassen unter 10 ms, was der maximal zulässigen Zeitdauer für einen Übertragungszyklus bei CAN entspricht.

Die elektromagnetische Verträglichkeit ist eine Herausforderung, die einen Einsatz von Ethernet bisher erschwert, da die im Heim- und Produktionsbereich übliche Schirmung von Kabeln zu hohe Kosten für den Einsatz im Fahrzeug verursachen würden. Mit der Erprobung und Standardisierung von Ethernet mit einer Bandbreite von 100 Mbit/s über eine ungeschirmte Zweidrahtleitung ist aber auch dieses Gegenargument nicht länger gültig [Bruc 10, Nave 13].

Ian Ritches formuliert in „Automotive High Speed Bus Networks“ [Rich 12] eine Prognose zur Integration von Ethernet in den nächsten Jahren für die interne Fahrzeugkommunikation. Der Bedarf an hochbandbreitiger Kommunikation in mehreren Anwendungsbereichen wird Ethernet demnach schnell ins Fahrzeug bringen. Vor allem Kamera-basierte Anwendungen und Diagnose und die Ablösung von MOST durch Ethernet werden die Treiber sein. Ritches schätzt, dass im Jahr 2020 bereits 121 Millionen Ethernet-Knoten in in diesem Jahr produzierten Fahrzeugen verbaut werden.

Gateways

Zwischen den grundlegend unterschiedlichen Netzen mit zueinander inkompatiblen und zudem nur bis auf verschiedenen Schichten ausgeprägten Protokollen sind Kommunikationsknoten für die Vermittlung von Daten notwendig. Entsprechend dem OSI-Modell muss die Konvertierung auf der Anwendungsschicht geschehen, also mittels Gateways. In Ermangelung einheitlicher Schnittstellen müssen dafür proprietäre Gateways eingesetzt werden.

Bei Navet et al. [Nave 05] findet sich eine kritische Betrachtung zu den heute eingesetzten Gateways auf der Anwendungsschicht. Der Bedarf an Kommunikation zwischen den Domänen nimmt stark zu und die heute dafür eingesetzten Gateways weisen viele Schwachstellen auf: Ein Gateway stellt einen *Single Point of Failure* dar. Ein solches Gateway ist eine komplexe Systemsoftware, da es Anwendungswissen benötigt. Jede Software-Änderung mit Anpassung der Schnittstelle einer Anwendung, die über das zentrale Gateway kommuniziert, impliziert auch Software-Änderungen am Gateway. Laut Aussage von Fahrzeugherstellern bewegt sich das zentrale Gateway heute an den Grenzen seiner Leistungsfähigkeit, es wird zusehends schwieriger, den zunehmenden Übersetzungsumfängen gerecht zu werden.

Ohne bereits eine konkrete Lösung vorzuschlagen, abstrahieren Navet et al. [Nave 05] zwei zusammenhängende Lösungsansätze. Zum einen soll die Trennung der Anwendungsdomänen nicht zum Hindernis für die zukünftige Anwendungsentwicklung werden. Dazu muss eine Technologie eingesetzt werden, die verschiedenen Kommunikationsanforderungen bezüglich Bandbreite, Latenz und Kommunikationsparadigmen gerecht wird. Zum anderen soll Middleware dafür eingesetzt werden, die Interoperabilität zwischen heutigen Subnetzen sicher zu stellen. Diese Vorschläge lassen sich leicht auf die konkrete Umsetzung durch ein Internet Protocol-basiertes Fahrzeugbordnetz und entsprechende Kommunikationsmiddleware übertragen. Dies beschreiben die Autoren folgendermaßen: „A classic approach for easing the integration of software-based components is to furnish an MW layer that provides common services and a common interface to application software components.“ [Nave 05] Ein klassischer Ansatz, die Integration von Software-basierten Komponenten zu vereinfachen, ist die Einführung einer Middleware-Schicht, die gemeinsame Dienste und Schnittstellen für Anwendungssoftware-Komponenten bereit hält.

Standardisierungen

Schmeiser [Schm 07] unterscheidet zwei Ansätze, um die Heterogenität der Netze zu abstrahieren. Beim ersten Ansatz steht den Anwendungsprogrammen eine API mit definierten Funktionsaufrufen in Form einer Bibliothek bereit. Darunter liegt eine Schicht zur Anpassung, welche die API auf eine konkrete Vernetzungstechnologie umsetzt. Diese Anpassungsschicht gibt Anwendungen die Möglichkeit zur Angabe von QoS-Parametern. Innerhalb des Systems können dann standardisierte Gateways zum Einsatz kommen. Diese

Lösung mit einem aufwendigen monolithischen Protokollblock und komplexer Gateway-Systemsoftware weist allerdings einige Nachteile auf. Die Architektur ist unflexibel, schwer zu warten und spätere Änderungen und Erweiterungen, wie beispielsweise das Hinzufügen von Anpassungen für weitere Vernetzungstechnologien, sind nur mit erheblichem Aufwand möglich.

Der zweite Ansatz ist eine hierarchische Protokollfamilie. Die gestellten Anforderungen werden entsprechend dem OSI-Modell auf mehrere Schichten aufgeteilt. Zum Datenaustausch zwischen den heterogenen Netzen wurden bisher Gateways eingesetzt, die bekanntlich eine vollständige Umsetzung der Anwendungsdaten auf Schicht 7 vornehmen mussten. Mit einer hierarchischen Protokollfamilie werden die Rechnernetze durchgängig verbunden. Hier werden heterogene Netze statt durch Gateways nun durch Router gekoppelt, die Pakete auf der Vermittlungsschicht weiterleiten können. Das Durchlaufen des Protokoll-Stacks nur bis zur Schicht 3 bedeutet weniger Overhead in den reinen Koppelelementen des Rechnernetzes. In den Routern ist keine komplette Interpretation der Anwendungsdaten notwendig. Lediglich die Header der eingehenden Pakete müssen analysiert werden, um das Paket weiterleiten zu können.

In der Automobilindustrie gibt es diverse Bestrebungen zur Standardisierung. Im Folgenden soll mit AUTOSAR ein Ansatz exemplarisch genannt werden, der sich mit der Einführung von IP-basierten Vernetzungstechnologien beschäftigt und entsprechende Systemsoftware in seinen spezifizierten Umfang aufnehmen könnte. AUTOSAR (AUTomotive Open System ARchitecture) soll die Integration und Wiederverwendung von Software auf verschiedenen Steuergeräten erleichtern. Für die einheitliche Kommunikation im Fahrzeug gibt es dafür eine gemeinsame signalbasierte Schnittstelle für CAN und FlexRay. Die Kodierung erfolgt durch Konvention in einfacher, aber effizienter binärer Form. Die Bestrebungen zur Standardisierung laufen weiter und bieten die Möglichkeit, Ethernet unter derselben standardisierten Schnittstelle zu betreiben. Detaillierte Informationen zu AUTOSAR finden sich bei Stefan Bunzel [Bunz 11] und Simon Fürst [Furs 09].

3.2. Das Internet und der TCP/IP-Protocol-Stack

Die logische Trennung zwischen Anwendungen und Vernetzungstechnologien bei gleichzeitiger De-Facto-Festlegung auf ein einziges Protokoll auf der Vermittlungsschicht hat das enorme Wachstum des Internets begünstigt. Im Gegensatz zur Entwicklung des Kommunikationsbordnetzes im Fahrzeug, stand die Kommunikation zwischen heterogenen Teilnetzen dabei früh im Fokus.

Wenige standardisierte Protokolle bilden das Gerüst des Internets und damit auch für ein IP-basiertes Fahrzeugbordnetz. Ein wesentlicher Grund für den Erfolg des Internets liegt in der Durchsetzung von IP als einziges Protokoll auf der

Vermittlungsschicht, die damit als *Konvergenzschicht* dient. Zahlreiche Vernetzungstechnologien, welche die unteren beiden Schichten implementieren, können IP-Pakete transportieren. Die Vernetzungstechnologien unterscheiden sich damit in Eigenschaften wie Bandbreite, maximaler Übertragungsstrecke oder Zuverlässigkeit, nicht aber darin, dass sie alle einen verbindungslosen Dienst mit globaler Adressierung unterstützen. Oberhalb von IP auf der Transportschicht haben sich im Wesentlichen mit TCP und UDP zwei Protokolle durchgesetzt. Darüber wiederum gibt es auf der Anwendungsschicht (Sitzungs- und Darstellungsschicht haben sich im TCP/IP-Referenzmodell nicht etabliert) zahlreiche Protokolle wie Hypertext Transfer Protocol (HTTP) für Webseiten, File Transfer Protocol (FTP) für Dateitransfer, Internet Message Access Protocol (IMAP) für E-Mails und viele weitere.

Im Unterschied zum Internet sind in einem Fahrzeugbordnetz die Eigenschaften der Vernetzungstechnologien nicht unbekannt. Steuerungsmöglichkeiten, welche Anwendung über welche Technologie kommunizieren, blieben auch in einem IP-basierten Fahrzeugbordnetz möglich. Der Fokus liegt darauf, eine durchgängige, verteilte Kommunikation zu erreichen, in der eine Kommunikation über unterschiedliche Vernetzungstechnologien nahtlos möglich ist. Außerdem muss eine Migration von Anwendungen auf andere Vernetzungstechnologien bei veränderten Anforderungen einfach umsetzbar sein. Bevor es aber um die mögliche Einführung des Internet Protocols ins Fahrzeugbordnetz geht, sollen im Folgenden die Protokolle auf Vermittlungs- und Transportschicht beschrieben werden, um die Vorteile einer unveränderten Wiederverwendung herauszustellen.

Internet Protocol

Das Internet Protocol (IP) [Post 81a] ist ein verbindungsloses Protokoll der Vermittlungsschicht. Es verbindet Systeme in Paket-orientierten Kommunikationsnetzen. Dafür werden Datenpakete von einer Quelle zu einer Senke übertragen, die jeweils durch eine Adresse fixer Länge (IP-Adresse) identifizierbar sind. Das Protokoll erlaubt die Fragmentierung von Paketen in kürzere Pakete, falls eine Vernetzungstechnologie nur Datenrahmen übertragen kann, die kleiner sind als die maximal erlaubte Länge eines IP-Pakets.

Es gibt Implementierungen des Standards, die speziell für den Einsatz mit eingeschränkten Ressourcen entwickelt wurden, wie beispielsweise der *lightweight IP-Stack* (lwIP) [Dunk 01].

Transmission Control Protocol

TCP [Post 81b] ist ein verbindungsorientiertes Protokoll der Transportschicht, das die Ende-zu-Ende-Kommunikation zwischen zwei Prozessen herstellt und dabei eine zuverlässige Datenübertragung gewährleistet. TCP ist Byte-Strom-basiert, Protokolle höherer Schichten können Daten in diesen Byte-Strom

schreiben und auslesen. Dazu beschreibt TCP, wie ein TCP-Socket implementiert werden muss, um über definierte Kommunikationsprimitive Daten aus dem Socket auslesen oder in den Socket schreiben zu können. TCP paketiert diese Daten mit einem Header, der die beiden kommunizierenden Prozesse adressiert. Übertragungsfehler können durch eine 16-bit Checksumme erkannt werden. Eine Flusskontrolle wird durch einen Sliding Window Algorithmus und die Vergabe von Sequenznummern zur Reihenfolgesicherung der Pakete erzielt. Jedes TCP-Paket muss vom Empfänger durch Acknowledgments quittiert werden. Somit wird eine zuverlässige Übertragung ermöglicht. Der Staukontrollmechanismus von TCP beruht auf Timeouts. Die variable Größe des Sliding Windows legt fest, wie viele Datagramme maximal übertragen werden dürfen, bevor bereits versendete Pakete bestätigt wurden. Beim Ausbleiben der Acknowledgments wird die Senderate reduziert, bis die erwarteten Acknowledgments empfangen wurden.

Das Timeout-Verhalten wurde für die Übertragung innerhalb von großen, räumlich weit verteilten Rechnernetzen wie dem Internet entwickelt. Die Annahmen über Latenzzeiten, die den Timeouts zu Grunde liegen, sind daher für das Fahrzeugbordnetz ungeeignet. Es bleibt damit zu prüfen, inwieweit ein IP-basiertes Fahrzeugbordnetz tatsächlich auf TCP und erhältliche Standard-Implementierungen zurückgreifen könnte oder ob an dieser Stelle spezifische Anpassungen notwendig sind.

User Datagram Protocol

UDP [Post 80] ist ein verbindungsloses Protokoll der Transportschicht. UDP arbeitet Paket-orientiert. Es adressiert wie TCP den sendenden und empfangenden Prozess, die Datenübertragung ist aber aus Perspektive des Senders direkt nach dem Senden erledigt. Das Protokoll bietet weder einen Mechanismus zur Reihenfolgesicherung noch zur Erkennung von verlorenen oder duplizierten Datenpaketen. Lediglich beim empfangenden Prozess findet eine Fehlererkennung statt, dafür wird eine 16-bit arithmetische Prüfsumme genutzt.

UDP wird daher in Umgebungen eingesetzt, in denen Verluste von Paketen unwahrscheinlich sind (beispielsweise aufgrund der eingesetzten Vernetzungstechnologie und -topologie) oder für Anwendungen, die mit einer unvollständigen Menge der empfangenen Datenpakete arbeiten können (beispielsweise für Audio- oder Video-Übertragung). Falls einzelne Eigenschaften wie die Erkennung fehlender Pakete oder eine Reihenfolgesicherung dennoch erforderlich sind, müssen sie auf einer höheren Kommunikationsschicht oder durch die Anwendung selbst implementiert werden. Aufgrund seiner Einfachheit ist UDP ein sehr schlank implementierbares Protokoll, das auch gut für Multicast- und Broadcastübertragungen geeignet ist.

Auch für UDP existieren Varianten des Protokolls, die von speziellen Anforderungen durch bestimmte Anwendungen ausgehen. Ein Beispiel ist das Lightweight User Datagram Protocol (UDP-Lite) [Fair 04]. Jenes liefert die

Datenpakete aus, auch wenn die Checksumme der Pakete ungültig ist. Damit eignet es sich für die Übertragung von Multimedia-Daten, wenn der verwendete Audio- oder Video-Codec (z.B. H.264, MPEG4) trotz fehlerhafter Datenpakete gute Ergebnisse liefert.

3.3. Vision eines IP-basierten Fahrzeugbordnetzes

Rainer Steffen et al. [Stef 10] haben bereits 2010 einen Nachweis veröffentlicht, dass sich das heutige Fahrzeugbordnetz im Wesentlichen durch ein IP-basiertes Fahrzeugbordnetz ersetzen ließe. Darin finden sich einige Vorteile der Einführung von IP ins Fahrzeug bzgl. Vereinfachung der Fahrzeugkommunikation, Vorteile für den Entwicklungsprozess, Wiederverwendung und einfachere Wartung von Anwendungen und die Durchgängigkeit der Kommunikation nach außen.

Abbildung 3.2.: Das IP-basierte Fahrzeugbordnetz abstrahiert die verschiedenen Vernetzungstechnologien. In Anlehnung an [Stef 10].

Abbildung 3.2 zeigt eine schematische Darstellung des vorgeschlagenen Paradigmenwechsels bei der internen Kommunikation im Fahrzeugbordnetz. Anstelle der Kommunikation über eine Anwendungsschicht-Gateway (vergleiche Abschnitt 3.1 weiter oben) werden die verschiedenen Vernetzungstechnologien im Fahrzeug „IP-fähig“ gemacht. So können die Anwendungen unabhängig der zugrunde liegenden Vernetzungstechnologie direkt miteinander kommunizieren.

In Glas et al. [Glas 10] finden sich weiterführende Argumente für ein IP-basiertes Fahrzeugbordnetz und welche Forschungsgebiete dazu beitragen müssen. Auf Basis dieser Vorarbeiten sind nachfolgend die wichtigsten Argumente für ein IP-basiertes Kommunikationsbordnetz aufgelistet:

IP-B1. Das Internet Protocol bietet ein einheitliches Adressierungsschema.

IP-B2. Die Einführung eines einzigen Protokolls für die Interdomänenkommunikation reduziert die Komplexität.

IP-B3. Komplexe Anwendungsschicht-Gateways werden durch einfache IP-Router bzw. IP-fähige Switches als Koppelelemente zwischen IP-Teilnetzen ersetzt.

IP-B4. Standard-Hardware, die auf IP aufbaut, ist heute schon deutlich günstiger als vergleichbare proprietäre Hardware bei gleicher oder sogar höherer Bandbreite.

IP-B5. Die Kommunikation wird unabhängig von der physikalischen Vernetzungstechnologie. Damit lassen sich Anwendungen einfacher wiederverwenden, sie sind einfacher zu warten und einzelne Funktionen können auf unterschiedliche Steuergeräte verteilt werden.

IP-B6. IP bietet als standardisiertes Protokoll eine große Menge an existierenden Standards wie TCP und UDP als Transportprotokolle und eine große Auswahl an Anwendungsprotokollen. So ist beispielsweise heute der gemeinsame Zugriff auf eine Festplatte über einen Bus eine komplexe und fehleranfällige Aufgabe. Mit dem Einrichten eines Network File System (NFS) in einem IP-basierten Fahrzeugbordnetz ist dies einfach erledigt.

IP-B7. Das Internet Protocol macht es einfacher, neue Anwendungen ins Fahrzeug zu bringen, gerade solche, die über unterschiedliche Anwendungsdomänen hinweg kommunizieren.

IP-B8. IP ermöglicht Technologieunabhängigkeit. Wenn alle Kommunikation auf IP-basiert, kann man ohne Software-Änderung Ethernet oder einen beliebigen anderen asynchronen Bus verwenden.

IP-B9. Die Verwendung von IP für die interne Kommunikation erlaubt auch eine vereinfachte Durchgängigkeit nach außen für die Anbindung an ein Backend-System, ans Internet und für die Car2X-Kommunikation.

IP-B10. Diagnose über IP ist bereits standardisiert. Ein paralleler Stack mit proprietären Protokollen bedeutet zusätzlichen Aufwand.

IP-B11. Security-Standards für IP sind langjährig erprobt.

Die genannten Vorteile kommen voll zum Tragen, wenn die gesamte interne Fahrzeugkommunikation IP-basiert ist. Dafür wäre die Einführung des Internet Protocols als Revolution notwendig. Ein historisch gewachsenes Kommunikationsbordnetz würde aus einer Gesamtsystemsicht heraus verändert.

In der Praxis muss die Einführung als Evolution verlaufen. Die Gründe für eine schrittweise Migration hin zu einem IP-basierten Fahrzeugbordnetz liegen in der Wirtschaftlichkeit. Das etablierte Kommunikationsbordnetz stößt teilweise an die Grenzen seiner Leistungsfähigkeit, ist andererseits aber als Gesamtsystem stabil und erprobt. Nach einem Baukastenprinzip versucht die Automobilindustrie gleiche Teile zwischen verschiedenen Baureihen einzusetzen, dies gilt insbesondere auch für Steuergeräte und Vernetzungstechnologien, unter anderem um die aufwändige Absicherung von Software-intensiven Komponenten zu vereinfachen. Die Kosten für die etablierten Vernetzungstechnologien sind oftmals niedrig, weil bereits hohe Skaleneffekte zum Tragen kommen. Eine Ausnahme hierbei bildet MOST, da die Technologie noch nicht bei den Volumenherstellern verwendet wird und kein offener Wettbewerb etabliert ist.

Es muss folglich einen Migrationspfad geben, der die Koexistenz zwischen existierenden Vernetzungstechnologien für das Fahrzeug und IP-basierter Kommunikation sichert. Die bei Zinner et al. [Zinn 11] vorgestellte Lösung hierfür ist die Neuentwicklung von Gateways zwischen dem Internet Protocol und den im Fahrzeug etablierten Vernetzungstechnologien.

Im Folgenden soll eine Migration beschrieben werden, die heutigen Indizien folgt, aber selbstredend nicht exakt der tatsächlichen zukünftigen Entwicklung entsprechen kann. CAN bleibt weiterhin der Standardbus. Allerdings werden Steuergeräte, die heute am CAN hängen, mehr und mehr Aufgaben bekommen und damit Kandidaten für die Anbindung mit einer leistungsfähigeren Vernetzungstechnologie wie Ethernet. Die Einführung von Ethernet als „Systembus“ wird folglich auch das Internet Protocol für die interne Fahrzeugkommunikation etablieren. Damit halten offene, standardisierte Technologien Einzug ins Fahrzeug, die nach und nach mehr Anwendungen abdecken werden. Heute wird Ethernet für Diagnostics over Internet Protocol (DoIP) und für das Update von Massendaten wie Navigationskartenmaterial eingesetzt. Im nächsten Schritt werden teure, Bandbreiten-intensive Punkt-zu-Punkt-Anbindungen von Kameras und Monitoren ersetzt werden. FlexRay und MOST werden derzeit für Anwendungen mit Echtzeitanforderungen und hohen Anforderungen an die verfügbare Bandbreite verwendet. MOST kann zwar IP-Pakete übertragen, wahrscheinlicher ist aber die kurzfristige Ersetzung dieser vergleichsweise teuren Technologie. FlexRay könnte ebenfalls mittelfristig durch Ethernet/IP ersetzt werden.

Der Trend zu leistungsfähigeren Steuergeräten, die umfangreichere Anwendungen beherbergen, wird den Bedarf an Bandbreite weiter steigern. Um die einfachen, günstigen Steuergeräte nicht vom Innovationspotential auszunehmen, die der Informationsaustausch im Fahrzeug bietet, könnten leistungsfähige Steuergeräte als Domänenhauptrechner fungieren. Sie würden die Interdomänenkommunikation über das Internet Protocol übernehmen und gegenüber den leistungsschwachen Steuergeräten als Gateway fungieren, das nun auf die Domäne beschränkt deutlich weniger komplex wäre. Das Konzept von Domänenhauptrechnern findet sich bei Lim et al. [Lim 11a]. Ethernet dient in dieser

Arbeit als Backbone für die Vermittlung zwischen den Anwendungsdomänen.

Neben der internen Kommunikation, wird das Fahrzeug zu einem Knoten im Internet. Viele der zukünftigen Innovationen gerade auf dem Gebiet der aktiven Sicherheit werden durch Datenaustausch mit einem Daten-Backend oder der Kommunikation mit Verkehrsinfrastruktur möglich. Gerade die dafür benötigten Telematik-Steuergeräte erscheinen prädestiniert dafür, ihre ohnehin IP-basierten Daten direkt an mehrere mächtige Steuergeräte intern weiterzugeben.

Unabhängig davon, welche Migrationsschritte zuerst kommen, sind die Trends zur Einführung von Ethernet und zur Kommunikation mit der Umwelt klar erkennbar. Diese Trends werden das Fahrzeugbordnetz verändern. Vor allem ist auch bei einer schrittweisen Migration davon auszugehen, dass Anwendungen aus verschiedenen Anwendungsdomänen auf eine IP-basierte Kommunikation umgestellt werden. Daher muss die zu entwickelnde Systemsoftware auch Anforderungen aus mehreren Anwendungsdomänen betrachten.

3.4. Verfügbare IP-basierte Middleware-Lösungen

Der folgende Abschnitt betrachtet die Fragestellung, ob es bereits Konzepte für Kommunikationsmiddleware gibt, die im Fahrzeug eingesetzt werden können. Dafür werden im Folgenden Standards und Produkte betrachtet, welche die Serialisierung der Daten für die Übertragung erlauben und komplette Lösungen, die eine umfangreiche Funktionalität bieten. Auch wenn es nicht um die abschließende Softwareentwicklung für eine Kommunikationsmiddleware für die interne Fahrzeugkommunikation geht, muss also die Frage beantwortet werden, ob es existierende Lösungen gibt, die den besonderen Ansprüchen eines Fahrzeugbordnetzes gerecht werden. Im Forschungsprojekt Sicherheit in Eingebetteten IP-basierten Systemen (SEIS) wurden dazu etwa 30 Komplett- oder Teillösungen untersucht. Die meisten von ihnen zeigen sich bereits bei oberflächlicher Betrachtung als nicht zielführend. An dieser Stelle wird bewusst auf eine komplette Abhandlung verzichtet. Stattdessen liegt der Fokus auf wenigen Lösungen und für jene wiederum auf den wesentlichen Fragestellungen bezüglich Funktionsumfang und Lauffähigkeit auf leistungsschwachen, eingebetteten Plattformen, wie sie im heutigen Fahrzeugbordnetz vorherrschen.

Serialisierungssprachen

Zahlreiche Konzepte umfassen lediglich die plattformunabhängige Aufbereitung der Daten. Im Folgenden sind einige Serialisierungssprachen exemplarisch aufgeführt.

Abstract Syntax Notation One

Abstract Syntax Notation One (ASN.1) [Abst] ist eine reine Serialisierungssprache zur abstrakten Beschreibung von Datentypen. Der Standard wird weit verbreitet zur Datenkodierung im Telekommunikationsbereich (standardisiert durch die International Telecomunication Union) genutzt. Die Kodierung von Daten wird in einer Backus-Naur-Form (BNF)-ähnlichen Darstellung beschrieben. Bei ASN.1 handelt es sich nur um die Definition eines Datenaustauschformats.

YAML/JSON

YAML [YAML] und JavaScript Object Notation (JSON) [Croc 06] sind reine, einfache, unicode-basierte Serialisierungssprachen. Die Beschreibung ist angelehnt an agile Programmiersprachen. Das Ergebnis liegt in einem menschenlesbaren Format vor. Die grundsätzliche Annahme von YAML ist, dass sich jede beliebige Datenstruktur nur mit assoziativen Listen, Arrays und Einzelwerten darstellen lässt. Durch dieses einfache Konzept ist YAML wesentlich leichter von Menschen zu lesen und zu schreiben als beispielsweise XML, außerdem vereinfacht es die Weiterverarbeitung der Daten, da die meisten Programmiersprachen solche Konstrukte bereits integriert haben.

Protocol Buffers

Protocol Buffers [Prot] ist ein Format zur Serialisierung inklusive einer Schnittstellenbeschreibungssprache. Es wurde von Google entwickelt und teilweise unter einer 3-Klausel-BSD-Lizenz veröffentlicht. Hauptsächliche Entwurfskriterien der Protocol Buffers waren laut Beschreibung Einfachheit und Leistungsfähigkeit. Daher ist es als Binärformat konzipiert. Es wurde mit der Motivation einer Anwendung für die Speicherung und zum Austausch strukturierter Daten zwischen verschiedenen Rechenzentren entwickelt.

Protocol Buffers ist ein proprietäres Format, das vor allem gegenüber XML-basierten Serialisierungssprachen für eine deutlich bessere Performanz entwickelt wurde. Eine einfache Beschreibungssprache und die automatische Erzeugung von Code zum einfachen Zugriff auf entfernte Datenstrukturen lassen einen Einsatz im IP-basierten Fahrzeugbordnetz möglich erscheinen.

Lösungen für Kommunikationsmiddleware

Common Object Request Broker Architecture

Die Common Object Request Broker Architecture (CORBA) ist eine offene Spezifikation für eine objektorientierte Infrastruktur für verteilte Systeme [Mowb 98]. Den Kern des Konzepts bildet ein sogenannter Object Request Broker (ORB), der die plattformübergreifenden Protokolle und Dienste definiert. CORBA ist standardisiert und wird von der Object Management Group

(OMG) verwaltet [OMG]. CORBA-konforme Implementierungen vereinfachen das Erstellen verteilter Anwendungen in heterogenen Umgebungen.

Es existieren zahlreiche Arbeiten, die den Standard ergänzen. Beispielsweise erweitern Becker et al. [Beck 00] CORBA um ein generisches Dienstgüte-Management, das die Behandlung von Dienstgüteanforderungen in heterogenen Umgebungen flexibilisiert, anstatt nur eine begrenzte Menge an Dienstgüteparametern zu erlauben. Dieses Prinzip soll auch für die Kommunikationsmiddleware im IP-basierten Fahrzeugbordnetz betrachtet werden.

Eine Abwandlung des Standards speziell für eingebettete Systeme ist CORBA/e [CORB]. Ziel dieser Spezifikation ist es, den Entwicklern von Echtzeitsystemen die Möglichkeit zu geben, Ressourcen zu verwalten und die Vorhersagbarkeit des Systems zu beeinflussen. Dafür ist in CORBA/e die Funktionalität im Vergleich zu Standard-CORBA eingeschränkt. CORBA/e bietet die Möglichkeit, mittels der Compact- und Micro-Profile nach unten zu skalieren. Damit werden nach unten skalierbare Applikationen unterstützt. Dabei ist die Kommunikation zwischen Applikationen mit Compact- und Micro-Profilen als auch mit Standard-CORBA-Applikationen gewährleistet.

Es existieren zahlreiche Implementierungen des CORBA-Standards, wie beispielsweise TAO, MICO, JacORB, Orbacus, omniORB, Orbix, VisiBroker, NEO. Für CORBA/e ist die Auswahl deutlich kleiner, es existieren aber auch hierfür Implementierungen wie OIS und LegORB [Roma 00].

CORBA besitzt eine hohe Komplexität und wurde aus den ursprünglichen Anwendungsdomänen von anderen Technologien verdrängt bzw. konnte sich erst gar nicht etablieren (Enterprise Systeme, Web Services). Für den Einsatz in der internen Kommunikation im Fahrzeugbordnetz würde CORBA einen ORB auf jedem Mikrocontroller erfordern. Der Nachweis, dass dies auch für leistungsschwache, eingebettete Plattformen möglich ist, müsste auch hier noch erbracht werden. Die im Rahmen des SEIS-Projekts erfragten Expertenmeinungen zeigen, dass es derzeit keine Akzeptanz für eine Anwendung in der Automobilindustrie gibt.

Internet Communications Engine

Internet Communications Engine (Ice) ist eine objektorientierte Kommunikationsmiddleware. Ice wurde mit dem Ziel entwickelt, ein besseres CORBA zu werden, also mit vergleichbaren Entwicklungszielen und Anwendungsdomänen, aber mit deutlich weniger Komplexität. Ice bietet Werkzeuge, Schnittstellen und Bibliotheken für objektorientierte Client-Server-Anwendungen. Mit Ice-E gibt es analog zu CORBA/e eine eingeschränkte Variante des Standards. Diese zielt auf Consumer Electronics-Geräte als Plattform.

Ice bietet die Basisfunktionalität für die Kommunikation zwischen verteilten Anwendungen als entfernte Aufrufe zwischen Clients und Servern. Daneben bietet es weitere Mechanismen, beispielsweise um Server bei Gebrauch zu starten, Proxys einzurichten, Anwendungen zu konfigurieren und Updates für Anwendungen einzuspielen. Ice bietet hierzu eine Vielzahl an Diensten, die

selbst als Ice Server umgesetzt sind. Die meisten dieser Dienste sind für ein IP-basiertes Fahrzeugbordnetz nicht relevant, weil sie auf Umgebungen mit einer unbestimmten Anzahl an Kommunikationspartner abzielen und weiterhin andere Rahmenbedingungen annehmen, als sie im Fahrzeug vorherrschen. Das Prinzip der Ice Dienste kann allerdings als Vorbild für die Systemdienste dienen, welche die Kommunikationsmiddleware für das IP-basierte Fahrzeug bieten muss.

Apache Etch

Apache Etch [Apaca] ist ein plattform-, programmiersprachen- und transportunabhängiges RPC-Framework. Obwohl notwendige Bestandteile zu einer Service-orientierten Middleware (siehe Abschnitt 2.4) fehlen, wird eine intuitive Darstellung in der Schnittstellenbeschreibungssprache genutzt, die auf das Anbieten und Konsumieren von Diensten in einem Rechnernetz zielt. Apache Etch bietet ein umfangreiches Tooling an. Dies umfasst eine Schnittstellenbeschreibungssprache inklusive Compiler, der für unterschiedliche Plattformen Programmcode-Rahmen erzeugt. Die Apache Etch IDL wird als Network Service Definition Language (NSDL) bezeichnet und ermöglicht die Beschreibung von entfernten Aufrufen in einer an objektorientierten Programmiersprachen angelehnten Syntax. Dies ermöglicht ein intuitives Verständnis der Beschreibung von Schnittstellen und eine logische Kapselung in zusammengehörende Einheiten. Definiert werden Funktionsaufrufe mit Aufruf- und Rückgabeparametern. Als Parameter sind neben den üblichen Basisdatentypen auch Strukturen wie Arrays, Listen und Mengen möglich. Außerdem erlaubt die Schnittstellenbeschreibung auch die Definition von Ausnahmen (Exceptions) und Annotationen, die zusätzliche Eigenschaften eines entfernten Aufrufs spezifizieren. Über diese Annotationen kann ein Anwendungsentwickler beispielsweise abstrakt festlegen, ob es sich bei einem Aufruf um einen synchronen oder asynchronen Aufruf handelt. Die Menge der möglichen Annotationen ist erweiterbar und bietet daher eine gute Möglichkeit zur Anpassung.

Die Schnittstelle zwischen zwei Kommunikationspartnern wird in einer gemeinsamen Datei definiert, analog zu einem Vertrag zwischen zwei Parteien. Dieses Prinzip entspricht den üblichen Praktiken in der Automobilindustrie, in der die Entwicklung von Steuergeräten durch unterschiedliche Lieferanten geleistet wird.

Apache Etch bietet für die eigentliche Kommunikation einen schlanken und damit effizienten RPC-Mechanismus. Die Serialisierung ist binär, aber grundsätzlich austauschbar. Aktuell wird neben der binären Serialisierung als „Tagged Data Stream“ auch eine XML-basierte Serialisierung angeboten. Es können aber auch andere Serialisierungsvorschriften genutzt werden, sofern eine Abbildung des Funktionsumfangs möglich ist. Apache Etch bietet sich aufgrund seiner Einfachheit, modularen Architektur und freien Verfügbarkeit als Basis für prototypische Untersuchungen an.

Keine der vorgestellten Middleware-Lösungen wird heute im Fahrzeug eingesetzt, noch nimmt die entsprechende Dokumentation Bezug auf eine Eignung für einen solchen Einsatz. Bevor diese Arbeit auf Konzept und Lösung für eine Kommunikationsmiddleware, die für das IP-basierte Fahrzeugbordnetz geeignet ist, eingehen kann, müssen zunächst die besonderen Herausforderungen der Kommunikationsumgebung im Fahrzeug herausgearbeitet werden.

3.5. Zusammenfassung

Das heutige Kommunikationsbordnetz im Fahrzeug ist ein heterogenes Rechnernetz, das historisch gewachsen ist. Die nach und nach entwickelten Vernetzungstechnologien sind untereinander inkompatibel, so dass die Kommunikation durch ein Anwendungsschicht-Gateway übersetzt werden muss. Diese kritische Komponente kann durch die Einführung IP-basierter Kommunikation vereinfacht oder langfristig gar ersetzt werden. Das Internet Protocol bietet eine logische Trennung zwischen Anwendungen und Netzzugang und vereinfacht damit signifikant sowohl die interne Kommunikation über mehrere Anwendungsdomänen hinweg, als auch die Kommunikation des Fahrzeugs nach außen. Auch bei einer schrittweisen Migration ist davon auszugehen, dass Anwendungen aus verschiedenen Anwendungsdomänen auf eine IP-basierte Kommunikation umgestellt werden. Eine IP-basierte Kommunikationsmiddleware muss also die Anforderungen aus diversen Anwendungsdomänen – insbesondere für die Kommunikation zwischen denselben – erfüllen.

Verfügbare Lösungen für eine Kommunikationsmiddleware wurden für Kommunikationsumgebungen entwickelt und eingesetzt, die keine unmittelbaren Rückschlüsse auf deren Eignung für die interne Fahrzeugkommunikation erlauben. Einige der vorgestellte Lösungen zeigen aber Prinzipien, die beim Erarbeiten einer geeigneten Kommunikationsmiddleware helfen. Bevor das Konzept einer Kommunikationsmiddleware für das IP-basierte Fahrzeugbordnetz entworfen werden kann, müssen zunächst die speziellen Herausforderungen der Kommunikation im Fahrzeug herausgearbeitet werden. Diese Identifikation der funktionalen und nicht-funktionalen Anforderungen beginnt mit der näheren Betrachtung typischer Anwendungen und ist Gegenstand des folgenden Kapitels.

4. Anforderungen an die Kommunikationsmiddleware

Anwendungen in einem zukünftigen IP-basierten Fahrzeugbordnetz bilden eine Basis für die Anforderungen an die Kommunikationssoftware. Weitere Anforderungen werden aus der Literaturrecherche in den vorherigen Kapiteln und aus Expertengesprächen mit Fahrzeugherstellern und Zulieferern im Rahmen des Forschungsprojekts SEIS gesammelt. In diesem Kapitel werden diese Anforderungen schrittweise gefolgert, explizit ohne einen Anspruch auf Vollständigkeit zu erheben. Vielmehr werden diejenigen Anforderungen herausgestellt, die eine Untersuchung von Kommunikationsmiddleware für das Fahrzeugbordnetz aus wissenschaftlicher Perspektive fördern. Anschließend werden daraus Schlussfolgerungen für die prinzipiellen Rahmenbedingungen einer Entwicklung gezogen. Ein kompletter Anforderungskatalog ist erst für ein konkretes System mit final definierten Steuergeräten und Anwendungen möglich und daher bei einer nachgelagerten Software-Entwicklung noch zu erbringen. Die identifizierten Anforderungen und Schlussfolgerungen führen zum eigentlichen Konzept im folgenden Kapitel.

4.1. Typische Anwendungen im Fahrzeugbordnetz

Dieser Abschnitt führt einige typische Anwendungen auf, die Kommunikationsanforderungen an das IP-basierte Fahrzeugbordnetz stellen. Ziel ist es, einige heutige und zukünftige Anwendungen zu umreißen, die stellvertretend auf die allgemeinen Anforderungen schließen lassen. Ein Beitrag hierzu wurde bereits in Weckemann, Lim und Herrscher [Weck 11] veröffentlicht.

Auch wenn sich die Vernetzungstechnologien ändern, bleiben viele der heutigen Anwendungsfälle bestehen. Viele Anwendungen bei CAN dienen dem reinen Verteilen von Informationen. Ein Beispiel hierfür ist das Verteilen der aktuellen Fahrtgeschwindigkeit. Auch wenn bei CAN jeder Kommunikationspartner prinzipiell jede Nachricht lesen kann, ist die Menge der Empfänger einer bestimmten Dateneinheit zur Entwicklungszeit statisch festgelegt. Da in einem IP-basierten Fahrzeugbordnetz nicht davon ausgegangen werden kann, dass die zugrundeliegende Vernetzungstechnologie eine solch einfache Verteilung an viele potentielle Empfänger bereits impliziert, stellt sich die Frage, inwieweit eine Kommunikationsmiddleware eine Punkt-zu-Multipunkt-Kommunikation

abbilden muss. Viele Anwendungen benötigen aktuelle Werte über den aktuellen Fahrzeug- oder Fahrzustand. So muss beispielsweise die Information über die aktuelle Fahrtgeschwindigkeit auch einem Cabrioverdeck-Steuergerät signalisiert werden, wenn vorgeschrieben ist, dass das Verdeck nur bis zu einer festgelegten Fahrtgeschwindigkeit geöffnet werden darf. Ein solches Verteilen von Informationen für viele Anwendungen muss in einem IP-basierten Fahrzeugbordnetz auf Middleware-Schicht signalbasiert umgesetzt werden.

Weitere heutige Anwendungen im Fahrzeug bewegen sich vom Fensterheber bis hin zur Motorsteuerung, in der Infotainmentdomäne von digitalem Streaming von Audio- und Video-Daten bis zur komplexen Steuerung von Audio-Verstärkern und dem Navigationssystem. Um zu zeigen, wie typische Anwendungen im IP-basierten Fahrzeugbordnetz kommunizieren, sind im Folgenden einige Beispiele beschrieben.

Fahrerassistenzkameras unterstützen den Fahrer beispielsweise beim Einparken, indem sie ihm ein Video nach hinten (Rear View System) oder zusammengesetzt aus den Daten mehrerer Kameras rund um das Fahrzeug (Top View System) auf einem integrierten Monitor anzeigen. Aufgrund der hohen Bandbreitenanforderung für die Video-Übertragung werden die Kameras für die Vernetzung mit Ethernet vorgesehen. Andererseits benötigt ein solches Kamerasteuergerät lediglich einen leistungsschwachen Mikrocontroller, der den Bildwandler steuert. Das Video-Streaming kann von einem dedizierten Chip übernommen werden. Während heute ein analoges oder digitales Video-Signalkabel und ein CAN-Anschluss am Bildwandler für die Steuerung notwendig sind, wird in einer zukünftigen IP-basierten Vernetzung auf Basis von Ethernet nur eine Vernetzungstechnologie benötigt. Der zweite Vorteil in der IP-basierten Kommunikation ist, dass derselbe Video-Strom von mehreren Empfängern genutzt werden kann, was Kosten reduziert und die Topologie vereinfacht. Die Steuerung erfolgt standardisiert über die Kommunikationsmiddleware.

Typische Steuerfunktionen sind Start/Stopp, Helligkeitseinstellungen, Synchronisierung und ähnliches. Um auf die zahlreichen auf Basis des Internet Protocols verfügbaren, standardisierten Protokolle zurückgreifen zu können, muss die Kommunikationsmiddleware weitere Kommunikation auf demselben physikalischen Link zulassen. Die eigentliche Video-Übertragung kann als (komprimierter) Video-Strom über ein standardisiertes Protokoll wie das Real-Time Transport Protocol (RTP) [Schu 03] erfolgen, wie bei Hintermaier et al. [Hint 10] beschrieben.

