Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Fabian Kempf

Simulationsbasierte Analyse heterogener Fahrzeugnetzwerke:
Generierung, Simulation und Evaluation

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Computer Science

Fabian Kempf

Simulationsbasierte Analyse heterogener Fahrzeugnetzwerke: Generierung, Simulation und Evaluation

Masterarbeit eingereicht im Rahmen der Masterprüfung
im Studiengang Informatik
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Franz Korf
Zweitgutachter: Prof. Dr.-Ing. Andreas Meisel

Abgegeben am 26. September 2014

Fabian Kempf

Thema der Masterarbeit
Simulationsbasierte Analyse heterogener Fahrzeugnetzwerke: Generierung, Simulation und Evaluation

Stichworte
Echtzeit-Ethernet, TTEthernet, Bussysteme, Automotive Anwendungen, Domänenspezifische Sprache, Netzwerkgeneration, Netzwerksimulation, OMNeT++

Kurzzusammenfassung
Im klassischen Automobilnetz kommunizieren die Steuergeräte der verschiedenen Domänen überwiegend untereinander. Bei neuen Anwendungen sind Informationen von Systemen aus unterschiedlichen Domänen notwendig. Dies erfordert domänenübergreifende Kommunikation in einem heterogenen Netzwerk. Ein Ansatz für die Vernetzung der Domänen liegt in einem echtzeitfähigen Ethernetbackbone, das sowohl bandbreitenintensive als auch sicherheitskritische Anwendungen unterstützt. Um verschiedene Netztopologien des heterogenen Netzwerkes zu vergleichen, eignen sich simulationsbasierte Analysen. Ziel dieser Arbeit ist die Herausstellung charakteristischer Merkmale der Kommunikation im Automobil zur simulationsbasierten Analyse mit einem Backbone. Für die Konfiguration ist der Entwurf und die Umsetzung einer domänenspezifischen Sprache mit den Anforderungen an relevante Kommunikationssysteme im Automobil herauszustellen, die einen gesamtheitlichen Netzwerkbeschreibungsansatz erlaubt. Dieser Ansatz ermöglicht die einfache Analyse komplexer heterogener Fahrzeugnetzwerke, von der Generierung über die Simulation bis hin zur Evaluierung.

Title of the paper
Simulation based analysis of heterogeneous in-car networks: Generation, simulation and evaluation

Keywords
Real-time Ethernet, TTEthernet, Bussystems, Automotive Applications, Domain-specific language, Networkgeneration, Networksimulation, OMNeT++

Abstract
In common automotive networks, the communication of control units is limited by isolated domains. Modern applications use information from different domains and thus, a global communication is required. A real-time ethernet backbone, which provides high data rates as well as safety-critical traffic transmission is a promising approach as underlaying network. For a real-world deployment, a detailed analysis of the different networks and their interaction on a shared backbone is necessary. Simulation based analyses are best suited to address this problem. This thesis classifies common in-car networks and simulates a real-world deployment with an underlaying real-time ethernet backbone. For this purpose, a domain-specific language is developed, that allows a description of the involved network technologies. This approach enables the generation, simulation and evaluation of complex heterogeneous in-car networks.

Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die durch ihre Unterstützung zum Gelingen dieser Masterarbeit beigetragen haben.

Mein besonderer Dank gilt Herrn Prof. Dr. Franz Korf für die Förderung des CoRE-Projekts, ohne das diese Arbeit nicht entstanden wäre, die stets bereichernde Kritik und Geduld.

Ebenfalls möchte ich mich bei der CoRE-Gruppe der HAW für die konstruktiven Gespräche und die Diskussionsbereitschaft bedanken, durch die mir immer wieder neue Anstöße und Ideen für die Bearbeitung des Themas gegeben wurden.

Des Weiteren möchte ich mich bei Till Steinbach und den Mitwirkenden der Simulation bedanken, die für die Simulationsgrundlage dieser Arbeit sorgten und in Diskussionen die Lösung von Problemstellungen unterstützten.

Außerdem bedanke ich mich bei meinen Eltern, meiner Familie und meiner Freundin Nadine für die Unterstützung während des Studiums und der Entstehungszeit dieser Arbeit.

Inhaltsverzeichnis

1 Einleitung **1**

2 Hintergrund und verwandte Arbeiten **7**
2.1 Kommunikation im Automobil 7
2.1.1 Relevante Kommunikationssysteme im Automobil 8
2.1.2 Echtzeitfähiges Ethernet als Backbone in Automobilnetzwerken 12
2.1.3 Netzwerktopologien 20
2.2 Simulation heterogener Fahrzeugnetzwerke 21
2.2.1 Simulationsmodellierung 22
2.2.2 Verifikation und Validierung von Simulationsmodellen 23
2.2.3 Diskrete, ereignisbasierte Simulation 24
2.2.4 OMNeT++ als Simulationsumgebung 26
2.2.5 Netzwerkprotokollframeworks für OMNeT++ 28
2.3 Domänenspezifische Sprachen zur Konfigurationsgenerierung 29
2.3.1 Grundlagen domänenspezifischer Sprachen 29
2.3.2 Framework zur Entwicklung von Programmiersprachen und DSLs . . . 30
2.4 Verwandte und vergleichbare Arbeiten 32

3 Anforderungen zur Evaluierung heterogener Fahrzeugnetzwerke **34**
3.1 Datenmodellierung 35
3.1.1 Kommunikationsnachrichten als Analysegrundlage 35
3.1.2 Aufbau einer Kommunikationsmatrix 36
3.1.3 Klassifizierung von Nachrichten und Endgeräten 37
3.1.4 Topologien für ein Echtzeit-Ethernet-Backbone 39
3.2 Simulation heterogener Netzwerke 41
3.2.1 Benötigte Simulationskomponenten für heterogene Netzwerke 42
3.2.2 Analysemethoden zur Evaluierung 43
3.2.3 Metriken für die simulationsbasierte Analyse 44
3.2.4 Simulationskonfiguration und Generierung 45

4 Konzept und Architektur **48**
4.1 Konzept zur vereinheitlichten Analyse 48
4.1.1 Ausarbeitung eines Datenmodells 48

4.1.2 Klassifizierung der Kommunikationsnachrichten . . . 50
4.1.3 Abstraktionsstufen von Endgeräten . . . 54
4.1.4 Topologievariationen eines Echtzeit-Ethernet-Backbones . . . 56
4.2 Architektur und Komponenten . . . 60
4.2.1 Simulationskomponenten des Modells . . . 60
4.2.2 Konzept zur Generierung der Simulationskonfiguration . . . 66
4.2.3 Modell zur Konfigurationsgenerierung . . . 71
4.3 Konzept zur Evaluation heterogener Fahrzeugnetzwerke . . . 73
4.3.1 Metriken als Analysegrundlage . . . 73
4.3.2 Aufzeichnung und Auswertung der Metriken . . . 77

5 Umsetzung einer Lösung zur Generierung, Simulation und Evaluation 80
5.1 Domänenspezifische Sprache zur Generierung von Fahrzeugnetzwerken . . . 80
5.1.1 NedG-File Format . . . 80
5.1.2 Validationsregeln . . . 84
5.1.3 Interpretation der Sprache . . . 85
5.1.4 Generieren von Konfigurationsdateien . . . 87
5.2 Simulation und Evaluation . . . 88
5.2.1 Aufzeichnung der Metriken . . . 89
5.2.2 Ausgabe und Evaluierung der Metriken . . . 90

6 Validierung des Generierungsprozesses 92
6.1 Konfiguration eines Time-Triggered Ethernet Backbones . . . 92
6.2 Konfiguration eines Bussystems . . . 95
6.3 Konfiguration einer heterogenen Lösung . . . 97

7 Fallstudie zur Untersuchung heterogener Netztopologien 100
7.1 Hierarchisches Netzwerk mit abstrahierten Namen . . . 100
7.2 Simulationsergebnisse . . . 102
7.3 Vergleich mit einer weiteren Topologie . . . 106
7.4 Bewertung der domänenspezifischen Sprache . . . 107

8 Zusammenfassung, Fazit und Ausblick 108
8.1 Zusammenfassung der Arbeit . . . 108
8.2 Fazit zur Analyse heterogener Netzwerke . . . 110
8.3 Ausblick auf zukünftige Arbeiten . . . 110

A Validierungsbeispiele der entwickelten DSL 112

Literaturverzeichnis 117

Abbildungsverzeichnis

1.1 Bordnetz eines modernen Automobils 2
1.2 Zielsetzung zur Evaluation heterogener Fahrzeugnetzwerke 4

2.1 Steuergeräte am Lowspeed- und Highspeed-CAN 9
2.2 Alternative Ansätze zur Implementierung von Echtzeit-Ethernet im OSI-Modell 14
2.3 Eingeführte Servicelayer von Time-Triggered Ethernet 15
2.4 Format eines TT- und RC-Frames bei TTEthernet 16
2.5 Nachrichtenklassen von Time-Triggered Ethernet 18
2.6 Time-Triggered Ethernet Toolchain 19
2.7 Ausgewählte Netzwerktopologien 20
2.8 Simulationspipeline 22
2.9 Zustandsübergänge in kontinuierlichen und diskreten Systemen 25
2.10 Klassifizierung von Simulationsmodellen 25
2.11 Modellstruktur in OMNeT++ 27
2.12 Konkrete und abstrakte Syntax einer Sprache 30
2.13 Syntax von Feature Diagrammen 32

3.1 Anforderung für die Evaluierung unterschiedlicher Topologien 34
3.2 Klassische Netzwerktopologie im Automobil mit zentraler Gatewayarchitektur 39
3.3 Physikalischer Graph einer generischen Topologie 40
3.4 Abstraktion verschiedener Netzwerkspezifikationen auf ein Modell 41
3.5 Benötigte Simulationsmodelle verschiedener Komponenten eines heterogenen Netzwerkes 42