Ein vergleichbarer Anwendungsfall aus der Infotainmentdomäne ist ein Audio-Verstärker, übernommen aus Rahmani et al. [Rahm 07]. Ein typischer externer Verstärker wird heute über analoge Audio-Kabel und CAN zur Steuerung angebunden. Alternativ kann sowohl die digitale Übertragung von Audio-Strömen als auch die Steuerung über MOST erfolgen. In einem IP-basierten Fahrzeugbordnetz wird auch hier das Audio-Streaming von einem standardisierten Protokoll abgedeckt, während die Steuerung des Verstärkers über die Kommunikationsfunktionen einer Kommunikationsmiddleware erfolgt. Die

entsprechenden Einstellungen eines Verstärkers sind Regelung der Lautstärke, Stummschaltung, Fading und Mixing von verschiedenen Audio-Quellen und Anpassungen am Equalizer. Abbildung 4.1 zeigt schematisch, wie ein Audio-Verstärker und mehrere Kameras via Ethernet in einem IP-basierten Fahrzeugbordnetz an das zentrale Steuergerät der Infotainmentdomäne (Head Unit) angeschlossen werden. Dabei wird derselbe physikalische Link sowohl für die Steuerung durch die Kommunikationsmiddleware als auch für das eigentliche Streaming der Audio- bzw. Kamera-Daten durch ein standardisiertes Protokoll genutzt.

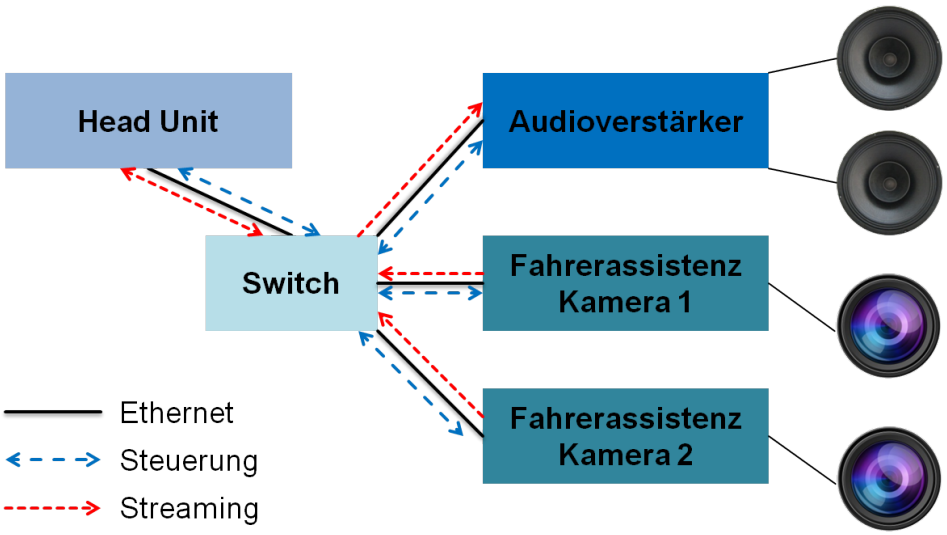

Abbildung 4.1.: Steuerung und Streaming zwischen Head Unit, Audio-Verstärker und Kameras über denselben physikalischen Link.

Auch ein Telematiksteuergerät hat vergleichbare Anforderungen. Ein Telematiksteuergerät kapselt diverse Verbindungstechnologien des Fahrzeugs nach außen, der Anwendungsfall ist übernommen aus Steffen et al. [Stef 10]. In einem IP-basierten Fahrzeugbordnetz kann das Fahrzeug ohne Aufwand für die Übersetzung zwischen nicht-kompatiblen Protokollen eine direkte Verbindung ins Internet aufbauen, über Mobilfunk (GPRS, UMTS, LTE) oder ein lokales Wireless Local Area Network (WLAN), beispielsweise in der heimischen Garage oder in einer öffentlichen Tiefgarage. Durch die Verbindung ins Internet können eine Vielzahl unterschiedlicher Dienste nahtlos ins Fahrzeug übernommen werden, beispielsweise Voice over IP (VoIP), Video-Konferenzen, Internet-TV und Musik-Streaming-Dienste.

In ein solches Telematiksteuergerät werden, neben der Technologien für eine Verbindung ins Internet, weitere funkbasierte Technologien integriert. Der Empfang von Funksignalen für Radio und TV sowie die Positionierung über ein Global Navigation Satellite System (GNSS) – wie das amerikanische Global Positioning System (GPS) oder das europäische Galileo – wird ebenfalls vom Telematiksteuergerät übernommen. Die zunehmende Integration mehre-

rer sogenannter Frontends für den Funkempfang in ein Steuergerät liegt in der dadurch erreichten Reduzierung von Antennen- und Verkabelungskosten begründet. Durch den weltweiten Einsatz der Fahrzeuge bei starken regionalen Unterschieden in den verwendeten Funkstandards besteht ein Trend zum Software-defined Radio, wie allgemein in Tuttlebee [Tutt 99] oder speziell für die Automobilindustrie in Stolz et al. [Stol 12a] dargestellt. Das bedeutet, dass die Funktionalität beim Empfang von Funksignalen zusehends von Software- anstelle von Hardware-Umfängen erbracht wird. Die dafür benötigte Steigerung der Rechenleistung wird durch die gewonnene Flexibilität aufgewogen, da die gleiche Hardware in mehr Regionen und damit in höherer Stückzahl verbaut werden kann. Ein integriertes Telematiksteuergerät kapselt somit eine Vielzahl von unterschiedlichen Diensten vom Internet-Zugang bis zur Positionierung des Fahrzeugs. Die Kommunikationsmiddleware muss diese Dienste steuern. Falls keine kontinuierliche Datenübertragung benötigt wird, muss sie zusätzlich die Abfrage einzelner Datensätze mittels eines einfachen Aufrufs anbieten. Dies entspricht einem entfernten Funktionsaufruf, bei dem eine parametrisierte Anfrage ein typisiertes Ergebnis zurück liefert.

Sowohl im Fall der Fahrerassistenzkameras, des Audio-Verstärkers, als auch des Telematiksteuergeräts übernimmt die Kommunikationsmiddleware die Übertragung von Steuerbefehlen und einfachen Anfragen, um dem Anwendungsprogrammierer Abstraktion von der tatsächlichen Verteilung der Steuergeräte zu bieten. Daneben gibt es bereits zahlreiche Protokolle für die Datenübertragung, wie beispielsweise RTP. Für jenes ist entsprechend ein Steuerungsprotokoll definiert, das auf die Steuerung von RTP-Strömen optimiert ist – das RTP Control Protocol (RTCP) [Schu 03]. Auch der Audio Video Bridging Standard (AVB IEEE 1722) spezifiziert mit den Erweiterungen in IEEE 1722.1 unter anderem die Steuerung von AVB-Geräten [IEEE 10a, IEEE 10b]. Eine Untersuchung der Performance von AVB im Fahrzeug findet sich in Lim et al. [Lim 12]. Die Kommunikationsmiddleware muss die Verwendung von standardisierten Protokollen und Technologien, die bereits auf Basis des Internet Protocols verfügbar sind, im Fahrzeugbordnetz zulassen.

Die Erwartung an zukünftige Anwendungen lassen auf weitere Anforderungen schließen. Nach der Infotainmentdomäne erscheint die Fahrerassistenz als geeigneter Kandidat für die IP-basierte Vernetzung. Die voranschreitende Automatisierung der Fahraufgabe durch autonom bremsende und beschleunigende Geschwindigkeitsregelungssysteme (Tempomat) hin zum gänzlich autonomen Fahren benötigt ein Umfeldmodell über die nähere Umgebung um das Fahrzeug. Dieses Umfeldmodell muss neben der Infrastruktur (z.B. Anzahl der Fahrspuren) auch Informationen über andere Verkehrsteilnehmer wie nachfolgende Fahrzeuge und Fußgänger am Straßenrand als Objekte repräsentieren.

Diese Informationen können nur durch ein Zusammenspiel zahlreicher Sensoren gewonnen werden, die schließlich zu einem gemeinsamen Modell fusioniert werden. Dieses Umfeldmodell muss dann in einer Repräsentation durch komplexe Datenstrukturen mit Dienstgüteanforderungen mehreren Empfän-

gern zur Verfügung gestellt werden. Diese Dienstgüte bezieht sich in diesem Beispiel vor allem auf strikte Latenzanforderungen an die Kommunikation. Sowohl bei der Generierung als auch bei der Verteilung des Modells sind dabei zahlreiche Steuergeräte beteiligt. Dieser Anwendungsfall betrifft damit mehrere Anwendungsdomänen, weil ein komplexes Zusammenspiel zwischen Erfassung in der Fahrerassistenzdomäne, Reaktionen des Motors und des Fahrwerks und Erhalt der Aufmerksamkeit des Fahrers in der Infotainmentdomäne gefordert sind.

Eine zunehmende Anzahl von Assistenzsystemen erhöht die Sicherheit und macht das Reisen komfortabler. Die meisten Anwendungen sind aber speziell für einen Kontext gedacht. So ist beispielsweise die Rückfahrkamera nur beim Ein-/Ausparken oder Rückwärtsfahren relevant. In der Regel sind im heutigen Fahrzeugbordnetz die meisten Steuergeräte im Fahrbetrieb dennoch ständig aktiv. Aus Gründen der Energieeffizienz müssen Konzepte entwickelt werden, die Steuergeräte im Ruhezustand erlauben, aber eine schnelle Reaktionsfähigkeit im Bedarfsfall wiederherstellen. Aus Kommunikationsperspektive muss dies im IP-basierten Fahrzeugbordnetz geeignet umgesetzt werden.

Die Anbindung von mitgebrachten elektronischen Geräten des Kunden wird zunehmend wichtiger für die Fahrzeughersteller, wie schon in Steffen et al. [Stef 10] ausgeführt. CE-Geräte, wie beispielsweise ein Smartphone oder ein Tablet, sind heute ein ständiger Begleiter nicht nur jüngerer Zielgruppen. Die Erwartungshaltung der Kunden verlangt nach einer – gegebenenfalls an die Fahraufgabe angepassten – Nutzung der Dienste auf dem eigenen CE-Gerät im Fahrzeug. Die Anbindung erfolgt heute über USB (hohe Bandbreite) oder Bluetooth (drahtlose Verbindung). WLAN (IEEE 802.11) kombiniert die Vorteile hoher Bandbreite und bequemer drahtloser Nutzung und wird sich für die Integration von CE-Geräten etablieren.

Vergleichbar zur Verbindung des Telematiksteuergeräts ins Internet, können auch hier durch eine IP-basierte Vernetzung zahlreiche Dienste des CE-Gerätes für das Angebot im Fahrzeug einfacher genutzt werden, als dies bei proprietären Technologien der Fall wäre. Andererseits entstehen durch die Anbindung von potentiell unsicheren Geräten durch die einfache Integration auch Security-Risiken, die für die interne Fahrzeugkommunikation in diesem Ausmaß neu sind. Auch wenn die interne Kommunikation im Fahrzeug im Fokus für die IP-basierte Kommunikationsmiddleware steht, ist die Anbindung von CE-Geräten damit ein wichtiger Anwendungsfall für die Anforderungen an die Sicherheit im IP-basierten Fahrzeugbordnetz.

4.2. Funktionale Anforderungen an die Kommunikationsmiddleware

Die prinzipiellen funktionalen Bausteine der Kommunikationsmiddleware sind eine Schnittstellenbeschreibungssprache (Interface Definition Language), ein

oder mehrere Kommunikationsparadigmen, die ein Koordinationsmodell implementieren, Marshalling für den einheitlichen Datenaustausch und Systemdienste, welche die Kommunikation im verteilten System aus Anwendungssicht vereinfachen. Dies wurde in Abschnitt 2.3 allgemein eingeführt. Im Folgenden geben diese Bestandteile die Struktur vor, um die funktionalen Anforderungen detailliert zu beschreiben.

4.2.1. Anforderungen an die Schnittstellenbeschreibungssprache

Die Anforderungen an die IDL sind nicht scharf zu umfassen, da die Diskussion im Wesentlichen auf subjektiven Präferenzen fußt, wie prägnante Schnittstellen beschrieben werden sollen. An dieser Stelle werden Expertenmeinungen wiedergegeben. Wichtig dafür ist, dass die Anforderungen an die Schnittstellenbeschreibungssprache weitestgehend unabhängig von der Architektur der Kommunikationsmiddleware und damit vom Kern der vorliegenden Arbeit sind. Da bei der Implementierung und Evaluation des Konzepts die Definition von Schnittstellen erforderlich ist, wird im Folgenden dennoch auf diese Anforderungen eingegangen.

Die Kommunikation muss eine Schnittstellenbeschreibungssprache mit definierter Syntax und Semantik bieten, welche die Beschreibung von Schnittstellen unabhängig von konkreten Programmiersprachen erlaubt. Sie abstrahiert den Funktionsumfang, den die Kommunikationsmiddleware für die Kommunikation zwischen Anwendungen bietet. In ihr muss daher die Nutzung des Koordinationsmodells und anderer Systemdienste beschreibbar sein, außerdem welche Parameter ein Kommunikationspartner übergibt und welche Parameter er als Antwort erwartet. Da die Beschreibung der Schnittstelle letztendlich auf konkreten Plattformen implementiert werden muss, bieten viele existierende Lösungen ein Tooling, das Schnittstellen automatisiert in Code-Rahmen für ausgewählte Programmiersprachen übersetzt. Die Anforderungen wurden im Rahmen des Forschungsprojekts SEIS diskutiert. An den Expertengesprächen waren Teilnehmer mehrerer deutscher Fahrzeughersteller, Zulieferer und Forschungseinrichtungen beteiligt. Die Anforderungen stellen damit eine subjektive, aber für die Automobilindustrie praxisnahe Einschätzung dar. Die gesammelten Anforderungen an die Schnittstellenbeschreibungssprache sind im Folgenden aufgelistet:

Anforderung 1 Verfügbarkeit einer IDL: Die Kommunikationsmiddleware muss eine IDL mit definierter Syntax und Semantik anbieten.

Anforderung 2 Beachtung bestehender Definitionskonzepte: Die IDL muss Anwendungsschnittstellen unter Berücksichtigung von bestehenden Konzepten beschreiben können. Bestehende Konzepte können z.B. CAN, FlexRay, MOST sein.

Anforderung 3 Skalierbarkeit und geringe Kopplung der IDL: Hinzunahme, Entfernung und Änderung von Komponenten sollen auf IDL-Ebene möglichst geringe Auswirkungen haben.

Anforderung 4 Verständlichkeit der IDL: Die IDL muss für einen Menschen leicht verständlich sein. Dies bedeutet insbesondere, dass sich die Syntax an verbreitete, höhersprachige Programmiersprachen anlehnen soll.

Anforderung 5 Einfache Datentypen: Die IDL muss mindestens die primitiven Datentypen boolean, byte, integer, float, string, enum, void unterstützen.

Anforderung 6 Strukturierte Datentypen: Die IDL muss mindestens Listen variabler und fester Länge, zusammengesetzte Datentypen sowie beliebige Schachtelungen dieser Datentypen unterstützen.

Anforderung 7 Wiederverwendung: Die IDL muss eine Methode zur Wiederverwendung von Schnittstellendefinitionen unterstützen. Dies kann z.B. auch Vererbung sein.

Anforderung 8 Modularisierung: Die IDL muss ein modulares Design ermöglichen.

Anforderung 9 Fehlerbehandlung: Die IDL soll die Kennzeichnung einer dedizierten Möglichkeit zur Fehlerbehandlung in der Schnittstelle ermöglichen.

Anforderung 10 Automatische Codegenerierung: Die Kommunikationsmiddleware muss auf der Grundlage einer definierten Schnittstelle automatisch Code generieren können.

Wie in Abschnitt 3.4 beschrieben, existieren zahlreiche Lösungen und damit Vorschläge zur Schnittstellenbeschreibung. Diese sind weitestgehend austauschbar. Für die Implementierung und Validierung der Konzepte wird demnach eine beliebige IDL gewählt, die obige Anforderungen erfüllt.

4.2.2. Anforderungen an das Koordinationsmodell

Laut der verwendeten Definition von Issarny (siehe Abschnitt 2.3) beschreibt das Koordinationsmodell das dynamische Verhalten der Kommunikationsmiddleware für die Bereitstellung von Kommunikation für Anwendungen im verteilten System. Es wird durch ein oder mehrere Kommunikationsparadigmen detailliert. Diese wiederum definieren, wie Kommunikation mit einer festgelegten Semantik hergestellt wird, wie in Abschnitt 2.4 ausgeführt. Als Hypothese für die nähere Untersuchung wird ein RPC-Framework angenommen, das auch signalbasierte Kommunikation wie in einer Nachrichten-orientierten

Kommunikationsmiddleware und Systemdienste für generelle Kommunikationsfunktionen bietet. Die Detaillierung dieser Anforderungen erfolgt durch Folgerungen aus den relevanten Anwendungen und einer Analyse heutiger Kommunikation im Fahrzeugbordnetz.

Wie in Abschnitt 3.1 eingeführt, werden bei CAN Daten auf Basis von einfachen Signalen übertragen. Jeder Kommunikationsknoten an einem CAN-Bus liest den Identifikator jedes übertragenen Signals. Die Entscheidung, ob das Signal für einen Knoten relevant ist, wird anhand einer statischen Liste mit CAN-IDs getroffen. Ein Signal beinhaltet eine Dateneinheit von maximal 8 Byte. CAN ist damit gut für die Verbreitung relativ kleiner Dateneinheiten an mehrere Empfänger geeignet. FlexRay funktioniert ähnlich, die wesentlichen Unterschiede sind die höhere Bandbreite, die deterministische Übertragung, die maximale Größe einer Dateneinheit von 64 Byte und die Unterscheidung in statisches und dynamisches Übertragungssegment. Während sich das dynamische Segment ebenfalls für die ereignisbasierte Kommunikation eignet, bietet sich das statische Segment für eine zyklische Übertragung in definierten Zeitintervallen an. Die Anwendungsgebiete von FlexRay im Unterschied zu CAN sind erstens verteilte Regelungen in der Antriebs- oder Fahrerassistenzdomäne, die eine höhere Bandbreite erfordern. Zweitens setzen sicherheitskritische Anwendungen wie beispielsweise ein elektronischer Bremseingriff auf FlexRay, weil sie eine verlässliche Übertragung in Echtzeit fordern.

Da die CAN- und FlexRay-Spezifikationen nur die beiden unteren Schichten des OSI-Modells definieren, lassen sich aus den Vernetzungstechnologien nicht direkt Rückschlüsse auf die tatsächliche praktische Nutzung ziehen. Dazu wurde eine Analyse eines bestehende Fahrzeugbordnetzes durchgeführt. Das Fahrzeugbordnetz ist für ein gewähltes Fahrzeugderivat in einer Bordnetzdatenbank spezifiziert, in der alle Kommunikationsbeziehungen zwischen Steuergeräten bis auf die Detaillierung einzelner Nachrichten gelistet sind. Diese Nachrichten sind statisch vordefiniert und werden auf Basis von Sendebedingungen zur Laufzeit mit aktuellen Werten versendet. Die Betrachtungen zeigen, dass sich die Anwendungsdomänen ähneln, es wurde daher zur Vereinfachung eine Anwendungsdomäne und ein Fahrzeugderivat ausgewählt. Zunächst wurde ausgewertet, welche Kommunikationsparadigmen in CAN, das als priorisierter Bus für eine Kommunikation auf Basis von Ereignissen konzipiert ist, tatsächlich in der Praxis verwendet werden. In der Bordnetzdatenbank lassen sich dafür drei Sendebedingungen unterscheiden. Neben der erwarteten *ereignisbasierten* und *zyklusbasierten* Kommunikation finden auch entfernte Abfragen von Daten *auf Anfrage* statt. Dies kann dadurch erreicht werden, dass eine Nachricht als Ereignis für das Senden bestimmter Daten definiert wird.

Abbildung 4.2 zeigt für eine Anwendungsdomäne und ein Fahrzeugderivat die Anzahl der statisch definierten Nachrichten je Sendebedingung. Mit 2460 *ereignisbasierten* und 2425 *zyklusbasierten* wird die Mehrzahl der Nachrichten so ausgetauscht, wie es für einen priorisierten Bus wie CAN zu erwarten ist. Mit 212 Nachrichten *auf Anfrage* folgt ein nicht vernachlässigbarer Anteil aber

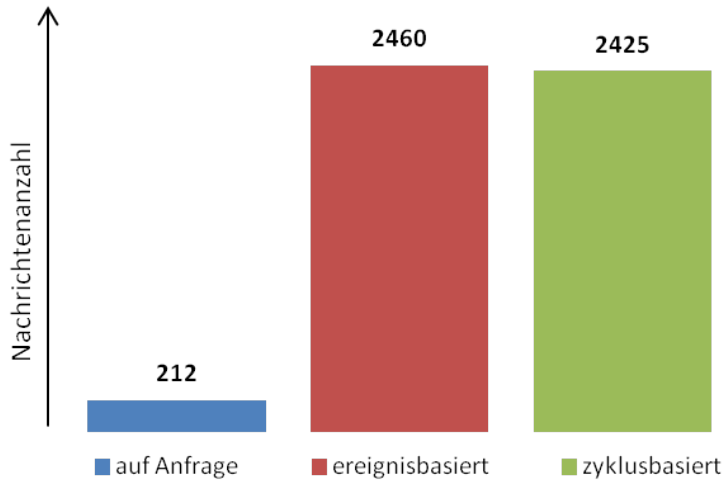

Abbildung 4.2.: Verteilung statisch definierter CAN-Nachrichten nach den Sendebedingungen *auf Anfrage*, *ereignisbasiert* und *zyklusbasiert*.

auch einem Kommunikationsparadigma, das für die Kommunikation über CAN unerwartet ist. In der Praxis wird CAN damit auch für ein funktionsbasiertes Kommunikationsparadigma verwendet, das über den konzipierten Umfang der Vernetzungstechnologie hinausgeht.

	CAN	FlexRay	MOST
Spezifikation	proprietär	proprietär	proprietär
Kommunikationskanal	priorisierter Bus	statisches und dynamisches Segment	Steuerungskanal, synchroner und asynchroner Kanal
Paradigmen	**ereignisbasiert**, zyklisch, Datenabfrage	**ereignisbasiert, zyklisch**, Datenabfrage	**funktionsbasiert**
Anwendung	(Sensor-) Daten übertragen	(Sensor-) Daten übertragen	Steuerung, Daten übertragen, Streaming, Massendaten

Tabelle 4.1.: Heterogenität bezüglich Kommunikationskanälen, Paradigmen und Anwendungen bei den heutigen Vernetzungstechnologien.

Bei MOST basiert die API auf Funktionsblöcken, welche nach dem Prinzip eines RPC (siehe Abschnitt 5.1.1) arbeiten. Diese API beherrscht typische Funktionen von MOST-Steuergeräten wie Audio-Geräten, GPS-Empfängern, Telefonie- und Telematiksystemen und ist damit für Multimedia-Anwendungen

	IP-basiert
Spezifikation	offene Standards und Technologien verfügbar
Kommunikationskanal	Kommunikationsmiddleware über einem verbindungslosen, paketorientierten Dienst
Paradigmen	**funktions- und signalbasiert**
Anwendung	**Steuerung, Daten übertragen**, Streaming, Massendaten

Tabelle 4.2.: Die entsprechenden Eigenschaften eines IP-basierten Fahrzeugbordnetzes, in dem eine Middleware mehrere Paradigmen abbildet und verschiedene Anwendungen unterstützt.

in der Infotainmentdomäne prädestiniert. Durch das Bereitstellen einer gemeinsamen API wird die Kompatibilität zwischen den Steuergeräten sichergestellt. Tabelle 4.1 zeigt einen Überblick zu den bei CAN, FlexRay und MOST verwendeten Kommunikationsparadigmen und -arten. CAN und FlexRay kommunizieren signalbasiert, auch wenn das Konzept eines Funktionsaufrufs *auf Anfrage* in der Praxis umgesetzt wird. Beide verteilen dabei hauptsächlich Daten im verteilten System und dienen der Steuerung von entfernten Anwendungen. MOST kommuniziert funktionsbasiert zur Steuerung von Anwendungen und bietet daneben hochbandbreitige Übertragung von Datenströmen und Massendaten. Tabelle 4.2 zeigt entsprechend die Eigenschaften eines IP-basierten Fahrzeugbordnetzes. Die Übertragung von Datenströmen und Massendaten wird über existierende, standardisierte Protokolle erfolgen. Für die Steuerung und für einfache Datenabfragen muss die Kommunikationsmiddleware dagegen mindestens die heute üblichen Kommunikationsparadigmen geeignet abbilden.

Damit lassen sich nun die Anforderungen an das Koordinationsmodell formulieren. Analog zum einfachen Verteilen von Daten bei CAN und FlexRay muss die Kommunikationsmiddleware signalbasierte Kommunikation ermöglichen.

Anforderung 11 Die Kommunikationsmiddleware muss ein signalbasiertes Kommunikationsparadigma umsetzen.

Falls keine kontinuierliche Datenübertragung benötigt wird, muss die Kommunikationsmiddleware zusätzlich die Abfrage einzelner Datensätze als einfache Datenabfrage anbieten. Zudem muss sie entfernte Anwendungen entsprechend der heutigen MOST-Funktionsblöcke steuern. Beides entspricht einem entfernten Funktionsaufruf, bei dem eine parametrisierte Anfrage ein typisiertes Ergebnis zurück liefert.

Anforderung 12 Die Kommunikationsmiddleware muss ein funktionsbasiertes Kommunikationsparadigma umsetzen.

Dieses funktionsbasierte Kommunikationsparadigma muss auch Möglichkeiten zur Behandlung von Übertragungsfehlern anbieten, die ansonsten jede Anwendung selbst behandeln muss. Dafür müssen Garantien umgesetzt werden, die über die Wiederholung eines Aufrufs im Fall eines Fehlers bei der Übertragung entscheiden. Diese Garantien werden als Aufrufsemantiken bezeichnet und lassen sich beispielsweise durch die Eigenschaften der Übertragungswiederholung als *maybe*-, *at-least-once*- und *at-most-once*-Übertragung unterscheiden, wie dies unter anderem in Tay et al. [Tay 90] aufgeführt ist.

Anforderung 13 Die Kommunikationsparadigmen müssen vorgegebene Aufrufsemantiken umsetzen.

Für die Übertragung von Datenströmen und Massendaten werden in einem IP-basierten Fahrzeugbordnetz existierende Protokolle genutzt. Ein Beispiel ist der beschriebene Anwendungsfall der Fahrerassistenzkameras. Jene verwenden mit RTP ein spezielles Protokoll zur Übertragung von Video-Strömen, während die Steuerung der Kameraeinstellungen durch die Kommunikationsmiddleware erfolgt.

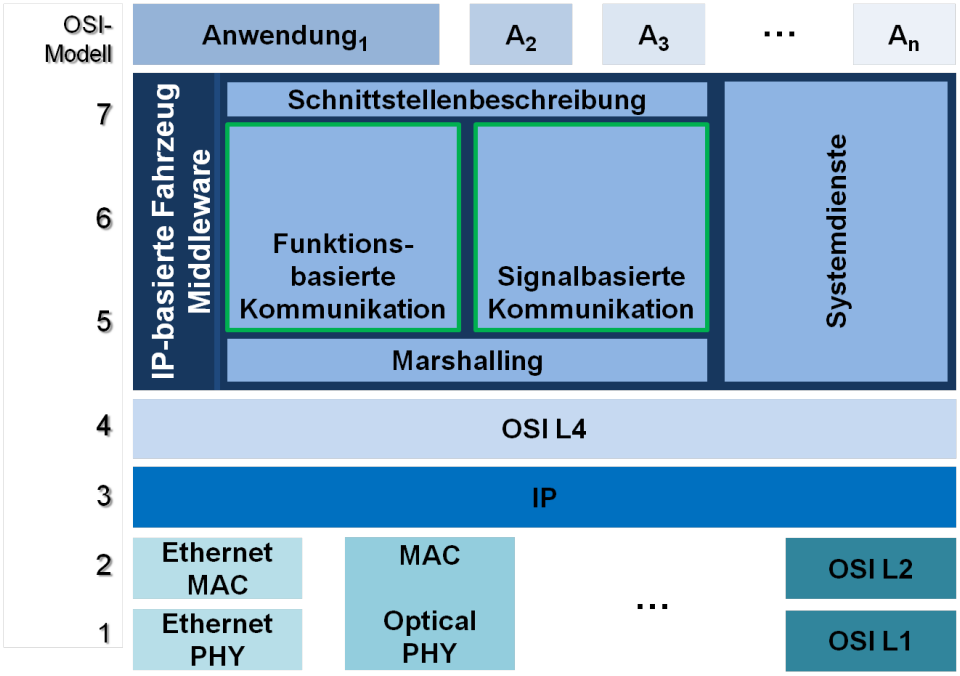

Abbildung 4.3.: Die Funktionalität der IP-basierten Kommunikationsmiddleware mit zwei grundlegenden Kommunikationsparadigmen.

Anforderung 14 Die Kommunikationsmiddleware muss IP-basierte Kommunikation außerhalb ihrer eigenen Implementierung erlauben bzw. ermöglichen, um den Vorteil der Wiederverwendung von standardisierten Protokollen des TCP/IP-Stacks nutzen zu können.

Abbildung 4.3 zeigt somit die zwei identifizierten Kommunikationsparadigmen, welche die Kommunikationsmiddleware anbieten muss.

4.2.3. Anforderungen an das Marshalling

Die Kommunikationsmiddleware muss Serialisierungsregeln implementieren, so dass verteilte Anwendungen auf unterschiedlichen Plattformen ein einheitliches Übertragungsformat produzieren und konsumieren können. Serialisierung bedeutet die Transformation einer komplexen Datenstruktur auf einer gegebenen Plattform und Programmiersprache in eine serielle Form von geordneten Daten, die beim Empfänger entsprechend der dortigen Plattform und Programmiersprache wiederum in eine komplexe Datenstruktur übersetzt werden können. Die Serialisierungsregeln lassen sich nach Eigenschaften optimieren, die einander widersprechen. Für das IP-basierte Fahrzeugbordnetz sind folgende Eigenschaften relevant:

- Mächtigkeit, also der Umfang der Regeln, welche Datentypen, Datenstrukturen und höhere Schnittstellenkonzepte wie beispielsweise Vererbung durch das Marshalling umgesetzt werden können.
- Performanz, also Ausführungszeit für Marshalling und De-Marshalling auf den beteiligten Plattformen bei Sender und Empfänger.
- Wartbarkeit, also Toleranz gegenüber Schnittstellenänderungen, insbesondere Rückwärtskompatibilität gegenüber Kommunikationsteilnehmern, die eine ältere Version der Schnittstelle verwenden.

Die Entscheidung für ein Serialisierungsformat hat einen erheblichen Einfluss auf die Leistungsfähigkeit der Kommunikationsmiddleware [Henn 08]. Die beiden widersprüchlichen Aspekte sind die resultierende Größe der serialisierten Daten und der Aufwand für das Marshalling bzw. De-Marshalling. Eine kleinere Datenmenge spart Bandbreite, geringerer Aufwand spart im Wesentlichen Rechenleistungen auf den beteiligten Steuergeräten. Für jede Umgebung muss entsprechend ein Kompromiss gefunden werden.

Ein weiteres von den Industrieexperten genanntes Unterscheidungskriterium hängt an der Frage, ob die serialisierten Daten in einem binären oder menschenlesbaren Format vorliegen sollen. Ein menschenlesbares Format verwendet einen Zeichensatz, um eine bessere Wartbarkeit und einfache Erweiterbarkeit zu bieten. Dagegen optimiert ein binäres Format die Darstellung und führt zu kleineren Paketgrößen. Für Wartbarkeit und Diagnose können allerdings auch entsprechende Tools eingesetzt werden, die binär-kodierte Daten in eine für Menschen leicht verständliche Form übersetzen. Ein strukturiertes, menschenlesbares Format wie XML erzeugt deutlich größere Datenpakete. Daher ist es für eingebettete Plattformen im Fahrzeugbordnetz nicht geeignet. Die optimierten Steuergeräte benötigen möglichst kurze Datenpakete, welche die Speicherverwaltung mit ihren begrenzten Puffergrößen handhaben kann. Der

Kompromiss zwischen Größe und Übersetzungsaufwand entscheidet sich nicht mit der Bandbreite der Vernetzungstechnologie, sondern als Abwägung zwischen Rechenleistung und Speicherbedarf in den Puffern der Netzwerkschnittstelle. Auf diese Einschränkungen wird weiter unten im Abschnitt 4.3 über nicht-funktionale Anforderungen näher eingegangen.

Wie bereits in Abschnitt 3.4 beschrieben, existieren zahlreiche Lösungen für die Serialisierung, die prinzipiell für den Einsatz im IP-basierten Fahrzeugbordnetz infrage kommen. Die Serialisierungsschicht einer Kommunikationsmiddleware kann ausgetauscht werden, insofern eine Abbildung des in der Anwendungsschnittstelle beschriebenen Kommunikationsumfangs möglich ist.

Anforderung 15 Die Kommunikationsmiddleware muss ein Marshalling von plattformabhängigen Daten in ein binäres Serialisierungsformat zur Übertragung anbieten, sowie die entsprechende Umkehrfunktion auf der Plattform des Empfängers.

Zu übertragende Informationen liegen in der einfachen Kommunikation über CAN und FlexRay meist als numerische Werte oder boolesche Zustände vor. Damit werden gewöhnliche einfache Datentypen wie beispielsweise ganze Zahlen (Integer) und Fließkommazahlen (Float) genutzt. Selbst Steuerungskommunikation in MOST geschieht meist über solch einfache Datentypen. Es kommen allerdings Text und Listen hinzu, beispielsweise bei der Übertragung einer Liste von Musiktiteln von einem externen Datenträger. Zukünftige Anwendungen werden allerdings Anforderungen an komplexere Datenstrukturen stellen.

Zukünftige Fahrerassistenzsysteme und Systeme zum automatisierten Fahren benötigen ein Umfeldmodell, das relevante Faktoren aus der direkten Umgebung des Fahrzeugs geeignet repräsentiert. Dazu werden die Daten mehrerer Sensoren zu einem stimmigen Umfeldmodell fusioniert. Ein solches Umfeldmodell beschreibt Objekte, die durch Werte wie Position, Bewegungsrichtung und weitere Eigenschaften beschrieben sind. Dies entspricht Objekten, wie sie in der objektorientierten Softwareentwicklung modelliert werden. Diese Objekte müssen als komplexere Datenstrukturen mehreren Empfängern zur Verfügung gestellt werden. Dafür ist die Erfüllung von definierten Dienstgüteanforderungen notwendig, da diese Informationen durch die Fahrerassistenzsysteme potentiell für den aktiven Eingriff in die Fahraufgabe verwendet werden.

Anforderung 16 Die Kommunikationsmiddleware muss ein Marshalling für gewöhnliche einfache Datentypen und für komplexe Datenstrukturen, wie sie in der objektorientierten Softwareentwicklung als Objekte gekapselt werden, beherrschen.

4.2.4. Anforderungen an Systemdienste

Neben den Funktionen der Schnittstellenbeschreibung, der Kommunikationsparadigmen und des Marshallings kann weitere Kommunikationsfunktionalität

in Form von Systemdiensten angeboten werden. Jene kapseln Aufgaben. Zum einen solche Aufgaben, die für den Betrieb des gesamten Kommunikationssystems notwendig sind. Zum anderen solche, die mehreren Anwendungen nutzen, so dass sie nur einmal im Rahmen der Middleware und nicht mit jeder neuen Anwendung implementiert werden müssen.

Wie zu Beginn dieses Kapitels im Abschnitt über typische Anwendungen betrachtet, müssen für ein zukünftiges Fahrzeugbordnetz aus Gründen der Energieeffizienz Konzepte entwickelt werden, die Steuergeräte im Ruhezustand erlauben, aber eine schnelle Reaktionsfähigkeit im Bedarfsfall wiederherstellen. Ein solcher Teilnetzbetrieb benötigt eine Verwaltung von Diensten hinsichtlich Verfügbarkeit und Betriebsmodus. Zwar entsteht durch die Anbindung von CE-Geräten die Möglichkeit, dass Dienste dynamisch verfügbar werden können, die über einen Service-Discovery-Mechanismus gefunden und im Netzwerk bekannt gemacht werden müssen. Im Gegensatz zu einer Umgebung, in der die Menge der Kommunikationspartner prinzipiell unbestimmt ist, lassen sich die Anforderungen an ein Service-Management im Fahrzeug aber abgrenzen. Die wichtigste Aufgabe für ein Service-Management ist nicht das Auffinden von Diensten oder gar verschiedener Implementierungen derselben Dienstschnittstelle, sondern die Verwaltung der bekannten Dienste bezüglich ihres Status. Dies wird vor allem dann benötigt, wenn ein Teilnetzbetrieb möglich sein soll, der eine Untermenge der Steuergeräte und damit potentielle Kommunikationspartner schlafen legt.

Anforderung 17 Die Kommunikationsmiddleware muss einen Systemdienst für Service-Management bereitstellen, dessen Hauptaufgabe das Monitoring von Zuständen im Sinne von Verfügbarkeit der teilnehmenden Dienste ist.

Ein Teilnetzbetrieb-Management kann somit als weiterer Systemdienst betrachtet werden. Die Zunahme von Komfort- und Sicherheitsfunktionen im Fahrzeug erhöht den Verbrauch elektrischer Energie. Mehrere Gründe verlangen dagegen eine effiziente Nutzung. Dies sind unter anderem Energiesparen für den Umweltschutz, steigende Kosten für die Lichtmaschine (elektrischer Generator) sowie Verlängerung der Lebensdauer und Erhalt der Startfähigkeit von Batterien. Für ein Elektrofahrzeug (Battery Electric Vehicle) (BEV) wirkt sich der Energieverbrauch von Komfort- und Sicherheitsfunktionen sogar direkt auf die maximale Reichweite aus, die das Fahrzeug bis zum nächsten Ladezyklus zurücklegen kann. Für einen sparsamen Umgang mit elektrischer Energie im Fahrzeug können einzelne Steuergeräte oder eine Gruppe von vernetzten Steuergeräten (ein Teilnetz) abgeschaltet bzw. schlafen gelegt werden. Aufgabe eines Teilnetzbetrieb-Managements ist dann das kontrollierte Aufwecken und Einschlafen von Steuergeräten.

Anforderung 18 Die Kommunikationsmiddleware muss einen Systemdienst für Teilnetzbetrieb-Management bereitstellen, der prüft, ob ein Teilnetz-

betrieb zur Erhöhung der Energieeffizienz in einem gegebenen Systemzustand sicher möglich ist, und den Teilnetzbetrieb geeignet verwaltet.

In Lim, Weckemann und Herrscher [Lim 11c] wurden Dienstgüteanforderungen für ein Switched Ethernet zur IP-basierten Kommunikation im Fahrzeug evaluiert. Unter Dienstgüte (Quality of Service) versteht man die Vorgabe, für bestimmte Datenübertragungen innerhalb eines Rechnernetzes bestimmte Parameter wie eine zugesicherte Bandbreite zu garantieren. Mögliche Parameter sind Bandbreite, Verzögerung, Jitter und Fehlerrate [Craw 98]. Heute ist es üblich, die Gewährleistung von Bandbreite rein durch eine geschickte Dimensionierung des Rechnernetzes und Anpassung der Topologie zu erreichen. Dies widerspricht aber der Flexibilisierung, die durch die Einführung des Internet Protocols für die fahrzeuginterne Kommunikation erreicht wird. Eine typische IP-basierte Vernetzungstechnologie wie Ethernet wird für verschiedene Arten von Daten genutzt werden, sowohl für entfernte Steuerung als auch für Multimedia-Daten. Wenn die zugrundeliegende Vernetzung für den Anwender transparent gehalten werden soll, dann muss es eine Möglichkeit geben, die Dienstgüte für eine Kommunikationsverbindung zu kennzeichnen. Ein Dienstgüte-Management muss Möglichkeiten für eine solche Kennzeichnung bieten und entsprechende Funktionen für die Umsetzung auf die konkrete Vernetzungstechnologie implementieren.

Anforderung 19 Die Kommunikationsmiddleware muss einen Systemdienst für das Dienstgüte-Management bereitstellen.

Um unerlaubtes Mitlesen oder Manipulieren von Informationen zu verhindern, muss die Kommunikation geschützt werden. Da sich der deutsche Begriff „Sicherheit“ im Fahrzeugkontext meistens auf die Sicherheit der Passagiere (im Englischen „Safety“) bezieht, wird im Folgenden der englische Begriff „Security“ verwendet. Durch die Einführung des Internet Protocols in das Fahrzeugbordnetz werden Security-Aspekte bedeutender. So ermöglicht die Öffnung der Fahrzeugkommunikation nach außen neue Angriffsszenarien. Diese Öffnung geschieht durch die Fähigkeit des Fahrzeugs, über zellularen Mobilfunk ins Internet zu gelangen oder mitgebrachte CE-Geräte anzubinden. Im ersten Fall helfen sich die Fahrzeughersteller, indem sie private anstatt öffentlicher Zugangspunkte wählen und damit die Verbindung ins Fahrzeug über ein eigenes Backend sichern, dafür aber die Funktionalität einschränken. Für die CE-Geräte-Anbindung besteht in der Regel eine API mit demselben Effekt. Die eingeschränkte Schnittstelle bietet ein hohes Maß an Kontrolle bei eingeschränkter Funktionalität. Sobald die Kundenerwartungen steigen, wird der Fahrzeugzugangspunkt aber direkt mit den Security-Herausforderungen konfrontiert werden, die vergleichbar für einen Router als Zugangspunkt eines Heimnetzwerks bestehen.