4.1 Klassifizierung von Nachrichten 52
4.2 Nachrichtenzahlen in den Klassen eines Endgerätes mit Sender- und Empfängerseite 54
4.3 Klassifizierungsstrategien von Endgeräten 55
4.4 Heterogenes Netzwerk mit zentraler Stern-Topologie 57
4.5 Heterogenes Netzwerk mit Double-star Backbone 58
4.6 Heterogenes Netzwerk mit Daisy-Chain Backbone 59
4.7 Heterogenes Netzwerk mit hierarchischer Topologie nach Domänen 60

4.8 Modellierung des TTEthernetHosts mit Unterstützung des AVB-Protokolls und dem physikalischen Port . . . 62
4.9 Modellierung eines CAN-Knotens . . . 63
4.10 Gatewaymodell . . . 65
4.11 Homogenes Graphmodell zur Übersetzung zwischen DSL und Konfigurationsdateien . . . 66
4.12 Graphmodell mit unterschiedlichen Komponenten als Knoten . . . 67
4.13 Features eines Netzwerkes . . . 69
4.14 Features eines Endgerätes . . . 70
4.15 Features einer heterogenen Nachricht . . . 71
4.16 Designstrategien für Gateways . . . 72
4.17 Präziser Aufzeichnungsmodus für Metriken . . . 78
4.18 Performanter Aufzeichnungsmodus für Metriken . . . 79

5.1 Überführung der Sprachelemente in das Graphmodell . . . 86
5.2 Bandwidth Allocation Gaps im Generierungsprozess . . . 87
5.3 Linkauslastung bei verschiedenen Aufzeichnungsmodi . . . 91

6.1 Validierung der Backbonegenerierung . . . 93
6.2 Linkauslastung im generierten Backbone . . . 94
6.3 Generierungsvalidierung eines CAN-Netzwerkes . . . 95
6.4 Heterogenes Netzwerk zur Validierung der DSL . . . 97
6.5 Latenzverhalten in einem heterogenen Netzwerk . . . 99

7.1 Hierarchisches, heterogenes Netzwerk mit hinzugefügten Kameras . . . 101
7.2 Linkauslastung eines stark belasteten Knotens in einer hierarchischen Netztopologie . . . 102
7.3 Latenzen kamerabasierter Daten in einem Empfänger . . . 103
7.4 Latenzen von Kameradaten an verschiedenen Knoten . . . 103
7.5 Latenzen aller Time-Triggered Nachrichten an einem Switchlink . . . 104
7.6 Veränderter Offset beeinflusst die Latenzen von Time-Triggered Nachrichten . 105
7.7 Daisy-Chain basierte Topologie mit Domäne für bandbreitenintensive Anwendungen . . . 106
7.8 Latenzvergleich beider Topologien . . . 107

Tabellenverzeichnis

3.1 Spezifikation von Signalen für Punkt-zu-Punkt Verbindungen nach Tindell / Burns (Tindell / Burns 1994). 35
3.2 Kommunikationsmatrix mit einer Sortierung nach Netzknoten. (Schäuffele / Zurawka 2013)[S. 84]. 37
3.3 Unterteilung von Bussystemen in SAE-Klassen (Reif 2009)[S. 14]. 38

4.1 Beispiele von Nachrichteneinteilungen auf die vier verschiedenen Klassen. . . 53

5.1 Überblick über die relevanten Metriken in Automobilnetzwerken und ihren Aufzeichnungspunkten in der Simulation. (vgl. Kempf 2014) 89

6.1 Aufteilung der Pakete in den einzelnen Modulen der Endgeräte. 96
6.2 Empfangene Nachrichten der einzelnen Systeme in dem heterogenen Netzwerk mit der Anzahl der Pakete pro Nachricht. 98

1 Einleitung

Der Leistungszuwachs der IT in einem Automobil ist durch einen ständig wachsenden Prozess gekennzeichnet. Aufgrund anpassbarer Hardware und mehr Rechenleistung ist es möglich, komplexere und innovativere Konzepte mit dem Fahrzeug zu verbinden. Bis zu 70 dedizierte Kleincomputer, auch elektronische Steuergeräte genannt, überwachen Funktionen des Fahrzeugs und unterstützen den Fahrer durch sogenannte Fahrerassistenzsysteme (vgl. Saad 2003)[S. 03]. Mit einer Vielzahl von Sensoren und Aktoren werden unterschiedliche Bereiche des Fahrzeugs ständig überwacht und gesteuert.

Zu den bisher eingesetzten Technologien wie der Fahrdynamikregelung (ESP) und dem Antiblockiersystem (ABS), die in nahezu jedes Automobil Einzug hielten und das Fahrverhalten unterstützen, sorgen vielfältige Sensoren in oberen Mittelklasse- und Oberklassewagen für einen Informationsgewinn außerhalb des Fahrzeugs. Durch Kameras, Radar- und Ultraschallsensoren kann die Umgebung erfasst werden. Sie ist für sicherheitskritische Systeme, wie Unfall-Erkennung und Fußgängerschutz bis hin zur autonomen Fahrzeugsteuerung, die alle Fahrerassistenzsysteme vereint, nutzbar. Alle diese Fahrerassistenzsysteme haben gemein, dass mehrere unabhängige Kleincomputer beteiligt sind und miteinander kommunizieren. Dabei präsentieren sie sich nach außen hin als ein System, das als *verteiltes System* bezeichnet wird (vgl. Tanenbaum / Steen 2008). Abbildung 1.1 zeigt das Bordnetz eines modernen Automobils.

Die Kommunikation zwischen den Systemen im Fahrzeug läuft über serielle Bussysteme, bei denen mehrere Teilnehmer gemeinsame Übertragungswege nutzen. Das Bordnetz, das die Gesamtheit aller Elemente der Hardware beschreibt (vgl. Winner / Hakuli / Wolf 2009)[S. 85], besteht aus differenten Bussystemen, bei denen jedes für unterschiedliche Aufgabenstellungen genutzt wird. Durch Anforderungen an die Kosten, die Datenrate, die Sicherheitsrelevanz und den Determinismus haben sich heute die Bussysteme CAN, LIN, MOST und FlexRay als Standard für die Datenkommunikation im Automobil etabliert (vgl. Marscholik / Subke 2011)[S. 03]. Allerdings spielt die domänenübergreifende Kommunikation eine immer größer werdende Rolle. Steuergeräte, die physikalisch zu unterschiedlichen Bussystemen gehören, werden meist über ein zentrales Gateway logisch miteinander verbunden und können über dieses Gateway Nachrichten austauschen. Durch die schnell ansteigende Anzahl der Steuergeräte, die sich nicht mehr einer genauen Domäne zuordnen lassen und den daraus resultierenden erhöhten Anforderungen an das Bordnetz ist ein dezentraler Ansatz mit einem bussystemverbindenden Fahrzeugbackbone besser geeignet. Dabei stellt sich die Frage,

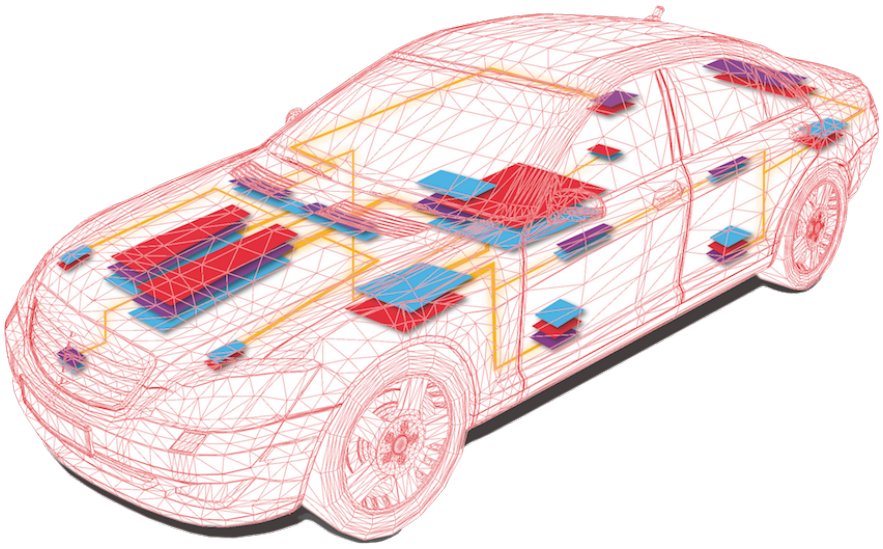

Abbildung 1.1: *Das Bordnetz eines modernen Automobils (Quelle: (CoRE-Arbeitsgruppe [a]))*

welches Kommunikationssystem sich dafür als sichere Alternative mit genügend Bandbreite und einem garantierten, deterministischen Nachrichtenverhalten eignet. Gerade bei dem Einsatz kamerabasierter Assistenzsysteme spielen beide Anforderungsarten eine übergeordnete Rolle.