Auch im heutigen Fahrzeugbordnetz sind Auslesen und Manipulieren von Daten im Fahrzeugbordnetz möglich. Die wesentlichen Änderungen in einem

IP-basierten Fahrzeugbordnetz sind neben dem einfacher erreichbaren Zugang auch die ungleich größere Anzahl an Personen, die ein ausreichendes Fachwissen über die Technologie haben. Allerdings betreffen die Fragen nach Security nicht nur die interne Kommunikation. Notwendig für ein sicheres Gesamtsystem ist auch eine sichere Datenhaltung auf den Steuergeräten, Updates für Steuergeräte-Software durch Einspielen von Security-Patches und so weiter. Dafür wird ein generelles Security-Framework benötigt. Dabei wird sich die Betrachtung darauf fokussieren, wie die Kooperation zwischen der Kommunikationsmiddleware und einem solchen Security-Framework erreicht werden kann.

Anforderung 20 Die Kommunikationsmiddleware muss einen Systemdienst für ein Security-Management bereitstellen, das einen flexiblen Schutz durch ein Security-Framework für das IP-basierte Fahrzeugbordnetz integriert.

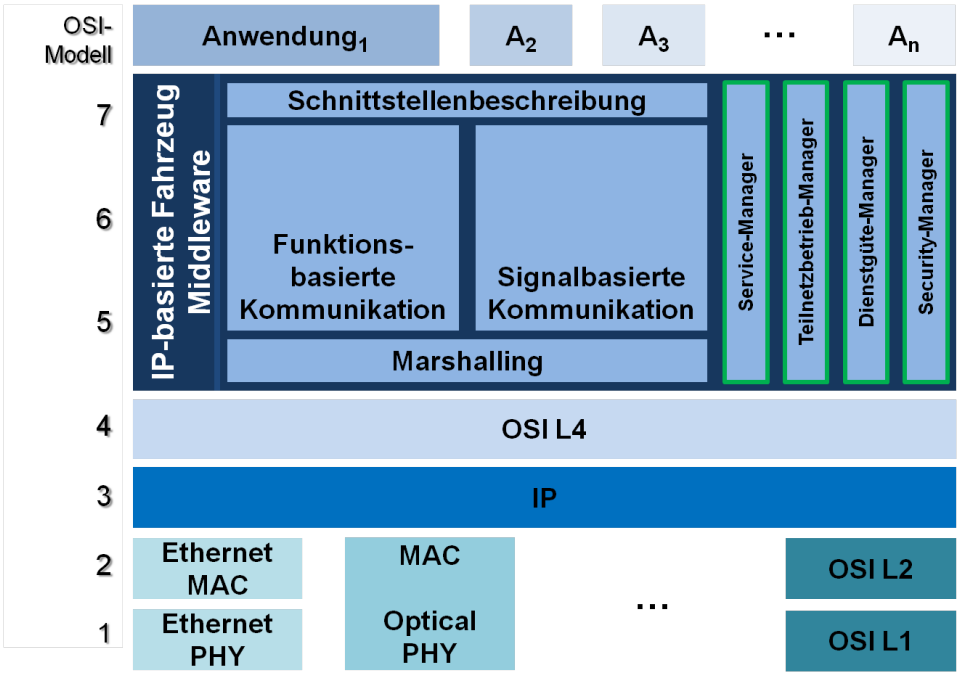

Abbildung 4.4.: Vier identifizierte Systemdienste ergänzen die IP-basierte Kommunikationsmiddleware.

Abbildung 4.4 zeigt die vier identifizierten Systemdienste. Auf die Systemdienste selbst und auch die Zusammenhänge zwischen Kommunikationsmiddleware und Security-Framework wird in Kapitel 6 eingegangen.

4.3. Nicht-funktionale Anforderungen in einem eingeschränkten, verteilten System

Nicht jede Anwendung im Fahrzeug hat dieselben Anforderungen an die Kommunikationsmiddleware. Vor allem Anwendungen auf leistungsfähigen Steuergeräten und hohem Funktionsumfang arbeiten auf komplexen und dynamischen Datenstrukturen und benötigen dadurch mächtigere Kommunikationsmechanismen. Dagegen gibt es viele leistungsschwache Steuergeräte mit Anwendungen, die einfache funktionale Anforderungen an die Kommunikation stellen. Die zunehmende Kommunikation zwischen Anwendungsdomänen zwingt diese unterschiedlichen Anwendungen zur Kommunikation, die im IP-basierten Fahrzeugbordnetz direkt ohne ein höherschichtiges Koppelelement erfolgen soll. Die Herausforderung liegt dabei in einer Skalierung bezüglich der verfügbaren Leistungsfähigkeit der Steuergeräte.

Anforderung 21 Die Kommunikationsmiddleware muss Interdomänenkommunikation zwischen Steuergeräten unterschiedlicher Leistungsfähigkeit ermöglichen.

Anwendungen und Steuergeräte werden unabhängig voneinander in evolutionären Schritten erweitert. Eine konzertierte Evolution des gesamten Steuergeräte-Verbunds ist aufgrund der unterschiedlichen Entwicklungszyklen von Steuergeräten und Fahrzeugderivaten aber nicht möglich. Bei einem Evolutionsschritt eines Steuergeräts steigen mit der Erhöhung der Funktionalität der Anwendung in der Regel auch die Kommunikationsanforderungen, während die bisherigen Kommunikationsverbindungen im Sinne einer Rückwärtskompatibilität unverändert weiter bestehen müssen. Bei einem Umzug einer Anwendung auf ein leistungsstärkeres Steuergerät darf die Kommunikationsfunktionalität erweitert werden, bestehende Funktionalität muss aber weiterhin gegeben sein.

Anforderung 22 Die Kommunikationsmiddleware berücksichtigt die Anwendungsmigration auf leistungsfähigere Steuergeräte unter Beibehaltung der Kompatibilität.

In heutigen Fahrzeugen wird eine Vielzahl verschiedener Steuergeräte mit unterschiedlicher CPU-Leistungsfähigkeit und Arbeitsspeicher verwendet. Der Bedarf an Kommunikation reicht von einer ECU auf einer 32-bit Plattform mit hoher Prozessortaktung mit einer leistungsschwächeren ECU auf einer 16-bit Plattform oder gar einem 8-bit Sensor mit einer entsprechenden Varianz an Arbeitsspeicher und permanentem Speicher. Wie in Abschnitt 4.1 über Anwendungen beschrieben, ist das maßgebliche Kriterium für die Anbindung eines Steuergeräts an das IP-basierte Fahrzeugbordnetz nämlich nicht die Rechenleistung, sondern der Bedarf an hoher Übertragungsbandbreite. Der Anwendungsfall einer Fahrerassistenzkamera zeigt deutlich, wie eine einfache Anwendung, die im Wesentlichen der Manipulation von Einstellungen dient, aufgrund

des Sendens eines Video-Stroms für die IP-basierte Kommunikation geeignet ist. Die Kommunikationsmiddleware muss allerdings auf allen Steuergeräten, die über das IP-basierte Fahrzeugbordnetz kommunizieren, eingesetzt werden, um den durch das Internet Protocol erhaltenen Konvergenzvorteil auszunutzen.

Anforderung 23 Eine Kommunikationsmiddleware muss auf allen Steuergeräten des IP-basierten Fahrzeugbordnetzes lauffähig sein.

Es gibt zudem Argumente für einen möglichst umfassenden Einsatz einer einzigen IP-basierten Kommunikationsmiddleware auf verschiedenen Ebenen.

1. Für einen Fahrzeughersteller ermöglicht eine übergreifende Lösung für das IP-basierte Fahrzeugbordnetz den Konvergenz-Vorteil, der erst durch die Einführung des Internet Protocols erreicht wird.

2. Für mehrere Fahrzeughersteller verringert eine gemeinsame, standardisierte Lösung die individuellen Infrastrukturkosten durch die gemeinsame Nutzung von nicht-wettbewerbsdifferenzierenden Technologien.

Anforderung 24 Eine Kommunikationsmiddleware soll bestehende Praktiken in der Automobilindustrie berücksichtigen, um als standardisierte Lösung herstellerübergreifend eingesetzt werden zu können.

4.4. Widersprüche in und Folgerungen aus den Anforderungen

Die funktionalen Anforderungen verlangen die Unterstützung zweier Kommunikationsparadigmen: Ein funktionsbasiertes Kommunikationsparadigma für die entfernte Steuerung von Anwendungen und gezielte Abfrage von Daten sowie ein signalbasiertes Kommunikationsparadigma, um einfache Daten zu verteilen. Außerdem werden mehrere Systemdienste für die Kommunikation gefordert. Die nicht-funktionalen Anforderungen verlangen die Lauffähigkeit der Kommunikationsmiddleware-Implementierung auf leistungsschwachen Steuergeräten und die Durchgängigkeit der Kommunikation zwischen zwei beliebigen Kommunikationspartnern im IP-basierten Fahrzeugbordnetz.

Die Schwierigkeiten von leistungsschwachen Steuergeräten bzgl. ihrer Kommunikationsfähigkeiten sind im Folgenden aufgeführt:

- Das dynamische Verwalten einer Notifizierungsliste und von Diensttabellen macht es schwierig, die maximal benötigte Größe des Speichers zur Entwicklungszeit abzuschätzen. Dem ließe sich durch einen größeren Speichervorhalt entgegnen, was jedoch den allgemeinen Entwicklungszielen einer Herstellkosten-getriebenen Entwicklung von eingebetteten Plattformen widerspricht.

- Die Übertragung von großen und dynamischen Datenstrukturen (z.B. Listen) erfordert auch dynamische Speicherallokierung. Die meisten leistungsschwachen, eingebetteten Plattformen setzen aber auf eine strikt statische Speicherallokierung.
- Eine Aufrufsemantik wie *at-most-once* oder *at-least-once* benötigt entweder ein verbindungsorientiertes Transportprotokoll wie TCP, das die einmalige Übertragung eines Datenpakets bereits sicherstellt, oder die Kommunikationsmiddleware muss ein entsprechendes Session-Handling implementieren. Beide Lösungen benötigen zusätzlichen Speicher und ein Betriebssystem, das Multi-Threading unterstützt.
- Ein Security-Framework (z.B. für Authentifizierung und Verschlüsselung) benötigt ebenfalls signifikant mehr Speicher und Rechenleistung. Die sichere Speicherung von Schlüsseln benötigt sichere Hardware.

Eine Implementierung einer Kommunikationsmiddleware, die alle funktionalen Anforderungen vollumfänglich erfüllt, kann auf den leistungsschwachen Steuergeräten im Fahrzeug nicht lauffähig sein. Mit der Anforderung nach einer einzigen Kommunikationsmiddleware für alle Steuergeräte des IP-basierten Fahrzeugbordnetzes muss eine skalierbare Lösung gefunden werden. Dies kann nicht einfach durch Einschränkung der Funktionalität einer Anwendung auf den tatsächlich benötigten Funktionsumfang geschehen. Zum einen sind aufgrund der starken Heterogenität in den Leistungscharakteristika der Plattformen strukturelle Änderungen notwendig, wie ein und dasselbe Kommunikationsparadigma Plattform-angepasst umgesetzt werden kann. Zum anderen ist die Verwaltung der Komplexität, die durch pro Anwendung optimierte Implementierungen entstehen würde, für das Industrieumfeld mit der hohen Anzahl an Varianten und Derivaten ungeeignet. Gesucht werden vielmehr Lösungen für klar umrissene Klassen von Steuergeräten und Anwendungen. Es sind also strukturelle Anpassungen an den funktionalen Bestandteilen der Kommunikationsmiddleware notwendig, ohne die Interoperabilität aufzugeben. Die Kommunikationsmiddleware muss in unterschiedlichen, interoperablen Ausprägungen verfügbar sein.

4.5. Beurteilung verfügbarer IP-basierter Middleware-Lösungen

In Abschnitt 3.4 wurden mit CORBA, the Internet Communications Engine (Ice) und Apache Etch drei verfügbare Lösungen mit einem adäquat erscheinendem Funktionsumfang vorgestellt.

Die Betrachtung zeigt, dass keine der Lösungen für den Einsatz in heterogenen, größtenteils aus eingebetteten Systemen bestehenden Umgebungen unmittelbar geeignet ist. Die verfügbaren Lösungen erfüllen die speziellen Anforderungen im Fahrzeugbordnetz nicht. Gerade die Heterogenität – sowohl

was die Anwendungssoftware als auch die sie beherbergenden, eingebetteten Plattformen angeht – wird von diesen Lösungen unzureichend adressiert. Ebenso wird dort, wo Interoperabilität gefordert wird, nicht auf den Zwang zur Einfachheit bei den dazwischen liegenden Koppelelementen geachtet. Gesucht ist eine Lösung, die ohne Anwendungsschicht-Gateway auskommt, das zwischen zwei heterogenen Plattformen übersetzt.

Somit bietet sich keine existierende Lösung uneingeschränkt für den Einsatz im Fahrzeugbordnetz an. Eine Software-Entwicklung mit dem Ziel einer einsetzbaren Lösung ist aber auch nicht beabsichtigt. Das vorgeschlagene Konzept zeigt Prinzipien auf, die unabhängig einer bestimmten Lösung sind und damit eine fundierte Grundlage für die anforderungsgerechte Software-Entwicklung einer Kommunikationsmiddleware für das IP-basierte Fahrzeugbordnetz bieten.

4.6. Zusammenfassung

Aus bestehenden und zukünftigen Anwendungen, der Kommunikation im heutigen Fahrzeugbordnetz und in Expertengesprächen wurden die Anforderungen an eine Kommunikationsmiddleware für ein IP-basiertes Fahrzeugbordnetz formuliert.

Viele Anwendungen benötigen aktuelle Fahrwerte. Die aktuelle Fahrgeschwindigkeit muss beispielsweise dem Cabrioverdeck-Modul mitgeteilt werden, wenn vorgeschrieben ist, dass das Verdeck nur bis zu einer festgelegten Fahrtgeschwindigkeit geöffnet werden darf. Komplexere Anwendungsfälle wie eine Kamerabild-gestützte Einparkhilfe benötigen Fahrerassistenzkameras, die auf einer leistungsschwachen Plattform einfache Steuerungsbefehle verarbeiten müssen, während auf derselben Vernetzungstechnologie über ein standardisiertes Protokoll ein Video-Strom übertragen wird.

Aus Anwendungen und einer Analyse der heutigen Bordnetzdatenbank wird gefolgert, dass die Kommunikationsmiddleware über ein funktionsbasiertes und ein signalbasiertes Kommunikationsparadigma verfügen muss. Vor allem aus Gesprächen mit Experten bei Herstellern und Zulieferern stammen die Anforderungen an eine Schnittstellenbeschreibungssprache mit definierter Syntax und Semantik, die bestehende Definitionkonzepte wie heute beispielsweise bei CAN und MOST beachten muss. Die Plattform-unabhängige Kommunikation wird durch ein Marshalling erreicht, das einfache Datentypen und komplexe Datenstrukturen wie Objekte aus einer objektorientierten Softwareentwicklung einheitlich serialisiert bzw. de-serialisiert. Dafür wird ein binäres Serialisierungsformat bevorzugt, um kleine Nutzdatengrößen zur Berücksichtigung eingeschränkter Speicherressourcen und eine gute Performanz bei der Übersetzung in den Steuergeräten zu ermöglichen. Schnittstellenbeschreibung, Koordinationsmodell und Marshalling stellen dabei die Basisfunktionalität der Kommunikationsmiddleware für die Steuerung entfernter Aufrufe und für die Informationsverteilung und -abfrage dar.

Daneben ermöglichen weitere Systemdienste, konzertierte Kommunikationsaufgaben für das Gesamtsystem zu erbringen oder wiederverwendbare Funktionen für einzelne Anwendungen zu kapseln. Ein Service-Management muss ein Monitoring von Zuständen im Sinne von Verfügbarkeit der teilnehmenden Dienste ermöglichen. Dies vereinfacht die Verwaltung von Variantenvielfalt bei der Fahrzeugproduktion. Außerdem erlaubt ein Service-Management eine dynamische Dienstnutzung bei der Anbindung externer CE-Geräte oder für Backend-Anwendungen. Es ermöglicht zudem einen Teilnetzbetrieb, bei dem zur Steigerung der Energieeffizienz ausgewählte Steuergeräte in einen Ruhezustand versetzt werden können. Für die eigentliche Verwaltung eines Teilnetzbetriebs muss ein weiterer Systemdienst als Teilnetzbetrieb-Management bereitstehen, der definierte Randbedingungen prüft und das Schlafenlegen und Aufwecken verwaltet. Weitere Systemdienste bieten Anwendungen die Möglichkeit, für Kommunikationsbeziehungen bestimmte Dienstgüte- oder Sicherheitskriterien einzurichten. Die entsprechenden Systemdienste werden als Dienstgüte bzw. Security-Management bezeichnet. In beiden Fällen gibt es für IP-basierte Kommunikationsnetze zahlreiche existierende Protokolle. Die Kommunikationsmiddleware muss die flexible Nutzung von entsprechenden Frameworks integrieren.

Bei der Betrachtung nicht-funktionaler Anforderungen vor allem hinsichtlich der Leistungsfähigkeit potentieller Steuergeräte zeigt sich, dass nicht alle Steuergeräte alle funktionalen Anforderungen umsetzen können. Es gibt Anwendungsbeispiele, in denen auch kleine, eingebettete Plattformen mit einem leistungsschwachen, 8-Bit-basierten Prozessor und nur wenig Arbeits- bzw. Programmspeicher potentielle Kandidaten für eine IP-basierte Kommunikation sind. Eine Kommunikationsmiddleware muss auch diese Klasse leistungsschwacher Steuergeräte unterstützen.

Vor allem ergibt sich durch die Prognose einer zunehmenden Kommunikation über Anwendungsdomänen hinweg die Anforderung, Kommunikation zwischen allen Anwendungen des IP-basierten Kommunikationsbordnetzes unabhängig der heterogenen Leistungsfähigkeit der beherbergenden Steuergeräte zu gewährleisten. Die Kommunikationsmiddleware muss also auf allen Steuergeräten des IP-basierten Fahrzeugbordnetzes – also insbesondere auf den Leistungsschwächsten – lauffähig sein. Eine Implementierung, die alle funktionalen Anforderungen vollumfänglich erfüllt, ist für die nicht-funktionalen Anforderungen ungeeignet. Vor allem ein dynamisches Verwalten von Kommunikationspartnern und das Übertragen von dynamischen Datenstrukturen ist auf eingebetteten Plattformen, die in der Regel auf statisch vorgegebene Strukturen optimiert sind, nicht umsetzbar.

Die Anforderung nach einer einzigen Kommunikationsmiddleware für alle Steuergeräte des IP-basierten Fahrzeugbordnetzes verlangt demnach eine skalierbare Lösung. Eine einfache Einschränkung der Funktionalität einer Anwendung auf den tatsächlich von ihr benötigten Funktionsumfang ist dabei nicht zielführend. Zum einen sind aufgrund der starken Heterogenität in den Leis-

tungscharakteristika der Plattformen strukturelle Änderungen notwendig, wie ein und dasselbe Kommunikationsparadigma Plattform-angepasst umgesetzt werden kann. Zum anderen verursachen pro Anwendung optimierte Implementierungen eine hohe Anzahl von Software-Varianten. Die Verwaltung dieser Komplexität ist für das Industrieumfeld mit ihren vielen unterschiedlichen Fahrzeugderivaten ungünstig. Es sind daher strukturelle Anpassungen an den funktionalen Bestandteilen der Kommunikationsmiddleware notwendig, ohne die Interoperabilität aufzugeben.

Keine existierende Middleware-Lösung bietet sich uneingeschränkt für den Einsatz im Fahrzeugbordnetz an, wobei einige selbstverständlich als Basis für eine Weiterentwicklung genutzt werden könnten. Das vorgeschlagene Konzept soll auch keine komplett entwickelte Lösung zum Ziel haben. Es sollen Prinzipien aufgezeigt werden, die unabhängig von einer bestimmten Lösung sind und damit die Grundlage für eine anforderungsgerechte Entwicklung einer Kommunikationsmiddleware für das IP-basierte Fahrzeugbordnetz bilden.

Im folgenden Kapitel wird zunächst vorgeschlagen, wie sich die einzelnen Bestandteile einer Kommunikationsmiddleware für das IP-basierte Fahrzeugbordnetz umsetzen lassen und wie ein modulares Konzept, die identifizierten Widersprüche in den funktionalen und nicht-funktionalen Anforderungen auflösen kann. Die Kommunikationsmiddleware muss in unterschiedlichen, interoperablen Ausprägungen verfügbar sein.

5. Eine skalierbare Kommunikationsmiddleware für das Fahrzeugbordnetz

In diesem Kapitel wird eine Kommunikationsmiddleware vorgestellt, die den Anforderungen in einem IP-basierten Fahrzeugbordnetz entspricht. Dafür müssen einerseits die Anforderungen an die funktionalen Bestandteile einer Kommunikationsmiddleware umgesetzt werden. Andererseits muss das Konzept die Herausforderungen lösen, die besonders durch die nicht-funktionalen Anforderungen entstehen.

Ein modulares Konzept adressiert die identifizierten Widersprüche in den funktionalen und nicht-funktionalen Anforderungen, die durch die Heterogenität der Steuergeräte vor allem bezüglich ihrer Leistungsfähigkeit besteht. Die Kommunikationsmiddleware muss in unterschiedlichen, interoperablen Ausprägungen verfügbar sein. Jene müssen die jeweiligen Anforderungen und die Leistungsfähigkeit berücksichtigen. Gleichzeitig muss jede Anwendung im IP-basierten Fahrzeugbordnetz mit jeder anderen kommunizieren können, ohne auf ein Koppelelement mit Anwendungswissen zurückgreifen zu müssen. Ein besonderes Augenmerk liegt dabei auf der kleinsten Ausprägung der entworfenen Kommunikationsmiddleware. Anhand einer prototypischen Implementierung und Evaluation wird die prinzipielle Tragfähigkeit des Konzepts validiert.

5.1. Die Umsetzung des Koordinationsmodells

Das Koordinationsmodell beschreibt das dynamische Verhalten der Middleware für die vermittelte Kommunikation. Es kommt dabei mindestens ein Kommunikationsparadigma zur Anwendung. Diese Kommunikationsparadigmen folgen einer definierten Semantik und stellen somit das eigentliche Protokoll dar, nach dem die Middleware Kommunikation herstellt. Mit den im vorherigen Kapitel identifizierten Anforderungen lassen sich die in Abschnitt 2.4 beschriebenen Klassen für Kommunikationsmiddleware nun näher untersuchen.

Für transaktionale Middleware und Tuplespace-basierte Middleware gibt es keine Beispiele im heutigen Fahrzeugbordnetz. Die Konsistenz zwischen aktuellen Werten bei verschiedenen Kommunikationspartnern, die transaktionale Middleware herstellt, gehört nicht zu den geforderten Eigenschaften. Der Overhead in der Kommunikation, der durch den hohen Aufwand dieses Paradigmas

entsteht, lässt sich damit für die interne Fahrzeugkommunikation nicht rechtfertigen. Gegebenenfalls würde dies bei einer Kommunikation zwischen einem Fahrzeug mit der Verkehrsinfrastruktur oder zwischen Fahrzeugen (Car2X) zur Verbesserung der aktiven Sicherheit relevant. Dies liegt aber außerhalb des Fokus der vorliegenden Arbeit.

Tuplespace-basierte Middleware ist für die Weitergabe von Daten an mehrere Empfänger gut geeignet. Anhand von Untersuchungen mit Data Distribution Service (DDS) [OMG 07] wurden die Möglichkeiten einer Tuplespace-basierten Middleware betrachtet. Dabei zeigt sich schnell, dass sich das hohe Abstraktionsniveau schlecht für die funktionsbasierte Kommunikation und für die Diagnose von Steuergeräten eignet.

Nachrichtenorientierte Middleware entspricht prinzipiell der einfachen signalbasierten Kommunikation, wie sie beispielsweise via CAN erfolgt. Allerdings fehlen nachrichtenorientierter Middleware mit der fehlenden Sicherung synchroner Kommunikation und der einheitlichen Handhabung heterogener Daten funktionale Umfänge wie beispielsweise die Unterstützung eines funktionsbasierten Kommunikationsparadigmas und die Garantie von Aufrufsemantiken.

Ein RPC-Framework bietet einfache, aber gut strukturierte Kommunikation. Es ermöglicht entfernte Aufrufe in Form von parametrisierten Funktionen und definierten Aufrufsemantiken. Objekt- und komponentenorientierte Middleware sind die logischen Weiterentwicklungen des RPC-Frameworks. Die Software-Entwicklung in der Fahrzeugindustrie ist aber noch stark durch eingebettete Systeme geprägt. Dies wird sich aufgrund von Stabilität- und Safety-Anforderungen in naher Zukunft auch nicht ändern. So könnte Objekt- und Komponenten-orientierte Middleware nur eine isolierte Lösung für mächtige Anwendungen sein, aber keine Lösung darstellen, die für das gesamte IP-basierte Fahrzeugbordnetz die Heterogenität abstrahiert. Service-orientierte Middleware als nächste Evolutionsstufe übersteigt mit der einhergehenden Komplexität die Möglichkeiten im Fahrzeugbordnetz beträchtlich.

In Kapitel 4 wurde vom Koordinationsmodell gefordert, sowohl ein signalbasiertes, als auch ein funktionsbasiertes Kommunikationsparadigma umzusetzen. Beide Kommunikationsparadigmen müssen durch geeignete Kommunikationsmechanismen umgesetzt werden. Ein RPC-Framework, das auch signalbasierte Kommunikation wie in einer Nachrichten-orientierten Middleware und Systemdienste für allgemeine Kommunikationsfunktionen unterstützt, erfüllt diese Anforderungen. Die folgenden Abschnitte erläutern, wie diese beiden Kommunikationsparadigmen geeignet umgesetzt werden können und wie damit auch ein minimales Koordinationsmodell konstruiert werden kann.

5.1.1. Funktionsbasiertes Kommunikationsparadigma

Die geforderte funktionsbasierte Kommunikation muss Anfragen für ausgewählte Informationen und die Steuerung entfernter Anwendungen ermöglichen.

RPCs sind in der klassischen Definition nach Birrell [Birr 84] synchrone Interaktionen zwischen genau einem Client und einem Server. Moderne Lösungen (wie beispielsweise Apache Etch, siehe Abschnitt 3.4) bieten zusätzlich die Möglichkeit für asynchrone Aufrufe. Für dieses weit verbreitete Konzept existieren viele unterschiedliche Implementierungen, die sich in Zielsetzung, Aufrufsemantiken und weiteren Eigenschaften unterscheiden [Tay 90]. Man unterscheidet unter anderem zwischen synchronen und asynchronen Funktionsaufrufen [Anan 92]. Synchrone Funktionsaufrufe unterbrechen den lokalen Prozess und warten, bis die Verarbeitung der aufgerufenen Funktion abgeschlossen ist. Nach asynchronen Funktionsaufrufen kann der lokale Prozess weiterlaufen. Die Antwort der asynchronen Funktion wird zwischengespeichert und kann später abgeholt werden. Dies kann einfach dadurch erreicht werden, indem der Client aus Middleware-Sicht über Server-Funktionalität verfügt. So kann der Server die Antwort analog zu einem Aufruf in die entgegengesetzte Richtung senden. Ein RPC ist eine komplexe Interaktion, die eine Kommunikationsmiddleware zur Verfügung stellen kann. Dies entspricht in etwa der Abstraktion eines heutigen MOST-Funktionsaufrufs.

Die Formate eines entfernten Funktionsaufrufs und gegebenenfalls der Antwort darauf werden in einer IDL definiert. Werkzeuge erzeugen aus den beschriebenen Schnittstellen direkt einen Code-Rahmen mit Funktionsrümpfen, welche als *Stubs* und *Skeletons* bezeichnet werden. Ein RPC-Framework kann damit beliebige Programmiersprachen und Plattformen bedienen. Es muss lediglich ein entsprechender Compiler für die Zielplattform verfügbar sein, um die Definition zur Entwurfszeit in die Zielprogrammiersprache zu übersetzen. Eine Laufzeitkomponente der Kommunikationsmiddleware ist für die Funktionalität des entfernten Funktionsaufrufs verantwortlich. Dies beinhaltet das Marshalling, bei dem die Aufrufparameter in ein serielles Format übersetzt werden. Beim Marshalling der Parameter werden die plattform- und sprachspezifischen Anwendungsdaten in ein plattformunabhängiges Serialisierungsformat gebracht. Die Übersetzung zur Laufzeit muss ebenfalls für alle Zielplattformen implementiert sein.

Dieses funktionsbasierte Kommunikationsparadigma muss definierte Aufrufsemantiken bieten. Aufrufsemantiken definieren Garantien über die Häufigkeit eines einzelnen Aufrufs im Fall eines Fehlers bei der Nachrichtenübertragung. Es wird zwischen den Semantiken *maybe*, *at-least-once* und *at-most-once* unterschieden, wie sie beispielsweise in Tay et al. [Tay 90] aufgeführt sind. Eine Implementierung dieser Semantiken benötigt Timeouts und das Verwalten von Listen, mit Identifikatoren von versendeten Nachrichten und erhaltenen Bestätigungen. Die Entscheidung für höherwertige Semantiken, die über die beliebige Übertragung einer *maybe*-Semantik hinausgehen, hängt auch davon ab, ob das Kommunikationsparadigma synchron oder asynchron zwischen den beteiligten Kommunikationspartnern erbracht wird. Die asynchrone Übertragung bei gleichzeitiger Garantie einer höherwertigen Aufrufsemantik verlangt, dass die Kommunikationsprozesse multi-threaded betrieben werden [Anan 92].

Durch die Entsprechung zur Steuerungskommunikation in MOST und der in CAN praktisch nachgewiesenen Verwendung von gezielten Informationsanfragen, eignet sich der Remote Procedure Call für die Umsetzung des funktionsbasierten Kommunikationsparadigmas.

5.1.2. Signalbasiertes Kommunikationsparadigma

Die Anforderungen an das Koordinationsmodell richten sich weiterhin an ein signalbasiertes Kommunikationsparadigma, das ein Verteilen von Informationen zyklisch oder bei definierten Ereignissen abbildet.

Gemäß den Betrachtungen der relevanten Anwendungen dient CAN oftmals dem reinen Verteilen von Informationen, beispielsweise dem Verteilen der aktuellen Fahrtgeschwindigkeit an viele Empfänger. Allerdings entspricht das Verteilen von kleinen Datenblöcken auch den Einschränkungen von CAN mit seiner maximalen Nutzdatengröße von 8 Byte. Eine typische IP-basierte Vernetzungstechnologie wie Ethernet weist im Vergleich zu CAN eine deutlich höhere Nutzdatengröße und Bandbreite auf. Bevor eine gute Umsetzung der signalbasierten Kommunikation vorgeschlagen werden kann, müssen Fragen nach einer geeigneten Abbildung des Konzepts zum Verteilen von Informationen beantwortet werden, die diese Unterschiede in den verwendeten Vernetzungstechnologien geeignet berücksichtigen.

In einem IP-basierten Fahrzeugbordnetz kann nicht allgemein davon ausgegangen werden, dass die zugrundeliegende Vernetzungstechnologie eine einfache Verteilung von Informationen (wie bei CAN) an viele potentielle Empfänger bereits impliziert. Die einfachste Möglichkeit, die prinzipiell die Kommunikation in CAN nachbildet, ist das Versenden von Broadcasts an alle Empfänger. Ethernet gilt exemplarisch als typische Vernetzungstechnologie in einem zukünftigen IP-basierten Fahrzeugbordnetz. Selbstverständlich ermöglicht auch eine Vernetzungstechnologie wie Ethernet Broadcasts. Jene würden allerdings den Vorteil verspielen, dass die Bandbreite von 100 MBit/s in einem Switched Ethernet für jede Verbindung zwischen einem Kommunikationspartner und einem Switch unabhängig zur Verfügung steht. Bei einem Broadcast würden auch alle Verbindungen belastet, für deren Empfänger die Informationen irrelevant sind. Broadcasts sind in einem Switched Ethernet daher ineffizient. Um keine Verbindungen unnötig zu belasten, bietet das Internet Protocol die Übertragung als Multicast mit einer festgelegten Empfängergruppe. Multicasting bezeichnet allgemein eine Übertragung von einem Sender zu einer festgelegten Gruppe von Empfängern. IP-Multicasts müssen dabei auf die zugrundeliegende Vernetzungstechnologie umgesetzt werden. Eine eingehende Betrachtung muss demnach auf dieser Schicht stattfinden. Dazu müssten in einem Switched Ethernet in den Koppelelementen (Switches) Multicast-Gruppen verwaltet werden. Dies müsste allerdings statisch zur Produktionszeit des Fahrzeugs geschehen und erfüllt nicht die Anforderungen an dynamische Gruppen von Empfängern, wie sie durch wechselnde Dienstzustände und einen Teil-

netzbetrieb in den Anforderungen 17 und 18 gefordert werden. Es muss also geklärt werden, inwieweit eine Kommunikationsmiddleware diese Punkt-zu-Multipunkt-Kommunikation abbilden kann.

Um darauf schließen zu können, wie eine geeignete Umsetzung des signalbasierten Kommunikationsparadigmas erfolgen muss, wurde analysiert, wie CAN-Telegramme in einem Switched Ethernet generell übertragen werden können. Daran lassen sich Optimierungspotentiale testen und die Anforderungen an die Kommunikationsmiddleware detaillieren. Die erste Untersuchung fragt, ob sich eine Aggregation von CAN-Nutzdaten in einem Ethernet-Rahmen lohnt. Dafür wurden reale Daten aus der Bordnetzdatenbank für den CAN der Body-Domäne in einem ausgewählten Fahrzeugderivat untersucht. Die Eigenschaften von CAN und Ethernet bezüglich Optionen bei der Übertragung und Nutzdatengröße unterscheiden sich deutlich. CAN spezifiziert eine einfache, anhand eines Identifikators priorisierte Übertragung, während Ethernet in seinem Protokoll-Header und -Tail neben den Adressen der Kommunikationspartner zahlreiche weitere Übertragungsoptionen erlaubt. Die Nutzdatengröße bei CAN ist maximal 8 Byte, bei Ethernet ist eine Übertragung von 40 Byte bis zu 1500 Byte in einem Frame möglich.

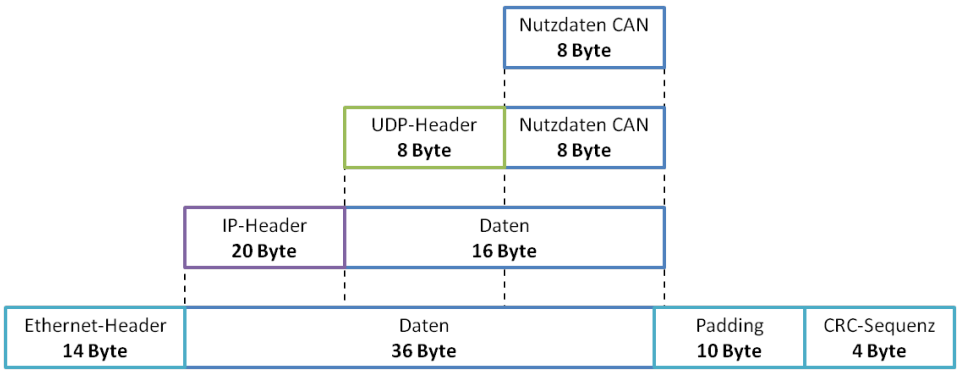

Abbildung 5.1.: Die maximale Nutzlast eines CAN-Frames in einem Ethernet-Frame.

Abbildung 5.1 zeigt, wie sich ein maximaler Datenblock aus einem CAN-Rahmen in einer IP-basierten Übertragung über Ethernet einordnen würde. Die Anwendung bzw. die Kommunikationsmiddleware würde den Datenblock an die Transportschicht geben. Der Einfachheit halber und der ungesicherten Übertragungseigenschaften in CAN entsprechend, wird UDP als Transportprotokoll verwendet. Dadurch werden mit dem UDP-Header die Ports der sendenden und empfangenden Anwendung mit 8 Byte hinzugefügt. Auf der nächsten Schicht im Protokollstack fügt IP auf der Vermittlungsschicht die IP-Adresse der sendenden und empfangenden Netzschnittstelle (oder eine Multicast-Adresse) mit weiteren 20 Byte hinzu. Ethernet-Header und -Tail benötigen schließlich zusammen 18 Byte. Da dabei aber die spezifizierte Mindestgröße eines Ethernet-Rahmens nicht erreicht wird, findet ein Auffüllen des

Rahmens (Padding) um weitere 10 Byte statt. Die 8 Byte Nutzdaten stehen damit 56 Byte Protokoll-Overhead gegenüber. Die Nutzdaten machen damit nur ein 1/8 des Ethernet-Rahmens aus. Zusammen mit der Prämisse, dass wenige große Nachrichten effizienter als viele kleine Nachrichten sind [Henn 08], ergibt sich die Fragestellung, ob eine Aggregation von CAN-Telegrammen im Falle einer Migration auf Ethernet-Frames möglich ist.

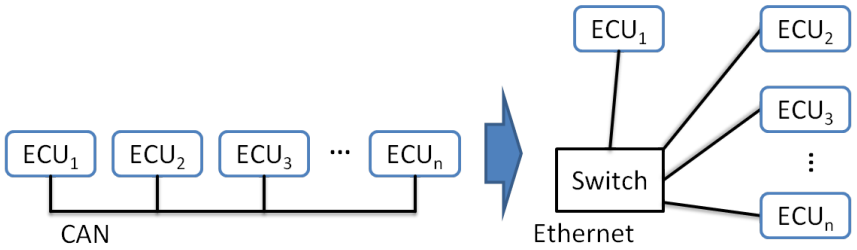

Abbildung 5.2.: Die Topologien für die Simulation einer Migration von CAN auf Ethernet.

Dafür wurde betrachtet, welche Steuergeräte mit welchen anderen Steuergeräten in einer Kommunikationsbeziehung stehen. Die nähere Betrachtung der Bordnetzdatenbank zeigt für zwei ausgewählte Steuergeräte, dass häufig mehrere verschiedene Nachrichten ausgetauscht werden. Diese werden teilweise zyklisch verteilt, teilweise basiert der Nachrichtenaustausch auf demselben auslösenden Ereignis. Dies ermöglicht die Aggregation kurzer CAN-Daten in einem einzigen Ethernet-Frame.

Dazu wurde eine Simulation für die Migration auf Ethernet durchgeführt. Als Basis für die Auswertung dient die einfache Migration, so dass jedes CAN-Telegramm mit unveränderter Zykluszeit in einem Ethernet-Frame übertragen wird. Zum Vergleich wird das Aggregationskonzept betrachtet, das Nachrichten mit ähnlicher Zykluszeit oder gleicher Sendebedingung in einem einzigen Ethernet-Frame sendet. Für die Simulation wird angenommen, dass alle CAN-Steuergeräte einfach durch Ethernet-Steuergeräte ersetzt werden. In der Praxis wird eine Migration zu einer teureren Vernetzungstechnologie wie Ethernet wahrscheinlich erst erfolgen, wenn die Aufgaben mehrerer Steuergeräte in einem leistungsfähigeren Steuergerät gebündelt werden. Dies wäre allerdings dem Prinzip der Effizienzsteigerung durch Aggregation sogar zuträglich. Da keine realistischen Annahmen über eine solche Bündelung gemacht werden konnten, fiel die Entscheidung auf eine einfache 1:1-Ersetzung.

Die Migration von einer Busvernetzung bei CAN zu Ethernet nimmt einen Switch als Koppelelement an, der die betrachteten Steuergeräte über eine Ethernet Punkt-zu-Punkt-Verbindung vernetzt, wie in Abbildung 5.2 schematisch dargestellt. Die Simulation wurde mit den beschriebenen Annahmen mit dem Framework für Rechnernetzsimulationen OMNeT++ [OMNe] durchgeführt.

Abbildung 5.3 zeigt folgendes: Der rote Balken zeigt die Datenrate von

7126 kbit/s am Switch für die Simulation ohne Verwendung eines Aggregationskonzept. Der blaue Balken zeigt die Datenrate von 3354 kbit/s unter Verwendung des beschriebenen Aggregationskonzepts. Damit kann grundsätzlich gezeigt werden, dass eine einfache Migration signalbasierter Kommunikation Potentiale zur effizienten Ausnutzung der verfügbaren Bandbreite verschenkt. Eine Aussage über die real möglichen Effizienzgewinne lässt sich aufgrund der vagen Annahmen an Migration und Verteilung der Steuergeräte nicht ableiten. Generell ist bei der Übertragung, auch unter Verwendung eines Aggregationskonzepts, die Datenrate aufgrund der großen Ethernet-Rahmen sehr hoch. Die Migration der in CAN üblichen Kommunikation durch häufiges Senden ohne Abhängigkeit zum tatsächlich vorhandenen Informationsbedarf ist damit nicht zielführend. Gesucht ist ein Kommunikationsmechanismus, der diesen Informationsbedarf bei der Umsetzung des signalbasierten Kommunikationsparadigmas berücksichtigt.