Ethernet bezeichnet den am weitesten verbreiteten Standard für lokale Netze (vgl. Institute of Electrical and Electronics Engineers 2005). Sowohl die technische Planung als auch die Instandhaltung sind im Gegensatz zu anderen Kommunikationssystemen vergleichsweise kostengünstig (vgl. GE Fanuc Intelligent Platforms 2009). Durch die Spezifizierung von Übertragungsraten zwischen 10 Mb/s und 100 Gb/s unterstützt es derzeitige sowie zukünftige Anforderungen hinsichtlich der erforderlichen Bandbreite. Allerdings fehlt Ethernet in seiner ursprünglichen Form das deterministische Verhalten, das für sicherheitskritische Anwendungen unerlässlich ist. Dieses zeichnet sich dadurch aus, dass die Übertragung präzise, schnell und vor allem zeitlich punktgenau stattfindet. Beispielsweise muss sich ein Airbag im Fahrzeug je nach Geschwindigkeit zwischen 10 ms und 40 ms explosionsartig aufblasen (vgl. Christ / Büttner 2012). Bei zu früher oder nicht rechtzeitiger Auslösung sind unbeabsichtigte Effekte auf den Fahrzeuginsassen nicht auszuschließen. Daher ist es notwendig, dass die Ereignisauslösung genau im richtigen Zeitpunkt eintrifft und nicht – wie so häufig falsch angenommen – so schnell wie möglich.

Für Fahrzeugnetzwerke ist neben der zeitgesteuerten Kommunikationsart, bei der viele Applikationen eine regelmäßige Übertragung kleiner Nachrichten benötigen, eventbasierter Verkehr

notwendig. Durch sporadisch auftretende, asynchrone Nachrichten können Alarmsituationen und Systemaktualisierungen durchgeführt werden. Daher haben sich zur Übertragung von Echtzeitdaten mehrere Varianten von Echtzeit-Ethernet gebildet. Time-Triggered Ethernet (vgl. Steiner 2008) ermöglicht als eine Erweiterung zu Ethernet sowohl zeitgesteuerte als auch eventbasierte Kommunikationsnachrichten und bietet zudem eine vollständige Abwärtskompatibilität zum Ethernetstandard.

Mit einem leistungsstarken Fahrzeugbackbone stellt sich die Frage, wie die bevorstehende Entkopplung von Funktionen und ihrer physikalischen Position die zukünftige Netzstruktur im Automobil beeinflusst. Zur Untersuchung unterschiedlicher Netztopologien und um Leistungsdaten spezifischer Szenarien zu erlangen, ohne große Kosten für neue Hardware zu investieren, spielen Simulationstechniken eine immer größer werdende Rolle (vgl. Jasperneite / Watson 2008). Wegen der Möglichkeit jedes Bussystem zu modellieren und mit anderen Modellen kombinieren zu können, ist die Simulation ein geeignetes Werkzeug zum Testen und Analysieren von Netzwerkprotokollen. Dabei steht die heterogene Herangehensweise über verschiedene Bussysteme hinweg im Vordergrund dieser Arbeit. Diese erfordert einen ganzheitlichen Ansatz, bei dem nicht mehr jede Domäne einzeln betrachtet wird, sondern mit einem Generierungsansatz eine Aufteilung in verschiedene Domänen stattfindet. Hierdurch wird die Evaluierung komplexer Topologien ermöglicht.

Problemstellung und Zielsetzung

Ziel dieser Arbeit ist die Herausstellung charakteristischer Merkmale relevanter Kommunikationssysteme im Automobil, damit diese über ein Backbone miteinander kommunizieren und simulationsbasiert analysiert werden können. Dafür ist der Entwurf und die Umsetzung einer domänenspezifischen Sprache mit den Anforderungen an relevante Kommunikationssysteme im Automobil herauszustellen, die einen gesamtheitlichen Ansatz einer übersichtlichen Netzwerkbeschreibung erlaubt. Dieser Ansatz ermöglicht die einfache Analyse von komplexen heterogenen Fahrzeugnetzwerken, von der Generierung über die Simulation bis hin zur Evaluierung.

Im Fahrzeug der Zukunft wird es möglich sein, dass ausgewählte Endgeräte Informationen anderer Endgeräte erhalten und aufbereiten können. Aufgrund des immer weiter steigenden Bedarfs nach domänenübergreifender Kommunikation ist es notwendig, den Fokus auf eine ganzheitliche Betrachtung des Netzwerkes zu legen und verschiedene Kommunikationssysteme miteinander zu kombinieren. Diese simulationsbasierte Analyse heterogener Fahrzeugnetzwerke lässt sich in verschiedene Teilaspekte untergliedern, die in Abbildung 1.2 zusammengefasst werden:

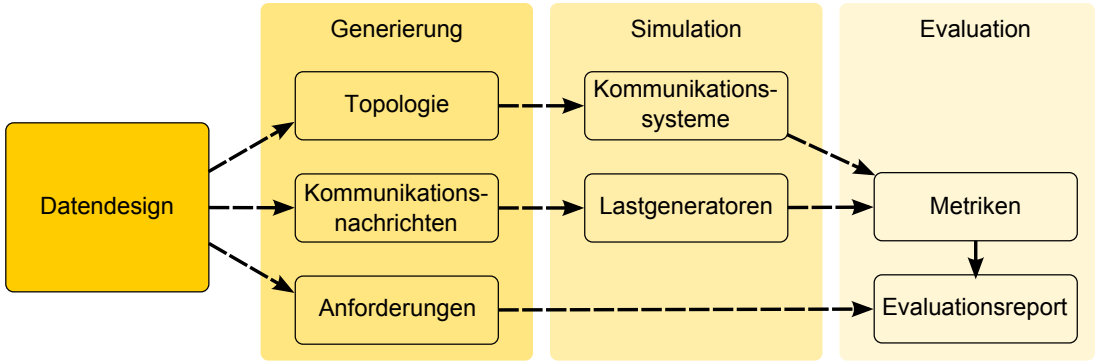

Abbildung 1.2: *Zielsetzung zur Evaluation heterogener Fahrzeugnetzwerke. Ein Datendesign ist die Grundlage für die Simulation. Aus diesem können Topologie und Kommunikationsnachrichten für die Bussysteme und Lastgeneratoren der Simulation abgeleitet werden. In der Evaluationsphase werden die aufgezeichneten Metriken der Simulation und Anforderungen des Datendesigns benötigt.*

Realistisches Datendesign: Die gesteigerten Anforderungen an ein Fahrzeug finden sich in einer großen Anzahl elektronischer Komponenten wieder. Diese tauschen über eine Vielzahl von Nachrichten Informationen untereinander aus. Jede Nachricht stellt unterschiedliche Anforderungen an das Netzwerk dar, die es zu spezifizieren gilt. Ist anschließend für jede Nachricht die passende Information zusammengetragen, kann das Netzwerk mit verschiedenen Topologien evaluiert werden. Eine standardisierte Anzahl an Komponenten, wie sie miteinander verbunden sind und mit welchen Nachrichten sie kommunizieren, ist dabei entscheidend für den Vergleich mit anderen Ansätzen und einer erfolgreichen Evaluierung.

Simulation unterschiedlicher Kommunikationssysteme: Da im Automobil derzeitig mehrere Bussysteme eingesetzt werden, sich die überwiegenden Simulationen allerdings lediglich mit einem Bussystem beschäftigen, liegt der Fokus auf der ganzheitlichen Betrachtung einer domänenübergreifenden Kommunikation über mehrere Bussysteme hinweg. Dafür sind geeignete Modelle der Quell- und Zielbussysteme sowie die Simulationsmodelle eines echtzeitfähigen Backbones nötig, das diese Netzwerke miteinander verbindet.

Simulationsgenerierung: Um eine Vielzahl von Topologien mit dem gleichen Datendesign zu simulieren, ist es notwendig, das Erstellen der Simulation partiell zu automatisieren. Dabei sind die Nachrichten, die ein Endsystem zu mehreren anderen Endsystemen verschickt, bei jeder Simulation gleich. Die Endgeräte werden allerdings über verschie-

dene Topologien miteinander verbunden, simuliert und ausgewertet. Das Problem, das hierbei entsteht, ist, dass die Konfigurationsdateien der Simulation bei jeder Topologieveränderung angepasst werden müssen, und Konfigurationsfehler erst bei der Simulation oder bei der Evaluierung auffallen. Gerade bei der Planung eines Netzwerkes mit verschiedenen Kommunikationssystemen ist es notwendig, den Netzwerkdesigner mit einer vereinheitlichten Darstellung zu unterstützen und verschiedene Kommunikationssysteme zu verknüpfen. Dazu ist es erforderlich, die Eigenschaften der einzelnen Kommunikationssysteme festzuhalten, Gemeinsamkeiten zu finden und anschließend in einer geeigneten Form dem Designer zur Verfügung zu stellen.

Analyse heterogener Netzwerke: Der letzte Teilaspekt liegt in der Analyse der während der Simulation aufzuzeichnenden Metriken und der Entwicklung eines Evaluierungsreports, mit dem das Verhalten des heterogenen Netzwerkes gezeigt werden kann. Schon während des Generierungsprozesses sollten die relevanten Metriken einschließlich ihrer jeweiligen Anforderungen festgelegt werden. Die Evaluierungsphase beschäftigt sich anschließend mit der Aufbereitung der unterschiedlichen Metriken, also der Bewertung, Einhaltung der Anforderungen und Erstellung eines Analyseberichts.

Inhaltlicher Aufbau der Arbeit

Die Arbeit gliedert sich in neben der Einleitung in sieben weitere Kapitel. Zunächst werden im zweiten Kapitel Hintergrundinformationen zu bisherigen Kommunikationssystemen gegeben und die Schwierigkeiten bei echtzeitkritischen Datenübertragungen beleuchtet. Dazu wird mit Time-Triggered Ethernet als denkbarem Fahrzeugbackbone der Zukunft eine Möglichkeit vorgestellt, die deterministischen Datenverkehr zulässt. Zudem werden notwendige Begriffe der Simulation erläutert, die eine Modellierung in der Konzeption und Umsetzung ermöglichen. Mit der Einführung zu domänenspezifischen Sprachen werden die Besonderheiten bei der Simulationsgenerierung herausgearbeitet.