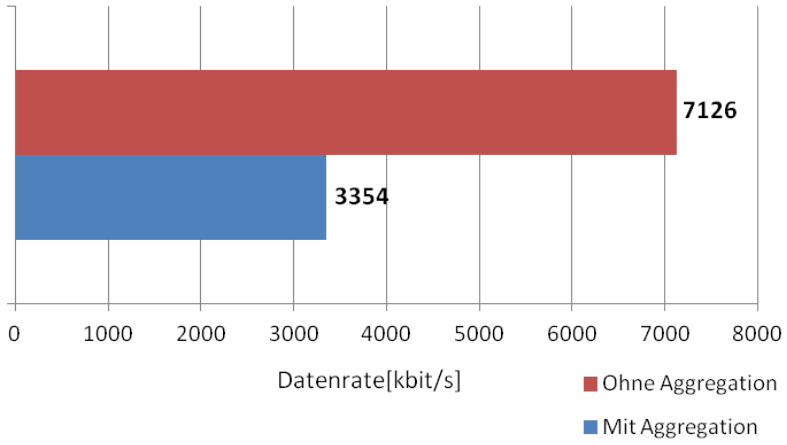

Abbildung 5.3.: Effizienzgewinn durch Aggregation mehrerer CAN-Nachrichten in einem Ethernet-Frame.

Als zweite Untersuchung, wie eine einfache Verteilung von Informationen an viele potentielle Empfänger in einem IP-basierten Fahrzeugbordnetz erreicht werden kann, richtet sich direkt nach dem Einsatz von Multicasts bzw. ob es sich lohnt, auf einer Vernetzungstechnologie wie Switched Ethernet Multicast-Gruppen zu verwalten.

Dazu wurde wiederum ein Auszug aus der Bordnetzdatenbank analysiert. Dies zeigt direkt, dass es in einem realen Nutzungsszenario auch auf CAN nur wenige Punkt-zu-Multipunkt-Nachrichten gibt. Abbildung 5.4 zeigt die Verteilung, wie viele der untersuchten Nachrichten relativ an wie viele Empfänger gehen. Knapp die Hälfte der Nachrichten wird dabei an genau einen Empfänger gesendet. Für Nachrichten mit wenigen Empfängern macht es aus Perspektive des notwendigen Kommunikationsaufwands keinen großen Unterschied, ob sie

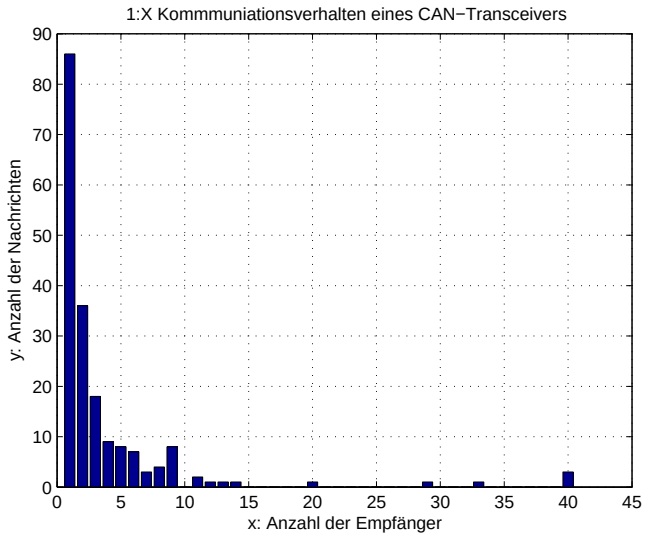

Abbildung 5.4.: Häufigkeiten von Punkt-zu-Multipunkt-Beziehungen auf Basis der Bordnetzdatenbank.

als Multicast oder als wenige Unicasts gesendet werden. Die Nachrichten mit 40 Empfängern stellen für diese Auswertung die maximale Anzahl an Empfängern dar und entsprechen auf diesem Datensatz einem Broadcast an alle Teilnehmer auf dem Bus.

Offenbar gibt es keinen umfangreichen Bedarf für die Verwaltung von Multicast-Gruppen. Dazu kommt, dass die Analyse sich auf den Betriebszustand eines fahrbereiten Fahrzeugs bezieht. Die Abhängigkeiten zum Betriebszustand, beispielsweise wenn das Fahrzeug gerade erst geöffnet, der Motor aber noch nicht gestartet wurde, verändern die Kommunikationsbeziehungen, weil nicht alle Steuergeräte in jedem Zustand aktiv sind. Dies kann über eine dynamische Veränderung der Empfängerliste behandelt werden. Eine solche Dynamik lässt sich besser in einem Konzept in der Kommunikationsmiddleware auf den einzelnen Steuergeräten als für die einfachen Koppelelemente des Rechnernetzes umsetzen. Statische Multicast-Gruppen auf Ebene der Vernetzungstechnologie lohnen sich für die Migration heutiger CAN-Kommunikation oder allgemeiner zur Umsetzung des signalbasierten Kommunikationsparadigmas nicht. Aufgrund der Variantenvielfalt von Ausstattungen und Fahrzeugderivaten benötigt eine signalbasierte Kommunikation auf einer typischen IP-basierten Vernetzungstechnologie wie Ethernet Möglichkeiten zur dynamischen Anpassung pro Fahrzeug.

Deshalb und aufgrund der Forderung nach einem Kommunikationsmechanismus, der den vorhandenen Informationsbedarf bei der Umsetzung des signalbasierten Kommunikationsparadigmas berücksichtigt, wurde die Entscheidung für ein Konzept zur dynamischen Anmeldung von Diensten für bestimmte Daten mit anschließenden Benachrichtigungen getroffen. Dafür muss noch die

Frage nach dem Umfang eines solchen Konzepts und dem Grad der benötigten Dynamik beantwortet werden.

Die Empfängerliste ändert sich zur Laufzeit aufgrund von Bordnetzzustand, Teilnetzbetrieb und hinzukommenden Kommunikationsteilnehmern wie CE-Geräten. Da eine Verwaltung von Multicast-Gruppen in den einfachen Koppelelementen des Fahrzeugbordnetzes ungeeignet ist, muss das signalbasierte Kommunikationsparadigma durch ein dynamisches Verwalten von Empfängern und mehrfacher Unicast-Übertragung durch die Kommunikationsmiddleware umgesetzt werden. Dazu muss eine dynamische Anmeldung von Diensten für definierte Daten möglich sein. Die Umsetzung des signalbasierten Kommunikationsparadigmas soll daher durch Notifizierung erfolgen, bei dem sich Kommunikationsteilnehmer dynamisch für ausgewählte Informationen an- und abmelden können. Die Literatur behandelt in diesem Zusammenhang oft einen Publish/Subscribe-Mechanismus [Tane 06]. Bei diesem gibt es allerdings keine fixe Kommunikationsbeziehung zwischen Sender und Empfänger. Der Sender übermittelt unbekannten Empfängern Nachrichten mittels inhaltlicher Adressierung, meist als Thema bezeichnet. Der Vorteil des Entkoppelns von Sender und Empfänger, der in einer besseren Skalierbarkeit und vereinfachten Dynamik bei der Netzwerktopologie resultiert, ist für den automotiven Einsatz unerheblich. Außerdem widerspricht ein Entkoppeln von Sender und Empfänger Anforderung 13, die definierte Aufrufsemantiken fordert.

Im Fahrzeug muss sich eine Anwendung mit ihrer Adresse als Empfänger (Abonnent) bei einem Sender (Notification Provider) anmelden. Es findet also ein Abonnieren einer bestimmten Nachricht – bei Eintreffen einer Änderungsbedingung oder mit einer definierten Zykluszeit – bei einer anderen Anwendung statt. Sender und Empfänger haben damit eine Kommunikationsbeziehung hergestellt. Dies soll unter dem Begriff Notifizierung behandelt werden. Da gebräuchlicher, wird im Folgenden meist synonym der englische Begriff *Notification* verwendet. Das verwendete Notification-Konzept beinhaltet damit das Anmelden für bestimmte Nachrichten bei einem Sender (Abonnement), das entsprechende Abmelden (Abonnement-Kündigung), sowie Vorschriften für Schnittstellen, die ein Abonnent implementieren muss, um abonnierte Notifications empfangen zu können.

Zusammenfassend kann das signalbasierte Kommunikationsparadigma in einem IP-basierten Fahrzeugbordnetz durch mehrfache Unicast-Übertragung durch die Kommunikationsmiddleware umgesetzt werden. Dazu muss eine dynamische Anmeldung von Diensten für definierte Daten möglich sein. Mit den Aussagen über eine Umsetzung der beiden identifizierten Kommunikationsparadigmen ist der funktionale Umfang für die eigentliche Kommunikation bestimmt. Im Folgenden muss nun beantwortet werden, wie sich dieser Umfang auf unterschiedlich leistungsfähigen Steuergeräten umsetzen lässt.

5.2. Skalierbarkeit durch binärkompatible Interoperabilität

Anforderungen 21 und 23 verlangen, dass Interdomänenkommunikation zwischen Steuergeräten unterschiedlicher Leistungsfähigkeit möglich und die Kommunikationsmiddleware auf allen Steuergeräten des IP-basierten Fahrzeugbordnetzes lauffähig ist.

Nur eine skalierbare Lösung kann die Anforderung nach einer einzigen Kommunikationsmiddleware für alle Steuergeräte des IP-basierten Fahrzeugbordnetzes lösen. Eine einfache Einschränkung der Funktionalität einer Anwendung auf den tatsächlich von ihr benötigten Funktionsumfang ist dabei nicht zielführend. Zum einen sind aufgrund der starken Heterogenität in den Leistungscharakteristika der Plattformen strukturelle Änderungen notwendig, wie ein und dasselbe Kommunikationsparadigma Plattform-angepasst umgesetzt werden kann. Zum anderen ist die Verwaltung der Komplexität, die durch pro Anwendung optimierte Implementierungen entstehen würde, für das Industrieumfeld mit der hohen Anzahl an Varianten und Derivaten ungeeignet. Es sind strukturelle Anpassungen an den funktionalen Bestandteilen der Kommunikationsmiddleware notwendig, ohne die Interoperabilität aufzugeben.

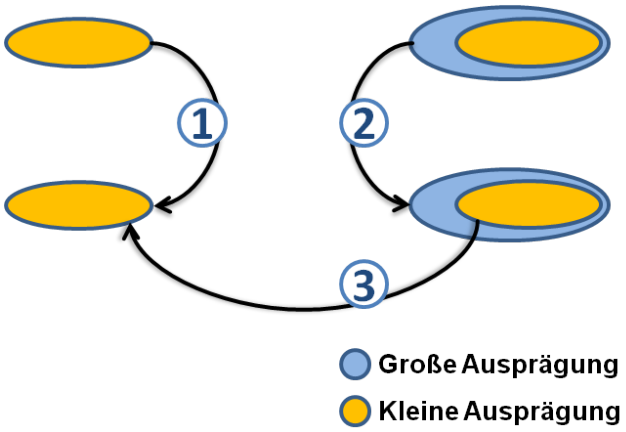

Abbildung 5.5.: Schematische Darstellung der Interaktion verschiedener, interoperabler Ausprägung einer Kommunikationsmiddleware.

Ein Konzept, in dem eine Kommunikationsmiddleware in unterschiedlichen, interoperablen Ausprägungen vorliegt, muss beantworten, wie sich Interoperabilität mit minimalem Aufwand umsetzen lässt. Für die Kommunikation sind einheitliche Serialisierungsregeln und mindestens ein gemeinsamer Kommunikationsmechanismus notwendig. Beides wird durch Anforderung 23 eingeschränkt. Demnach müssen alle Steuergeräte des IP-basierten Fahrzeugbordnetzes an der Kommunikation teilnehmen können. Der minimale funktionale Umfang, der leistungsschwachen Steuergeräten die Teilnahme an der Kommunikation ermöglicht, wird im nächsten Abschnitt detailliert beschrieben. Der

kleinste gemeinsame Nenner, den auch leistungsschwache Steuergeräte handhaben können, wird allerdings die funktionalen Anforderungen anspruchsvoller Anwendungen untereinander nicht erfüllen. Anspruchsvolle Anforderung fordern die Übertragung komplexer Datenstrukturen und ein dynamisches Verhalten. Dafür müssen die Serialisierungsregeln und Kommunikationsmechanismen skalieren. Dann muss das Koordinationsmodell aber auch Aussagen treffen, welche Ausprägungen der Kommunikationsmiddleware in welchem funktionalen Umfang interoperabel kommunizieren können. Diese Unterscheidung führt dazu, dass eine Anwendung mit einer Anwendung derselben Ausprägung anders kommuniziert als mit einer Anwendung in einer leistungsschwächeren Ausprägung. Abbildung 5.5 skizziert dieses Konzept.

Um für Anwendungen eine Abstraktion der Kommunikation zu bieten, muss die Schnittstellenbeschreibungssprache diese Unterscheidung definierbar machen. Die mögliche Kommunikation zwischen zwei betrachteten Anwendungen wird durch die gemeinsame Kommunikationsschnittstelle festgelegt. Somit muss eine Anwendung zur Laufzeit kein Wissen darüber besitzen, welche Ausprägung der Kommunikationsmiddleware ihrem Kommunikationspartner zur Verfügung steht. Der Anwendungsentwickler soll von der Umsetzung der Kommunikation entlastet werden. Die Transparenz, welche Mächtigkeit der Kommunikation zwischen Anwendungen auf bestimmten Steuergeräteklassen zugrunde liegt, wird durch die gemeinsame Beschreibung der Schnittstelle bereits auf Ebene der IDL hergestellt.

Das Konzept muss entsprechend auf Systemdienste erweitert werden. Auch jene müssen modular aufgebaut sein, um den nicht-funktionalen Anforderungen gerecht werden zu können. Die Kommunikationsmiddleware ist hierbei das Bindeglied, das entsprechend der Mächtigkeit der Basisfunktionalität, die zwischen bestimmten Steuergeräteklassen möglich ist, geeignete Module von Systemdiensten einsetzt.

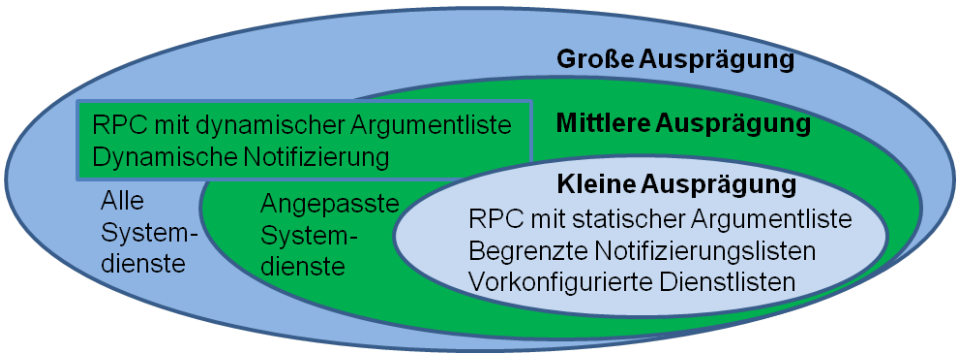

Abbildung 5.6.: Schematische Darstellung der Forderung nach funktionaler Inklusion für unterschiedliche Ausprägungen.

Eine Kommunikationsmiddleware muss mächtigen Anwendungen auf leistungsstarken Steuergeräten einen hohen Umfang an Kommunikationsfunktionalität bieten und gleichermaßen die Einschränkungen leistungsschwacher

Steuergeräte berücksichtigen. Dies wird als funktionale Skalierbarkeit bezeichnet. Die funktionalen Anforderungen an eine Kommunikationsmiddleware für das IP-basierte Fahrzeugbordnetz variieren mit Anwendungsmächtigkeit und Steuergeräteleistungsfähigkeit.

Diese funktionale Skalierung lässt nun auch eine genauere Beschreibung von Anforderung 22 nach einer möglichen Anwendungsmigration auf leistungsfähigere Steuergeräte ohne Verlust an Kommunikationskompatibilität zu. Mit unterschiedlich mächtigen Ausprägungen einer Kommunikationsmiddleware stellt sich dies als eine Forderung nach funktionaler Inklusion dar. Dies ist in Abbildung 5.6 schematisch dargestellt. Der funktionale Umfang einer Ausprägung muss eine Obermenge einer kleineren Ausprägung sein. Damit lassen sich bei der Migration einer Anwendung auf ein leistungsfähigeres Steuergerät die Kommunikationsfunktionen erweitern, ohne die bestehenden Schnittstellen oder den entsprechenden Programmcode anpassen zu müssen.

5.2.1. Minimales Koordinationsmodell und Basis-Serialisierung

Gemäß der Einschränkungen von leistungsschwachen Steuergeräten muss ein minimaler funktionaler Umfang gefunden werden, der IP-basierte Kommunikation zwischen oder mit dieser Steuergeräteklasse ermöglicht. Das Konzept muss damit auch untersuchen, wie sich die Anforderungen an das Koordinationsmodell möglichst durch einen einzigen, gegebenenfalls vereinfachten Kommunikationsmechanismus umsetzen lassen. Dieser Abschnitt behandelt beides: Erstens, wie sich ein möglichst einfacher RPC-Mechanismus für die Einschränkungen von eingebetteten Plattformen eignet. Zweitens, wie sich damit auch ein Notification-Konzept umsetzen lässt, das RPCs in allen Teilen der Protokollimplementierung nutzt.

Leistungsschwache Steuergeräte sind vor allem dadurch eingeschränkt, dass sie nicht auf dynamischem Speicher arbeiten können. Dies verhindert Berechnungen auf großen und dynamischen Datenstrukturen und ein dynamisches Verwalten von Notifizierungslisten. Fehlendes Multi-Threading eingebetteter Betriebssysteme erschwert die Garantie von Aufrufsemantiken, welche die Übertragung einer Nachricht sicherstellen bzw. Duplikate verhindern. Security-Mechanismen wie für Authentifizierung und Verschlüsselung benötigen eine signifikante Hochrüstung des verfügbaren Speichers und der Rechenleistung. Vor allem durch Berücksichtigung der Einschränkungen beim Arbeitsspeicher von eingebetteten Systemen ergibt sich die Vorgabe, nur statische Daten zu übertragen. Die maximalen Längen der Nutzdaten müssen dadurch bei der Beschreibung der Schnittstelle bereits angegeben werden. Eine Ausprägung der Kommunikationsmiddleware für leistungsschwache Steuergeräte muss auf statischen Daten beruhen. Dies vereinfacht neben der Reservierung von Puffern auch das Marshalling der Daten für die Übertragung und damit die Komplexität des RPCs selbst.

Für die Einschränkung einer Kommunikationsbeziehung auf statische Daten ist allein maßgeblich, ob einer der beiden Kommunikationspartner mittels der Ausprägung für leistungsschwache Steuergeräte kommuniziert. Dies kann bereits zur Entwicklungszeit überprüft werden, wenn die Schnittstelle in Programmcode kompiliert wird. Eine Prüfung zur Laufzeit entfällt damit.

Der Notifizierungsmechanismus wird für leistungsschwache Steuergeräte auf feste Listen zur Entwicklungszeit beschränkt. Dies wird durch die Einschränkung auf statische Speicherverwaltung bedingt. Bei einem dynamischen Anmelden von Diensten müsste eine maximal mögliche Anzahl von Empfängern zur Entwicklungszeit festgelegt werden. Damit ist die gewählte Umsetzung des in Anforderung 11 geforderten signalbasierten Kommunikationsparadigmas durch ein dynamisches An- und Abmelden für definierte Daten nicht direkt möglich. Dies lässt sich konzeptuell dadurch lösen, dass ein leistungsstarkes Steuergerät für alle im Gesamtsystem benötigten Daten bei einem leistungsschwachen Steuergerät statisch notifiziert ist. Eine dynamische Notifizierung durch andere Anwendungen kann nun beim leistungsstarken Steuergerät erfolgen. Dies geht einher mit dem Konzept von Domänenhauptrechnern, die Interdomänenkommunikation für leistungsschwache Steuergeräte derselben Anwendungsdomäne kapseln. Damit werden die Informationen zwar nicht auf der Vermittlungsschicht verteilt, aber es handelt sich bei Domänenhauptrechnern auch nicht um Anwendungsschicht-Gateways, da kein Übersetzungsaufwand zwischen nichtkompatiblen Protokollen notwendig ist.

Auch wenn die leistungsstarken Steuergeräte hinsichtlich ihrer Kommunikationsfunktionalität nicht eingeschränkt werden müssen, sollen Überlegungen über eine einfache Umsetzung des Notification-Konzepts angestellt werden, um eine möglichst schlanke Implementierung der Kommunikationsmiddleware zu ermöglichen. Das Notification-Konzept verlangt im Fahrzeugbordnetz kein Entkoppeln von Sender und Empfänger, sondern setzt beide in ein temporäre Beziehung, bei der die Adressen der Kommunikationspartner für die interne Kommunikation bekannt sind. Damit handelt es sich um Kommunikation zwischen bekannten Partnern, die auf einem festgelegten Protokoll und damit auf einer bekannten Menge von entfernten Aufrufen beruht. Dies lässt sich durch Remote Procedure Calls herstellen, die das Protokoll implementieren. Es handelt sich dabei um eine generische Anwendungsschnittstelle, welche die Kommunikationsmiddleware bereits neben der eigentlichen Anwendungsschnittstelle implementiert. Durch die Zurückführung des Notification-Konzepts auf eine definierte Reihe von RPCs ist implizit auch die Forderung nach einer festgelegten Semantik bei der Erzeugen von Notifizierungslisten gegeben.

Es lässt sich damit ein Notification-Konzept umsetzen, das Remote Procedure Calls in allen Teilen der Protokollimplementierung nutzt. Abbildung 5.7 visualisiert diese Verfeinerung des Konzepts, indem der funktionale Baustein *Notifizierung* nun auf dem RPC aufsetzt.

Als Transportprotokoll wird das verbindungslose UDP verwendet. TCP wurde für die Kommunikation im Internet entwickelt. Es ist für das

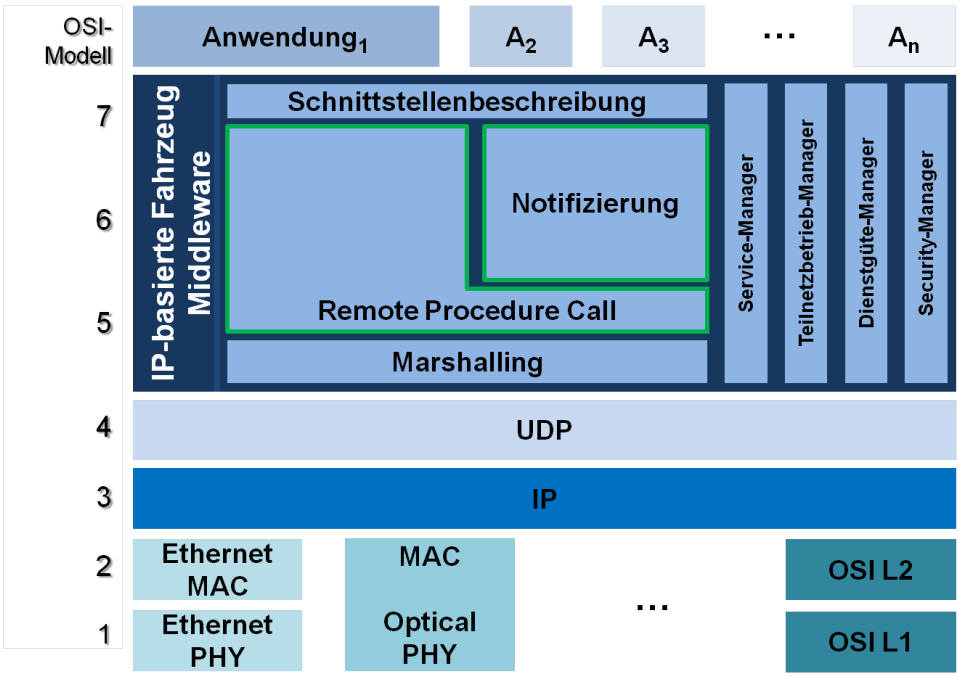

Abbildung 5.7.: Ein Notifizierungsmechanismus auf Basis von Remote Procedure Calls.

Fahrzeugbordnetz ungeeignet aufgrund der Timeouts, die für weite Übertragungsentfernungen und für weniger verlässliche Rechnernetze als das gut kontrollierbare Fahrzeugbordnetz gedacht sind. Anpassungen der Timeouts wären zwar möglich und wurden in zahlreichen Arbeiten für unterschiedliche Anwendungsgebiete untersucht, aber wenn existierende TCP-Implementierungen nicht direkt verwendet werden können, kann die Funktionalität einer Verbindungsorientierung gleichwertig durch Aufrufsemantiken durch die Kommunikationsmiddleware implementiert werden. Für leistungsschwache Steuergeräte ist zudem aufgrund der Speichereinschränkungen bereits eine umfangreiche TCP-Implementierung zu groß. Es gibt reduzierte TCP-Implementierungen speziell für eingebettete Plattformen. Die Analyse soll sich aber nicht an konkreten Umsetzungen ausrichten, sondern prinzipielle Umsetzungen untersuchen. Daher wird uneingeschränkt UDP als verbindungsloses Transportschichtprotokoll verwendet und die Kommunikationsmiddleware übernimmt die Sicherstellung von Aufrufsemantiken.

Zusammenfassend ermöglichen interoperable Ausprägungen die Unterstützung heterogener Steuergeräte mit unterschiedlicher Leistungsfähigkeit, ohne den Vorteil einer einzigen Lösung aufzugeben. Interoperabilität wird durch ein einheitliches Serialisierungsformat (statische Interoperabilität) und einen gemeinsamen Kommunikationsmechanismus (dynamische Interoperabilität) gewährleistet.

5.2.2. Klassifizierung interoperabler Spezifikationen

Dieser Abschnitt beschreibt die funktionalen Bestandteile verschiedener Ausprägungen der Kommunikationsmiddleware unter Berücksichtigung der nichtfunktionalen Eigenschaften.

Zunächst stellt sich die Frage nach der Anzahl von Ausprägungen. Zahlreiche Ausprägungen erlauben eine Annäherung an eine pro Steuergerät optimale Anpassung. Dagegen ist die Verwaltung der Komplexität, die durch viele Ausprägungen entstehen würde, für das Industrieumfeld mit seiner hohen Anzahl an Varianten und Derivaten ungeeignet. Gesucht werden daher Lösungen für klar umrissene Klassen von Steuergeräten und Anwendungen. Die tatsächliche Anzahl lässt sich erst für eine finale Entwicklung einer Kommunikationsmiddleware und den Einsatz im Fahrzeug bestimmen. Die Prinzipien lassen sich gut anhand von drei Ausprägungen demonstrieren. Eine Ausprägung mit minimalem Umfang für leistungsschwache Steuergeräte und eine Ausprägung mit maximalem Umfang veranschaulichen das Leistungsspektrum der Kommunikationsmiddleware. Anhand einer Ausprägung mit ausgewählter Teilfunktionalität lässt sich allgemein erklären, wie eine Abstufung zwischen Ausprägungen ermöglicht und die Koordination dennoch erreicht werden kann. Diese Ausprägungen werden im Folgenden als Low-, Mid- und High-Ausprägung bezeichnet. Die Definition einer Ausprägung richtet sich nach der Leistungsfähigkeit von Steuergeräten und entsprechend der geforderten Kommunikationsmächtigkeit.

High-Ausprägung

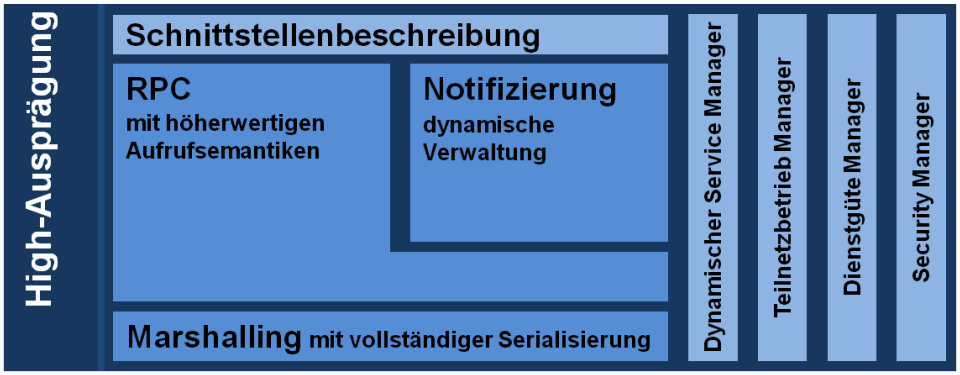

Abbildung 5.8.: Funktionale Bestandteile der High-Ausprägung.

Die High-Ausprägung bietet das gesamte Spektrum an möglichen Kommunikationsmechanismen, Aufrufsemantiken und Systemdiensten. Abbildung 5.8 zeigt dies detailliert. Der RPC erlaubt die Übertragung von strukturierten Daten und Listen dynamischer Größe und bietet dabei Garantien über das Verhalten im Fehlerfall mit einer *at-least-once*- oder *at-most-once*-Aufrufsemantik. Durch die Möglichkeit der Verwaltung von dynamischen Listen kann auch ein

dynamisches Ab- und Anmelden von Diensten für Notifications erfolgen. Die Systemdienste, die in Kapitel 6 noch beschreiben werden, stehen in vollem Umfang zur Verfügung.

Low-Ausprägung

Unter den verschiedenen Ausprägungen ist diejenige mit minimalem Funktionsumfang – vor dem Hintergrund der Leistungseinschränkung eingebetteter Steuergeräte im Fahrzeug – offensichtlich die kritischste und damit auch die interessanteste für die Untersuchung. Die Low-Ausprägung muss ausreichende Funktionalität bieten, um interoperabel im IP-basierten Fahrzeugbordnetz kommunizieren zu können und dabei den strengen Anforderungen an geringe Leistungsfähigkeit und optimierte Speichernutzung gerecht zu werden.

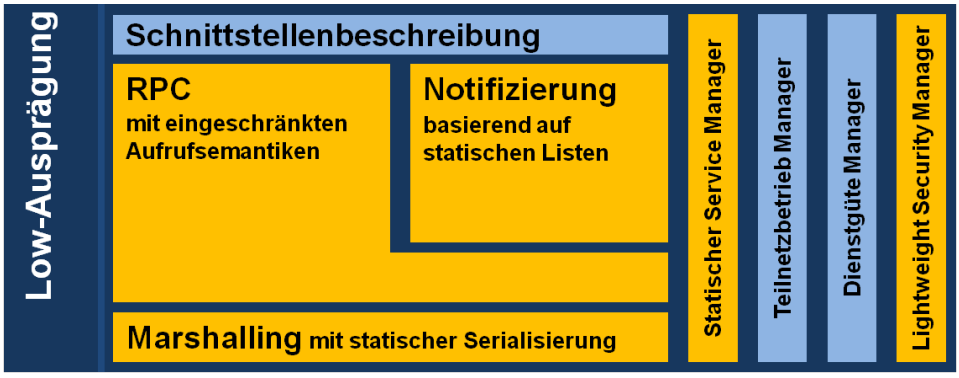

Abbildung 5.9.: Funktionale Bestandteile der Low-Ausprägung.

Abbildung 5.9 zeigt die funktionalen Bestandteile der Low-Ausprägung. Den Einschränkungen durch geringe Rechenleistung und Speicher wird mit dem oben beschriebenen minimalen Koordinationsmodell und einer auf den Basisumfang eingeschränkten Serialisierung begegnet. Die Aufrufsemantiken können keine Garantien geben, weil eine Haltung und Überprüfung von Nachrichtenlisten nur sehr eingeschränkt möglich ist.

Dynamische Notifications sind nicht möglich. Die Kommunikationspartner werden daher statisch festgelegt. Das Anmelden bei einem anderen Steuergerät ist als einfacher, entfernter Funktionsaufruf uneingeschränkt möglich. In der Regel beherbergen leistungsschwache Steuergeräte Anwendungen, die Sensoren kapseln und daher selten Empfänger von Daten sind.

Die Teilnahme am Teilnetzbetrieb ist dagegen unkritisch. Ist das leistungsschwache Steuergerät in der Rolle des Empfängers, so meldet ein bekannter Sender seinen Status, bevor er sich schlafen legt bzw. aufgewacht ist, und es ist nur ein Überschreiben des aktuellen Status in der statischen Liste der Kommunikationspartner notwendig. Ist das leistungsschwache Steuergerät in der Rolle des Senders, so muss jedem Empfänger aus der statischen Notifizierungsliste

eine Benachrichtigung gesendet werden, bevor sich das Gerät schlafen legt bzw. nachdem es aufgewacht ist. Auch dies ist ohne Einschränkung möglich.

Für das Security-Management schränkt die mangelnde Leistungsfähigkeit die Möglichkeit, bestimmte Security Ziele zu erreichen, stark ein. Prototypische Implementierungen [Weyl 10] zeigen, dass moderne Security Mechanismen, die auf asymmetrische Schlüssel beruhen, nicht auf leistungsschwachen, eingebetteten Steuergeräten eingesetzt werden können. Allerdings ist es durchaus möglich, symmetrische Schlüssel auf den für IP-basierte Kommunikation infrage kommenden Steuergeräten einzusetzen. Die Ergebnisse hierin zeigen, dass beispielsweise der Advanced Encryption Standard (AES) sich für symmetrische Verschlüsselung auf leistungsschwachen Steuergeräten eignet. Zusätzlich kann die Rechnernetztopologie sicherstellen, dass ein leistungsschwaches Steuergerät durch ein leistungsstärkeres vor Angriffen von außen geschützt ist.

Mid-Ausprägung

Die Mid-Ausprägung bietet eine echte Untermenge der geforderten Kommunikationsfunktionalität. Sie eignet sich für Steuergeräte, die nicht alle Kommunikationsmechanismen benötigen oder nur eingeschränkt auf Systemdienste zurückgreifen müssen.

External-Ausprägung

Die Prinzipien interoperabler Ausprägungen einer Kommunikationsmiddleware lassen sich anhand von drei Ausprägungen erläutern. Dies bezieht sich auf die interne Fahrzeugkommunikation, die im Fokus der vorliegenden Arbeit steht. Unter den genannten, generellen Vorteilen eines IP-basierten Fahrzeugbordnetzes spielt die Öffnung des Fahrzeugs nach außen eine wichtige Rolle. An dieser Stelle soll daher kurz der Vollständigkeit halber ausgeführt werden, dass sich das Konzept über die interne Vernetzung hinaus, auch auf Ausprägungen in der Peripherie des Fahrzeugs erweitern lässt.

Da Kommunikationsteilnehmer außerhalb des Fahrzeugs vermutlich mit genau einem dedizierten Endpunkt (In/Out-Gateway) im Auto verbunden sind, könnte für diesen Kommunikationsweg prinzipiell auch eine spezielle Kommunikationsmiddleware eingesetzt werden, die am In/Out-Gateway auf interne Middleware-Kommunikation übersetzt wird. Eine einheitliche Kommunikationsmiddleware hätte auch hier den Vorteil, die Übersetzung im In/Out-Gateway zu vermeiden und die offensichtlichen Vorteile einer Durchgängigkeit – wie beispielsweise Einfachheit, Wartbarkeit und größere Wissensbasis – zu erhalten.

Eine Möglichkeit der CE-Geräte-Anbindung wäre in Form einer speziellen Kommunikationsanwendung auf dem mobilen Endgerät möglich, welche die Kommunikationsschnittstelle ins Fahrzeug für andere Anwendungen kapselt, wie beispielsweise in Bouard et al. [Boua 12] aus Security-Perspektive beschrieben. Die Anwendungen sind damit auf den Umfang dieser öffentlichen Schnitt-

stelle beschränkt. Diese External-Ausprägung für CE-Geräte muss auf die verschiedenen mobilen Plattformen portiert werden. Der Nutzer muss sich dann für seine Plattform eine „Fahrzeug-App“ installieren. Die Durchgängigkeit der Vernetzung würde damit auf CE-Geräte ausgedehnt, die neue Anwendungen ins Fahrzeug bringen.

5.3. Prototypische Implementierung der Kommunikationsmiddleware

Es gibt drei grundsätzliche Methoden zur vergleichenden Leistungsanalyse von verteilten Anwendungen und Protokollen: Die mathematische Analyse, die Simulation und die Messung. Aufgrund des hohen Praxisbezugs der vorliegenden Arbeit, wird das Konzept durch Messungen evaluiert. Dazu wird zunächst ein Prototyp implementiert, der anschließend hinsichtlich der identifizierten Anforderungen untersucht wird. Der Prototyp zur Überprüfung des Konzepts wurde auf Basis einer existierenden Lösung entwickelt. Diese wurde entsprechend um fehlende Bestandteile aus dem erarbeiteten Konzept erweitert, um die unterschiedlichen Ausprägungen zur Veranschaulichung der Eigenschaften und für eine praktische Evaluation umzusetzen.

5.3.1. Ein bestehendes RPC-Framework als Grundlage

Für ein tieferes Verständnis und als Proof-of-Concept wurde eine Low-Ausprägung entsprechend der Anforderungen aus Kapitel 4 implementiert. Als Basis dient Apache Etch, das bereits in Abschnitt 3.4 neben anderen verfügbaren Lösungen beschrieben wurde. Die als Open-Source-Software verfügbare Kommunikationsmiddlewarelösung Apache Etch bietet sowohl einen bestehenden Umfang, der vielen funktionalen Anforderungen weitestgehend genügt, als auch die Möglichkeit, das Framework zu erweitern. Apache Etch bietet eine modulare, modifizier- und erweiterbare Architektur, ein effizientes, binäres Serialisierungsformat und eine intuitiv verständliche Schnittstellenbeschreibungssprache. Das Framework diente bereits für prototypische Einzeluntersuchungen im Rahmen der Anforderungs- und Konzeptfindung. Dies wurde teilweise bereits in Weckemann, Lim und Herrscher [Weck 11] veröffentlicht.

Apache Etch eignet sich aus mehreren Gründen für den Prototyp. Die Lösung wurde als Apache Incubator Projekt entwickelt und wird als Open-Source-Software unter einer Apache Lizenz 2.0 [Apacb] verwaltet, die kaum Einschränkung an die Verwendung der öffentlichen Code-Basis stellt. Der RPC-Mechanismus ist bereits für verschiedene Programmierumgebungen implementiert, so dass ein Testen gegenüber einer leistungsstarken Plattform als High-Ausprägung ohne großen Entwicklungsaufwand möglich ist. Außerdem ist das Marshalling in der existierenden Code-Basis modular umgesetzt, was Veränderungen hinsichtlich der beabsichtigten Einschränkung für leistungsschwache

Steuergeräte erleichtert.

5.3.2. Eigene Erweiterungen am RPC-Framework

Die prototypischen Implementierungen dienen dem Zweck, das vorgestellte Konzept zu überprüfen. Für die Angleichung der existierenden Lösung, mussten einige Erweiterungen für Apache Etch implementiert werden, die in diesem Abschnitt vorgestellt werden.

Apache Etch ist zwar prinzipiell unabhängig von der zugrundeliegenden Transportschicht, aber ohne ein verlässliches, verbindungsorientiertes Transportprotokoll müssen Aufrufsemantiken vom RPC-Framework konzipiert und entwickelt werden. Der verfügbare Programmcode und die Anwendungsbeispiele setzen auf TCP auf. Wenn eine Etch-Client eine Verbindung zu einem Server aufbaut, wird zunächst eine TCP-Verbindung etabliert. Diese Verbindung bleibt auch bestehen, nachdem der Aufruf erfolgt ist. Dies vereinfacht zum einen den Aufwand für den nächsten Aufruf des Clients beim selben Server. Noch wichtiger ist aber zum anderen die damit für den Server verbundene Möglichkeit, Aufrufe auf dem Client tätigen zu können. Dies macht die Kommunikation bidirektional und ermöglicht weitere Antworten nach der erwarteten ersten Antwort des Servers.

Eine Umstellung des untenliegenden Transportprotokolls auf UDP fordert eine Design-Entscheidung. Die Kommunikationsmiddleware kann eine verbindungsorientierte Kommunikation implementieren. Dann kann das RPC-Framework weiterhin wie gerade beschrieben genutzt werden – inklusive der Eigenschaften späterer Antworten. Andererseits erlaubt eine Verlagerung der Verbindungsorientierung in die Kommunikationsmiddleware eine flexible Anpassung von Timeouts und Ähnlichem im Vergleich zur Übernahme einer existierenden TCP-Implementierung. Den Leistungseinschränkungen eines leistungsschwachen Steuergeräts wird dieser Weg nicht gerecht. Falls beim Wechsel auf UDP auf Aufrufsemantiken verzichtet bzw. die Mächtigkeit eingeschränkt wird, fallen wir auf eine uni-direktionale Kommunikation zurück. Beide Möglichkeiten wurden für den Prototyp implementiert. Dafür wurde das Apache Etch-Modul *Stream Based Transport* durch ein Modul *Datagram Based Transport* ersetzt. In der Schnittstelle müssen Funktionen eine Timeout-Annotation angeben. Eine *at-most-once*-Aufrufsemantik kann damit von der Kommunikationsmiddleware als ein synchroner Aufruf umgesetzt werden. Bei einem Solchen wartet der Aufrufer für das als Timeout angegebene Intervall auf die Antwort. Falls keine Antwort während des Intervalls eintrifft, wird der Aufruf erneut gesendet. *At-most-once* über UDP benötigt damit das Halten einer Session durch die Kommunikationsmiddleware.

Die beiden Kommunikationsparadigmen können durch einfache Funktionsaufrufe abgedeckt werden. Dabei handelt es sich um Unicast-Kommunikation zwischen genau einem Sender und einem Empfänger. Dafür wurde Apache Etch um eine Broadcast-Unterstützung auf Basis von UDP erweitert. Die Funk-

tionalität ist dafür in einem Subdienst gekapselt. Alle Anwendungen dieses „BroadcastDirectoryService“ nutzen dabei den gleichen UDP-Port. Die Systemdienste, die im folgenden Kapitel detailliert betrachtet werden, benötigen ebenfalls eine Broadcast-Kommunikation.

Apache Etch wurde nicht für eingebettete Plattformen entwickelt. Eine Untersuchung muss zeigen, dass die Kommunikationsmiddleware dennoch auf Automotive-qualifizierten Steuergeräten lauffähig ist. Dies gilt in besonderem Maße für die Low-Ausprägung. Der Proof-of-Concept muss dabei zwei Aspekte zeigen:

1. Die Lösung ist für eingebettete Plattformen geeignet. Der Nachweis nach der Eignung für eingebettete Plattformen lässt sich im Wesentlichen dadurch erbringen, dass der Prototyp einer Low-Ausprägung für eine typische Automotive-qualifizierte Plattform optimiert wird, so dass er gerade das vorgeschlagene minimale Koordinationsmodell umsetzt. Diese Implementierung kann für sich bezüglich ihres Ressourcenbedarfs untersucht werden. Dafür wurde eine Ausprägung implementiert, die ein minimales Koordinationsmodell, wahlweise als Etch-Client oder -Server, mit eingeschränkten Apache Etch-Serialisierungsregeln umsetzt.

2. Die Lösung ist mit leistungsfähigeren Ausprägungen interoperabel. Der Nachweis der Interoperabilität geschieht bei der Implementierung implizit, da die implementierte Low-Ausprägung mit einer High-Ausprägung, also der verfügbaren Etch-Lösung, kommuniziert.

5.3.3. Implementierungsentscheidungen für die Low-Ausprägung

Das existierende Apache Etch-Framework unterstützt die Generierung von Programmrümpfen aus den definierten Schnittstellen als Programmiersprachen, neben Java auch C/C++. Dennoch ist die Architektur durch die ursprünglichen Entwicklungsziele durch objektorientierte Aspekte geprägt. Dies hat eine gute Modularisierung des Frameworks zum Vorteil, da es die Weiterentwicklung beziehungsweise Änderungen leicht macht. Allerdings darf die Unterstützung der Programmiersprache C nicht suggerieren, dass das bestehende Framework für leistungsschwache Plattformen wie eingebettete Systeme ausgelegt ist. Ein Entwicklungsziel für die prototypische Low-Ausprägung war es daher, eine grundsätzliche Vereinfachung der Architektur des Frameworks zu entwickeln, das für eingebettete Plattformen geeignet ist. Für die Anwendungen auf solchen Plattformen ist C für die Programmierung gut geeignet. Da dies außerdem bereits durch die existierende Code-Generierung unterstützt wird, was eine Wiederverwendung von Code erlaubt, wurde entschieden, dass für die kleine Ausprägung eine Codegenerierung in C adäquat und ausreichend ist. Da diese Codegenerierung aus denselben Schnittstellen erfolgen kann, wie

für die bisherigen Programmiersprachen-Bindings üblich, wird diese prototypische Umsetzung im Folgenden als *EmbeddedC Binding* bezeichnet.

Das *EmbeddedC Binding* setzt zwingend auf einer Transportschicht mit verbindungslosem UDP auf, entsprechend der Nutzung eines existierenden Bindings als große Ausprägung.

Zu Beginn der Entwicklung wurden einfache Design-Richtlinien gesetzt:

- Einfache Architektur: Die Implementierung umfasst keine Abbildung der Apache Etch-Architektur, sondern lediglich die Teilmenge der Funktionalität, die für die Interoperabilität mit bestehenden Bindings notwendig ist, in einer vereinfachten Architektur. Dies sind ein einfacher RPC-Mechanismus und statische Notifizierungslisten auf Basis der Apache Etch-Serialisierungsvorschriften, die auf Felder und Strukturen festgelegter maximaler Länge eingeschränkt sind.

- Optimierung auf kleine Anwendungsschnittstellen: Eine Anwendung auf einem eingebetteten System wird im Allgemeinen nur wenige Funktionen anbieten.

- Kein dynamisches Speichermanagement: Es sind nur statische Allokationen und ein statischer Prozessorstack verfügbar.

- Transportschicht: Ein einfacher, verbindungsloser Transport über UDP ist ausreichend.

- Aufrufsemantiken: Es werden keine Garantien für die Ausführung gewährleistet, die Kommunikationsmiddleware bietet damit lediglich eine *Maybe*-Aufrufsemantik.

Eine Optimierung für kleine Anwendungsschnittstellen meint Überlegungen, die eine Verbesserung der Leistungsfähigkeit versprechen, wenn eine Schnittstelle nur aus wenigen Funktionen besteht. Dies lässt sich am Beispiel des Marshallings erläutern. Bei einer Schnittstelle mit vielen Funktionen greifen viele unterschiedliche Funktionen auf dieselben Serialisierungsregeln zurück. Daher lohnt es sich, jene in einer Bibliothek zu sammeln, so dass sie von allen Funktionen genutzt werden können. Dagegen bleiben bei wenigen Funktionen in einer Schnittstelle im Allgemeinen viele Serialisierungsregeln ungenutzt. Eine Bibliothek mit allen Regeln würde signifikant mehr Speicher beanspruchen als notwendig. In diesem Fall lohnt es sich zu jeder Funktion direkt die Serialisierungsregeln zu binden, die sie entsprechend ihrer Parameter benötigt. Dieser Ansatz wird durch einfache Makros umgesetzt. Der Ansatz impliziert damit Redundanz, wenn unterschiedliche Funktionen dieselben Typen von Parametern haben. Diese Redundanz ist allerdings für kleine Schnittstellen vernachlässigbar im Vergleich zu einer Bibliothek, welche die gesamten Serialisierungsregeln kapselt.

Für leistungsschwache, eingebettete Plattformen würde eine dynamische Speicherverwaltung selbst einen relativ hohen Bedarf an Systemressourcen beanspruchen. Außerdem würde in einem System mit wenig verfügbarem Arbeitsspeicher die im Laufe des Betriebs dadurch bedingte Speicherfragmentierung zunehmen. Dies wiederum würde das System destabilisieren. Als Konsequenz muss eine dynamische Speicherallokierung bei einem solchen System strikt ausgeschlossen werden. Für das *EmbeddedC Binding* werden nur statische Allokationen und ein statischer Prozessorstack verwendet. Da alle Anwendungsschnittstellen zur Entwicklungszeit feststehen, entstehen dadurch kaum Einschränkungen für entsprechende Anwendungen im Automotive-Umfeld. Die Apache Etch-Serialisierungsregeln werden entsprechend eingeschränkt, so dass Felder eine maximale Länge nicht überschreiten dürfen. Dies bedingt eine Anpassung der Schnittstellenbeschreibungssprache. Die Einschränkungen werden so gegenüber dem Anwendungsprogrammierer verbindlich.

5.3.4. Ein Binding für eingebettete Plattformen

Das *EmbeddedC Binding* basiert nicht auf der existierenden Apache Etch-Architektur. Dennoch dient ein knapper Einblick in diese Architektur dem Verständnis hinsichtlich der Entwicklung des neuen Bindings und dessen Interoperabilität mit den bestehenden Bindings.

Das Apache Etch-Framework generiert eine Menge von Klassen. Diese sind, ohne Anspruch auf Vollständigkeit, ein *Mailbox Manager*, ein *Messagizer* und ein *Packetizer*. Diese dienen der Modularisierung der Architektur, so dass Klassen während der Entwicklung einfach ausgetauscht werden können. Die Klassen übernehmen nacheinander die notwendigen Arbeitsschritte zum Senden bzw. beim Empfangen einer Nachricht. Auf diesen modularen Aufbau verzichtet das *EmbeddedC Binding*. Die gesamte Funktionalität wird in einer einfachen Laufzeitbibliothek (Runtime Library) konzentriert. Die aus einer Schnittstelle als Stubs und Skeletons generierten Code-Rahmen sind auf die Programmiersprache C beschränkt. Die Optimierung auf leistungsschwache Plattformen unter der Annahme von Schnittstellen mit nur wenigen Funktionen wird durch die Verwendung von Preprocessor-Makros umgesetzt. Auf diese Weise werden Aufrufe und serialisierte Parameter direkt an den Funktionscode gebunden. Somit werden die Allokation von Arbeitsspeicher und die Notwendigkeit, Daten im Speicher zu verschieben, minimiert. Dies implementiert die weiter oben beschriebene Idee, dass einzelne Code-Fragmente für kleine Anwendungsschnittstellen effizienter sind als eine Bibliothek, welche die gesamte Marshalling-Funktionalität beherrscht.

Dies soll durch folgendes Beispiel veranschaulicht werden. Eine Funktion `setParameters` besitzt zwei Parameter, eine Fließkommazahl `float fFreq` und eine ganze Zahl `int iVolume`. Als Rückgabewert wird eine ganze Zahl erwartet. Diese Funktion wird zu einem Funktionsstub für einen Client kompiliert, entsprechend der Marshalling-Funktionalität für einen ausgehenden, entfernten

Funktionsaufruf:

```
int setParameters(float fFreq, int iVolume)
{
  // Generate Etch Message:
  [...]
  ETCHCALL (0x7d6effad,
            ETCH_REMOTEHOST,
            ETCH_PORT)
    ETCHPUTVAL   (MESSAGEID, _etchMsgID)
    ETCHPUTFLOAT (0x79a81cdd , fFreq)
    ETCHPUTVAL   (0x3b4383bf , iVolume)
  ETCHCALLEND
}
```

In diesem Code-Beispiel bezeichnen Wörter in Großbuchstaben die Namen von Preprocessor-Makros. Diese definieren die Aneinanderreihung der einzelnen Nachrichtenbestandteile für die zu sendende Nachricht. `ETCHCALL` initialisiert den entfernten Aufruf durch ein Leeren des Sendepuffers und durch ein Einsetzen des Apache Etch-Protokoll-Headers und der UDP-Socket-Adressen. Die folgenden `ETCHPUTxxx`-Makros serialisieren die definierten Parameter gemeinsam mit ihren eindeutigen Identifikatoren und erhöhen gleichzeitig die Zähler für die Anzahl der Parameter und die Paketlänge. Abschließend finalisiert `ETCHCALLEND` die Nachricht im Sendepuffer und übergibt sie an eine Funktion, welche die Nachricht an den UDP-Stack mit dem Primitiv *UDP Send Packet* übergibt. Hier erkennt man, wie der Marshalling-Overhead linear mit der Anzahl der Funktionen einer Anwendungsschnittstelle ansteigt.

Für einen ankommenden entfernten Funktionsaufruf stellt das *EmbeddedC Binding* entsprechend eine Laufzeitbibliothek entsprechend der Apache Etch-Klasse *Packet Parser* zur Verfügung. Wenn der untenliegende *UDP Packet Dispatcher* eine empfangene Nachricht übergibt, überprüft die Bibliothek zunächst den Header, ob es sich um eine Apache Etch-Nachricht handelt. Anschließend wird der Identifikator des Aufrufs mit einer Liste der registrierten Aufrufe verglichen, die aus der Schnittstellenbeschreibung generiert wurde. Bei einer Übereinstimmung erfolgt ein Aufruf auf den hinterlegten Pointer der lokalen Funktion. Die De-Serialisierung der Aufrufparameter innerhalb der Nachricht geschieht entsprechend der Serialisierung durch Preprocessor-Makros, die an den identifizierten Funktionsaufruf gebunden sind. Die Parameter werden direkt in die für die lokalen Funktionsvariablen vorgesehenen Speicherorte kopiert.

5.3.5. Erweiterungen an der Schnittstellenbeschreibungssprache

Die Apache Etch-Schnittstellenbeschreibungssprache erlaubt Zeichenketten und Arrays flexibler Länge. Entsprechend der geforderten Einschränkungen

bezüglich Marshalling und Speicherallokation wurde die Schnittstellenbeschreibungssprache erweitert.

Eine statische Allokation von Speicher bedingt eine festgelegte Länge von Feldern, damit der Empfangsspeicher im Vorhinein in der richtigen Größe bereitgestellt werden kann. Eine Begrenzung der Länge ist ohnehin sinnvoll. Steuerungsdaten in CAN und FlexRay sind auf 8 Byte bzw. 64 Byte begrenzt. Damit kommen die meisten heutigen Anwendungen mit dieser Begrenzung aus. Der Empfangspuffer muss so gewählt werden, dass er die längste mögliche Nachricht empfangen kann. Dies würde zu einer hohen Belastung des Arbeitsspeichers führen, um lange Nachrichten für wenige Ausnahmen zu erlauben, während die große Mehrheit der Nachrichten kurz sind. Für Zeichenketten und Arrays oder gar Kombinationen aus beiden gibt es prinzipiell zwei Ansätze.

Im ersten Ansatz definiert man global eine maximale Länge für alle Zeichenketten und Arrays. Dies ist zwar eine einfache Möglichkeit, sie ist allerdings unflexibel. Außerdem verschwendet sie Puffer im Fall von kurzen Feldern. Außerdem wird eine Ausnahmeregelung für Zeichenketten benötigt, die länger als die global vorgegebene maximale Länge sind, beispielsweise durch das Aufteilen auf mehrere Pakete. Diese in Apache Etch nicht vorgesehene Fragmentierung müsste in geeigneter Form gekennzeichnet werden.

Der zweite – und präferierte – Ansatz legt die maximale Länge für jedes betreffende Feld einzeln fest. Für eine konkrete Funktion und ihre Parameter gibt es eine Erwartungshaltung, die diese Kennzeichnung im Einzelnen ermöglicht. Die maximale Länge des Feldes wird demnach einzeln in der Schnittstelle definiert. Die folgenden drei Beispiele demonstrieren den Gebrauch der dafür eingeführten Annotationen:

```
int[30]         numbers
string(64)      status
string(20)[10] programList
```

Dabei beschreibt `int[30]` ein Array mit einer maximalen Anzahl von 30 ganzzahligen Werten. `string(64)` kennzeichnet eine Zeichenkette die aus maximal 64 Buchstaben besteht. Die Verwendung von runden Klammern bei Zeichenketten erlaubt die Unterscheidung zur entsprechenden Notation für Arrays. Somit wird eine Kombination aus beiden Annotationen möglich. Entsprechend bedeutet `string(20)[10]` eine Array mit einer maximalen Anzahl von 10 Zeichenketten, die jeweils aus bis zu 20 Buchstaben bestehen können.

Wie im verfügbaren Apache Etch-Framework erlaubt auch das *EmbeddedC Binding* die Definition von Strukturen, insofern alle Zeichenketten und Arrays innerhalb der Struktur mit maximalen Längen annotiert sind. Dagegen ist es offensichtlich nicht möglich, eine dynamische Struktur wie eine Liste oder einen Baum abzubilden.

Das *EmbeddedC Binding* generiert eine Laufzeitbibliothek in ANSI-C Code, der nicht auf eine bestimmte Plattform beschränkt ist. Allerdings sind auf

den meisten eingebetteten Betriebssystemen keine Standard IP/UDP-Socket Schnittstellen verfügbar. Stattdessen werden veränderte IP/UDP-Stacks, die speziell für die Plattform und beherbergte Anwendungen angepasst werden, verwendet. Das *EmbeddedC Binding* arbeitet daher auf einer beliebig gewählten Implementierung des IP/UDP-Stacks. Die Funktionalität für die Primitive *UDP Send Packet* und *UDP Received Packet* sind durch eine generische Funktion und einen *Dispatcher* so gekapselt, dass eine einfache Adaption auf ein gewähltes Betriebssystem möglich ist.

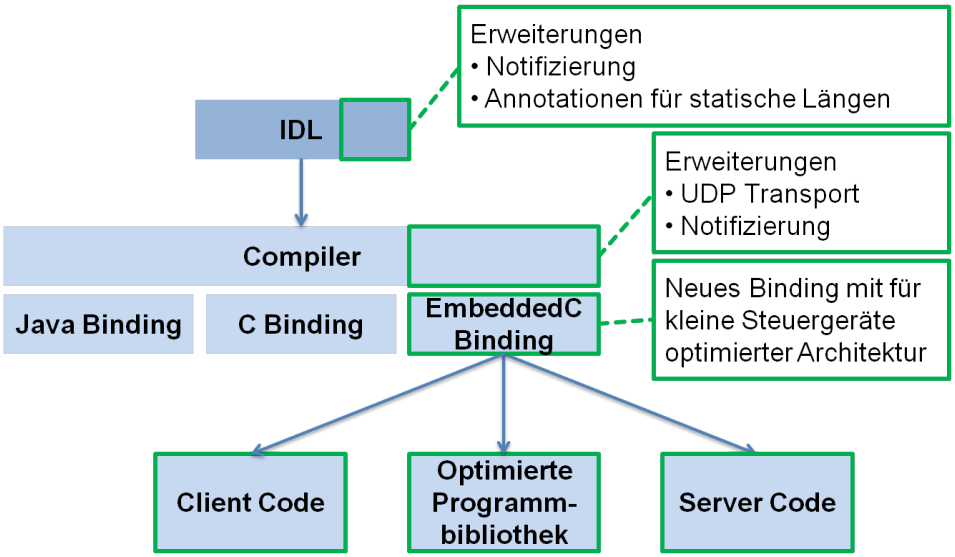

Abbildung 5.10.: Erweiterungen am existierenden Umfang von Apache Etch für die prototypische Evaluation.

Abbildung 5.10 fasst die wesentlichen Erweiterungen an Apache Etch zusammen. Die meisten Erweiterungen betreffen das neu entwickelte *EmbeddedC Binding*. Für die Evaluation waren zusätzliche Erweiterungen bezüglich Transport über UDP und eine Umsetzung des signalbasierten Kommunikationsparadigmas durch einen Notifizierungsmechanismus notwendig.

5.4. Evaluation der prototypischen Kommunikationsmiddleware

Das vorgestellte Konzept wird durch Messungen an einem Prototyp evaluiert. Die Low-Ausprägung der Kommunikationsmiddleware für leistungsschwache Steuergeräte ist dabei kritisch. Dies bezieht sich sowohl darauf, ob das Konzept auf einer solchen Ausprägung umsetzbar ist und wie sich die Interoperabilität im Versuch darstellt. Zunächst werden die interessanten Messgrößen bestimmt. Henning [Henn 08] beschreibt Leistungsuntersuchungen für Kommunikationsmiddleware. Diese lassen sich gemäß der hier angestellten Betrachtungen des

Fahrzeugbordnetz und der Anforderungen an eine Kommunikationsmiddleware für ein IP-basiertes Fahrzeugbordnetz auf Messungen im Fahrzeugkontext übertragen. Henning beschreibt, wie kritisch in IT-Systemen die Leistungsfähigkeit der Kommunikationsunterstützung ist. Im IT-Umfeld ist dies meist eine Herausforderung der Skalierung, wenn Server tausende von Clients bedienen müssen. Im Fahrzeug ist die Anzahl der Kommunikationspartner nicht nur deutlich kleiner, sondern auch in der Größenordnung bekannt. Die Kritikalität in der Leistungsfähigkeit entsteht durch die Leistungsschwäche der beteiligten Steuergeräte, die unter hohem Kostendruck entsprechend ihrer Anwendungen optimiert sind.

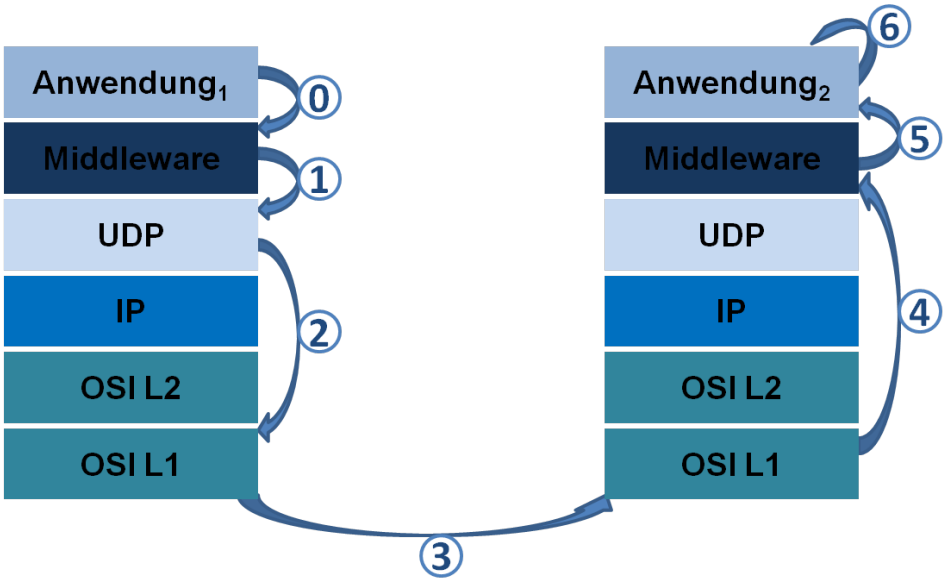

Abbildung 5.11.: Einteilung der Kommunikation von einer Anwendung zu einer anderen in mehrere Zeitintervalle.

Für die Leistungsfähigkeit der Kommunikationsmiddleware ist der Bedarf an Rechenleistung messbar. Henning beschreibt dies vereinfachend: Je mehr CPU-Zyklen für Marshalling und De-Marshalling von der Kommunikationsmiddleware genutzt werden, desto weniger CPU-Zyklen stehen für die Anwendung zur Verfügung. Damit dauert es entsprechend länger, bis eine Nachricht durch die Kommunikationsmiddleware vermittelt wird und der eigentlichen Anwendung zur Verfügung steht. Da die direkte Messung von CPU-Zyklen sehr aufwändig ist, messen wir diesen Aspekt indirekt über die Latenz. Die Latenz beschreibt die Zeitdauer zwischen Aufruf der entfernten Funktion und dem Erhalt der Antwort durch eine Anwendung. Die Latenz lässt sich damit, wie in Abbildung 5.11 gezeigt, in mehrere Zeitintervalle aufteilen:

1. Ein Intervall beim Client, in der die Kommunikationsmiddleware die Kommunikation im Auftrag der Anwendung übernimmt und für das Senden vorbereitet.

2. Die Abarbeitung im UDP/IP-Stack beim Client.
3. Einem Intervall der Nachricht im Rechnernetz inklusive der Abarbeitung in Koppelelementen wie Switches und Routern.
4. Dem Entgegennehmen und Bearbeiten der Nachricht durch den UDP/IP-Stack des Servers.
5. Einem Intervall für das De-Marshalling durch die Kommunikationsmiddleware beim Server.
6. Die Bearbeitung der Anfrage durch eine Server-Anwendung.

Nach der Bearbeitung durch den Server nimmt die Antwort den entsprechenden Rückweg.

Zur Validierung des Konzepts muss der Aufwand bewertet werden, den die Kommunikationsmiddleware verursacht. Dazu wird dieser Anteil mit den Anteilen einer einfachen Server-Anwendung, der Verweildauer im Rechnernetz und im UDP/IP-Stack verglichen. Rechnernetz und UDP/IP-Stack werden vereinfachend zusammengenommen betrachtet. Für die Evaluation setzt sich die Latenz somit aus drei wesentlichen Teilen zusammen:

- Bearbeitungsdauer der Kommunikationsmiddleware.
- Bearbeitungsdauer im UDP/IP-Stack inklusive Verweildauer im Rechnernetz.
- Bearbeitungsdauer der Anfrage durch eine Server-Anwendung.

In Bezug auf den Speicherbedarf, geht Henning auf mobile Endgeräte ein, um diesen Aspekt der Leistungsfähigkeit darzustellen. Ein höherer Speicherbedarf bedingt demnach direkt höhere Kosten für zusätzliche Hardware, einer der kritischsten Aspekte in der kostensensitiven Fahrzeugentwicklung.

Skalierbarkeit lässt sich in diesem Zusammenhang auf die Qualität des Konzepts zurückführen. Eine effiziente Kommunikationsmiddleware wird automatisch besser skalieren, weil sie der Anwendung einen geringeren Teil der begrenzten Ressourcen nimmt. Die Tauglichkeit der Low-Ausprägung für eingebettete Plattformen, die typisch für ein Fahrzeugbordnetz sind, wird daher hinsichtlich Speicherbedarf und Ausführungsverhalten der Kommunikationsmiddleware in Aufrufen pro Zeitintervall evaluiert.

5.4.1. Versuchsaufbau

Der Versuchsaufbau orientiert sich an einem realistischen Anwendungsfall eines Audio-Verstärkers in einem IP-basierten Fahrzeugbordnetz. Abbildung 5.12 veranschaulicht den Versuchsaufbau inklusive der jeweils verwendeten Hardware, Betriebssysteme und Bindings der Kommunikationsmiddleware.

Die Evaluation und Messungen wurden auf einer eingebetteten Plattform durchgeführt. Auf dieser läuft die Server-Anwendung, in Abbildung 5.12 auf der rechten Seite. Die Plattform rechnet mit einer Automotive-qualifizierten Freescale MPC5668 Fado CPU. Die CPU basiert auf der 32-bit Power-Architektur mit einer Taktung von 116 MHz und hat Zugriff auf einen Arbeitsspeicher von 592 kB SRAM und einen permanenten Speicher von 2 MB Flash Speicher. Für den Versuchsaufbau und die Leistungsmessungen wurde als Betriebssystem eCos [eCos] verwendet. eCos ist ein konfigurierbares Betriebssystemen mit Echtzeiteigenschaften, das für eingebettete Anwendungen geeignet ist. eCos unterscheidet nicht zwischen Betriebssystem und Anwendung. Beide werden beim Kompilieren gelinkt und als ein einziger Prozess gestartet. eCos bietet einige wählbare Treiber und Programmierschnittstellen. So ist beispielsweise eine einfache Implementierung eines IP-basierten Stacks bereits als Programmierschnittstelle für den Netzzugriff vorhanden. Dieser Stack basiert auf dem *lightweightIP-Stack* (lwIP) [Dunk 01]. Der Stack beinhaltet eine TCP-Implementierung. Er lässt sich aber so konfigurieren, dass UDP als Transportprotokoll verwendet wird, wie es dem vorgestellten Konzept entspricht.

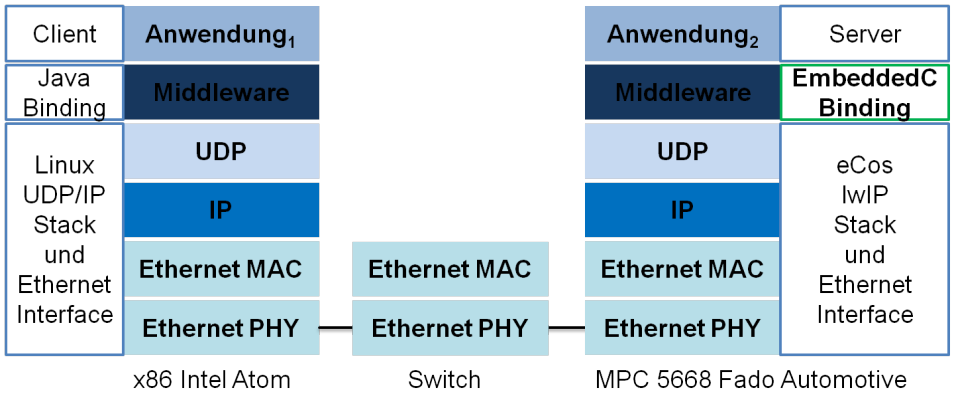

Abbildung 5.12.: Versuchsaufbau bestehend aus einem handelsüblichen PC und einem eingebetteten System mit einer Automotive-qualifizierten MPC5668 Fado CPU.

Auch wenn die MPC5668 Fado CPU nicht die minimale Plattform darstellt, die für ein IP-basiertes Fahrzeugbordnetz infrage kommt, eignet sie sich für den Versuchsaufbau. Die Plattform stellt mit ihren Leistungscharakteristika eine realistische Umgebung für die Low-Ausprägung der Kommunikationsmiddleware dar. Andererseits ist für die Messung der Leistungsfähigkeit der Kommunikationsmiddleware, im Sinne von beanspruchter Rechenleistung, zusätzlicher Programmcode für die Messung selbst auf der eingebetteten Plattform notwendig, was auf einer leistungsschwächeren Plattform nicht möglich wäre.

Das MPC5668 Board ist über 100 MBit/s Ethernet über einen handelsüblichen Ethernet Switch an einen PC angeschlossen, in Abbildung 5.12 in der Mitte.

Der PC basiert auf einer x86-Architektur mit einer Intel Atom CPU, die auf 1.6 GHz getaktet ist, in Abbildung 5.12 auf der linken Seite. Der PC emuliert den Hauptrechner der Infotainmentdomäne (Head Unit) als Kommunikationspartner. Die Head Unit nutzt die verfügbare Apache Etch-Kommunikationsmiddleware. Für sie wird die Schnittstelle durch das *Apache Etch Java Binding* erzeugt. Auf dem MPC5668 Board wird das entwickelte *EmbeddedC Binding* ausgeführt. Die Quellcode-Rahmen werden für beide Kommunikationspartner wie für Apache Etch üblich aus derselben Schnittstellendatei generiert.

Als Beispielanwendung wird die Schnittstelle eines Audio-Verstärkers verwendet. Die Interaktion des Nutzers erfolgt dabei an der Head Unit. Die Schnittstelle beschreibt daher die Kontrollfunktionen der Head Unit mit typischen Funktionen wie Start/Stopp, Senderauswahl und Lautstärkenregelung. Das MPC5668 Board repräsentiert den Audio-Verstärker, der die Steuerbefehle entgegen nimmt und entsprechende Regelungen vornimmt.

5.4.2. Speicherbedarf der Low-Ausprägung

Gemäß den identifizierten nicht-funktionalen Anforderungen ist Speicherbedarf die kritischste Anforderung. Das Designziel für eine Kommunikationsmiddleware im IP-basierten Fahrzeugbordnetz ist es diesbezüglich, eine funktionale Anwendungsabstraktion zu bieten, dabei aber nur einen geringen Speicherbedarf an die optimierten, eingebetteten Plattformen zu stellen.

Eine direkte Messung des Arbeitsspeichers ist nicht möglich, da das *EmbeddedC Binding* temporär zusätzlichen Speicher auf dem Prozessorstack nutzt. Daraus folgt, dass die Messungen eine obere Grenze für den durch den Anwendungsprozess statisch allokierten Arbeitsspeicher darstellen. Die Größe des Programmcodes im permanenten Speicher lässt sich durch das Linken von Betriebssystem und Anwendung auch nicht direkt messen. Dies ist aber möglich, indem das Betriebssystem zwar mit IP-Stack, aber einmal ohne und einmal mit den Umfängen der Kommunikationsmiddleware auf der eingebetteten Plattform installiert wird. Der Bedarf der Kommunikationsmiddleware an permanentem Speicher ergibt sich folglich als Differenz. Die Messungen wurden unter Nutzung des Unix Programms *size* basierend auf einer einzelnen Funktion der Anwendungsschnittstelle durchgeführt:

```
f: short getMaximumStartUpVolume()
```

Die Speicherbelegung einzelner Komponenten wurde mithilfe der Sektionen der Binary Object Datei identifiziert. Die Sektion *.text* beschreibt dabei den Programmspeicher. Die Sektionen *.data* und *.bss* ergeben addiert den statischen Bedarf an Arbeitsspeicher. Alle anderen Komponenten konnten durch Aktivierung und Deaktivierung in der eCos-Konfiguration identifiziert und heraus gerechnet werden. Tabelle 5.1 und entsprechend Abbildung 5.13 zeigen eine Übersicht der Ergebnisse.

Komponente	Arbeitsspeicher	Programmspeicher
Betriebssystemkern	13 kB / 3%	28 kB / 13%
lwIP (inklusive TCP)	48 kB / 13%	59 kB / 27%
EmbeddedC Bibliothek	5 kB / 1%	5 kB / 2%
Programmcode	310 kB / 82%	97 kB / 44%
Andere	≈ 1 kB / 0%	30 kB / 14%
Gesamt	377 kB	219 kB

Tabelle 5.1.: Absolute und relative Werte für den statisch allokierten Arbeitsspeicher (SRAM) und die Programmspeicherbelegung (Flash) aufgeteilt nach Komponenten.

Die EmbeddedC Bibliothek als prototypische Umsetzung der Low-Ausprägung nimmt für die einfache Anwendungsschnittstelle lediglich 5 kB des Programmspeichers ein. Im Vergleich zum Betriebssystemkern mit 28 kB und dem verwendeten IP-Stack mit 59 kB ist dies ein offensichtlich kleiner Anteil an der Speicherbelegung. Die Kommunikationsmiddleware benötigt demnach knapp 5% mehr Programmspeicher als für einen direkten Zugriff auf die Primitive der Transportschicht benötigt würden und bietet dafür die gewünschte Kommunikationsabstraktion. Für den Bedarf an Arbeitsspeicher ergibt sich ein vergleichbares Bild. Lediglich 5 kB werden von der Kommunikationsmiddleware beansprucht. Dagegen benötigen das Betriebssystem 13 kB und der IP-Stack 48 kB.

Die vorgestellten Ergebnisse zeigen die Arbeits- und Programmspeicherbelegung am Beispiel einer Anwendung, deren Schnittstelle nur eine Funktion umfasst. Die Werte ermöglichen damit eine Aussage über den minimalen Speicherbedarf der Kommunikationsmiddleware und damit über eine untere Schranke. Im nächsten Schritt wurde untersucht, wie sich das Hinzufügen von Funktionen zu einer Schnittstelle auswirkt. Dazu wurde die Versuchsreihe mit je einer zusätzlichen Funktion in der Schnittstelle wiederholt. Dabei zeigt sich, dass der Bedarf an Programmspeicher linear anwächst. Dies entspricht den Erwartungen, da das gewählte Marshalling an jede Funktion die individuell benötigte Marshalling-Funktionalität bindet. Das Hinzufügen einer Funktion mit einem Parameter führt zu ca. 700 Byte mehr Programmspeicherbelegung. Somit lässt sich der Bedarf an Programmspeicher als Funktion angeben, die lediglich eine konstante Grundfunktionalität beschreibt und ansonsten mit der variablen Mächtigkeit der Menge der Funktionen in der Schnittstelle wächst. Der Programmspeicherbedarf m_p berechnet sich aus der entwickelten Apache Etch *EmbeddedC Bibliothek* und der Anzahl von Funktionen n_{Funktion} in der Schnittstelle.

$$m_p = 5\,\text{kB} + 0.7\,\text{kB} \cdot n_{\text{Funktion}}$$

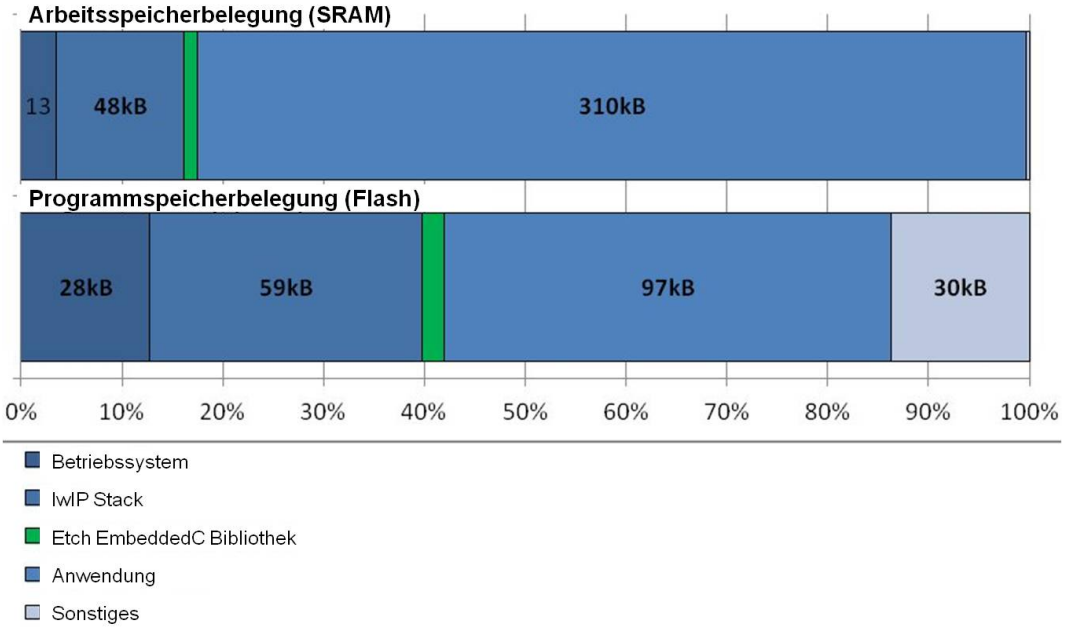

Abbildung 5.13.: Statisch allokierten Arbeitsspeicher (SRAM) und die Programmspeicherbelegung (Flash) aufgeteilt nach Komponenten.

Die prototypische Umsetzung einer konzeptionellen Low-Ausprägung zeigt, dass mit nur wenig Programm- und Arbeitsspeicherbelegung, die für Anwendungen auf leistungsschwachen Steuergeräten geeignete, funktionale Kommunikationsabstraktion erbracht werden kann.

5.4.3. Ausführungsverhalten der Low-Ausprägung

Die Untersuchung des Ausführungsverhaltens misst die Latenz im verteilten System. Die Latenz lässt sich in die Bearbeitungsintervalle der Kommunikationsmiddleware, im UDP/IP-Stack und Rechnernetz, sowie durch die Server-Anwendung unterteilen.

Die Evaluation erfolgt dabei wiederum über wiederholte Kommunikation zwischen einer Head Unit als Client und einem Audio-Verstärker als Server. Die beiden Kommunikationspartner werden wie oben durch denselben PC und das eingebettete MPC5668 Fado Board repräsentiert. Der Versuch besteht aus aneinandergereihten, entfernten Funktionsaufrufen derselben Funktion:

```
g: boolean setMaximumStartUpVolume
           (short maximumStartUpVolume)
```

Der Funktionsaufruf wird 1000-fach wiederholt, so dass der Empfang der Antwort eines Aufrufs durch die Client-Anwendung umgehend den nächsten Aufruf auslöst. Anschließend lässt sich die durchschnittliche Zeitdauer für einen

Aufruf ermitteln. Die Bildung des Durchschnitts ist nicht nur in der unterschiedlichen Zeitdauer zwischen zwei Aufrufen begründet. Für die Messung müssen Zeitstempel gespeichert werden, wofür die Systemuhr des PCs (Head Unit) genutzt wird. Die Bearbeitungszeit setzt sich demnach nicht nur aus der Ausführung der eigentlichen Kommunikation und der Funktionsberechnung zusammen, sondern wird durch die Messung selbst beeinflusst. Die Zeitstempel werden von der Systemuhr entsprechend des CPU-Takts erzeugt. Damit hat die Uhr eine theoretische Granularität von 10 ns. In der Praxis ist die Granularität durch verpasste Uhrticks geringer. Für die Messung genügen aber zwei Zeitstempel, zu Beginn und nach dem tausendsten Aufruf. Auf diese Weise ist der Aufwand für das Setzen der Zeitstempel für die Messung vernachlässigbar. Die Ausführungszeit pro Aufruf wird anschließend als Durchschnitt aller Aufrufe ermittelt. Die Ergebnisse sind in Tabelle 5.2 aufgeführt:

Ausführungszeit pro RPC [μs]	RPCs pro Sekunde [1/s]
450	2200

Tabelle 5.2.: Ausführungszeit eines entfernten Funktionsaufrufs.

Die durchschnittliche Ausführungszeit von 450 μs pro RPC stellt dabei den Best Case dar, da ein kontinuierliches Wiederholen des Aufrufs die Re-Initialisierung des Server-Prozesses und die Re-Konfiguration des Rechnernetzes über das Address Resolution Protocol (ARP) vermeidet. Da eine längere Ausführungsdauer durch Re-Initialisierungen und Re-Konfigurationen aber der Anwendung bzw. dem Rechnernetz angerechnet werden müssen, ist das Ergebnis zur Bewertung der Kommunikationsmiddleware valide.

Für weitere Messungen wurde im nächsten Schritt die Anwendungsschnittstelle modifiziert. Wie erwartet zeigen die Messergebnisse, dass die Anzahl der Funktionen in einer Schnittstelle keinen signifikanten Einfluss auf die Ausführungsdauer hat, da sich lediglich die Anzahl der Funktionsidentifikatoren erhöht, welche die Kommunikationsmiddleware in einer Liste verwaltet. Bei der geringen üblichen Anzahl von Funktionen auf leistungsschwachen Steuergeräten stellt das Suchen in einer Liste keinen erheblichen Rechenaufwand dar.

Außerdem wurde feststellt, dass auch eine zusätzliche Last auf der genutzten Vernetzungstechnologie keine signifikante Auswirkung auf die Performanz der Kommunikationsmiddleware hat. Um dies zu prüfen, wurde auf derselben Netzschnittstelle ein RTP-Strom mit Audio-Daten gesendet. Ein RTP-Paket wird dabei alle 7 ms gesendet. Dies änderte nichts an der Ausführungsdauer für einen entfernten Funktionsaufruf.