Das dritte Kapitel beschäftigt sich mit den Anforderungen einer Evaluierung von heterogenen Fahrzeugnetzwerken. Dieses Kapitel unterteilt die Notwendigkeit zur Datenmodellierung und die daraus resultierende Simulationsanalyse in zwei separate Abschnitte. Die Datenmodellierung verdeutlicht dabei die wachsende Komplexität und die fehlende vereinheitlichte Datenbasis, um Simulationsergebnisse besser vergleichen zu können. Die Simulation als konkretes Beispiel zur Evaluierung dient darauf aufbauend der Analyse benötigter Modellierungsschritte und der Notwendigkeit, die Simulation durch unterstützende Konfigurationsgenerierung in den Produktionsprozess zu integrieren.

Im vierten Kapitel wird im Bezug auf das vorangegangene das Konzept hinter der Analyse und die Arbeitsschritte der Generierung, der Simulation und der Evaluierung erläutert. Dazu

zählen unter anderem die Entwicklung eines Datenmodells, das sich zur Evaluierung eignet, sowie die Herausarbeitung und Auswertung geeigneter Metriken.

Aufbauend auf den Vorbetrachtungen wird im fünften Kapitel eine konkrete Implementierung einer domänenspezifischen Sprache zur Beschreibung von heterogenen Netzwerken und der Generierung von Simulations- und Auswertungskonfigurationen erläutert.

Das sechste Kapitel beschäftigt sich mit der Validierung der domänenspezifischen Sprache und beantwortet somit die Frage, ob sich die DSL zur Generierung heterogener Systeme eignet und funktionsfähige Simulationsmodelle beschrieben werden können.

Im siebten Kapitel soll anhand eines Anwendungsbeispiels die Evaluierung eines größeren heterogenen Netzwerkes verdeutlicht werden, bevor im letzten Kapitel 8 nach einer Zusammenfassung der Ergebnisse ein Fazit gezogen wird und die Arbeit mit dem Ausblick abschließt.

2 Hintergrund und verwandte Arbeiten

In diesem Kapitel werden die Grundlagen für die simulationsbasierte Analyse heterogener Systeme erläutert. Zu Beginn werden die Kommunikationssysteme im Fahrzeug mit ihren charakteristischen Ausprägungen erklärt und dargestellt, warum ein Time-Triggered Ethernet als Alternative im Fahrzeugbackbone zulässig ist. Anschließend werden Simulationsgrundlagen im Hinblick auf die diskrete, ereignisbasierte Simulation, die sich für die Modellierung von Netzwerken besonders eignet, vermittelt. Zur Erstellung komplexer, heterogener Netzwerke mit unterschiedlichsten Topologien wird im darauffolgenden Kapitel auf eine domänenspezifische Sprache zum Ansprechen unterschiedlicher Simulationsmodelle eingegangen. Zuletzt werden verwandte und vergleichbare Arbeiten vorgestellt, die sich mit der Simulation zeitkritischer Kommunikationssysteme auseinandergesetzt haben und die darlegen, wie sie bei der Konfiguration komplexer Netzwerke vorgegangen sind.

2.1 Kommunikation im Automobil

Die Kommunikation im Automobil ist derzeitig in verschiedene Domänen untergliedert. Jede Domäne hat dabei andere Charakteristika, so dass sich unterschiedliche Anforderungen an das Kommunikationssystem, das die Systeme miteinander verbindet, ergeben. Dadurch hat sich in jeder Domäne ein anderes Kommunikationssystem etabliert. Während das MOST-Bussystem für hohe Datenraten optimiert ist und somit Bandbreite für Entertainmentanwendungen zur Verfügung stellen kann, ist der zeitgesteuerte LIN-Bus besonders kostengünstig einzusetzen, kann jedoch keine hohen Datenraten liefern. Jedes Bussystem hat somit unterschiedliche Aufgaben, und kein Bussystem kann alle Anforderungen gleichzeitig erfüllen. (vgl. Marscholik / Subke 2011)[S. 74]

Die folgenden Abschnitte beschäftigen sich mit den relevanten Bussystemen im Automobil und mit dem Netzwerkprotokoll Time-Triggered Ethernet (TTE), das eine Lösung für ein zukünftiges, intelligentes Backbone im Automobil zwischen den Domänen darstellt (vgl. Steinbach / Korf / Schmidt 2010).

2.1.1 Relevante Kommunikationssysteme im Automobil

Bussysteme sind für die Datenkommunikation im Automobil unerlässlich. Durch die Einsparungen an Kabeln, Kosten und Fahrzeuggewicht, die Integration weiterer elektronischer Systeme, das Updaten von Software und dem Wegfall der n:m-Verkabelungen konnte der Anteil der verbauten Komponenten innerhalb eines Fahrzeugs steigen. Jedes Bussystem wird dabei in unterschiedlichen Domänen eingesetzt, weil es verschiedenen Auswahlfaktoren in dem jeweiligen Bereich hinsichtlich der Bandbreite, der Störsicherheit, der Zahl der adressierbaren Knoten und der Verkabelungstopologie genügt (vgl. Dohmke 2002). Nachfolgend werden die relevanten Bussysteme im Automobil mit ihren protokollspezifischen Eigenschaften und den Nachrichteneigenschaften beschrieben. Diese sind für eine abstraktere Netzwerkbeschreibung und somit für die Möglichkeit heterogene Netzwerke zu simulieren, notwendig.

Lin-Bus

Das *Local Interconnect Network* (LIN) (LIN-Administration) ist ein serielles Bussystem, das für den kostengünstigen Einsatz im Automobil für Sensoren, Aktoren und einfache Steuergeräte vorgesehen ist. Die Kostenreduktion wird dabei durch eine Verbindung mit nur einem Draht erzielt, mit dem eine Übertragungsrate von bis zu 20 kBit/s möglich ist. Es handelt sich um ein Master-Slave Bussystem, bei dem der Master einen Schedule aller Nachrichten der Slaves besitzt. Soll ein entsprechender Slave seine Nachricht übertragen, legt der Master einen Header mit einer speziellen Nachrichtenadresse auf den Bus. Anschließend kann der Slave die Datenbytes übertragen.

Nachfolgend werden die protokollspezifischen Eigenschaften gemäß Streichert / Traub (Streichert / Traub 2012) aufgezählt:

- **Name:** Name des LIN-Busses
- **Baudrate:** Übertragungsgeschwindigkeit des Lin-Busses - bis zu 20 kBit/s, empfohlen: 2400 bit/s, 9.600 bit/s, 19.200 bit/s
- **Version:** Durch die unterschiedlichen Versionen des Protokolls kann sichergestellt werden, dass die Steuergeräte einheitlich am LIN-Bus konfiguriert werden.

Für die Nachrichten, die über den Bus übertragen werden, gelten folgende Eigenschaften:

- **Name:** Name der Nachricht
- **Länge:** Länge des Datenfeldes - 2, 4 und 8 Byte + 3 Byte Kontrollinformation
- **Identifier:** Kennung der Nachricht - 1 bis 64

CAN-Bus

Der „Controller Area Network - Bus“ (CAN-Bus) (Robert Bosch GmbH) stellt das verbreitetste Bussystem im Automobil dar. Es ist ein serielles Bussystem, das bei den Feldbussen eingeordnet wird. Das seit 1981 von Bosch und Intel entwickelte Kommunikationsprotokoll dient zur Datenkommunikation zwischen Steuergeräten im Automobil. Das Bussystem hat sich in drei Ausführungen etabliert. Die schnellste stellt der *Highspeed-CAN* dar, der mit einer Brutto-Datenrate von 500 kBit/s die Daten zwischen den Steuergeräten des Antriebsstranges überträgt. Die etwas langsamere Variante, der *Lowspeed-CAN*, wird zur Übertragung verschiedener Komfortfunktionen wie der Klimaanlage genutzt. Abbildung 2.1 zeigt ein Kommunikationsbeispiel mit beiden Bussystemen.

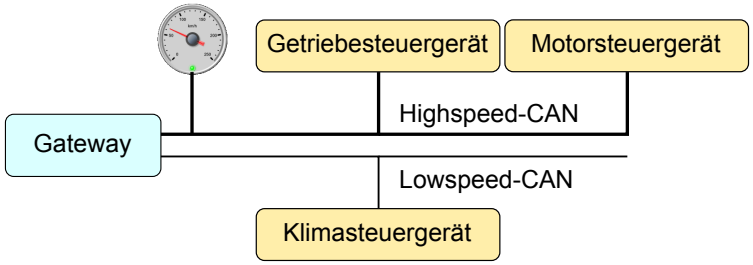

Abbildung 2.1: *Verschiedene Steuergeräte, die über den Highspeed- und den Lowspeed-CAN kommunizieren. Über ein Gateway sind die beiden Bussysteme verbunden (vgl. Marscholik / Subke 2007)[S. 39])*

Neben den beiden genannten Varianten zählt noch der *Single-Wire-CAN* mit einer Datenrate bis zu 16 kBit/s für Steuergeräte mit niedriger Sicherheitsrelevanz zu den CAN-Bus Ausführungen.

Für die Datenübertragung innerhalb des CAN-Systems wird ein dezentrales Verfahren für den Buszugriff eingesetzt. Dieses wird Carrier Sense Multiple Access / Collision Resolution Verfahren (CSMA/CR) genannt. Jedes Endsystem kann gleichberechtigt an der Kommunikation teilnehmen. Dazu gibt es eine bitweise Arbitrierungsphase, bei der jedes System beim Senden seines Identifiers den Bus überwacht. Wenn zwei Systeme gleichzeitig ihren Identifier senden, überschreibt bei unterschiedlichen Bits das dominante Bit das rezessive. Dadurch weiß das System mit dem rezessiven Bit, dass der Bus die Nachricht eines höher priorisierten Systems überträgt.