Des Weiteren werden die Ausführungszeiten der verschiedenen beteiligten Komponenten Server-seitig unterschieden. Diese bestehen chronologisch aus Empfangen einer Nachricht durch den IP-Stack, De-Marshalling durch die Kommunikationsmiddleware, Bearbeitung der Anfrage durch die Anwendung, Marshalling der Antwort und Senden der Nachricht durch den IP-Stack. Da-

für werden Zeitstempel innerhalb von IP-Stack, Kommunikationsmiddleware und Anwendung genutzt. Die Zeitstempel werden wiederum gemäß der Systemuhr am PC gesetzt. Auch hier ist die in der Praxis geringere Granularität der Uhr im Vergleich zur theoretischen Auflösung von 10 ns zu beachten, weswegen erneut der Durchschnitt aus wiederholten Durchläufen gebildet wird. Zusätzlich muss in diesem Versuch aber auch der Messfehler beachtet werden, der durch die zahlreichen Zeitstempel pro einzelnem Aufruf entsteht. Dafür wurde die Versuchsfolge sowohl mit allen benötigten Zeitstempeln, als auch mit nur zwei Zeitstempeln je zu Beginn und zu Ende eines Aufrufs durchgeführt. Dadurch lässt sich das Intervall annähernd bestimmen, dass durch das Setzen der Zeitstempel selbst verursacht wird. Dieser Einfluss wurde als $2\,\mu s$ kalkuliert. Außerdem werden die Zeitstempel zunächst gepuffert und erst bei vollem Puffer auf der RS232 Debug Netzschnittstelle kommuniziert, um durch die Messung verursachte Last auf der Ethernet Schnittstelle zu vermeiden.

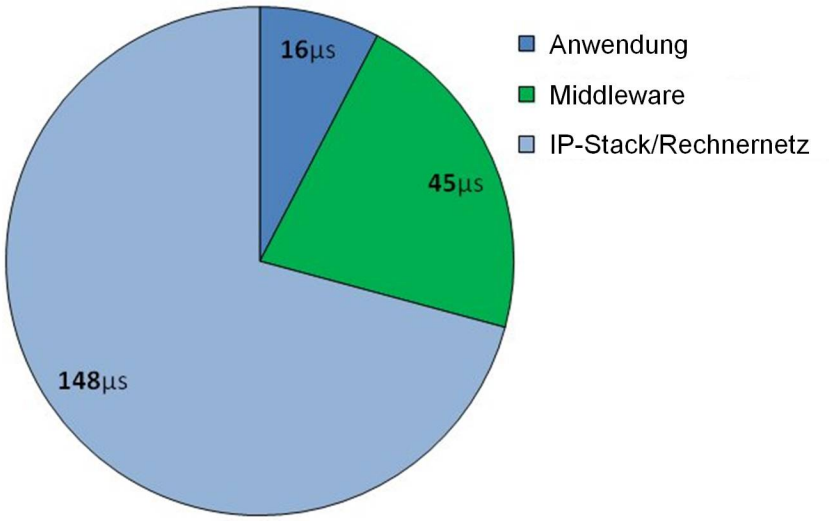

Abbildung 5.14.: Ausführungsdauer der unterschiedlichen Komponenten für einen entfernten Funktionsaufruf.

Die Ergebnisse sind in Abbildung 5.14 visualisiert. Die Empfangs- und Sendezeiten sind wiederum aggregiert, um die bisher verwendete Aufteilung in die Komponenten Kommunikationsmiddleware, Anwendung und IP-Stack/Rechnernetz zu verwenden. Die Ausführung des lokalen Funktionsaufrufs `g` benötigt durchschnittlich $16\,\mu$s (in $> 90\%$ der Fälle sind es $\approx 8\,\mu$s). Aufgrund von Ausreißern von mehr als $400\,\mu$s, die durch Task Switches verursacht werden, liegt die Standardabweichung bei $63\,\mu$s. Die benötigte Ausführungsdauer der Kommunikationsmiddleware beträgt durchschnittlich $45\,\mu$s bei einer Standardabweichung von $3\,\mu$s. Das längste Intervall bilden lwIP-Stack und Netztreiber mit durchschnittlich $148\,\mu$s bei einer Standardabweichung von $3\,\mu$s. Dies summiert sich zu einer durchschnittlichen Zeit von $210\,\mu$s, die ein Aufruf auf dem MPC5668 Board verbringt. Dabei nimmt die Kommunikati-

onsmiddleware nur ein geringes Intervall ein.

Dies lässt sich in Bezug zur maximal erlaubten Ende-zu-Ende-Verzögerung für Steuerungsdaten setzen, wie sie in CAN und FlexRay typisch sind. Diese Anforderung ist $Delay_{\text{max,Steuerungsdaten}} <= 10ms$ [Rahm 08]. Das Ausführungsverhalten der Kommunikationsmiddleware schränkt damit auch in einer Low-Ausprägung die Anforderungen an die Ende-zu-Ende-Verzögerung im Fahrzeugbordnetz nicht ein.

5.4.4. Interoperabilität unterschiedlicher Ausprägungen

Die oben beschriebenen Versuche zeigen implizit, dass das prototypisch umgesetzte *EmbeddedC Binding*, Marshalling und De-Marshalling der Anwendungsdaten korrekt umsetzt. Der erfolgreiche Austausch zwischen dem entwickelten Binding und der eingesetzten, vorab verfügbaren Kommunikationsmiddleware validiert die Konformität des eingeschränkten funktionalen Umfangs zum vorgegebenen Apache Etch-Serialisierungsformat. Die Nachrichten wurden dazu als verteilte Funktionsaufrufe von der Head Unit an den Audio-Verstärker gesendet. Das Audio-Streaming wurde als RTP-Strom realisiert, um eine realistische Zusatzbelastung der Netzschnittstelle zu erzeugen. Trotz dieser Zusatzbelastung, die durch die Kommunikationsmiddleware manipuliert wird, aber ansonsten unabhängig von ihr ist, zeigte sich die prototypische Low-Ausprägung auf dem MPC5668 Board als uneingeschränkt interoperabler Kommunikationspartner.

5.4.5. Fazit der Evaluation

Das vorgestellte Konzept einer einfachen, binärkompatiblen Interoperabilität zwischen unterschiedlich funktionalen Ausprägungen einer Kommunikationsmiddleware ist für eine Umgebung wie das IP-basierte Fahrzeugbordnetz valide. Eine prototypische Low-Ausprägung, welche die strengen nichtfunktionalen Anforderungen in einem von eingebetteten Systemen geprägten Rechnernetz adressiert, kann die funktionalen Minimalanforderungen an typische Anwendungskommunikation erfüllen. Die Low-Ausprägung ermöglicht eine interoperable Kommunikation mit prinzipiell mächtigeren Ausprägungen innerhalb von klar definierten, gemeinsamen Funktionsumfängen. Das einfache, minimale Koordinationsmodell bildet die Basis dafür, dass die Kommunikationsmiddleware auf leistungsschwachen Steuergeräten lauffähig ist. Die Messungen zeigen dabei, dass eine prototypische Implementierung in einer realistischen Versuchsumgebung, den verfügbaren Speicher nur geringfügig belastet. Das Ausführungsverhalten liegt deutlich über dem der heute im Fahrzeug eingesetzten Vernetzungstechnologien bei gleichzeitig besserer Kommunikationsabstraktion.

Eine Kommunikationsmiddleware für das IP-basierte Fahrzeugbordnetz muss in unterschiedlichen, interoperablen Ausprägungen vorliegen, die den je-

weiligen Anforderungen und Fähigkeiten von Anwendungen bzw. Steuergeräten entsprechen. Da diese Ausprägungen auf derselben funktionalen Basis kompatibel sind, lassen sie sich als eine einzige Kommunikationsmiddleware betrachten, die trotz der heterogenen Leistungsfähigkeit für die Abstraktion der gesamten internen, IP-basierten Kommunikation eingesetzt werden kann.

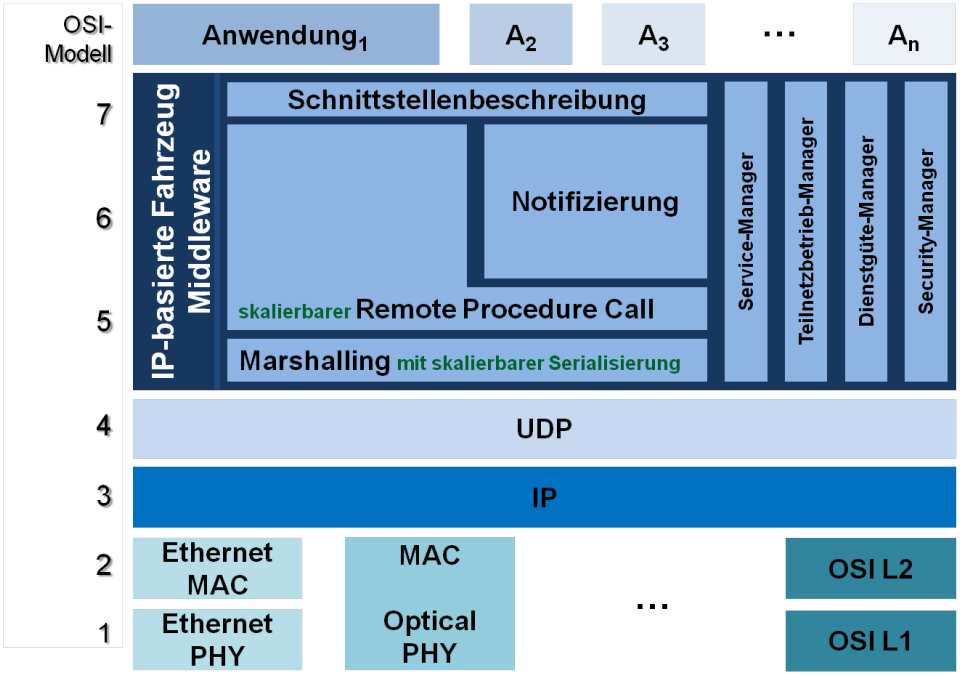

Abbildung 5.15.: Skalierbare Kommunikationsmiddleware auf Basis einer skalierbaren Serialisierung.

Abbildung 5.15 zeigt schematisch, wie verschiedene Ausprägungen einer Kommunikationsmiddleware die Basisfunktionalität bilden. Auf der Basis von gemeinsamen skalierbaren Serialisierungsregeln ermöglicht ein ebenfalls skalierbarer Kommunikationsmechanismus die Interoperabilität. Eine Schnittstellenbeschreibungssprache, die eine Kennzeichnung der Einschränkungen ermöglicht, bildet die verständliche, einheitliche Abstraktionsschicht gegenüber dem Anwendungsentwickler.

5.5. Zusammenfassung

Eine Kommunikationsmiddleware, die den Anforderungen in einem IP-basierten Fahrzeugbordnetz entspricht, muss in mehreren interoperablen Ausprägungen vorliegen. Dieses Konzept der binärkompatiblen Interoperabilität adressiert die identifizierten Widersprüche in den funktionalen und nichtfunktionalen Anforderungen, die durch die Heterogenität in der Leistungsfähigkeit der Steuergeräte besteht. Die Ausprägungen müssen den jeweiligen An-

forderungen und Fähigkeiten von Anwendungen bzw. Steuergeräten entsprechen. Gleichzeitig muss jede Anwendung im IP-basierten Fahrzeugbordnetz mit jeder anderen kommunizieren, ohne auf ein Koppelelement mit Anwendungswissen zurückzugreifen. Strukturelle Anpassungen an den funktionalen Bestandteilen ermöglichen eine einzige Kommunikationsmiddleware, welche die Interoperabilität zwischen Steuergeräten unterschiedlicher Leistungsfähigkeit im IP-basierten Fahrzeugbordnetz erlaubt.

Die Umsetzung der signal- und funktionsbasierten Kommunikationsparadigmen kann durch einen einzigen Kommunikationsmechanismus erfolgen. Das funktionsbasierte Paradigma wird direkt durch Remote Procedure Calls umgesetzt, die je nach beteiligten Ausprägungen mit dynamischen oder nur statischen Daten operieren. Das signalbasierte Paradigma wird durch ein darauf aufbauendes Notification-Konzept implementiert und ermöglicht somit eine stärker am aktuellen Informationsbedarf orientierte Umsetzung, als dies im heutigen Fahrzeugbordnetz durch das einfache Verteilen von Daten der Fall ist. Das Notification-Konzept definiert das Anmelden für bestimmte Nachrichten bei einem Sender, das entsprechende Abmelden sowie Vorschriften für Schnittstellen, die ein Abonnent implementieren muss, um abonnierte Notifications empfangen zu können. Interoperabilität wird bereits durch eine minimale Übereinstimmung der funktionalen Bestandteile möglich. Sie wird durch ein einheitliches (teils auf statische Felder eingeschränktes) Serialisierungsformat (statische Interoperabilität) und einen gemeinsamen Kommunikationsmechanismus (dynamische Interoperabilität) gewährleistet.

Das Prinzip verschiedener Ausprägungen einer Kommunikationsmiddleware, die für die Umgebung mit eingebetteten Systemen wie das Fahrzeugbordnetz geeignet sind, lässt sich anhand von drei Ausprägungen demonstrieren. Eine Ausprägung mit minimalem Umfang für leistungsschwache Steuergeräte und eine Ausprägung mit maximalem Umfang veranschaulichen das Leistungsspektrum der Kommunikationsmiddleware. Eine weitere Ausprägung dient der Veranschaulichung, wie sich eine Teilmenge der Funktionalität in das Konzept eingliedern lässt.

Das Konzept wurde durch Leistungsmessungen an einer prototypischen Implementierung evaluiert. Dazu wurde auf Basis des als Open-Source-Software verfügbaren RPC-Frameworks Apache Etch eine Low-Ausprägung entwickelt. Diese Low-Ausprägung setzt die funktionalen Anforderungen eines minimalen Koordinationsmodells mit den gegebenen Serialisierungsregeln von Apache Etch um. Aus Sicht des Apache Etch-Frameworks ist die Low-Ausprägung ein weiteres Binding, das speziell für eingebettete Systeme Code-Rümpfe und eine Laufzeitbibliothek generiert. Dieser Umfang wird als *EmbeddedC Binding* bezeichnet. Der Versuchsaufbau orientiert sich an einem Anwendungsfall eines Audio-Verstärkers. Diese Anwendung läuft auf einem Automotive-tauglichen, eingebetteten MPC5668 Fado Board, das über Ethernet mit einer emulierten Head Unit (Hauptrechner der Infotainmentdomäne) kommuniziert.

Die Kommunikationsmiddleware benötigt nur knapp 5% mehr Programm-

speicher als für einen direkten Zugriff auf die Primitive der Transportschicht benötigt würden. Für den Bedarf an Arbeitsspeicher ergibt sich ein vergleichbares Bild. Lediglich 5 kB werden von der Kommunikationsmiddleware beansprucht. Dagegen benötigen das Betriebssystem 13 kB und der IP-Stack 48 kB. Bei der Untersuchung, wie der Speicherbedarf sich in Abhängigkeit der Anzahl von Funktionsdefinitionen der Anwendungsschnittstelle verhält, zeigt sich ein lineares Anwachsen der Programmspeicherbelegung. Dies entspricht den Erwartungen, da das Konzept für ein Marshalling bei der Low-Ausprägung an jede Funktion die individuell benötigte Marshalling-Funktionalität bindet. Die durchschnittliche Ausführungszeit für eine entfernten Funktionsaufruf beträgt 450 μs. Davon verbringt der Aufruf durchschnittlich 210 μs auf dem MPC5668 Fado Board. Dieses Intervall teilt sich auf in durchschnittlich 16 μs für den eigentlichen lokalen Funktionsaufruf, durchschnittlich 45 μs für die Abarbeitung durch die Low-Ausprägung der Kommunikationsmiddleware und durchschnittlich 148 μs im lwIP-Stack und Netztreiber. Die Kommunikationsmiddleware nimmt damit nur ca. ein Viertel der Zeit für das Senden und Empfangen ein, die dreifache Zeitdauer wird durch den darunter liegenden UDP/IP-Stack benötigt. Die Kommunikationsmiddleware verursacht damit einen vernachlässigbaren Zuwachs an Speicher- und Rechenleistungsbedarf, bietet aber die gewünschte Kommunikationsabstraktion gegenüber dem Anwendungsentwickler.

Im folgenden Kapitel wird der Bezug zu allgemeinen Systemdiensten hergestellt. Es wird erklärt, wie jene entwickelt werden, um das Konzept einer funktional skalierbaren Kommunikationsmiddleware für das IP-basierte Fahrzeugbordnetz modular zu erweitern.

6. Modulare Systemdienste für globale und wiederkehrende Kommunikationsaufgaben

In diesem Kapitel werden Systemdienste vorgestellt, die das Konzept einer funktional skalierbaren Kommunikationsmiddleware für das IP-basierte Fahrzeugbordnetz erweitern. Solche Systemdienste kapseln Aufgaben, die für das gesamte Kommunikationssystem grundlegend oder für Anwendungen wiederverwendbar sind, so dass sie nur einmal im Rahmen der Kommunikationsmiddleware und nicht von jeder Anwendung selbst implementiert werden müssen. Im Folgenden werden vier Systemdienste vorgestellt, die in einem IP-basierten Fahrzeugbordnetz benötigt werden. Weil sie allesamt eine verwaltende Aufgabe erbringen, werden sie jeweils als Management-Dienst bezeichnet. Die folgenden Abschnitte beschreiben somit Service-, Teilnetzbetrieb-, Dienstgüte- und Security-Management.

6.1. Ein Service-Management

Service-Discovery oder Dienstsuche bezeichnet die Aufgabe, Dienste in einem Netzwerk dynamisch anzubieten und zu suchen. Service-Discovery auf der Ebene von Geräten, die sich im Rechnernetz an- bzw. abmelden und dabei beispielsweise eine Geräteadresse beziehen, ist im alltäglichen Betrieb eines Fahrzeugs nicht relevant. Die verbauten Steuergeräte hängen von der Ausstattung des konkreten Fahrzeugs ab und ändern sich nach der Produktion nicht mehr. Eine Dienstsuche auf Ebene von Steuergeräten ist demnach nur ein einziges Mal zur Produktionszeit relevant, aber für die Beherrschung der Variantenvielfalt auch notwendig. Ohne ein solches dynamisches Verfahren müsste für jede mögliche Ausstattungskombination für jedes Steuergerät die Menge der Kommunikationspartner gepflegt werden. Dies zur Produktionszeit zu steuern, ist aus Perspektive einer Software-Logistik sehr komplex.

Andererseits gelten für eine einmalige Dienstsuche zur Produktionszeit andere Rahmenbedingungen als für die interne Kommunikation im laufenden Betrieb des Fahrzeugs. So steht ein Diagnosezugang zur Verfügung, der auf alle Steuergeräte zugreifen kann, und der gesamte Vorgang ist nicht zeitkritisch. Die Service-Discovery zur Feststellung der in einer konkreten Ausstattung vorhandener Anwendungen bzw. Steuergeräte muss nicht von einer Kommunikationsmiddleware erbracht werden, die ansonsten für die spezifischen Anforderungen

der internen Fahrzeugkommunikation entworfen wurde. Das dynamische An- und Abmelden von Steuergeräten wird im Folgenden daher nicht betrachtet. Im Folgenden wird davon ausgegangen, dass die Dienste auf den verbauten Steuergeräten prinzipiell schon bekannt sind.

Die Motivation für einen Systemdienst für ein Service-Management im Fahrzeug ergibt sich daher auf anderer Ebene. Es gibt für das Fahrzeugbordnetz verschiedene Betriebsmodi, die sich allgemein aus dem Fahrzeugzustand ergeben. Ist das Fahrzeug abgeschlossen, dann sind in der Regel alle Steuergeräte ausgeschaltet. Beim Öffnen des Fahrzeugs werden erste Steuergeräte gestartet, damit die Zeitdauer zum Aufstarten vor dem Fahrer verborgen bleibt. Typische Aufstartzeiten von leistungsschwachen Steuergeräten liegen bei 1-3 Sekunden. Der Hauptrechner der Infotainmentdomäne (Head Unit) benötigt 10-15 Sekunden bis er bedienbar und bereit für die Kommunikation mit anderen Steuergeräten ist.

Daneben werden noch die Zustände „Zündung an“ und „Motor an“ unterschieden. Diese Fahrzeugzustände führen bereits dazu, dass Steuergeräte und Dienste zu einem gegebenen Zeitpunkt verfügbar oder nicht verfügbar sein können. Ein Teilnetzbetrieb, wie weiter oben beschrieben, erweitert diese Anforderung, so dass auch im laufenden Betrieb, Dienste in einen Status wechseln, in dem sie vorübergehend nicht verfügbar sind. Lediglich im Fall einer External-Ausprägung der Kommunikationsmiddleware ist es möglich, dass auch ein Steuergerät (in diesem Fall ein CE-Gerät des Fahrers) dynamisch ins Fahrzeugbordnetz hinzukommt. Dies geschieht aber an einem bekannten, zentralen Zugangspunkt und kann entsprechend gekapselt werden. Ein Beispiel für mehrere externe Geräte, die einen Dienst gemäß einer vorab bekannten Anwendungsschnittstelle anbieten, sind Bildschirme, die über eine IP-basierte Kommunikationsschnittstelle verfügen. Auf solchen *IP-Displays* könnten Audio- und Video-Daten per Streaming über das IP-basierte Fahrzeugbordnetz ausgegeben werden.

Das Service-Management im IP-basierten Fahrzeugbordnetz muss also keine unbekannten Dienste anhand einer Dienstbeschreibung finden, sondern den Status von Diensten erkennen und vermitteln. Ein aktiv werdender Dienst führt hierzu ein sogenanntes Announcement im Netzwerk durch, in dem er das Vorhandensein des Dienstes ankündigt. Die eigentliche Dienstanfrage erfolgt, abhängig von der Architektur, direkt per Anfrage an ein zentrales Dienstverzeichnis oder als Broadcast im Rechnernetz. Der suchende Teilnehmer sendet dazu einen Identifikator des gewünschten Dienstes. Wird der gesuchte Dienst als aktiv zurückgemeldet, kann der Teilnehmer die eigentliche Interaktion mit dem Dienst direkt starten. Für den Fall, dass ein verbundener Dienst nicht mehr verfügbar ist, müssen geeignete Strategien zur Erkennung und Fehlerbehandlung implementiert werden, beispielsweise ein erneutes Starten der Dienstsuche.

Apache Etch besitzt keinen Systemdienst für Service-Management. Es gibt grundsätzlich zwei Ansätze, um dies zu ergänzen. Erstens kann man ein

bestehendes, dediziertes Konzept wie Zeroconf [Zero], beispielsweise in der Implementierung von Apple namens *Bonjour*, adaptieren und nutzen. Bonjour erscheint aufgrund seiner relativen Einfachheit für die Nutzung im IP-basierten Fahrzeugbordnetz geeignet [Verv 08]. Im zweiten Ansatz wird das RPC-Framework selbst um ein Service-Management erweitert, das auf den verfügbaren Funktionsaufrufen basiert. Im architekturalen Aufbau von Apache Etch kann dies als eigenständiger Dienst erfolgen, der an andere Dienste gebunden werden kann. Damit wird es möglich, die Funktionalität des Service-Managements gemäß den identifizierten Anforderungen umzusetzen, um auch diese Prinzipien besser zu verstehen und demonstrieren zu können.

Dazu wurde eine prototypische Implementierung eines *Apache Etch Service-Managements* entwickelt. Dies umfasst die Speicherung der angebotenen Dienste, eine Behandlung von Broadcast-Nachrichten, eine Anmeldung an einem zentralen Dienstverzeichnis und das Verwalten von verschiedenen Dienstzuständen. Ein Teilnetzbetrieb erfordert die Unterscheidung von „Dienst schläft“ und „Dienst ist nicht verfügbar“ sowie einen standardisierten Mechanismus zum „Aufwecken“ des Dienstes bzw. des zugeordneten Steuergeräts. Das Schlafen legen kann dabei über die vorhandenen Kommunikationsmechanismen der Kommunikationsmiddleware erfolgen.

Für ein Konzept, das ein Service-Management gemäß den beschriebenen Rahmenbedingungen im Fahrzeugbordnetz beschreibt, lassen sich grundsätzlich zwei unterschiedliche Ansätze anwenden: Ein zentrales und ein dezentrales Service-Management.

Der erste Ansatz basiert auf einem zentralen Dienstverzeichnis. Das Dienstverzeichnis ist dabei eine Anwendung, die den Zustand sämtlicher bekannter Dienste im IP-basierten Fahrzeugbordnetz kennt und überwacht. Die Anwendung, welche die Rolle des Dienstverzeichnisses einnimmt, muss dabei auf einem Steuergerät laufen, das idealerweise kurze Aufstartzeiten hat und selbst kein Kandidat für einen Ruhezustand im Rahmen eines Teilnetzbetriebs ist. Als Aufstart eines Steuergeräts wird dabei der Übergang von einem nicht-betriebsbereiten in einen betriebsbereiten Zustand bezeichnet. Dies erfolgt typischerweise beim Fahrzeugstart („Zündung an“).

Daneben ist ein dezentraler Ansatz möglich, bei dem jede Anwendung selbst ein eigenes Dienstverzeichnis pflegt, das alle für die Anwendung relevanten Dienste umfasst. Die Kommunikation sich ändernder Dienstzustände muss ohne koordinierende Komponenten über Broadcasts erfolgen.

Die beiden Ansätze bieten unterschiedliche Vorteile, wie sie allgemein typisch für den Vergleich zwischen zentralen und dezentralen Verfahren sind. Der zentrale Ansatz nutzt eine Komponente, die alle Dienstzustände kennt. Anfragen werden stets nur an diese bekannte Komponente gestellt. Damit treten keine Inkonsistenzen auf, der am zentralen Dienstverzeichnis bekannte Dienstzustand einer Anwendung ist gültig. Die Announcements und Anfragen sind Unicasts zwischen einer Anwendung und dem Dienstverzeichnis. Es werden somit keine ineffizienten Broadcasts benötigt. Außerdem ist für den Weckmechanismus ei-

ne zentrale Komponente zwingend notwendig, die jenen außerhalb der üblichen Kommunikationswege auslösen kann.

Ein dezentraler Ansatz schafft Unabhängigkeit von einer zentralen Komponente. Ein zentrales Dienstverzeichnis kann ausfallen oder temporär nicht erreichbar sein, vor allem wenn es beim Aufstart des Fahrzeugbordnetzes mehr Zeit benötigt als andere Anwendungen. Letzteres ist sogar wahrscheinlich. Eine Voraussetzung des zentralen Dienstverzeichnisses verlangt, dass es auf einem Steuergerät läuft, das selbst kein Kandidat für den Teilnetzbetrieb ist. In diese Kategorie fallen leistungsstarke Steuergeräte, die mehrere Anwendungen beherbergen und dafür mehr Zeit für den Aufstart benötigen. Der dezentrale Ansatz ermöglicht dann bereits Kommunikation zwischen schnell startenden Anwendungen, ohne dass bereits der gesamte Verbund bereit ist.

Eine Kombination beider Ansätze nutzt die jeweiligen Vorteile. Deshalb wurde prototypisch einen hybriden Ansatz implementiert. Das *Apache Etch Service-Management* kombiniert die Vorteile beider Ansätze: Es ermöglicht schnelle Kommunikation beim Aufstart ohne zentrales Dienstverzeichnis, danach ein einfaches Verwalten der Dienste mit einem zentralen Dienstverzeichnis. Um dem Nachteil von in einem Switched Ethernet ineffizienten Broadcasts zu begegnen, soll dabei folgende Strategie implementiert werden: Wenn möglich wird eine Zustandsmeldung eines Dienstgebers bzw. eine Zustandsabfrage eines Dienstnehmers stets per Unicast über das zentrale Dienstverzeichnis abgewickelt. Die Möglichkeit über Broadcasts zu kommunizieren wird nur genutzt, wenn das zentrale Dienstverzeichnis (noch) nicht verfügbar ist.

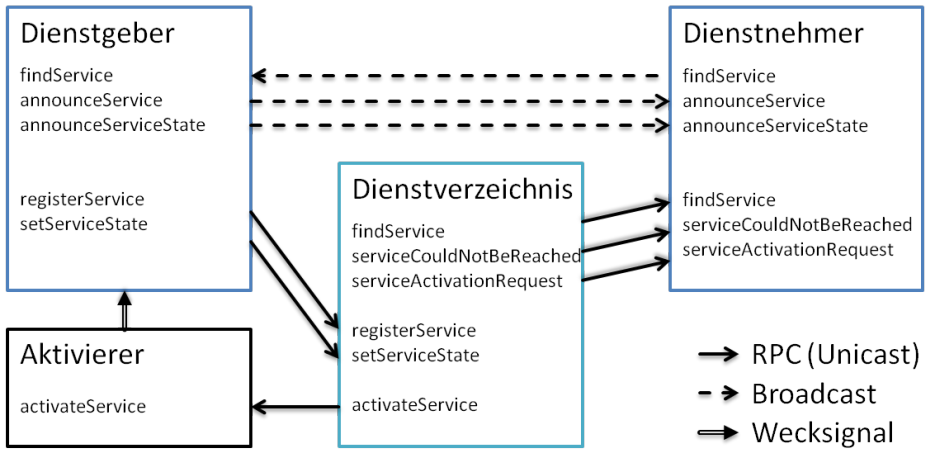

Abbildung 6.1.: Architektur des prototypischen Service-Managements.

Abbildung 6.1 zeigt die Architektur des entwickelten *Apache Etch Service-Managements*. Dienstgeber und Dienstnehmer sind dabei Rollen, eine Anwendung kann im Allgemeinen sowohl Dienstgeber als auch Dienstnehmer sein. Ein Dienstnehmer verwaltet dabei ein dezentrales Dienstverzeichnis (nicht explizit in der Grafik aufgeführt), das die Dienstzustände der relevanten Dienste überwacht. Das aufgeführte Dienstverzeichnis ist eine im IP-basierten Fahr-

zeugbordnetz zentrale Anwendung, welche die Betriebszustände aller Dienstgeber kennt und entsprechende Anfragen von Dienstnehmern beantwortet. Der Dienstgeber macht nach der Initialisierung seinen Status im gesamten IP-basierten Rechnernetz mittels Broadcast bekannt. Diese Informationen bekommen sowohl das Dienstverzeichnis als auch alle interessierten Dienstnehmer. Falls ein Dienstnehmer nicht bereits über einen solchen Broadcast den Status eines relevanten Dienstgebers bekommt, bieten sich ihm mehrere Alternativen zur Dienstsuche: Er kann einen Dienst dezentral per Broadcast suchen oder er kann zentral eine direkte Anfrage an das Dienstverzeichnis richten. Da ein Dienstgeber gegebenenfalls erst nach dem Dienstnehmer aktiv wird, funktioniert die zweite Möglichkeit auch asynchron. Das Dienstverzeichnis speichert eine nicht erfüllbare Anfrage und meldet einen aktiv werdenden Dienst einem entsprechenden Dienstnehmer.

Die Vorteile der hybriden Lösung werden dabei vor allem bei einem gleichzeitigen Aufstart aller Steuergeräte sichtbar, wie er bei Inbetriebnahme des Fahrzeugs typisch ist. Als Evaluation zeigt Tabelle 6.1 das Verhalten der hybriden Lösung abhängig von der Reihenfolge, in der ein ausgesuchter Dienstgeber (DG), ein ausgesuchter Dienstnehmer (DN) und das zentrale Dienstverzeichnis (DV) ihre Betriebsbereitschaft erlangen. Da die betrachteten Komponenten unabhängig voneinander aufstarten, ergeben sich $3! = 6$ mögliche kombinierte Reihenfolgen. Die Strategie ist, Unicasts an das Dienstverzeichnis gegenüber Broadcasts im gesamten IP-basierten Fahrzeugbordnetz zu bevorzugen. Dafür wird angegeben, wie viele Unicasts bzw. Broadcasts idealerweise notwendig sind, bis der beteiligte Dienstnehmer und das zentrale Dienstverzeichnis den Status des Dienstgebers kennen. Dabei zählt eine Unicast immer als synchroner Aufruf mit Anfrage und zugehöriger Bestätigung bzw. Antwort.

Da das zentrale Dienstverzeichnis davon ausgehen muss, dass es bereits Nachrichten verpasst hat, wird es seine Bereitschaft stets mit einem Broadcast im gesamten IP-basierten Fahrzeugbordnetz verbreiten. Dieser eine Broadcast wird in der Tabelle aus zwei Gründen zur Vereinfachung der Darstellung nicht aufgeführt. Erstens findet er unabhängig der Reihenfolge genau einmal statt und würde daher in allen Kombinationen einmal vorkommen. Zweitens stellen Dienstnehmer und Dienstgeber Rollen dar, die mehrfach vorhanden sind. Für die Bewertung des Broadcast-Bedarfs durch das Service-Management fällt das zentrale Dienstverzeichnis als einzelne Anwendung daher nicht ins Gewicht.

Tabelle 6.1 strukturiert die Betrachtung. Die Reihenfolge (DV, DG, DN) beschreibt dabei den Idealfall. Der Dienstgeber meldet seinen Status direkt beim bereiten Dienstverzeichnis, der Dienstnehmer erfragt den bekannten Status direkt beim Dienstverzeichnis. Wenn ein Dienstnehmer vor dem zentralen Dienstverzeichnis bereit wird ((DN, DV, DG), (DG, DN, DV) und (DN, DG, DV)), muss er einen Broadcast senden, weil es möglich ist, dass der gesuchte Dienstgeber bereits aktiv ist. Wenn ein Dienstgeber vor dem zentralen Dienstverzeichnis bereit wird ((DG, DV, DN), (DG, DN, DV) und (DN, DG, DV)), muss er einen Broadcast senden, weil es möglich ist, dass bereits ein Dienst-

Reihenfolge	Verhalten	Multicasts / Broadcasts
DV, DG, DN	DV meldet seine Bereitschaft. DG sendet seinen Status an DV. DN erfragt den Status beim DV und bekommt eine positive Antwort.	2 / 0
DV, DN, DG	DV meldet seine Bereitschaft. DV bekommt eine Anfrage des DN und merkt diese vor, da DG nicht bekannt ist. DG sendet seinen Status an DV. DV sendet danach eine asynchrone Nachricht bezüglich des Status an DN.	3 / 0
DG, DV, DN	DG erreicht DV nicht. Daraufhin sendet es einen Broadcast. DV meldet seine Bereitschaft. DV bekommt den Status des DG. DN erfragt den Status beim DV und bekommt eine positive Antwort.	3 / 1
DN, DV, DG	DN erreicht DV nicht. Daraufhin sendet es einen Broadcast, der unbeantwortet bleibt. DV meldet seine Bereitschaft. DN erfragt den Status beim DV, jenes merkt sich die Anfrage vor. DG sendet seinen Status an DV. DV sendet danach eine asynchrone Nachricht bezüglich des Status an DN.	4 / 1
DG, DN, DV	DG erreicht DV nicht. Daraufhin sendet es einen Broadcast. DN erreicht DV nicht. Daraufhin sendet es einen Broadcast, der von DG direkt beantwortet wird. DV meldet seine Bereitschaft. DG sendet seinen Status an DV.	4 / 2
DN, DG, DV	DN erreicht DV nicht. Daraufhin sendet es einen Broadcast, der unbeantwortet bleibt. DG erreicht DV nicht. Daraufhin sendet es einen Broadcast, der von DN bestätigt wird. DV meldet seine Bereitschaft. DG sendet seinen Status an DV.	4 / 2

Tabelle 6.1.: Verhalten der hybriden Lösung abhängig vom Aufstartverhalten eines Dienstgebers (DG), eines Dienstnehmers (DN) und des zentralen Dienstverzeichnisses (DV).

nehmer wartet. In den beiden ungünstigsten Fällen ((DG, DN, DV) und (DN, DG, DV)) sind demnach zwei Broadcasts notwendig, wobei der jeweils erste unbeantwortet bleibt.

In dieser Analyse des Verhaltens abhängig von der Aufstartreihenfolge stel-

len Dienstnehmer und Dienstgeber Rollen dar. In einem konkreten Fahrzeug kommen diese Rollen für mehrere Anwendungen und damit Steuergeräten zum Einsatz. Alle Steuergeräte, die schneller aufstarten als dasjenige, das das zentrale Dienstverzeichnis beherbergt, müssen über Broadcasts die Dienstbereitschaft bekanntgeben bzw. erfahren. Es ist daher kritisch, dass das zentrale Steuergerät nicht nur auf einem Steuergerät läuft, das selbst niemals schläft, sondern auch schnell aufstartet. Da es sonst keine weiteren Anforderungen an das zentrale Dienstverzeichnis bezüglich der Partitionierung im Fahrzeugbordnetz gibt, außer dass es den anderen Steuergeräten statisch bekannt ist, wählt man im Idealfall aus der Menge der nicht-schlafenden Steuergeräte dasjenige, mit der kürzesten Aufstartzeit.

6.2. Ein Teilnetzbetrieb-Management

Ein effizientes Energie-Management kann zur Reduzierung des Kraftstoffverbrauchs bzw. bei einem Elektrofahrzeug zur Erhöhung der Reichweite beitragen. Dazu werden temporär unbenutzte Steuergeräte schlafen gelegt und erst im Bedarfsfall wieder geweckt. Dafür ist ein kontrollierter Übergang in einen energiesparenden Ruhemodus notwendig, so dass die beherbergte Anwendung nach dem Aufwecken möglichst nahtlos weiterarbeiten kann. Die Verwaltung und Kommunikation verschiedener Dienstzustände wird bereits durch das oben beschriebene Service-Management angeboten. Dies beschreibt aber nur, wie ein Teilnetzbetrieb aus Kommunikationsperspektive hergestellt werden kann bzw. wie man Steuergeräte wieder reaktiviert. Es bleibt aber die Frage, wann ein Teilnetzbetrieb möglich ist, im Sinne einer Prüfung, ob ein Schlafenlegen von Steuergeräten sicher möglich ist. Außerdem müssen nach dem Aufwecken von Steuergeräten gegebenenfalls Kommunikationsbeziehungen erneut aufgebaut werden.

Ein Steuergerät ist teilnetzbetriebfähig, wenn es in einen Ruhezustand gesetzt (Schlafenlegen) und über einen externen Weckmechanismus wieder in den Betriebszustand gesetzt werden kann. Implizit darf es keinen Dienst auf dem betreffenden Gerät geben, der jederzeit aktiv sein muss. Das Schlafenlegen eines Geräts könnte prinzipiell über inhärente Kommunikationsmechanismen geschehen, falls das Security-Konzept den Aufruf einer entsprechenden Betriebssystemfunktion zulässt. Die Frage ist dann allerdings, welcher Dienst (auf einem fremden Steuergerät) darüber entscheiden darf, dass sich ein Steuergerät schlafen legt. Diese Beziehungen müssen zur Entwicklungszeit festgelegt werden. So kann beispielsweise eine Rückfahrkamera sowohl durch eine Nutzerinteraktion über die Mensch-Maschine-Schnittstelle, als auch durch das Einlegen des Vorwärtsgangs deaktiviert werden. Das Kamera-Steuergerät kann sich daraufhin selbst schlafen legen. Vor dem Schlafenlegen muss diese Absicht an den Teilnetzbetrieb-Manager oder mögliche Dienstnehmer über inhärente Kommunikationsmechanismen bekannt gegeben werden.

Die Umsetzung einer Komponente, die einen Teilnetzbetrieb im Sinne aktiver

Änderungen der Betriebszustände von Steuergeräten verwaltet, ist dagegen nicht zweckmäßig. Es gibt keinen Grund, den aktuellen Bedarf an aktiven Anwendungen zentral zu sammeln und zu kontrollieren. Damit bleiben für ein Teilnetzbetrieb-Management lediglich zwei funktionale Bestandteile. Erstens das Aufwecken von schlafenden Diensten und zweitens das Überwachen von Dienstzuständen. Das Aufwecken kann bei einem Steuergerät im Ruhezustand, dessen Kommunikationsschnittstelle inaktiv ist, nicht über die gewöhnlichen Kommunikationsmechanismen erfolgen. Dafür gibt es diverse Konzepte, wie ein Aufwecken signalisiert werden kann. Die Lösungen reichen vom Verbau einer speziellen Weckleitung über elektronische Sicherungen bis zu speziellen Verfahren über die Vernetzungstechnologie.

Somit wird der Teilnetzbetrieb-Manager einem zentralen Dienstverzeichnis im IP-basierten Fahrzeugbordnetz gleichgesetzt, das über die Kommunikationsmiddleware ansprechbar ist und über dedizierte Weckmechanismen ein schlafendes Steuergerät aufwecken kann.

6.3. Ein Dienstgüte-Management

Die Einführung des Internet Protocols für die fahrzeuginterne Kommunikation ermöglicht die flexible Nutzung einer Vernetzungstechnologie. Eine typische IP-basierte Vernetzungstechnologie wie Ethernet wird für verschiedene Arten von Daten genutzt werden, sowohl für die Steuerung entfernter Anwendungen als auch für die Übertragung von Multimedia-Daten. Die zugrundeliegende Vernetzung soll im IP-basierten Fahrzeugbordnetz für den Anwender durch die Kommunikationsmiddleware transparent gehalten werden. Jene muss daher eine Möglichkeit bieten, die Dienstgüte für eine Kommunikationsverbindung zu kennzeichnen. Die Dienstgüte muss schließlich auf die konkrete Vernetzungstechnologie umgesetzt werden.

In Lim, Weckemann und Herrscher [Lim 11c] wurden Dienstgüteanforderungen für ein Switched Ethernet zur IP-basierten Kommunikation im Fahrzeug evaluiert. Darin werden vier Dienstgüteklassen unterschieden. Die höchste Priorität ist für Steuerungsdaten vorgesehen, wie sie in CAN und FlexRay typisch sind. Dies entspricht den Funktionsaufrufen der Kommunikationsmiddleware selbst. Zweitens haben von Kameras erzeugte Datenströme eine hohe Priorität, da sie für Fahrerassistenzsysteme die sicherheitskritische Fahraufgabe unterstützen. Danach folgen als dritte Klasse sonstige Video- und Audio-Ströme und als vierte Klasse alle weiteren nicht-zeitkritischen Massendaten (Bulk Data). Eine vergleichbare Einteilung findet sich auch bei Steffen et al. [Stef 10] und bei Rahmani et al. [Rahm 09].

Typische Angaben, welche Dienstgüte in der internen Fahrzeugkommunikation gefordert wird, betreffen die Ende-zu-Ende-Verzögerung (End-to-End Delay) $Delay_{\max}$, also die maximale Zeitdauer, die für die Datenübertragung erlaubt ist. Tabelle 6.2 listet diese maximale Zeitdauer für die unterschiedlichen Klassen auf. Für Steuerungsdaten liegt diese Anforderung bei

Datenklasse	Ende-zu-Ende-Verzögerung
Steuerungsdaten	$Delay_{\text{max,Steuerungsdaten}} <= 10ms$
Kameradaten	$Delay_{\text{max,Kameradaten}} <= 45ms$
Multimedia-Daten	$Delay_{\text{max,Multimedia-Daten}} <= 150ms$

Tabelle 6.2.: Maximal erlaubte Ende-zu-Ende-Verzögerung für verschiedene Datenklassen.

$Delay_{\text{max,Steuerungsdaten}} <= 10ms$, für Kameradaten bei $Delay_{\text{max,Kameradaten}} <= 45ms$ [Rahm 08]. Für Multimedia-Daten wird von einer maximal erlaubten Ende-zu-Ende-Verzögerung von $Delay_{\text{max,Multimedia-Daten}} <= 150ms$ ausgegangen [Wolf 97].

Bakken beschreibt in seinem Enzyklopädie-Eintrag „Middleware“ [Bakk 01] den Zusammenhang zwischen Kommunikationsmiddleware und Dienstgüte. Demnach soll die Kommunikationsmiddleware die Dienstgüteanforderungen auf hoher Ebene von Anwendungen entgegennehmen und dann an einen Ressourcen-Manager auf einer niederen Ebene weitergeben. Kommunikationsmiddleware eignet sich als Anwendungsabstraktion der Dienstgüte. Dieselbe Abstraktion, die sie Anwendungen gegenüber für die Basisfunktionalität zur Kommunikation im verteilten System erzielt, können auf Dienstgüteanforderungen ausgedehnt werden. Dazu werden grundlegende Dienstgüteanforderungen generalisiert und damit unabhängig von den auf den Vernetzungstechnologien angewandten Verfahren gemacht. Dies erlaubt auch aus Perspektive der Dienstgüte eine Austauschbarkeit von Vernetzungstechnologien und Dienstgütemechanismen.

Für die Priorisierung von Dienstklassen gibt es prinzipiell zwei Ansätze. Erstens kann eine Priorisierung durch eine statische Konfiguration der Vernetzungstechnologie erreicht werden. Am Beispiel von Ethernet kann dies beispielsweise durch die Nutzung unterschiedlicher Ports an den Switches mit jeweils fest zugeordneter Priorität erfolgen. Nachteilig ist dabei, dass auf diese Weise nur Steuergeräte und nicht feingranular auf Anwendungs- oder gar Funktionsebene differenziert werden kann. Dies ist im zweiten Ansatz durch eine Weitergabe der Anforderung im Protokoll-Stack nach unten über die Vermittlungsschicht möglich. Eine konkrete Dienstgüteanforderung wird dazu durch den Protokollstack nach unten gereicht und erst von der genutzten Vernetzungstechnologie aufgelöst. Dazu sind für jede genutzte Vernetzungstechnologie und für jede abstrakte Dienstgüteanforderung entsprechende Protokolle notwendig.

Die Beschreibung der Dienstgüteanforderungen kann gemeinsam mit der Definition der Funktionsaufrufe in der Schnittstellenbeschreibungssprache erfolgen. Auf diese Weise kann der Anwendungsprogrammierer die Kommunikationsfunktionalität selbst und die Eigenschaften der Kommunikation in einer einzigen zentralen Definition pflegen. Die Umsetzung der abstrakten Dienstgüteanforderungen soll aber als eigenständiger Systemdienst gekapselt werden.

Dieser übersetzt die Dienstgüteanforderungen auf die Vernetzungstechnologien für das IP-basierte Fahrzeugbordnetz.

Der Ansatz der Trennung von Kommunikationsschnittstelle und Dienstgüte-Management ermöglicht das unabhängige Vornehmen von Dienstgüteeinstellungen für Anwendungen während des Aufstarts durch einen eigenständigen Task. Außerdem können Anwendungen ohne spezielle Dienstgüteanforderungen bereits vor dem Setzen bzw. der Überprüfung von Dienstgüteeinstellungen eingesetzt werden. Die Kapselung als eigenständiger Systemdienst ermöglicht noch eine weitere Möglichkeit. Wird eine Datenübertragung nicht über entfernte Aufrufe mithilfe der Kommunikationsmiddleware initialisiert, sondern durch eigenständige Anwendungsschichtprotokolle wie RTP für die Übertragung von Kamera-Strömen erbracht, so kann zumindest die Dienstgüteanforderung für die Verbindung über eine direkte Ansprache des Dienstgüte-Managements delegiert werden.

Werden heute übliche Vernetzungstechnologien im Fahrzeug für die Übertragung von IP-Paketen genutzt, sind die Möglichkeiten für die Erbringung einer bestimmten Dienstgüte gering. So nutzt beispielsweise CAN als priorisierter Bus für die Priorisierung statisch vergebene Identifikatoren. Andererseits würde die Bandbreite für die Übertragung von Kamera- oder Multimedia-Daten auch nicht ausreichen, weswegen eine Unterscheidung ohnehin nicht anwendbar ist. Für eine Vernetzungstechnologie wie Ethernet ergibt sich aber die Fragestellung, wie Dienstgüte in einem Switched Ethernet für die interne Fahrzeugkommunikation garantiert werden kann. Eine ausführliche Behandlung der Anforderungen und Verfahren auf Ebene der Vernetzungstechnologie finden sich in mehreren Beiträgen von Lim [Lim 11a, Lim 11b, Lim 11c, Lim 12].

6.4. Ein Security-Management

Der Schutz von Informationen im Fahrzeug betrifft nicht nur die Kommunikation, sondern auch das Verwalten von gespeicherten Daten. Mittels durchgängiger Vernetzung über das Internet Protocol und die Öffnung der Kommunikation nach außen entstehen neue Angriffsszenarien. Mögliche Angriffsziele sind die Manipulation von Daten und Funktionen durch das Einschleusen von Schad-Software, das Freischalten von nicht autorisierten Funktionen sowie das Ausspähen privater Informationen. Die Fragestellung nach einer geeigneten Herstellung einer sicheren Kommunikation lassen sich nur in Bezug auf ein generelleres Security-Framework beantworten. Die grundlegenden Überlegungen hierzu wurden in Weckemann und Weyl [Weck 10] bereits veröffentlicht.

Grundlegende Forschung für ein Security-Framework im Fahrzeug wurde im Rahmen des EVITA-Projekts [Weyl 10] beschrieben. Im EVITA-Projekt wurde eine Architektur entwickelt, die modulare Security-Dienste für Steuergeräte im Fahrzeug bietet. Dazu werden Steuergeräte einerseits Zonen unterschiedlichen Schutzbedarfs zugeordnet. Andererseits wird auch hier beachtet, dass die Leistungsfähigkeit von Steuergeräten teils stark eingeschränkt ist. Die Security-

Dienste können für unterschiedlich leistungsfähige Steuergeräte und unterschiedliche Security-Zonen adaptiert werden. Da sich mit einem modularen Security-Baukasten die verfügbaren Protokolle und Security-Mechanismen von Gerät zu Gerät unterscheiden, muss für je zwei Geräte ein Protokoll selektiert werden, das beide Kommunikationspartner implementieren. Der Schutz einer Verbindung zwischen zwei Kommunikationspartnern erfolgt mit dem Security-Protokoll, das die gewünschten Security-Eigenschaften erfüllt und auf beiden Kommunikationspartnern implementiert ist. Falls mehrere Protokolle diese beiden Bedingungen erfüllen, wird das höherwertige Protokoll verwendet.

Da die Anwendungsschnittstellen als Teil der Kommunikationsmiddleware in der IDL definiert werden, muss die Kommunikationsmiddleware die geforderten Security-Eigenschaften für eine Verbindung am Security-Framework anmelden. Dies erlaubt prinzipiell eine feingranulare Unterscheidung aus Security-Perspektive bis auf einzelne Funktionsaufrufe. Aus Sicht der Kommunikationsmiddleware werden aus einer definierten Schnittstelle nicht nur Programmcoderahmen für die eigentliche Kommunikation generiert. Es wird ebenfalls ein Schutz für diese Verbindung etabliert. Dazu muss das Security-Framework aussagen, welche Security-Protokolle zwischen zwei konkret gewählten Kommunikationspartnern möglich sind. Das Security-Framework stellt dafür sichere Kommunikationswege über unterschiedliche Protokolle bereit. Die Kommunikationsmiddleware ermöglicht die einheitliche Beschreibung der geforderten Security-Eigenschaften gemeinsam mit der Schnittstellenbeschreibung.

Die EVITA-Security-Architektur bietet modulare Security-Dienste, welche an die Bedürfnisse der Zonen innerhalb eines Fahrzeugbordnetzes angepasst und konfiguriert werden können. Diese Security-Dienste bieten dem Steuergeräte- und Anwendungsentwickler zonenübergreifend einheitliche Schnittstellen an, die aus Perspektive der Kommunikationsmiddleware über einen *Security-Stub* verwendet werden können. Das Security-Framework adaptiert die bereitgestellten Dienste entsprechend der geforderten Security-Eigenschaften. Dazu werden Security-Dienste von Security-Schnittstellen gekapselt, die wiederum konkrete Mechanismen als Plug-Ins realisieren. Neben der einheitlichen zonenübergreifenden Integration von Mechanismen über die vorgegebenen Schnittstellen hat eine solche modulare Security-Architektur den Vorteil, dass Mechanismen bedarfsgerecht durch die Wahl geeigneter Plug-Ins für die jeweiligen Zonen konfiguriert werden können, die Schnittstellen zur Applikation für den Entwickler aber gleich bleiben.

Außerdem ermöglicht der Plug-In-Mechanismus ein dynamisches Ersetzen von Security-Mechanismen, das ohne Auswirkungen auf die Anwendungen erfolgt. Diese Austauschbarkeit von Security-Mechanismen ist im Zusammenhang mit den kurzfristigen Zyklen aus Entdeckung von Security-Schwachstellen und entsprechenden Verbesserungen der Mechanismen unausweichlich [Weyl 10].

Abbildung 6.2 veranschaulicht, dass die Security-Anforderungen einer Anwendung in der eigentlichen Kommunikationsschnittstelle definiert werden,

während die entsprechenden Security-Mechanismen durch ein eigenständiges Security-Framework erbracht werden.

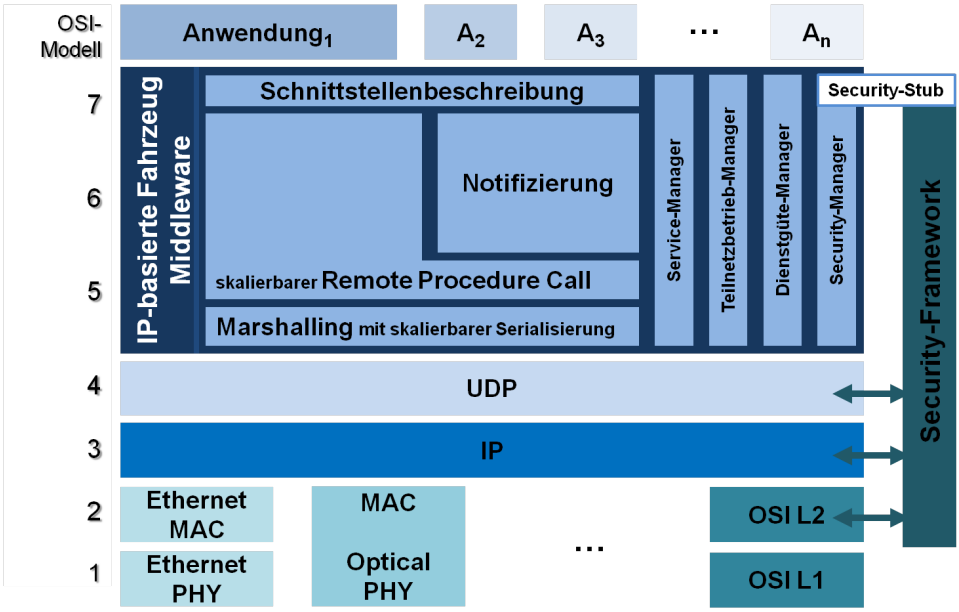

Abbildung 6.2.: Security-Management der Kommunikationsmiddleware auf Basis der Funktionalität eines modularen Security-Frameworks.

Die Kommunikationsmiddleware bietet damit Anwendungen einheitliche Security-Schnittstellen indirekt über den *Security-Stub* an. Dieser greift auf die Module zu, die vom Security-Framework angeboten werden. Werden Module auch auf den OSI-Schichten zwei bis vier benötigt, etwa zum Aufbau eines sicheren Kanals, so können Funktionen des Security-Frameworks auch direkt aufgerufen werden. Dieses modulare und skalierbare Security-Framework ergänzt die vorgestellte Kommunikationsmiddleware in unterschiedlichen Ausprägungen, weil es ebenfalls sowohl den Einschränkungen des jeweiligen Steuergeräts als auch den Security-Anforderungen der darauf laufenden Anwendungen gerecht werden kann.

Bouard et al. greifen in „Driving Automotive Middleware Towards a Secure IP-based Future“ [Boua 13] den Ansatz einer modularen Kommunikationsmiddleware und eines entsprechend skalierbaren, modularen Security Frameworks auf. Aus Sicht des Security-Frameworks übernimmt ein *Security Abstraction Layer* die Zuteilung von Anwendungskommunikation zu geeigneten Erweiterungen. Dafür muss bei der Beschreibung der Anwendungsschnittstellen eine Menge von Security-Anforderungen annotiert werden. Der *Security Abstraction Layer* bietet wiederum der Kommunikationsmiddleware eine definierte Anwendungsschnittstelle, an der für gegebene Security-Anforderungen die geeigneten Security-Protokolle eingerichtet werden. Dazu löst der *Security Abstraction Layer* die Abhängigkeiten zwischen den beteiligten Steuergeräten

auf, um das für die Kommunikationspartner gemeinsam verfügbare, am besten geeignete Protokoll zu bestimmen.

Das Konzept interoperabler Ausprägungen der Kommunikationsmiddleware für die interne Kommunikation im IP-basierten Fahrzeugbordnetz lässt sich auf die Kommunikation mit angebundenen CE-Geräten erweitern. Analog dazu erweitern Bouard et al. in „Automotive Proxy-Based Security Architecture for CE Device Integration“ [Boua 12] auch das Konzept eines skalierbaren Security-Frameworks. Dazu findet an einem ausgezeichneten Proxy eine strikte Entkopplung zwischen interner und externer Kommunikation statt. Auf diese Weise können nach außen in der CE-Industrie übliche und nach innen für die spezifischen Anforderungen im Fahrzeug angepasste Security-Protokolle verwendet werden, welche dieselben Security-Eigenschaften auf unterschiedlichen Wegen herstellen.

6.5. Allgemeine Anwendungsabstraktion für Systemdienste

Die beschriebenen Systemdienste übernehmen dedizierte Aufgaben im IP-basierten Fahrzeugbordnetz. Aus der vorgestellten Architektur und den Konzepten, wie sie ihre zugedachte Aufgabe erfüllen, ergeben sich die Möglichkeiten zur Nutzung der Basisfunktionalität der Kommunikationsmiddleware. Dienstgüte und Security-Management können aus derselben Anwendungsschnittstelle abgeleitet werden, wie die eigentliche Kommunikationsverbindung. Durch die Beschreibung von Anforderungen an die Security-Eigenschaften einer Verbindung und entsprechend von Dienstgüteanforderungen in einer gemeinsamen Definition mit den entfernten Funktionsaufrufen, wird eine zentrale Möglichkeit zur Definition der Kommunikationsbeziehung geschaffen. Die Schnittstelle beschreibt sowohl den eigentlichen entfernten Aufruf und annotiert an gleicher Stelle Eigenschaften der Kommunikationsbeziehung.

```
@Secure(Integrity)
int getCurrentVelocity()

@QoS(Latency=Class_2)
streamHandle startCameraStreaming()
```

Das Beispiel zeigt in Anlehnung an die Syntax der Apache Etch IDL zwei entfernte Aufrufe. `int getCurrentVelocity()` beschreibt eine Funktion, welche die aktuelle Geschwindigkeit des Fahrzeugs zurückgibt. Die Annotation `@Secure(Integrity)` beschreibt die Security-Anforderung, dass der entfernte Aufruf und die Antwort integer sein müssen. Es obliegt dem Security-Stub bzw. dem modularen Security-Framework, diese abstrakte Anforderung durch konkrete Security-Mechanismen abzubilden. Die Funktion

`streamHandle startCameraStreaming()` soll die Datenübertragung einer Fahrerassistenzkamera starten. Die Annotation `@QoS(Latency=Class_2)` legt für den aufzubauenden Video-Strom die definierte Dienstgüteklasse `Class_2` fest. Da im vorgestellten Modell die Nachrichten der Kommunikationsmiddleware stets automatisch in die höchste Dienstgüteklasse fallen, bezieht sich die Annotation auf die durch den Aufruf etablierte Kommunikation. Im Allgemeinen muss man aber unterscheiden, ob eine Annotation für den Aufruf selbst oder für den folgenden Aufbau einer Kommunikationsbeziehung gilt.

Das Service-Management ist selbst eine generische Anwendung, die von Anwendungen eingebunden wird, welche die Rolle eines Dienstnehmers oder -gebers einnehmen. Damit lässt sich das Service-Management wie jede andere Anwendung in der gegebenen Schnittstellenbeschreibungssprache beschreiben. Das Service-Management nutzt weiterhin die Basisfunktionalität aus entfernten Funktionsaufrufen und dasselbe Marshalling, das über die Kommunikationsmiddleware verbindlich genutzt wird. Auch das Teilnetzbetrieb-Management ist als zentrales Dienstverzeichnis eine Anwendung, welche die Basisfunktionalität der Kommunikationsmiddleware nutzt, wenn man vom notwendigen dedizierten Weckmechanismus absieht.

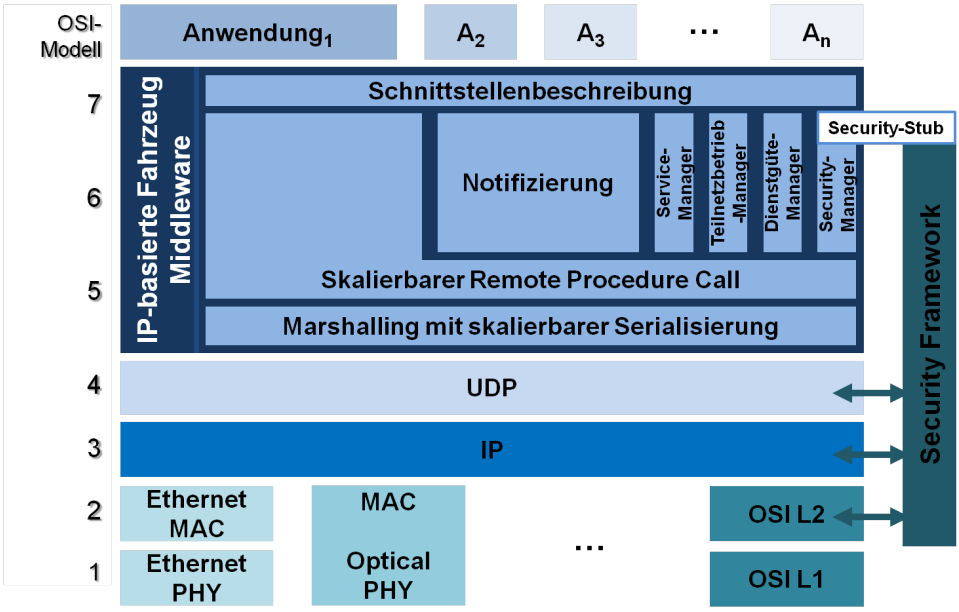

Abbildung 6.3.: Einheitliche Schnittstellenbeschreibung für die Basisfunktionalität zur Kommunikation und für Systemdienste.

Die Systemdienste können somit die Basisfunktionalität der IP-basierten Kommunikationsmiddleware mit Schnittstellenbeschreibung, Koordinationsmodell und Marshalling nutzen. Abbildung 6.3 veranschaulicht diese Eigenschaft: Alle Systemdienste werden durch die einheitliche Anwendungsschnittstelle abstrahiert und kommunizieren über die Vernetzungstechnologie auf Basis derselben Serialisierungsvorschriften über einfache, entfernte Aufrufe. Die

Kommunikationsmiddleware ist dazu geeignet, verschiedene Bestandteile für die eigentliche Kommunikation, ausgezeichnete Systemdienste wie ein Security-Management und Dienste für das Gesamtsystem wie ein Service-Management gegenüber dem Anwendungsentwickler einheitlich darzustellen.

6.6. Zusammenfassung

Systemdienste erweitern das Konzept einer funktional skalierbaren Kommunikationsmiddleware für das IP-basierte Fahrzeugbordnetz. Sie kapseln entweder Aufgaben, die für das gesamte Kommunikationssystem grundlegend sind, oder Aufgaben, die vielen Anwendungen nutzen. Somit müssen sie nur einmal im Rahmen der Kommunikationsmiddleware und nicht von jeder Anwendung selbst implementiert werden.

Das Service-Management im IP-basierten Fahrzeugbordnetz muss den Status von Diensten erkennen und vermitteln. Dies unterscheidet sich von einem Mechanismus für Service-Discovery, wie er in vielen anderen Bereichen eingesetzt wird. Meist ist es der Zweck von Service-Discovery, passende Dienste aus einer sich dynamisch ändernden Menge von Diensten auszuwählen. Die Dienste im Fahrzeug sind zur Produktionszeit prinzipiell schon bekannt, da die verbauten Steuergeräte nur von der Ausstattung des Fahrzeugs abhängen. Ein Service-Management wird zum einen aufgrund der sich im Fahrzeugbordnetz dynamisch ändernden Betriebsmodi und zum anderen für einen Teilnetzbetrieb, in dem sich nicht genutzte Steuergeräte zur Steigerung der Energieeffizienz temporär schlafen legen, benötigt.

Die prototypische Implementierung eines *Apache Etch Service-Managements* ermöglicht die Verwaltung von Dienstzuständen sowohl mit einem zentralen Dienstverzeichnis als auch dezentral. Dieser hybride Ansatz nutzt die Vorteile, dass ein zentrales Dienstverzeichnis ineffiziente Broadcasts im ordentlichen Betrieb einspart, während die dezentrale Kommunikation vor allem eine schnelle Bereitschaft schnell startender Dienste beim Aufstart des Fahrzeugbordnetzes ermöglicht. Die Untersuchungen zeigen, dass ein zentrales Dienstverzeichnis selbst auf einem möglichst schnell startenden Steuergerät beherbergt sein muss, um die kommunikationsintensive Phase beim Systemstart mit möglichst wenigen Broadcasts effizient zu ermöglichen.

Ein Teilnetzbetrieb-Manager kann als ein solches zentrales Dienstverzeichnis umgesetzt werden, das über die Kommunikationsmiddleware ansprechbar ist und zusätzlich über dedizierte Weckmechanismen ein schlafendes Steuergerät aufwecken kann.

Ein Dienstgüte-Management als Systemdienst der Kommunikationsmiddleware erlaubt dem Anwendungsentwickler die feingranulare Definition von Dienstgüteanforderungen. Die Anforderungen für eine zu etablierende Verbindung können als abstrakte Annotation für eine Funktion oder die Nutzung eines Anwendungsprotokolls beim Dienstgüte-Management angegeben werden. Jenes setzt dies auf die Vernetzungstechnologie um, z.B. durch unterschiedliche

Priorisierung von Daten in einem Switched Ethernet.

Ein Security-Management dient dem Schutz von Informationen im Fahrzeug. Allerdings betrifft dies nicht nur die Kommunikation, sondern das Gesamtsystem. Die Kommunikation kann also nur einen Teil zu einem gesicherten Gesamtsystem beitragen. Die Gewährleistung eines sicheren Gesamtsystems ist in Kooperation mit einem Security-Framework möglich, das neben der Kommunikation auch sichere Hardware, eine Einteilung des Fahrzeugbordnetzes in Zonen unterschiedlicher Schutzbedürftigkeit und weitere Aspekte umfasst. Im abgeschlossenen Forschungsprojekt EVITA [Weyl 10] wurde ein solches Security-Framework entworfen, das auch die eingeschränkte Leistungsfähigkeit von Steuergeräten im Fahrzeug berücksichtigt. Die Kommunikationsmiddleware ermöglicht, dass die geforderten Security-Anforderungen in der Anwendungsschnittstelle gemeinsam mit dem eigentlichen Aufruf definiert werden. Die Kommunikationsmiddleware bietet auf diese Weise einheitliche Schnittstellen für Anwendungen indirekt über einen Security-Stub an. Dieser greift auf die Module zu, die vom Security-Framework angeboten werden. Dieses modulare und skalierbare Security-Framework ergänzt die vorgestellte Kommunikationsmiddleware in unterschiedlichen Ausprägungen, weil es ebenfalls sowohl den Einschränkungen des jeweiligen Steuergeräts als auch den Security-Anforderungen der darauf laufenden Anwendungen gerecht werden kann.

Die vorgestellten Systemdienste können die Basisfunktionalität der IP-basierten Kommunikationsmiddleware aus Schnittstellenbeschreibung, Koordinationsmodell und Marshalling nutzen. Alle Systemdienste werden durch die einheitliche Anwendungsschnittstelle abstrahiert und kommunizieren auf Basis derselben Serialisierungsvorschriften über entfernte Aufrufe. Die Kommunikationsmiddleware ist dazu geeignet, verschiedene Bestandteile für die eigentliche Kommunikation, ausgezeichnete Kommunikationsdienste wie ein Security-Management und Dienste für das Gesamtsystem wie ein Service-Management gegenüber dem Anwendungsentwickler einheitlich darzustellen.

Systemdienste erhöhen die Abstraktion für Anwendungen. Sie erbringen Aufgaben, die nur nach einheitlichen Kriterien im IP-basierten Fahrzeugbordnetz erbracht werden können. Ein Service-Management kann nur aus einer Gesamtsicht auf den gesamten Steuergeräteverbund entworfen werden. Des Weiteren erbringen sie notwendige Bestandteile, sei es in Form von Erweiterungen bestehender Anwendungen oder als eigenständige Anwendung im Falle des zentralen Dienstverzeichnisses. Dienstgüte- und Security-Management kapseln abstrakte Anforderungen an die Kommunikation und ermöglichen angepasste Maßnahmen. Durch die damit mögliche Modularisierung ergänzen sie das Prinzip einer ebenfalls modularen Kommunikationsmiddleware in unterschiedlichen Ausprägungen.

Der Hauptteil der vorliegenden Arbeit wurde aus der Perspektive relevanter Anwendungen aufgebaut. Die Anwendungen stellen die Rahmen, an dem sich funktionale und nicht-funktionale Anforderungen erfassen lassen. Das folgende Kapitel soll nun einen Bogen spannen, indem das Konzept für ausgewählte

Anwendungen in einem möglichen IP-basierten Fahrzeugbordnetz verwendet wird. Abschließend wird aufgezeigt, wie das vorgestellte Konzept als Grundlage für eine Software-Entwicklung dient, die eine Standardisierung zum Ziel hat.

7. Das IP-basierte Fahrzeugbordnetz und seine Anwendungen

In diesem Kapitel wird abschließend ein kleiner Exkurs unternommen. Ausgewählte Anwendungen in einem möglichen IP-basierten Fahrzeugbordnetz wurden im Rahmen der Forschungsarbeit ausführlich betrachtet oder teilweise prototypisch umgesetzt. So wie die Erstellung des Konzepts auf Anforderungen aufbaut, die aus typischen Anwendungen abgeleitet wurden, wird im Folgenden gezeigt, wie die skalierbare Kommunikationsmiddleware einige ausgewählte Anwendungen bei der internen Fahrzeugkommunikation unterstützt. Die vorliegende Arbeit wird mit einem Ausblick auf eine mögliche Standardisierung abgeschlossen.

7.1. Untersuchte Anwendungen zur praktischen Demonstration des Erreichten

In einem Versuchsfahrzeug wurde ein prototypisches Setup aus für den Einsatz im Fahrzeug realistischen Komponenten aufgebaut. Der Aufbau besteht aus einem Steuergerät auf einer eingebetteten Plattform (x86-Intel Atom mit einer Taktung von 1600 MHz und einem Arbeitsspeicher von 667 MB DDR2) ähnlich der für die Evaluation des *EmbeddedC Bindings* genutzten Plattform. Der Hauptrechner der Infotainmentdomäne wird auf einem leistungsfähigen PC emuliert. Die Vernetzung erfolgt über 100 Mbit/s Ethernet. Die Übertragung findet dabei über eine ebenfalls Automotive-taugliche, ungeschirmte Zweidrahtleitung statt.

In Stolz, Weckemann und Lim „A Prototypical In-Car Entertainment Setup Using Software Defined Radio and Ethernet/IP-Based In-Vehicle Communication “ [Stol 12b] wurde am Anwendungsbeispiel eines Steuergeräts für den Rundfunkempfang dargestellt, wie sich eine Anwendung in einem IP-basierten Fahrzeugbordnetz verhält. Die prototypische Implementierung des Rundfunkempfängers ist als Software-defined Radio umgesetzt. Neben dem Audio-Streaming muss der Rundfunkempfänger Senderlisten kommunizieren und Steuerungsbefehle für eine Änderungen des aktuell empfangenen Senders entgegen nehmen. Der Aufbau zeigt als integrierter Demonstrator, dass sich Entertainment im Fahrzeug auch durch offene Standards und Lösungen aus

der CE-Industrie anstelle von proprietären Technologien der Fahrzeugindustrie realisieren lässt. Die Kommunikationsmiddleware beweist damit im praktischen Einsatz, wie sich durch die Abstraktion der IP-basierten Vernetzung eine einfache und schnelle Entwicklung von Kommunikationsschnittstellen ermöglichen lässt.

Ein IP-basierter Audio-Verstärker wird bereits in Weckemann et al. [Weck 12] beschrieben. Der in Abschnitt 5.4.1 beschriebene Versuchsaufbau wurde dazu im Kofferraum des Versuchsfahrzeugs verbaut. Als Verstärker-Steuergerät wurde die kleine Plattform basierend auf der Freescale MPC5668 Fado CPU verwendet. Diese wurde an den integrierten Verstärker des Versuchsfahrzeugs angebunden, um eine Musik-Wiedergabe über die Lautsprecher des Fahrzeugs zu ermöglichen. Ebenso wurde der für den Versuchsaufbau verwendete PC als Head Unit an den zentralen Monitor und die integrierte Bedieneinheit angeschlossen, um über eine emulierte fahrzeugtypische Bedienoberfläche eine Nutzersteuerung zu ermöglichen, die dem heute verwendeten System ähnelt.

Ein Anwendungsbeispiel, wie es erst zukünftig durch eine flexiblere Kommunikation in einem IP-basierten Fahrzeugbordnetz möglich wird, wurde in Endt und Weckemann „Remote Utilization of OpenCL for Flexible Computation Offloading using Embedded ECUs, CE Devices and Cloud Servers“ [Endt 12] beschrieben. Dabei wurde angenommen, dass es neben den eingebetteten Steuergeräten zukünftig auch Multi-Purpose-Rechner im Fahrzeug gibt. Über die Kommunikationsmiddleware wird festgestellt, ob andere leistungsfähige Steuergeräte oder angebundene CE-Geräte freie Ressourcen zur Verfügung stellen können. Dies würde beispielsweise ein paralleles Rechnen auf mehreren Grafikprozessoren erlauben. Zusätzlich muss Programmcode übertragen werden, um ein verteiltes, paralleles Rechnen zu ermöglichen. Die Kommunikationsmiddleware muss Möglichkeiten zur Verwaltung der eingesetzten Ressourcen bieten. Nach dem beschriebenen Konzept lässt sich dies durch eine Erweiterung in Form eines weiteren Systemdienstes umsetzen.

7.2. Standardisierung auf Basis des wissenschaftlichen Beitrags

Mit der Standardisierung von 100 Mbit/s Ethernet über eine ungeschirmte Zweidrahtleitung für den Einsatz im Fahrzeug wurde durch die OPEN Alliance der Grundstein für die Migration zu einem IP-basierten Fahrzeugbordnetz gelegt [OPEN].

Für einen baldigen, umfangreichen Einsatz von Ethernet im Fahrzeug muss eine Kommunikationsmiddleware-Lösung auf Basis von Ethernet/IP entwickelt werden. Die in der vorliegenden Arbeit identifizierten Anforderungen und das entworfene Konzept für eine skalierbare Lösung sind in die Entwicklung einer Kommunikationsmiddleware namens SOME/IP eingeflossen. Die Spezifikation

von SOME/IP [AUTO 13] greift die wesentlichen Aspekte bezüglich Koordinationsmodell und Marshalling auf. Als wesentliche Eigenschaft wird dabei die Unterstützung kleiner, eingebetteter Steuergeräte durch einfache Serialisierungsvorschriften heraus gestellt. SOME/IP ist funktional skalierbar und kann auf leistungsschwachen Steuergeräten ohne Betriebssystem (wie beispielsweise Fahrerassistenzkameras), auf AUTOSAR-Steuergeräten und auf größeren Steuergeräten der Infotainmentdomäne genutzt werden [Volk 13].

Die wissenschaftlichen Ergebnisse werden damit durch SOME/IP in Standardisierungsgremien verwertet, um eine Industrie-weite Basis für die zukünftige domänenübergreifende Anwendungskommunikation im Fahrzeugbordnetz zu schaffen. AUTOSAR ist eine Umsetzung einer komponentenorientierten Architektur. Einzelne Komponenten kommunizieren über einen sogenannten *Virtual Function Bus*, der durch die AUTOSAR-RTE (Run-Time Environment) implementiert ist. Unterhalb der RTE befinden sich verschiedene Schichten, die zusammen die AUTOSAR-BSW (Basic Software) bilden. Die BSW stellt Softwarekomponenten Dienste zur Verfügung, die über definierte Schnittstellen mittels der RTE aufgerufen werden können.

Die Aufgabe der RTE ist, die Kommunikation zwischen zwei Software-Komponenten umzusetzen. Diese Aufgabe umfasst die Kommunikation zwischen Softwarekomponenten in der Anwendungsschicht oder mit Diensten der BSW. Als Schnittstellen werden Systemprimitive auf Berkeley Sockets verwendet. Die Message-ID und die Länge werden vom Socket Adapter erzeugt. Die Client-ID und die Session-ID werden durch die RTE erzeugt. Verbindungen über die RTE werden zur Design-Zeit statisch konfiguriert. Eine Kommunikationsmiddleware wie SOME/IP kann daher unterhalb der RTE betrieben werden.

Eine Beschreibung für eine Implementierung von SOME/IP für die Ethernet-Spezifikation von AUTOSAR findet sich in Robert und Jayasudha [Robe 13].

7.3. Zusammenfassung

Ausgewählte Anwendungen in einem möglichen IP-basierten Fahrzeugbordnetz demonstrieren die Praxisrelevanz des vorgestellten Konzepts für eine Kommunikationsmiddleware im IP-basierten Fahrzeugbordnetz. Die Anwendungsbeispiele für einen Audio-Verstärker, ein Software-defined Radio und einen Dispatcher für die flexible Nutzung von Rechenleistung zeigen dabei die große Bandbreite der Einsatzmöglichkeiten. Von der einfachen Migration einer bestehenden Anwendung auf eine IP-basierte Vernetzung, über die Unterstützung der Entwicklung naheliegender Weiterentwicklungen bis hin zur Forschung für zukünftige Möglichkeiten zur flexibleren Gestaltung der gesamten Fahrzeugelektronik unterstützt die Kommunikationsmiddleware vielfältige Anwendungsbereiche.

Der Aufbau eines Versuchsfahrzeugs mit einem prototypischen Setup mit für den Einsatz im Fahrzeug realistischen Komponenten demonstriert das Zu-

sammenspiel der entwickelten Konzepte mit anderen aktuellen Bestrebungen zu offenen Standards. Fahrzeugelektronik und interne Vernetzung entwickeln sich weg von proprietären, hin zu offenen, standardisierten Technologien.

Die identifizierten Anforderungen und vorgestellten Konzepte sind in die Software-Entwicklung einer Lösung eingeflossen. Die Kommunikationsmiddleware-Lösung SOME/IP befindet sich derzeit in der Standardisierung im Rahmen von AUTOSAR.

8. Zusammenfassung und Fazit

In diesem Kapitel wird der wissenschaftliche Beitrag zusammengefasst. Kommunikationsmiddleware bietet Abstraktion auf Funktionsebene für Anwendungen im IP-basierten Fahrzeugbordnetz. Die vorgestellte Lösung sieht mehrere interoperable Ausprägungen der Middleware vor, die den Konflikt zwischen unterschiedlichen funktionalen Anforderungen einerseits und den sehr heterogenen Kommunikationspartnern andererseits lösen. Das Konzept der binärkompatiblen Interoperabilität erlaubt den Einsatz einer einzigen Kommunikationsmiddleware in interoperablen Ausprägungen. Der Entwurf dieser Ausprägungen zeigt, welche funktionalen Umfänge die Kommunikationsmiddleware für verschiedene Klassen von Steuergeräten benötigt und wie dies sowohl durch die Basisfunktionalität als auch durch erweiternde Systemdienste umgesetzt wird.

Das Elektroniksystem im Fahrzeug hat eine hohe Komplexität erreicht. Bis zu 80 Steuergeräte sind über bis zu sechs verschiedene Vernetzungstechnologien miteinander verbunden. Steuergeräte und dort beherbergte Anwendungen lassen sich logisch in Domänen aufteilen. In der Vergangenheit hat diese Unterteilung meistens eine isolierte Betrachtung der Kommunikation erlaubt. Allerdings wird ein wachsender Anteil neuer Innovationen durch reine Software-Anwendungen erreicht, welche die Kundenfunktionen auf den Steuergeräten verknüpfen und erweitern. Hierfür steigt der Bedarf an Kommunikation zwischen den Anwendungsdomänen. Außerdem öffnet sich die Fahrzeugkommunikation nach außen. Das Fahrzeug kommuniziert mit dem Smartphone des Fahrers und dem Internet. Für Kommunikation über verschiedene Anwendungsdomänen im Fahrzeug müssen heute Gateways eingesetzt werden, die zwischen den nicht-kompatiblen Protokollen übersetzen. Dies lässt sich durch den Einsatz des Internet Protocols für die interne Fahrzeugkommunikation verbessern, das für technologie- und domänenübergreifende Kommunikation entwickelt wurde. Die Konvergenzeigenschaft von IP ermöglicht Abstraktion von den eingesetzten Vernetzungstechnologien durch eine einheitliche Adressierung von Kommunikationspartnern. Neben dem durchgängigen Protokoll auf der Vermittlungsschicht setzt die effiziente Entwicklung eines komplexen verteilten Systems auch eine entsprechende Kommunikationsmiddleware voraus. Die Kommunikationsmiddleware kapselt die Kommunikationsaufgaben, die ansonsten von jeder Anwendung übernommen werden müssten. Neben der offensichtlichen Vereinfachung lässt sich darüber ein hohes Maß an Vereinheitlichung bis hin zur Standardisierung erreichen.

Die vorliegende Arbeit konzentriert sich auf die Anwendungskommunikation auf Basis einer IP-basierten Vernetzung innerhalb des Fahrzeugs. Eine Middleware muss die domänenübergreifende Kommunikation für die verteilten Anwendungen abstrahieren, um den steigenden Vernetzungsgrad dieser Anwendungen beherrschbar zu machen. Im Fahrzeug bestehen dafür spezielle Anforderungen. Zum einen werden in Fahrzeugen unterschiedliche Kommunikationsparadigmen genutzt. Zum anderen können sich die Kommunikationspartner in einem Fahrzeug hinsichtlich ihrer Ressourcen und ihrer Komplexität erheblich unterscheiden. Keine existierende IP-basierte Kommunikationsmiddleware erfüllt die identifizierten Anforderungen für den Einsatz im Fahrzeug. In dieser Arbeit wurde daher eine Kommunikationsmiddleware konzipiert, die für den Einsatz im Fahrzeug geeignet ist.

Heutige Vernetzungstechnologien im Fahrzeugbordnetz stellen bereits einen unterschiedlichen Umfang an Möglichkeiten zur Kommunikationsabstraktion bereit. Kommunikationsmiddleware für das IP-basierte Fahrzeugbordnetz muss mindestens den heutigen Umfang abdecken, um eine Migration bestehender Bordnetzkommunikation zu erlauben. Es wurde ein zukunftsfähiges Konzept entworfen, das weiterhin auch potentielle Anforderungen zukünftiger Anwendungen abdeckt. Die wissenschaftliche Untersuchung klärt, wie die Funktionsadressierung in einem IP-basierten Fahrzeugbordnetz gestaltet werden kann und welche Funktionalität die Systemsoftware für die Kommunikation bereitstellen muss. Im durch eingebettete Plattformen geprägten Fahrzeugbordnetz muss dies jeweils im Kontext der Verfügbarkeit von Ressourcen betrachtet werden. Die Mehrzahl der Steuergeräte sind eingebettete Systeme, mit wenig Rechenleistung und geringen Speicherressourcen. Durch den hohen industriellen Kostendruck ist die Leistungsfähigkeit eines Steuergerätes für den Bedarf der beherbergten Anwendungen optimiert. Geringer Bedarf an die Leistungsfähigkeit der Hardware geht allerdings in der Regel mit geringen funktionalen Anforderungen an die Kommunikation einher.