Die Nachrichten eines CAN-Netzwerkes bestehen aus vier verschiedenen Arten. Zwei davon werden zur Datenübertragung genutzt und sind somit für Applikationen interessant. Über sogenannte *Data-Frames* werden Daten an andere Knoten übermittelt. Jedes Data-Frame besteht aus einer Nachrichten-ID, der ein Sender zugeordnet wird, dem Payload, der die Daten

enthält, der Größe des gesamten Frames und dem RTR-Bit. Letzteres wird dazu verwendet, zwischen dem Data-Frame und dem erweiterten Data-Frame, nämlich dem *Remote-Frame* zu unterscheiden. Dieses stellt die zweite Art der Nachrichten dar. Es wird dazu genutzt, andere Teilnehmer aufzufordern, ihre Daten zu senden.

Für ein deterministisches Verhalten des Bussystems, bei dem Nachrichten zu definierten Zeiten verschickt werden, ist das Time-triggered CAN-Bus Protokoll (TTCAN) entwickelt worden. Dieses setzt auf höheren Protokollebenen an, wodurch über Zeitslots und einer Uhrensynchronisation zwischen den teilnehmenden Systemen eine Echtzeitsteuerung erreicht werden soll.

Nachfolgend werden die protokollspezifischen Eigenschaften gemäß Streichert / Traub (Streichert / Traub 2012) aufgezählt:

- **Name:** Name des CAN-Busses
- **Baudrate:** Übertragungsgeschwindigkeit des CAN-Busses (meist in kBit/s)
- **Version:** Typischerweise: CAN 2.0A, CAN 2.0B

Für die Nachrichten, die über den Bus übertragen werden, gelten folgende Eigenschaften:

- **Name:** Name der Nachricht
- **Identifier:** Kennung der Nachricht - Version 2.0A: 11 bit. Version 2.0B: 29 bit
- **Sendetyp:** Leitet sich aus der Anforderung der gemappten PDU ab
- **Periode:** Leitet sich aus der Anforderung der gemappten PDU ab
- **Entprellzeit:** Mindestabstand zwischen zwei aufeinanderfolgenden Nachrichten
- **Offset:** Leitet sich aus der Anforderung der gemappten PDU ab
- **Länge:** Länge des Datenfeldes

MOST-Bus

Für die Nutzung von Telematik und Multimediaanwendungen wird heutzutage in den meisten Fällen das „Media Oriented Systems Transport“-Bussystem (MOST Cooperation) genutzt. Es wurde speziell für die schnelle Datenübertragung, die hohen Anforderungen dieser Bereiche und die einfache Konfigurierbarkeit entwickelt. Aufgrund der überwiegenden Nutzung als Ringtopologie (2.1.3) lässt sich der Verkabelungsaufwand und somit das Gewicht im Automobil deutlich reduzieren. Derzeitig gibt es verschiedene Ausführungen, die Übertragungsraten von 24 Mbit/s bis zu 150 Mbit/s zulassen. Durch die Plug & Play Unterstützung von bis zu 64 Geräten ist die Skalierbarkeit und einfache Konfiguration gegeben (Dohmke 2002). Die

Standardisierung deckt die sieben Layer des OSI-Modells ab und stellt dem Anwender zahlreiche APIs zur Verfügung. Die Synchronisierung der Systeme wird über ein MOST-Gerät, das als Timing-Master kontinuierlich MOST-Frames in den Ring sendet, angestoßen. Über diese ist den anderen Systemen eine Nachsynchronisation ihrer Uhren möglich. Die Weiterentwicklung der Spezifikation wird von der MOST-Cooperation getrieben, zu der die meisten Fahrzeughersteller und Zulieferer gehören (vgl. Marscholik / Subke 2007)[S. 69] und ist frei zugänglich.

FlexRay

FlexRay (FlexRay Consortium) wurde als Bussystem für zeitkritische Datenkommunikation, die beispielsweise bei X-By-Wire-Systemen benötigt wird, entwickelt (vgl. Marscholik / Subke 2007)[S. 60]. Es ermöglicht neben der asynchronen auch synchrone Kommunikation über das TDMA-Verfahren. Dabei werden Nachrichten zu festgelegten Zeitpunkten innerhalb eines Kommunikationszyklus gesendet. Jeder Zyklus besteht aus einem statischen und einem dynamischen Segment, wobei Echtzeiteigenschaften nur im statischen Segment gegeben sind. Dieses wird in kleinere Slots mit fester Länge und unterschiedlichen IDs eingeteilt. In jedem Slot kann nur das Frame mit entsprechender ID gesendet werden. Zur Kommunikation sind synchrone Uhren in den einzelnen Systemen notwendig. Diese können durch unterschiedliche Taktgeneratoren voneinander abweichen und müssen über eine Uhrensynchronisation wieder ausgeglichen werden. Da es bei FlexRay keinen Timing-Master mit Referenzuhr (vgl. Marscholik / Subke 2007)[S. 68] gibt, wird die Synchronisation anhand von SYNC-Frames durchgeführt, die von mehreren Systemen im statischen Segment geschickt werden.

Durch die umfangreichen Konfigurierungsmöglichkeiten bei FlexRay werden die protokollspezifischen Eigenschaften gemäß Streichert / Traub (Streichert / Traub 2012) nachfolgend aufgezählt:

- **Name:** Name des FlexRay-Busses
- **Baudrate:** Übertragungsgeschwindigkeit (meist in MBit/s)
- **Version:** Durch die unterschiedlichen Versionen des Protokolls wird sichergestellt, dass nur passende Communication-Controller am FlexRay-Bus angebunden werden.
- **Zyklusdauer:** Länge eines FlexRay-Zyklus
- **Macrotick:** Dauer eines Macroticks
- **Statische Slots:** Anzahl an statischen Slots
- **Minislots:** Anzahl an Minislots im dynamischen Segment
- **Network Idle Time:** Länge der Network Idle Time

Für die Nachrichten, die über den Bus übertragen werden, gelten folgende Eigenschaften:

- **Name:** Name der Nachricht
- **Slot:** Nummer, in welchem Slot eine Nachricht versendet wird.
- **Cycle Repetition:** Beschreibt, ob eine Nachricht in jedem oder nur in jedem 2^n-fachen Zyklus versendet wird.
- **InCycle Repetition:** Gibt an, ob eine Nachricht mehrmals innerhalb eines Zyklus versendet werden soll.

Ethernet

Ethernet (Institute of Electrical and Electronics Engineers 2005) wird derzeitig im Automobil noch nicht großflächig eingesetzt. Durch die verfügbare, skalierbare Bandbreite und die Flexibilität hat es sich in vielzähligen Einsatzarten, wie in der Automatisierungstechnik, in der Bürokommunikation und in der Luftfahrt bereits etabliert. Ethernet beschreibt dabei eine Familie von Netzwerktechniken, die vorwiegend in lokalen Netzwerken eingesetzt wird. Dazu gehört die Definition der Adressierung und der Zugriffskontrolle für das Übertragungsmedium der einzelnen Pakete auf den ersten beiden Schichten des OSI-Schichtenmodells. Während zu Beginn das Netzwerk als Bus aufgebaut wurde, das mehrere Endgeräte verband, ist heute eine Sterntopologie mit einem zentralen Punkt als Verteilerstation für die einzelnen Pakete zuständig. Diese wird meistens im Fullduplexmodus mit separaten Kanälen für Senden und Empfangen betrieben und stellt Punkt-zu-Punkt-Verbindungen dar. Daher spielt die für die Kollisionserkennung notwendige „Carrier Sense Multiple Access with Collision Detection„ (CSMA/CD) Kollisionsauflösung keine entscheidende Rolle mehr, ist allerdings im Frame-Format immer noch gebräuchlich. Durch die Möglichkeit den Power over Ethernet Standard zu verwenden, das ein Verfahren beschreibt, wie sich ethernetfähige Geräte mit Strom versorgen lassen, ist es für den Einsatz im Fahrzeug prädestiniert.

2.1.2 Echtzeitfähiges Ethernet als Backbone in Automobilnetzwerken

Aufgrund der zusätzlichen Anforderungen, die künftig auf das Automobilnetz zukommen, ist es notwendig, eine heterogene Sicht auf das Netzwerk zu legen, bei dem die Kommunikation domänenübergreifend und in Echtzeit stattfinden kann. Ein Backbone bezeichnet dabei den Bereich, über den die meisten Daten transportiert werden, und der somit die höchste Bandbreite zur Verfügung stellen muss. Von den in Kapitel 2.1.1 genannten Kommunikationssystemen bietet Ethernet die besten Grundvorraussetzungen dafür.

Der heutige Standard für drahtgebundene Vernetzungstechnik ist Standard-Ethernet. “Jedes netzwerkfähige Produkt - vom einfachsten 10-Euro-Switch bis zum digitalen Videorecorder -

besitzt heute einen Fast-Ethernet-Anschluss für 100 Mbit/s" (vgl. Joerg Rech 2007). Wegen der steigenden Anzahl von Fahrerassistenzsystemen im Automobil wird auch in diesem Bereich eine größere Bandbreite gewünscht. Die bisher gezeigten Bussysteme können in ihren Domänen die benötigten Anforderungen abdecken. Allerdings wird es durch die domänenübergreifende Kommunikation und die zusätzlichen Bedürfnisse immer schwieriger, ohne eine leistungsstarke und kostengünstige Backbonearchitektur den zukünftigen Anforderungen zu genügen. Ethernet bietet als bereits erwähnter Standard für drahtgebundene Vernetzungstechnik die benötigten Eigenschaften wie hohe Bandbreite, Skalierbarkeit, Robustheit, Kostenvorteile und Verbreitung (Hank / Suermann / Müller 2012).

Die Eigenschaft der Echtzeitfähigkeit besitzt Ethernet in seiner ursprünglichen Form nicht. Das heißt, dass Nachrichten, die nur eine gewisse Latenz aufweisen dürfen, durch Nachrichten von anderen Geräten in den Switches verzögerbar sind. Daher können Systeme, die über ein Ethernet kommunizieren, keinen Echtzeitanforderungen genügen. Echtzeitanforderungen sind in verschiedene Typen untergliedert und werden nachfolgend beschrieben (Mall 2009):

Echtzeiteigenschaften in zeitkritischen Systemen

Zeitkritische Systeme unterliegen Anforderungen, die für einen reibungslosen Ablauf garantieren. Für jeden Prozess des Systems wird eine absolute Deadline (auch Deadline genannt) gesetzt, die den Zeitpunkt, an dem er spätestens abgeschlossen sein muss, definiert. Die relative Deadline hingegen beschreibt die Zeitspanne zwischen dem Zeitpunkt der Aktivierung des Tasks und der absoluten Deadline (vgl. Reif 2009). Hinsichtlich der Anforderung wird in drei verschiedene Arten der Echtzeit unterschieden. Diese lauten wie folgt:

Weiche Echtzeitanforderungen: Bei weichen Echtzeitanforderungen werden vorgegebene Reaktionszeiten erwartet, unterliegen jedoch keinen Deadlinegarantien, sodass ein Überschreiten der Zeitanforderung nicht zu Fehlern führt und das Ergebnis unter Umständen noch genutzt werden kann.

Feste Echtzeitanforderungen: Feste Echtzeitanforderungen belegen das System mit festen Zeitschranken. Bei Nichteinhalten der gesetzten Deadline wird das System in einen Fehlerzustand überführt, das einen Arbeitsabbruch des Systems impliziert.

Harte Echtzeitanforderungen: Sind harte Echtzeitanforderungen nötig, muss das System garantieren, dass definierte Reaktionszeiten niemals überschritten werden. Die Verletzung der Zeitschranken ist dabei nicht tolerierbar und kann zu Fehlern im System bis hin zum Schaden für System, Mensch und Umwelt führen.

Damit Ethernet die Determinismuseigenschaft verliehen werden kann und somit harte Echtzeitanforderungen erreicht werden, gibt es für die Implementierung von Echtzeit-Ethernet drei alternative Ansätze innerhalb des OSI-Modells (Jasperneite 2005):

Die erste Klasse beschreibt die Modifikation oberhalb der TCP/IP-Ebene auf Layer 5 - 7 und nutzt die Standardimplementierung der unteren vier Layer. Durch den TCP/IP Protokollstack können allerdings Verzögerungen von $> 100ms$ entstehen, die sich nur mit weichen Echtzeiteigenschaften vereinbaren lassen. Diese Lösung wird in Protokollen wie PROFINET V1 und Ethernet/IP genutzt.

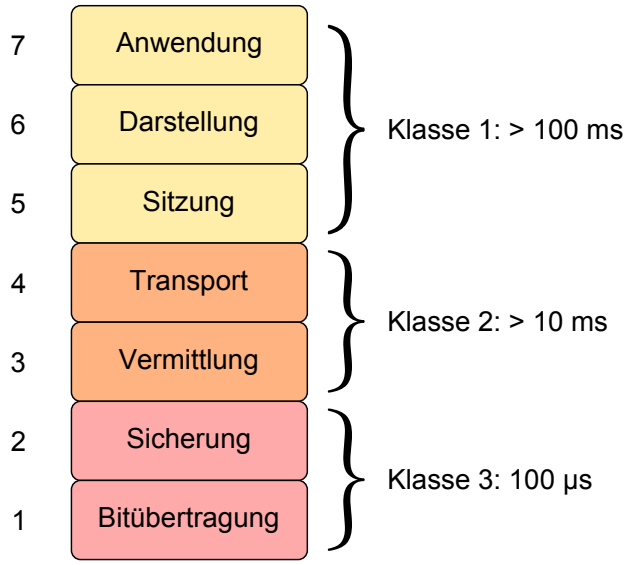

Abbildung 2.2: *Alternative Ansätze zur Implementierung von Echtzeit-Ethernet im OSI-Modell (Jasperneite 2005)*

.

Die zweite Klasse beschreibt eine Modifikation oberhalb der Datensicherungsschicht (über OSI-Layer 2). Wegen der Reduzierung des Protokollstacks ist ein Ende-zu-Ende Delay von $10ms$ zu erreichen.

Die dritte und letzte Klasse beschreibt eine Anpassung der Ethernetschicht selbst (Osi-Layer 1-2). Aufgrund der Einführung von Scheduling-Algorithmen auf Hardwareebene ist es möglich, Latenzen von $100\mu s$ und damit den benötigten Determinismus für harte Echtzeitanforderungen zu erreichen. Viele Implementierungen schränken Standardethernet dabei sehr ein. Mit Time-Triggered Ethernet existiert eine Lösung der Klasse drei, die vollständige Kompatibilität zum Standardethernet liefert.

Time-Triggered Ethernet

Time-Triggered Ethernet (TTE) wurde ursprünglich in der Real-Time Systems Group (vgl. Real Time Systems Group (RTS)) an der TU Wien im Jahr 2004 entwickelt. Nach einer Ausgliederung in die TTTech Computertechnik AG wird es in erweiterter Form kommerziell in der Automobil- und Flugzeugindustrie angeboten. Der Grundgedanke von TTE ist, unterschiedliche Kommunikationsarten gemeinsam über den gleichen Kanal zu übertragen. Das bedeutet, dass eine Übertragung von Nachrichten mit unterschiedlicher Priorisierung stattfindet. Dabei soll es in erster Linie möglich sein, eine zeitgesteuerte Kommunikation zu erlauben, wodurch das Verschicken von Nachrichten über einen festen Scheduler geschieht. Dies ist für ein deterministisches Netzwerk notwendig, bei dem keine Nachrichten auf unkontrollierbare Weise verzögert werden dürfen. Zweitens soll es möglich sein, genügend Bandbreite für bandbreitenintensive Anwendungen zur Verfügung zu stellen. Dieses spielt vor allem für zeitkritische Videostreams aus dem Bereich der weichen Echtzeitanforderung eine Rolle, die eine fehlerhafte Übertragung tolerieren, jedoch einer gewissen Bandbreite genügen müssen. Als letztes soll es möglich sein, wie im normalen Ethernetbetrieb jederzeit das Netzwerk frei zu skalieren und für nicht registrierte Anwendungen zu öffnen (vgl. Steiner 2008). Diese dürfen die deterministischen Nachrichten jedoch nicht beeinflussen.

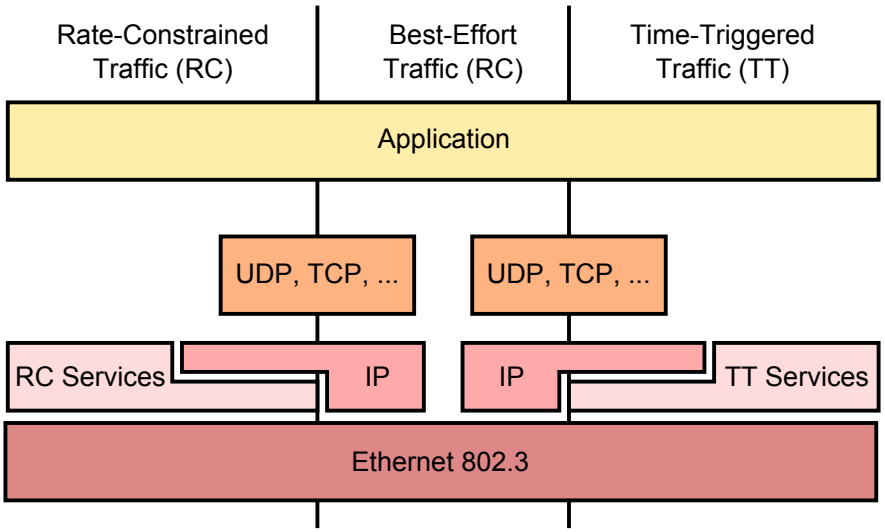

Abbildung 2.3: *Der eingeführte Servicelayer mit RC- und TT-Services fügt dem Ethernetprotokoll die benötigten Echtzeiterweiterungen für ein Backbone im Automotive Kontext zu (Steiner / Bauer / Hall u. a. 2009).*

Abbildung 2.3 zeigt den durch TTE definierten Servicelayer. Dieser baut direkt auf der Ethernetschicht 802.3 auf und erlaubt somit zeitgesteuerte Kommunikation oberhalb der unteren

beiden Layer (OSI-Layer 1-2). Dies bedeutet, dass Nachrichten aus höher gelegenen Protokollen –wie beispielsweise aus der Transportschicht– problemlos zeitgesteuert übertragen werden können. Nachfolgend wird die Verbindung über Virtual Links bei Time-Triggered Ethernet und das damit verbundene, gegenüber Standard-Ethernet veränderte, Frameformat beschrieben.

Virtual Links bei Time-Triggered Ethernet: Bei Ethernetverkehr wird anhand der Ziel-MAC-Address entschieden, an welchem Port ein Switch die Nachricht weiterzuleiten hat. Bei Time-Triggered Ethernet geschieht dieses mit Hilfe von Virtual Links (VL). VLs sind logische, unidirektionale, statische Verbindungen von einem Endsystem zu einer beliebigen Anzahl an Endsystemen auf verschiedenen Übertragungsmedien (Bob Pickles 2006). Somit wird eine 1-zu-1 oder 1-zu-n Beziehung dargestellt. Über einen sogenannten *Critical-Traffic Identifier* (CT-ID) wird die Zugehörigkeit eines Frames zu einem Virtual Link hergestellt. Die CT-ID ist Teil der Zieladresse, die bei einem TTE-Frame als *Critical-Traffic* bezeichnet wird. Abbildung 2.4 zeigt das Format bei zwei Traffic-Klassen von TTE.