Die Anwendungen auf den kleinen Steuergeräten bestehen oft nur aus der Steuerung und Datenweitergabe einfacher Aktuatoren und Sensoren. Das heutige System ist stark heterogen. Diese Heterogenität wird sowohl durch die unterschiedlichen Vernetzungstechnologien, als auch durch die eingesetzten Plattformen und die Mächtigkeit der darauf ausgeführten Anwendungen verursacht. Dies führt dazu, dass der identifizierte funktionale Umfang der Kommunikationsmiddleware auf kleinen Plattformen nicht ausführbar ist.

Der funktionale Umfang der Kommunikationsmiddleware lässt sich in zwei Aspekte unterteilen. Zum einen ist dies die Vermittlung der Kommunikation, also die Umsetzung von Kommunikationsparadigmen auf unterschiedlichen Plattformen. Zum anderen gibt es weitere kommunikationsrelevante Aufgaben, die durch Systemsoftware erbracht werden müssen. Diese Kapselung erfolgt durch Systemdienste. Für die Basisfunktionalität einer Kommunikationsmidd-

leware wurden die benötigten Kommunikationsparadigmen identifiziert. Die heute verwendeten Paradigmen lassen sich, trotz unterschiedlicher Mechanismen, durch zwei Paradigmen abbilden. Erstens wird ein signalbasiertes Kommunikationsparadigma benötigt, bei dem Informationen im Fahrzeugbordnetz an mehrere Empfänger verteilt werden. Zweitens wird ein funktionsbasiertes Kommunikationsparadigma gefordert, in dem Funktionen auf einem entfernten Steuergerät aufgerufen bzw. einfache Werte von einem entfernten Steuergerät abgefragt werden können. Daneben gibt es Bedarf für das Versenden von Datenströmen und Massendaten, der in einem IP-basierten Fahrzeugbordnetz aber bereits durch existierende Protokolle abgedeckt ist.

Gemäß den identifizierten Anforderungen wurde entschieden, das funktionsbasierte Kommunikationsparadigma durch einfache Remote Procedure Calls abzubilden. Ein auf dem Internet Protocol aufsetzendes Transportprotokoll bietet bereits die Adressierung einer Anwendung. Ein Multiplexing des Aufrufs auf unterschiedliche Funktionen der Anwendung kann darauf aufsetzend durch einen in der Anwendungsschnittstelle definierten Identifikator erreicht werden. Auf Basis dieser Funktionsadressierung lassen sich entfernte Funktionsaufrufe durchführen. Eine höherwertige Abstraktion, die eine service- oder komponentenbasierte Middleware ermöglicht, ist für die Anforderungen der internen Fahrzeugkommunikation nur in der Infotainmentdomäne wünschenswert, aber nicht zwingend notwendig. Ihr Einsatz würde eine für das Ausführungsverhalten in anderen Domänen kritische Komplexität erzeugen.

Das signalbasierte Kommunikationsparadigma wird durch ein Notification-Konzept implementiert. Die Analysen zeigen, dass eine einfache Abbildung der signalbasierten Kommunikation aus dem heutigen Fahrzeugbordnetz in ein zukünftiges IP-basiertes Fahrzeugbordnetz ineffizient ist. Dazu wurde eine Migration heutiger realer Kommunikation auf Ethernet betrachtet. Ethernet gilt dabei als eine wahrscheinliche Vernetzungstechnologie für ein IP-basiertes Fahrzeugbordnetz. Zum einen steigt der relative Protokoll-Overhead aus Nutzdatengröße zu Protokollinformationen bei einer IP-basierten Kommunikation beträchtlich. Zum anderen sind Multicast-Gruppen in einem Switched Ethernet ineffizient. Daraus wird gefolgert, dass die Umsetzung des signalbasierten Kommunikationsparadigmas sich stärker am aktuellen Informationsbedarf orientieren muss, als dies im heutigen Fahrzeugbordnetz durch das einfache Verteilen von Daten der Fall ist. Das Notification-Konzept erlaubt das dynamische Anmelden für ausgewählte Daten und das gezielte Verteilen dieser Daten an angemeldete Anwendungen basierend auf Änderungsbedingungen. Wir zeigen, dass sich prinzipiell beide Kommunikationsparadigmen durch einen einzigen Mechanismus abbilden lassen. Das Notification-Konzept funktioniert dabei als ein Mechanismus, der die einfachen RPCs des funktionsbasierten Kommunikationsparadigmas wiederverwendet.

Neben der Umsetzung der Kommunikationsparadigmen muss die Middleware Plattformunabhängigkeit ermöglichen. Marshalling ermöglicht dies durch ein verbindliches, serielles Datenformat für die Übertragung und die Seriali-

sierung strukturierter Daten. Diese Serialisierung muss auf allen für die IP-basierte Kommunikation relevanten Plattformen möglich sein. Das serielle Datenformat hat direkten Einfluss auf die Mächtigkeit und die Performanz der Kommunikationsmiddleware. Gemäß den identifizierten Anforderungen muss im Fahrzeugbordnetz ein effizientes, binäres Serialisierungsformat gewählt werden, das eine hohe Performanz bei geringem Ressourcenbedarf in den beteiligten Steuergeräten ermöglicht.

Um trotz der Heterogenität der verfügbaren Ressourcen der Steuergeräte ein einheitliches Gesamtsystem zu ermöglichen, wird das Konzept der binärkompatiblen Interoperabilität eingeführt. Dies wird möglich, da auf den kleinen Plattformen auch nur ein eingeschränkter funktionaler Umfang benötigt wird. Das vorgestellte Interoperabilitätskonzept ermöglicht die Kommunikation zwischen den Teilnehmern ohne zusätzliche Gateway-Komponente und erhält damit den durch IP eingeführten Vorteil der Durchgängigkeit trotz der vorhandenen Heterogenität.

RPCs operieren abhängig von den beteiligten Ausprägungen mit dynamischen oder nur statischen Daten und bieten unterschiedliche Aufrufsemantiken. Ein skalierbares Marshalling berücksichtigt darüber hinaus die unterschiedlichen Anforderungen und Leistungsfähigkeit der beteiligten Steuergeräte. Hiermit wird die Kommunikation zwischen allen Anwendungen im IP-basierten Fahrzeugbordnetz durchgängig ermöglicht.

Auf dieser Basis wurde die Lösung um wichtige Systemdienste erweitert. Diese Dienste implementieren Funktionen, die nur in der Kooperation mehrerer Komponenten erbracht werden können oder kapseln allgemeine Kommunikationsfunktionalitäten zur einfachen Wiederverwendung. Zwei für die Anwendung im Fahrzeug wichtige Systemdienste wurden prototypisch dargestellt: Ein Service-Management ermöglicht die Verwaltung von Diensten in unterschiedlichen Zuständen, ein Security-Management bildet Security-Ziele auf die bestmögliche Kombination von implementierten Security-Protokollen der beteiligten Kommunikationspartner ab. Darüber hinaus wird ein Dienstgüte-Management beschrieben, das die definierten Anforderungen einer Anwendung auf die verwendete Vernetzungstechnologie umsetzt. Ein Teilnetzbetrieb-Management überwacht den Zustand im Gesamtsystem und kann Teilnetze bedarfsgerecht schlafen legen oder aufwecken. Diese Systemdienste sind selbst skalierbar und lassen sich damit an das Konzept unterschiedlicher Ausprägungen der Kommunikationsmiddleware anpassen.

Durch Leistungsmessungen an den im Rahmen dieser Arbeit entstandenen Prototypen konnte gezeigt werden, dass die konzipierte Kommunikationsmiddleware für den Einsatz auf eingebetteten Systemen im Fahrzeug geeignet ist. Der Versuchsaufbau orientiert sich an typischen Anwendungsfällen für die Fahrzeugkommunikation und verwendet Automotive-qualifizierte, eingebettete Rechenplattformen. Insbesondere wird nachgewiesen, dass mit dem vorgestellten Konzept auch leistungsschwache Steuergeräte ins System eingebunden werden können. Die IP-basierte Kommunikationsmiddleware ist

damit auf allen relevanten Steuergeräten im Fahrzeug durchgängig einsetzbar.

Zusammenfassend abstrahiert die Schnittstellenbeschreibungssprache den gesamten funktionalen Umfang der Kommunikationsmiddleware, sowohl bezüglich der Basisfunktionalität für die eigentliche Kommunikation, als auch zur Beschreibung der Kommunikationsanforderungen an die Systemdienste. Sowohl das funktionsbasierte, als auch das signalbasierte Kommunikationsparadigma werden auf der Basis einfacher RPCs umgesetzt, die in unterschiedlichen Ausprägungen dynamische oder statische Eigenschaften verwenden und unterschiedliche Aufrufsemantiken unterstützen. Modulare Systemdienste erweitern dieses Konzept auf Grundlage der Basisfunktionalität und lassen sich für den Anwendungsprogrammierer einheitlich mit der übrigen Kommunikationsfunktionalität beschreiben.

Die modulare, skalierbare Architektur der vorgeschlagenen Kommunikationsmiddleware ermöglicht die in Abschnitt 1.2 referenzierte Forderung von Pretschner und Broy [Pret 07], schlanke und hoch optimierte Middleware-Teile einzusetzen.

Die wissenschaftlichen Ergebnisse werden derzeit in Standardisierungsgremien verwertet, um einen breiten Einsatz einer Kommunikationsmiddleware für die zukünftige domänenübergreifende Anwendungskommunikation im Fahrzeugbordnetz zu schaffen. Neben diesen praktischen Schritten in Richtung einer Software-Entwicklung bietet die Thematik weiteres Potential für eine ausgiebige Untersuchung. Die vorliegende Arbeit geht von einer zukünftigen Entwicklung aus, die viele Aspekte des Internet Protocols als Basis für die Vernetzung im Fahrzeug annimmt. In der Praxis wird sich in den nächsten Jahren zeigen, welche Ideen die Kosten- und Nutzenanalysen überstehen und den Weg ins Fahrzeug tatsächlich finden. In einer Übergangsphase wird ein gemischtes Kommunikationssystem zwischen „alten“ fahrzeugtypischen Vernetzungstechnologien und „neuen“ Vernetzungstechnologien aus anderen Umgebungen entstehen. Die Migration für dieses System zu begleiten, bietet aus Sicht der Anwendungskommunikation, aus Security-Perspektive und für weitere Aspekte viele neue Herausforderungen.

Literaturverzeichnis

[Abst] "Abstract Syntax Notation One (ASN.1) Recommendations (Link kontrolliert am 30.03.2014)". Website: http://www.itu.int/ITU-T/studygroups/com17/languages/.

[Anan 92] A. L. Ananda, B. H. Tay, E. K. Koh. "A survey of asynchronous remote procedure calls". *SIGOPS Oper. Syst. Rev.*, Vol. 26, Nr. 2, pp. 92–109, Apr. 1992.

[Apaca] "Apache Etch Homepage (Link kontrolliert am 30.03.2014)". Website: http://incubator.apache.org/etch.

[Apacb] "Apache Software Foundation Homepage, Apache License Version 2.0 (Link kontrolliert am 30.03.2014)". Website: http://www.apache.org/licenses/LICENSE-2.0.html.

[AUTO 13] AUTOSAR Development Partnership. "SOME/IP Serialization Protocol". Tech. Rep., AUTOSAR, 2013.

[Bakk 01] D. Bakken. "Middleware". In: *Encyclopedia of Distributed Computing*, Kluwer Academic, Dodrecht, The Netherlands, 2001.

[Beck 00] C. Becker, K. Geihs. "Generic QoS-support for CORBA". In: *Fifth IEEE Symposium on Computers and Communications*, pp. 60 – 65, Antibes-Juan les Pins, 2000.

[Bell 11] L. L. Bello. "The Case for Ethernet in Automotive Communications". *Special Issue on the 10th International Workshop on Real-time Networks (RTN 2011)*, Vol. 8, Nr. 4, 2011.

[Bern 87] P. Bernstein, V. Hadzilacos, N. Goodman. *Concurrency control and recovery in database systems.* Vol. 370, Addison-wesley New York, 1987.

[Birr 84] A. D. Birrell, B. J. A. Y. Nelson. "Implementing Remote Procedure Calls". *ACM Transactions on Computer Systems*, Vol. 2, Nr. 1, pp. 39–59, 1984.

[Boua 12] A. Bouard, B. Glas, A. Jentzsch, A. Kiening, T. Kittel, F. Stadler, B. Weyl. "Driving automotive middleware towards a secure ip-based future". In: *Proc. of 10th conference for Embedded Security in Cars (Escar'12)*, Berlin, Germany, Nov. 2012.

[Boua 13] A. Bouard, J. Schanda, D. Herrscher, C. Eckert. "Automotive proxy-based security architecture for ce device integration". In: C. Borcea, P. Bellavista, C. Giannelli, T. Magedanz, F. Schreiner, Eds., *Mobile Wireless Middleware, Operating Systems, and Applications*, pp. 62–76, Springer Berlin Heidelberg, 2013.

[Bruc 10] R. Bruckmeier. "Ethernet for Automotive Applications". In: *Freescale Technology Forum*, Orlando, 2010.

[Bunz 11] S. Bunzel. "AUTOSAR - the Standardized Software Architecture". *Informatik-Spektrum*, Vol. 34, Nr. 1, pp. 79–83, 2011.

[CORB] "CORBA/e Resource Page (Link kontrolliert am 30.03.2014)". Website: http://www.corba.org/corba-e/index.htm.

[Craw 98] E. Crawley, R. Nair, B. Rajagopalan, H. Sandick. "RFC2386: A Framework for QoS-based Routing in the Internet". 1998.

[Croc 06] D. Crockford. "RFC 4627: The application/json Media Type for JavaScript Object Notation (JSON)". 2006.

[Dahl 11] E. Dahlman, S. Parkvall, J. Skold. *4G: LTE/LTE-Advanced for Mobile Broadband: LTE/LTE-Advanced for Mobile Broadband (Google eBook)*. Academic Press, 2011.

[Daou 06] R. Daoud, H. Amer, H. Elsayed, Y. Sallez. "Ethernet-Based Car Control Network". In: *Canadian Conference on Electrical and Computer Engineering*, pp. 1031–1034, Ottawa, Ont., 2006.

[Ditz 03] M. Ditze, R. Bernhardi, G. Kämper, P. Altenbernd. "Porting the Internet Protocol to the Controller Area Network". In: *2nd International Workshop on Real-time LANs in the Internet Age*, Porto, Portugal, 2003.

[Dunk 01] A. Dunkels. "Design and Implementation of the lwIP TCP/IP Stack". Tech. Rep., Swedish Institute of Computer Science, 2001.

[eCos] "eCos Reference Manual (Link kontrolliert am 30.03.2014)". Website: http://ecos.sourceware.org/docs-latest/ref/ecos-ref.html.

[Emme 00] W. Emmerich. "Software engineering and middleware: a roadmap". In: *Proceedings of the Conference on The Future of Software Engineering*, pp. 117–129, ACM, New York, NY, USA, 2000.

[Endt 12] H. Endt, K. Weckemann. "Remote Utilization of OpenCL for Flexible Computation Offloading using Embedded ECUs, CE Devices and Cloud Servers". In: M. S. Koen De Bosschere, Erik H. D'Hollander, Gerhard R. Joubert, David Padua, Frans Peters,

Ed., *Advances in Parallel Computing (Volume 22) – Applications, Tools and Techniques on the Road to Exascale Computing*, pp. 133 – 140, Ghent, Belgium, 2012.

[Fair 04] G. Fairhurst, M. Degermark, S. Pink. "RFC 3828: The Lightweight User Datagram Protocol (UDP-Lite)". 2004.

[Flex 05] FlexRay Consortium. "FlexRay Communications System Protocol Specification Version 2.1". 2005.

[Frey 07] R. Freymann. "Anforderungen an das Automobil der Zukunft". In: *2nd Mobility Forum*, München, Deutschland, 2007.

[Fric 11] E. Frickenstein. "Das IP-basierte Bordnetz kommt". 2011.

[Furs 09] S. Fürst, J. Mössinger, S. Bunzel. "AUTOSAR - A Worldwide Standard is on the Road". 14th International VDI Congress Electronic Systems for Vehicles, Baden-Baden, 2009.

[Glas 10] M. Glass, D. Herrscher, H. Meier, M. Piastowski, P. Schoo. "SEIS - Security in Embedded IP-based Systems". *ATZelektronik worldwide*, Vol. 2010-01, pp. 36–40, 2010.

[Grot 10] G. Grotewold, K. Weckemann, P. Correia, A. Camek. "Secure Middleware for IP-Based In-Vehicle Communication". *EE Times Automotive*, July 2010.

[Grze 08] A. Grzemba. *MOST - The Automotive Multimedia Network*. Franzis Verlag, Poing, 2008.

[Henn 08] M. Henning. "The Rise and Fall of CORBA". *Communications of the ACM*, Vol. 51, 2008.

[Herr 11] D. Herrscher. "Chancen und Herausforderungen der vernetzten Mobilität. Das Fahrzeug als IP Knoten.". In: *Future Internet Konferenz*, Berlin, Germany, 2011.

[Hint 10] W. Hintermaier, E. Steinbach. "A System Architecture for IP-camera based Driver Assistance Applications". In: *IEEE Intelligent Vehicles Symposium (IV)*, pp. 540–547, 2010.

[IEEE 10a] IEEE Institute of Electrical and Electronics Engineers. "IEEE P1722.1/D0.12 - Draft Standard for Standard for Standard Device Discovery, Connection Management and Control Protocol for IEEE 1722 Based Devices". 2010.

[IEEE 10b] IEEE Institute of Electrical and Electronics Engineers. "IEEE P1722/D2.4 - Draft Standard for Layer 2 Transport Protocol for Time Sensitive Applications in a Bridged Local Area Network". 2010.

[IEEE 10c] IEEE Institute of Electrical and Electronics Engineers. "IEEE Standard 802.11p-2010 - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicles". 2010.

[IEEE 12] IEEE Institute of Electrical and Electronics Engineers. "IEEE Standard 802.3-2012 - Standard for Ethernet". 2012.

[Inte 94] International Organization for Standardization. "Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model". 1994.

[Issa 07] V. Issarny, M. Caporuscio, N. Georgantas. "A Perspective on the Future of Middleware-based Software Engineering". In: *Future of Software Engineering (FOSE '07)*, pp. 244–258, IEEE, May 2007.

[Joha 05] K. H. Johansson, M. Törngren, L. Nielsen. "Vehicle Applications of Controller Area Network". In: *Handbook of Networked and Embedded Control Systems*, pp. 741–765, Springer, 2005.

[Kern 11] A. Kern, T. Streichert, J. Teich. "An automated data structure migration concept - From CAN to Ethernet/IP in automotive embedded systems (CANoverIP)". In: *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pp. 112–117, 2011.

[Kesk 09] U. Keskin. "In-vehicle communication networks: a literature survey". Tech. Rep., Technische Universiteit Eindhoven, 2009.

[Lim 11a] H.-T. Lim, B. Krebs, L. Völker, P. Zahrer. "Performance evaluation of the inter-domain communication in a switched Ethernet based in-car network". In: *IEEE 36th Conference on Local Computer Networks (LCN)*, pp. 101–108, Bonn, Germany, 2011.

[Lim 11b] H.-T. Lim, L. Völker, D. Herrscher. "Challenges in a Future IP/Ethernet-based In-Car Network for Real-Time Applications". In: *Proceedings of the 2011 ACM/EDAC/IEEE Design Automation Conference (DAC)*, San Diego, USA, 2011.

[Lim 11c] H.-T. Lim, K. Weckemann, D. Herrscher. "Performance Study of an In-Car Switched Ethernet Network Without Prioritization". In: T. Strang, Ed., *Nets4Cars/Nets4Trains*, Springer-Verlag, 2011.

[Lim 12] H.-T. Lim, D. Herrscher, M. J. Waltl, F. Chaari. "Performance Analysis of the IEEE 802.1 Ethernet Audio/Video Bridging Standard". In: *5th International ICST Conference on Simulation Tools and Techniques (SIMUTools 2012)*, Sirmione-Desenzano, Italy, 2012.

[Lind 08] P. Lindgren, S. Aittamaa, J. Eriksson. “IP over CAN, transparent vehicular to infrastructure access”. *5th IEEE Consumer Communications and Networking Conference (CCNC 2008)*, pp. 758–759, 2008.

[MOST] “MOST Homepage (Link kontrolliert am 30.03.2014)”. Website: http://www.mostcooperation.com.

[MOST 10] MOST Corporation. “MOST Specification Rev. 3.0”. 2010.

[Mowb 98] T. J. Mowbray, W. A. Ruh. *Inside CORBA: Distributed Object Standards and Applications*. Addison-Wesley Longman Publishing Co., Inc., 1998.

[Nave 05] N. Navet, Y. Song, F. Simonot-Lion, C. Wilwert. “Trends in Automotive Communication Systems”. *Proceedings of the IEEE*, Vol. 93, Nr. 6, pp. 1204–1223, June 2005.

[Nave 13] N. Navet, F. Simonot-Lion. “In-vehicle communication networks - a historical perspective and review”. In: R. Zurawski, Ed., *Industrial Communication Technology Handbook, Second Edition*, Crc Pr Inc, 2013.

[Noba 11] J. Nöbauer. “Is Ethernet the rising star for in-vehicle networks?”. In: *IEEE Conference on Emerging Technologies in Factory Automation*, Toulouse, France, 2011.

[Nolt 05] T. Nolte, H. Hansson, L. L. Bello. “Automotive communications-past, current and future”. In: *10th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, Catania, 2005.

[OMG] “OMG CORBA Homepage (Link kontrolliert am 30.03.2014)”. Website: http://www.omg.org/spec/CORBA/.

[OMG 07] OMG Specification. “Data Distribution Service for Real-time Systems”. 2007.

[OMNe] “OMNet++ Homepage (Link kontrolliert am 30.03.2014)”. Website: http://www.omnetpp.org/.

[OPEN] “OPEN Alliance Homepage (Link kontrolliert am 30.03.2014)”. Website: http://www.opensig.org/.

[Post 80] J. Postel. “RFC 768: User Datagram Protocol”. 1980.

[Post 81a] J. Postel. “RFC 791: Internet Protocol”. 1981.

[Post 81b] J. Postel. “RFC 793: Transmission Control Protocol”. 1981.

[Pret 07] A. Pretschner, M. Broy, I. Kruger, T. Stauner. "Software engineering for automotive systems: A roadmap". In: *FOSE '07 2007 Future of Software Engineering*, pp. 55–71, IEEE Computer Society, 2007.

[Prot] "Protocol Buffers Homepage (Link kontrolliert am 30.03.2014)". Website: https://code.google.com/p/protobuf/.

[Rahm 07] M. Rahmani, J. Hillebrand, W. Hintermaier, R. Bogenberger, E. Steinbach. *A Novel Network Architecture for In-Vehicle Audio and Video Communication*. IEEE, May 2007.

[Rahm 08] M. Rahmani, R. Steffen, K. Tappayuthpijarn, E. Steinbach, G. Giordano. "Performance analysis of different network topologies for in-vehicle audio and video communication". In: *Telecommunication Networking Workshop on QoS in Multiservice IP Networks*, pp. 179 – 184, IEEE, Venice, Italy, 2008.

[Rahm 09] M. Rahmani, K. Tappayuthpijarn, B. Krebs, E. Steinbach, R. Bogenberger. "Traffic Shaping for Resource-Efficient In-Vehicle Communication". *IEEE Transactions on Industrial Informatics*, Vol. 5, Nr. 4, pp. 414–428, Nov. 2009.

[Reif 06] K. Reif. *Automobilelektronik*. Vieweg+Teubner Verlag, Wiesbaden, 2006.

[Rich 12] I. Riches. "Automotive High Speed Bus Networks". Tech. Rep. January, Strategy Analytics, 2012.

[Robe 13] S. Robert, J. S. Jayasudha. "TCP / IP Stack Implementation for Communication over IP with AUTOSAR Ethernet Specification". *International Journal of Engineer ing and Innovative Technology*, Vol. 3, Nr. 1, pp. 176–179, 2013.

[Robe 91] Robert Bosch GmbH. "CAN Specification". 1991.

[Roma 00] M. Roman, D. Mickunas, F. Kon, R. Campbell. "LegORB and ubiquitous CORBA". In: *Proceedings of the IFIP/ACM Middleware 2000 Workshop on Reflective Middleware*, 2000.

[Schm 07] A. Schmeiser. *Protokollfamilie zur universellen Steuerdatenkommunikation in heterogenen Netzwerkumgebungen*. PhD thesis, Ulm University, 2007.

[Schu 03] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson. "RFC 3550: RTP - A Transport Protocol for Real-Time Applications". 2003.

[Somm 10] J. Sommer, S. Gunreben, F. Feller, M. Kohn, A. Mifdaoui, D. Sass, J. Scharf. "Ethernet - A Survey on its Fields of Application". *IEEE Communications Surveys & Tutorials*, Vol. 12, Nr. 2, pp. 263–284, 2010.

[Stef 10] R. Steffen, R. Bogenberger, J. Hillebrand, W. Hintermaier, A. Winckler, M. Rahmani. "Design and realization of an ip-based in-car network architecture". *1st International ICST Symposium on Vehicular Computing Systems*, 2010.

[Stol 12a] L. Stolz, M. Feilen, W. Stechele. "An Optimized Software-Defined Digital Audio Broadcasting (DAB) Receiver for x86 Platforms". *Proceedings of the 7th Karlsruhe Workshop on Software Radios (WSR 2012)*, 2012.

[Stol 12b] L. Stolz, K. Weckemann, H.-T. Lim, W. Stechele. "A Prototypical In-Car Entertainment Setup Using Software Defined Radio and Ethernet / IP-Based In-Vehicle Communication". In: *The First International Conference on Advances in Vehicular Systems, Technologies and Applications, VEHICULAR 2012*, pp. 7–10, Venice, 2012.

[Tane 06] A. Tanenbaum. *Distributed Systems: Principles and Paradigms*. Prentice Hall, 2nd Ed., 2006.

[Tay 90] B. H. Tay, A. L. Ananda. "A survey of remote procedure calls". *SIGOPS Oper. Syst. Rev.*, Vol. 24, Nr. 3, pp. 68–79, 1990.

[Tutt 99] W. H. W. Tuttlebee. "Software-defined radio: facets of a developing technology". *IEEE Personal Communications*, Vol. 6, Nr. 2, pp. 38–44, 1999.

[Verv 08] C. N. Ververidis, G. C. Polyzos. "Service discovery for mobile ad hoc networks: A survey of issues and techniques". *IEEE Communications Surveys and Tutorials*, Vol. 10, Nr. 1-4, pp. 30–45, 2008.

[Volk 13] L. Völker. "SOME/IP – Die Middleware für Ethernet-basierte Kommunikation". *HANSER automotive networks*, pp. 17–19, 2013.

[Weck 10] K. Weckemann, B. Weyl. "Aus der IT-Welt ins Auto – Sichere Kommunikation im Fahrzeug mit dem Internet Protocol". *Elektronik automotive*, Nr. 12, pp. 38–42, 2010.

[Weck 11] K. Weckemann, H.-T. Lim, D. Herrscher. "Practical Experiences on a Communication Middleware for IP-based In-Car Networks". In: *Proceedings of the Fifth International Conference on COMmunication System softWAre and middlewaRE*, Verona, 2011.

[Weck 12] K. Weckemann, F. Satzger, L. Stolz, D. Herrscher, C. Linnhoff-Popien. "Lessons from a Minimal Middleware for IP-Based In-Car Communication". In: *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2012.

[Weyl 10] B. Weyl, M. Wolf, F. Zweers, T. Gendrullis, S. I. Muhammad, H. Schweppe, Y. Roudier. "Secure on-board architecture specification". Tech. Rep., Evita Project, 2010.

[Wolf 97] L. Wolf, C. Griwodz, R. Steinmetz. "Multimedia Communication". *Proceedings of the IEEE*, Vol. 85, Nr. 12, pp. 1915 – 1933, 1997.

[YAML] "YAML Homepage (Link kontrolliert am 30.03.2014)". Website: http://www.yaml.org/about.html.

[Zero] "Zero Configuration Networking (Zeroconf) Homepage (Link kontrolliert am 30.03.2014)". Website: http://www.zeroconf.org/.

[Zimm 07] W. Zimmermann, R. Schmidgall. *Bussysteme in der Fahrzeugtechnik*. Vieweg, Wiesbaden, 2007.

[Zinn 11] H. Zinner, J. Nöbauer, T. Gallner, J. Seitz, T. Waas. "Application and realization of gateways between conventional automotive and IP/ethernet-based networks". In: *Design Automation Conference (DAC)*, pp. 1–6, New York, NY, 2011.

Begriffsdefinitionen

Anwendung Ein Programm, das die eigentliche Funktion eines Steuergerätes in Software implementiert.

Anwendungsdomäne Eine Anwendungsdomäne beinhaltet Steuergeräte mit ähnlichen Funktionen und Anforderungen. Aus der Sicht der Kommunikationsnetze werden die Steuergeräte einer Domäne in der Regel durch ein Teilnetz organisiert. Aktuelle Fahrzeuge beherbergen beispielsweise die Anwendungsdomänen Powertrain (Motorsteuerung), Chassis (Fahrgestell), Body and Cabin (Innenraum), Driver Assistance (Fahrerassistenzsysteme) und Infotainment.

Anwendungsschicht-Gateway/Application Layer Gateway Ein Anwendungsschicht-Gateway bezeichnet eine Infrastrukturkomponente, die zwischen unterschiedlichen Vernetzungstechnologien übersetzt und dafür selbst Anwendungswissen benötigt.

Aufstart Der Aufstart des Fahrzeugs bezeichnet den Übergang von einem nicht-betriebsbereiten in einen betriebsbereiten Status bzw. aus der Sichtweise eines Architekten für das Fahrzeugbordnetz als Übergang in einen Modus „Zündung an“.

Fahrzeugbordnetz Das Fahrzeugbordnetz ermöglicht die interne Kommunikation im Fahrzeug. Es besteht aus den verbauten Steuergeräten, die über die verschiedenen Vernetzungstechnologien verbunden sind. In der vorliegenden Arbeit ist damit stets ein Fahrzeugbordnetz zur Herstellung von Kommunikation gemeint, im Gegensatz zu einem Energiebordnetz, das Steuergeräte, Sensoren und Aktuatoren mit elektrischer Spannung versorgt.

Kommunikationsmechanismus Umsetzung eine Kommunikationsparadigmas.

Kommunikationsmiddleware Kommunikationsmiddleware vermittelt zwischen Anwendungen und verbirgt die darunter liegende Komplexität der Infrastruktur. Dazu stellt die Kommunikationsmiddleware eine Schnittstelle zur Anwendung bereit und greift intern auf Betriebssystemfunktionen zu, um ihre Funktionalität umzusetzen. Der Zusatz „Kommunikation“ in Kommunikationsmiddleware soll verdeutlichen, dass die Kommunikation im Fokus steht und nicht andere Themen wie z.B. Ressourcenmanagement.

Kommunikationsmiddlewarelösung Ein konkretes Softwareprodukt, das die funktionalen Anforderungen an eine Kommunikationsmiddleware umsetzt.

Kommunikationsparadigma Logisches Modell, wie Kommunikation zwischen Kommunikationspartnern stattfindet.

Koordinationsmodell Ein Koordinationsmodell beschreibt das dynamische Verhalten einer Middleware für die vermittelte Kommunikation. Es kommt dabei mindestens ein Kommunikationsparadigma zur Anwendung.

Marshalling Marshalling dient einer systemunabhängigen Aufbereitung der Anwendungsdaten für die Kommunikation. Die Übertragung kann nur in einer einheitlichen, serialisierten Form erfolgen.

Massendaten Der Begriff Massendaten meint große Datenmengen oder nicht näher klassifizierte Daten. Der Begriff entspricht dem gebräuchlicheren englischen Begriff Bulk Data.

Notifizierung/Notification Eine Notifizierung/Notification ist eine Information, die von einem Dienst ohne Angabe von konkreten Adressaten veröffentlicht wird. Interessenten dieser Nachricht können sich statisch oder dynamisch für eine spezifische Nachricht anmelden.

Programmierschnittstelle (siehe auch API) Die Programmierschnittstelle beschreibt die Funktionalität, die im OSI-Modell von einer Schicht der jeweils nächsthöheren Schicht (ihrer Anwendung) angeboten wird.

Rechnernetz Zusammenschluss von Rechnern über eine Vernetzungstechnologien. Für den englischen Begriff (Computer) Network wird in deutschen Übersetzungen oft auch der Anglizismus Netzwerk verwendet.

Schnittstellenbeschreibungssprache Eine Schnittstellenbeschreibungssprache ist eine deklarative, formale Sprache, die eine Syntax zur Beschreibung von Schnittstellen einer Software-Komponente bereitstellt. Eine IDL macht es in einem verteilten System möglich, Schnittstellen unabhängig von einer konkreten Plattform und Programmiersprache zu beschreiben, so dass sich ein oder mehrere Kommunikationsparadigmen realisieren lassen.

Single Point of Failure Mit dem englischen Begriff *Single Point of Failure* wird allgemein der Nachteil eines zentralen Elements in einem Verbund bezeichnet, da ein Ausfall dieses Elements den Ausfall des gesamten Verbundes impliziert.

Steuergerät/Electronic Control Unit (ECU) Ein Steuergerät besteht aus einem Mikrocontroller und mehreren Sensoren und Aktuatoren und kann meist über mindestens eine Vernetzungstechnologie mit anderen Steuergeräten kommunizieren.

Subnetz/Teilnetz In heutigen Fahrzeugen ergeben sich durch den Einsatz diverser Vernetzungstechnologien (CAN, FlexRay, MOST, ...) und der logischen Einteilung des Fahrzeugbordnetzes in Anwendungsdomänen mehrere Teilnetze. Die Teilnetze sind heute durch ein Application Layer Gateway untereinander verbunden.

Systemdienste Systemdienste kapseln kommunikationsrelevante Aufgaben, die nur in Koordination mehrerer Komponenten erbracht werden können oder für eine einfache Wiederverwendung geeignet sind.

Teilnetzbetrieb Unter Teilnetzbetrieb versteht man das partielle Abschalten von Steuergeräten oder ganzen Teilnetzen im Fahrzeugbordnetz, um beispielsweise die Energieeffizienz zu steigern, ohne die aus Anwendungssicht erforderliche Kommunikation zu beeinträchtigen.

Vernetzungstechnologie Eine Vernetzungstechnologie dient der Kommunikation zwischen Steuergeräten untereinander und von Steuergeräten zu einfachen Sensoren und Aktuatoren im Fahrzeugbordnetz.

Abkürzungsverzeichnis

ADAS Advanced Driver Assistance Systems 24
API Programmierschnittstelle (Application Programming Interface) . . 23
ARP Address Resolution Protocol 92
BEV Elektrofahrzeug (Battery Electric Vehicle) 52
CAN Controller Area Network 19
CE-Gerät Consumer Electronics-Gerät 3
DDS Data Distribution Service 62
DoIP Diagnostics over Internet Protocol 32
DSC Dynamic Stability Control 22
ECU Steuergerät (Electronic Control Unit) 1
FTP File Transfer Protocol 28
GNSS Global Navigation Satellite System 41
GPS Global Positioning System 41
GPRS General Packet Radio Service
HMI Mensch-Maschine-Schnittstelle (Human Machine Interface) 4
HTTP Hypertext Transfer Protocol 28
IDL Schnittstellenbeschreibungssprache (Interface Definition Language) 13
IMAP Internet Message Access Protocol 28
IP Internet Protocol 2
ISO International Standards Organization 9
LAN Local Area Network 24
LIN Local Interconnect Network 19
LTE Long Term Evolution
LVDS Low Voltage Differential Signaling 24
MOST Media Oriented Systems Transport 19
NFS Network File System 31
NSDL Network Service Definition Language 36

QoS	Dienstgüte (Quality of Service) ... 25
RPC	Remote Procedure Call ... 16
RTCP	RTP Control Protocol ... 42
RTP	Real-Time Transport Protocol ... 40
SEIS	Sicherheit in Eingebetteten IP-basierten Systemen ... 33
TCP	Transmission Control Protocol ... 11
TTP/C	Time-Triggered Protocol Class C ... 19
UDP	User Datagram Protocol ... 11
UMTS	Universal Mobile Telecommunications System
USB	Universal Serial Bus
VoIP	Voice over IP ... 41
WLAN	Wireless Local Area Network ... 41
XML	Extended Markup Language

A. Anhang: Zusammengefasster Anforderungskatalog

Anforderung 1 Verfügbarkeit einer IDL: Die Kommunikationsmiddleware muss eine IDL mit definierter Syntax und Semantik anbieten.

Anforderung 2 Beachtung bestehender Definitionskonzepte: Die IDL muss Anwendungsschnittstellen unter Berücksichtigung von bestehenden Konzepten beschreiben können. Bestehende Konzepte können z.B. CAN, FlexRay, MOST sein.

Anforderung 3 Skalierbarkeit und geringe Kopplung der IDL: Hinzunahme, Entfernung und Änderung von Komponenten sollen auf IDL-Ebene möglichst geringe Auswirkungen haben.

Anforderung 4 Verständlichkeit der IDL: Die IDL muss für einen Menschen leicht verständlich sein. Dies bedeutet insbesondere, dass sich die Syntax an verbreitete, höhersprachige Programmiersprachen anlehnen soll.

Anforderung 5 Einfache Datentypen: Die IDL muss mindestens die primitiven Datentypen boolean, byte, integer, float, string, enum, void unterstützen.

Anforderung 6 Strukturierte Datentypen: Die IDL muss mindestens Listen variabler und fester Länge, zusammengesetzte Datentypen sowie beliebige Schachtelungen dieser Datentypen unterstützen.

Anforderung 7 Wiederverwendung: Die IDL muss eine Methode zur Wiederverwendung von Schnittstellendefinitionen unterstützen. Dies kann z.B. auch Vererbung sein.

Anforderung 8 Modularisierung: Die IDL muss ein modulares Design ermöglichen.

Anforderung 9 Fehlerbehandlung: Die IDL soll die Kennzeichnung einer dedizierten Möglichkeit zur Fehlerbehandlung in der Schnittstelle ermöglichen.

Anforderung 10 Automatische Codegenerierung: Die Kommunikationsmiddleware muss auf der Grundlage einer definierten Schnittstelle automatisch Code generieren können.

Anforderung 11 Die Kommunikationsmiddleware muss ein signalbasiertes Kommunikationsparadigma umsetzen.

Anforderung 12 Die Kommunikationsmiddleware muss ein funktionsbasiertes Kommunikationsparadigma umsetzen.

Anforderung 13 Die Kommunikationsparadigmen müssen vorgegebene Aufrufsemantiken umsetzen.

Anforderung 14 Die Kommunikationsmiddleware muss IP-basierte Kommunikation außerhalb ihrer eigenen Implementierung erlauben bzw. ermöglichen, um den Vorteil der Wiederverwendung von standardisierten Protokollen des TCP/IP-Stacks nutzen zu können.

Anforderung 15 Die Kommunikationsmiddleware muss ein Marshalling von plattformabhängigen Daten in ein binäres Serialisierungsformat zur Übertragung anbieten, sowie die entsprechende Umkehrfunktion auf der Plattform des Empfängers.

Anforderung 16 Die Kommunikationsmiddleware muss ein Marshalling für gewöhnliche einfache Datentypen und für komplexe Datenstrukturen, wie sie in der objektorientierten Softwareentwicklung als Objekte gekapselt werden, beherrschen.

Anforderung 17 Die Kommunikationsmiddleware muss einen Systemdienst für Service-Management bereitstellen, dessen Hauptaufgabe das Monitoring von Zuständen im Sinne von Verfügbarkeit der teilnehmenden Dienste ist.

Anforderung 18 Die Kommunikationsmiddleware muss einen Systemdienst für Teilnetzbetrieb-Management bereitstellen, der prüft, ob ein Teilnetzbetrieb zur Erhöhung der Energieeffizienz in einem gegebenen Systemzustand sicher möglich ist, und den Teilnetzbetrieb geeignet verwaltet.

Anforderung 19 Die Kommunikationsmiddleware muss einen Systemdienst für das Dienstgüte-Management bereitstellen.

Anforderung 20 Die Kommunikationsmiddleware muss einen Systemdienst für ein Security-Management bereitstellen, das einen flexiblen Schutz durch ein Security-Framework für das IP-basierte Fahrzeugbordnetz integriert.

Anforderung 21 Die Kommunikationsmiddleware muss Interdomänenkommunikation zwischen Steuergeräten unterschiedlicher Leistungsfähigkeit ermöglichen.

Anforderung 22 Die Kommunikationsmiddleware berücksichtigt die Anwendungsmigration auf leistungsfähigere Steuergeräte unter Beibehaltung der Kompatibilität.

Anforderung 23 Eine Kommunikationsmiddleware muss auf allen Steuergeräten des IP-basierten Fahrzeugbordnetzes lauffähig sein.

Anforderung 24 Eine Kommunikationsmiddleware soll bestehende Praktiken in der Automobilindustrie berücksichtigen, um als standardisierte Lösung herstellerübergreifend eingesetzt werden zu können.