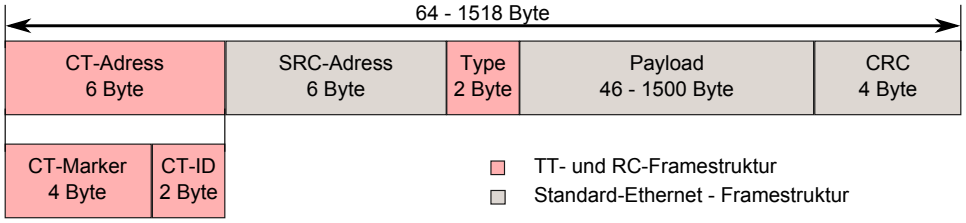

Abbildung 2.4: *Format eines TT- und RC-Frames bei TTEthernet (vgl. Bartols 2010)*

Jedes Endgerät und jeder Switch muss eine statische Konfiguration besitzen, in der steht, welche Virtual Links inwiefern unterstützt werden. Somit wird ein Pfad festgelegt, der bestimmt, auf welchen Ports ein Paket mit einer bestimmten CT-ID eintreffen darf und von welchen Ports es weitergeleitet wird.

Nachrichtenklassen: Time-Triggered Ethernet beschreibt drei verschiedene Nachrichtenklassen, die wie bereits erwähnt, für unterschiedliche Anforderungen genutzt werden. Die drei Klassen lauten: Time-triggered, rate-constrained und best-effort.

Time-Triggered-Traffic (TT): Time-Triggered (TU Wien 1997) ist die Nachrichtenklasse mit der höchsten Priorität und kann von Anwendungen mit hohen zeitkritischen Anforderungen genutzt werden. Die Klasse beruht auf dem „Time Division Multiplex Access"-Verfahren, das die Übertragung von Daten eines Senders nur in bestimmten Zeitabschnitten zulässt. Diese wiederholen sich zyklisch, so dass periodisch Zeitslots freigehalten werden. Durch diese Ordnung, die vor dem Einsatz des Systems statisch

konfiguriert werden muss, ist es möglich, Daten deterministisch zu übertragen und die Verzögerung von Nachrichten zu garantieren. Damit das System reibungslos funktioniert, werden synchronisierte interne Uhren auf den teilnehmenden Systemen benötigt. Durch den Einsatz von sogenannten *Protocol Control Frames* (PCF) ist eine Genauigkeit von unter einer Mikrosekunde erreichbar (vgl. Steiner 2008). Aufgrund der konstanten Latenz und des deterministischen Verhaltens ist ein Einsatz in zeitkritischen Systemen möglich.

Rate-Constrained-Traffic (RC): Die Rate-Constrained-Traffic-Klasse basiert auf dem ARINC-Standard 664, der auch Avionics Full Duplex Switched Ethernet (AFDX) (vgl. Aeronautical Radio Incorporated 2009) genannt wird. Durch den Einsatz von Virtual Links und der statischen Konfiguration ergänzt AFDX den Ethernet-Standard um eine deterministische Wegführung. Nach TT-Nachrichten besitzen Nachrichten der RC-Klasse die nächsthöhere Priorität und können gesendet werden, sobald kein Time-Triggered-Zeitfenster den Übertragunskanal blockiert. Zudem fügt das Protokoll eine Datenratenkontrolle durch sogenannte *Bandwidth Allocation Gaps* (BAG) hinzu. Jeder RC-Sender im Netzwerk besitzt eine Traffic-Shaping-Funktion, die nur eine gewisse Senderate erlaubt. Auf der Empfängerseite muss diese Rate, die hier ebenfalls als BAG angegeben wird, auch eingehalten werden. Bei Nichteinhaltung entfernt das System die Nachricht. Innerhalb der RC-Nachrichtenklasse ist es außerdem möglich, Prioritäten zwischen den Virtual Links zu vergeben. Durch das bandbreitenreservierende Verhalten und wegen der definierten Schranken für die Verzögerung von Nachrichten sind RC-Nachrichten eventbasierter Verkehr und eignen sich insbesondere für zufällig auftretende Verkehrsflüsse und bandbreitenintensive Anwendungen wie Videostreams.

Best-Effort-Traffic (BE): Unter Best-Effort-Traffic werden bei TTE die Standard-Ethernet Nachrichten verstanden. Sie stellen die am geringsten priorisierten Nachrichten dar, und die Übertragung findet nur statt, falls keine RC- und TT-Nachrichten zum Versenden anstehen. Zudem existieren keinerlei Qualitätszusicherungen, wodurch die Verzögerung der Pakete nicht gewährleistet werden kann, und ob die Pakete überhaupt beim Empfänger eintreffen. Der große Vorteil besteht in der einfachen Skalierung. Endgeräten, die keine zeitkritischen Anforderungen besitzen und nur temporär im Netzwerk teilnehmen, wird ohne statische Konfiguration eine Kommunikation mit anderen Netzteilnehmern ermöglicht. Dieses Verhalten ist bei den anderen beiden Verkehrsarten nicht gegeben.

Konfigurationsbeispiel: Abbildung 2.5 zeigt ein Time-Triggered Ethernet Netzwerk mit drei Electronic Control Units (ECU) und zwei Switches. Jeder Sender erzeugt Nachrichten mit einer Nachrichtenklasse, so dass im Switch alle drei Nachrichtenklassen gescheduled werden müssen. Die TT-Nachricht wird zyklisch von der dritten ECU versendet, und das Zeitfenster befindet sich am Anfang des Zyklus. Der Abstand bleibt im Switch vorhanden. Die ECU mit dem RC-Verkehr sendet zeitgleich zu der TT-Nachricht ebenfalls eine Nachricht.

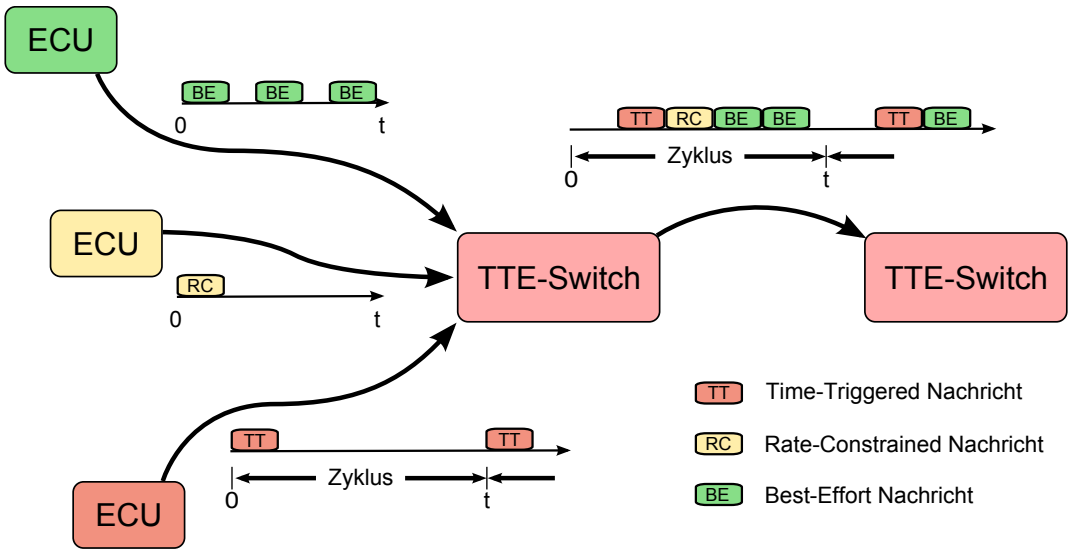

Abbildung 2.5: *Die drei Nachrichtenklassen bei Time-Triggered Ethernet (vgl. Steinbach 2011). Jede unterstützt unterschiedliche Anforderungen.*

Zudem sendet die ECU mit dem Best-Effort-Verkehr drei Nachrichten innerhalb des TT-Zyklus. Beim Switch sieht der Schedule wie folgt aus: Für die TT-Nachricht sind feste Zeitfenster definiert, so dass diese direkt weitergeleitet werden kann. Als nächstes wird die RC-Nachricht übertragen, bis anschließend die Weiterleitung der am niedrigsten priorisierten BE-Nachrichten zum zweiten Switch ansteht. Die letzte Best-Effort-Nachricht passt zudem nicht zwischen die zweite Best-Effort-Nachricht und den Time-Triggered-Zeitslot, so dass sie erst im nächsten Zyklus Platz findet.

TTEthernet-Toolchain: Bei Time-Triggered Ethernet müssen die Geräte des Netzwerkes vor der Nutzung konfiguriert werden. Zur Unterstützung des Entwicklers stellt TTTech (TTTech Computertechnik AG) vier verschiedene Design- und Konfigurationstools bereit. Alle Entwicklertools basieren auf offenen XML Datenformaten, damit eine Kompatibilität zu externen Tools und Umgebungen gewährleistet ist. Abbildung 2.6 zeigt die Time-Triggered Ethernet Toolchain.

$^{TTE}Plan$**;** Ist ein Netzwerkplanungstool, mit dem aus abstrakten Kommunikationsanforderungen wie Nachrichtenflüsse, Topologien und Timings, die Schedules und Konfigurationsdateien generiert werden können. Als Anforderungen werden dabei beispielsweise die Virtual Links, die Latenzanforderungen und die möglichen Framegrößen in einem „Network Description“-XML beschrieben, aus dem das Tool anschließend die

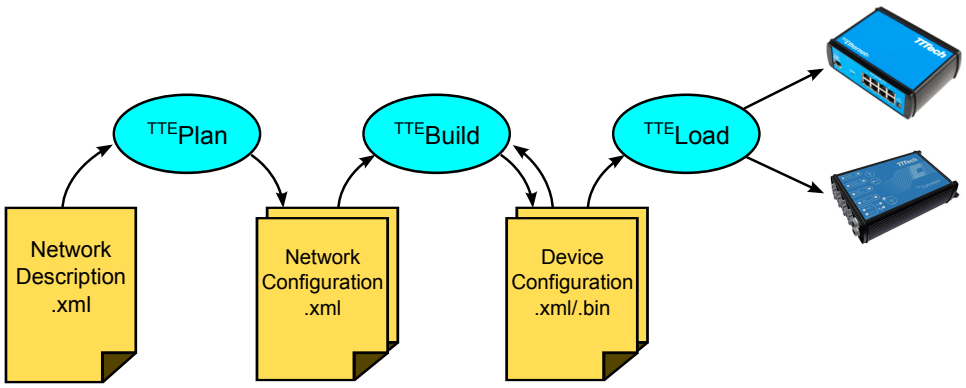

Abbildung 2.6: *Time-Triggered Ethernet Toolchain (vgl. TTTech Computertechnik AG)*

benötigten Time-Triggered-Traffic Schedules berechnet, den Rate-Constrained Traffic konfiguriert und anschließend eine „Network-Configuration“ erzeugt.

$^{TTE}Build$**:** Aus der generierten „Network-Configuration“ liest $^{TTE}Build$ die Topologie und gescheduledte Nachrichtenflüsse für jedes Gerät. Dazu kommen zusätzliche Konfigurationsparameter, die individuell für jedes Gerät festgelegt werden müssen. Anschließend wird für jedes Gerät eine XML Repräsentation gebildet, die jeden Parameter darstellt und anpassbar macht. Um anschließend die Funktionalität auf die Switches und Endsysteme zu laden, können die „Device-Configuration“ XML von $^{TTE}Build$ erneut eingelesen und als Binaries ausgegeben werden.

$^{TTE}Load$**:** Das letzte Tool für den Konfigurationsvorgang bildet $^{TTE}Load$. Mit dessen Hilfe ist es möglich, die erzeugten Binaries auf die Switches und Endsysteme zu laden.

$^{TTE}View$**:** Zur Analyse für Time-Triggered Ethernet Netzwerke stellt TTTech das Tool $^{TTE}View$ für Entwickler zur Verfügung. Es ermöglicht den Netzwerktraffic zu überwachen, aufzunehmen und zu zerlegen. Als Plugin für den Open-Source Ethernet Protokoll Analyzer Wireshark (Combs) bietet es eine Unterstützung für eine etablierte Software zur Analyse von Ethernetnetzwerken. Die Bereitstellung von Definitionen des TTEthernetprotokolls dient zur visuellen Aufbereitung, damit der aufgezeichnete Traffic einfach und offline analysiert werden kann.

2.1.3 Netzwerktopologien

Die Vernetzungsstruktur mehrerer Kommunikationspartner wird als Topologie bezeichnet. Neben der Punkt-zu-Punkt Verbindung als einfachste Form der Topologie haben sich verschiedene Varianten durchgesetzt, um Daten gleichzeitig an mehrere Empfänger übertragen zu können. Die Darstellung erfolgt dabei als Graph, wodurch eine gute grafische Darstellung möglich ist. Als Überblick werden einige mögliche Topologien in Abbildung 2.7 dargestellt und anschließend erläutert:

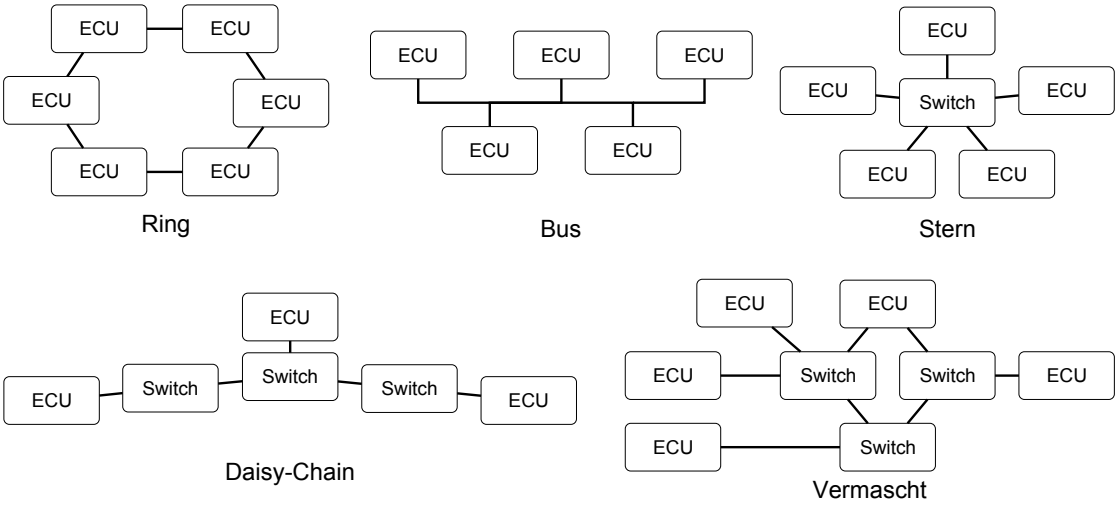

Abbildung 2.7: *Verschiedene Netzwerktopologien (Marscholik / Subke 2011), (Tanenbaum 2003)*

Ringtopologie: Die Ringtopologie stellt eine Erweiterung von Punkt-zu-Punkt-Verbindungen dar. Jeder Knoten wird mit zwei weiteren Knoten verbunden, wodurch eine direkte Kommunikation mit diesen möglich ist. Durch die Aneinanderreihung mehrerer Knoten, bis ein in sich geschlossener Kreis entsteht, kann die zu übertragende Information durch einmaliges Versenden jeden Knoten im Netzwerk erreichen. Allerdings muss jeder Knoten Informationen, die er gar nicht benötigt, sowohl empfangen als auch weiterleiten, wodurch eine erhöhte Datenübertragung stattfindet. Zudem entsteht in jedem Gerät eine zusätzliche Verzögerung, wodurch die Latenz erheblich ansteigt.

Bustopologie: Im Gegensatz zur Ringtopologie wird bei einer Bustopologie jeder Knoten mit dem gleichen Übertragungsmedium, nämlich dem Bus, verbunden. Auf diesem ist es jedem Knoten möglich, Daten zu senden und zu empfangen. Da es nur einem Knoten zur Zeit gestattet ist, seine Information zu übertragen, muss der Buszugriff zwischen

den Geräten geregelt sein. Der große Vorteil besteht in dem geringen Verkabelungsaufwand und in der bestehenden Sicherheit bezüglich des Ausfalls eines Gerätes.

Sterntopologie: Die Vorteile der Verkabelung und der Ausfallsicherheit von Endgeräten besitzen Teilnehmer einer Sterntopologie ebenfalls. Bei dieser Variante werden die Geräte mit einem zentralen Verteiler verbunden. Dieser arbeitet entweder so, dass alle Daten an alle Empfänger weitergeleitet werden oder die Daten zwischen seinen Links routet, so dass nur der gewünschte Empfänger die Daten erhält. Bei der genannten Variante, die als Switch bezeichnet wird, ist es erstmals möglich, eine höhere Übertragungsrate für die einzelnen Endgeräten zu gewährleisten, weil nur angeforderte Nachrichten auch tatsächlich zu dem Empfänger übertragen werden. Über mehrere Verteiler, die mit den gleichen Systemen verbunden sind, ist eine Ausfallsicherheit der zentralen Punkte möglich.

Daisy-Chain-Topologie: Werden mehrere Verteiler in Reihe geschaltet, um beispielsweise größere Entfernungen zurückzulegen und die zu übertragende Information auf den Switches aufzuteilen, wird von eine Daisy-Chain-Topologie gesprochen. Durch den Ausfall eines Verteilers sind Teile der Kommunikation unterbrochen, jedoch können Daten noch in den nun eigenständigen Subnetzen übertragen werden.

Vermaschte-Topologie: Vermaschte Topologien bestehen aus mehreren Topologievarianten und bieten ein Maß an Ausfallsicherheit durch redundante Verbindungen. Wegen einer dezentralen Aufteilung besitzt sie eine gute Verteilung der Last. Das ist für sicherheitskritische Umgebungen notwendig. Wird jeder Teilnehmer mit einem anderen Teilnehmer verbunden, bezeichnet dies ein vollständig vermaschtes Netzwerk, das überwiegend in der drahtlosen Netzwerkkommunikation wiederzufinden ist.

2.2 Simulation heterogener Fahrzeugnetzwerke

Simulationen dienen zur Verständnisbildung von Systemen. Aufgrund des Einsatzes schneller Rechner ist es möglich, das Verhalten komplexer Systeme frühzeitig abzubilden und somit wichtige Aussagen über eine spätere Prototypentwicklung zu treffen. Das *System* beschreibt dabei eine Menge von Entitäten beziehungsweise Objekten, die in der Realität existieren. Dieses System wird in der Simulation durch ein Modell dargestellt und als *Abbild* bezeichnet (vgl. Law / Kelton 1991). Um das System in ein Modell zu überführen, bedarf es einer genauen Systemanalyse, in der die Struktur und die Komponenten hinreichend spezifiziert werden, so dass relevante Aspekte für die Untersuchung modelliert sind. Simulationen spielen im Entwicklungsprozess neuer Techniken eine entscheidende Rolle, weil frühzeitig Annahmen ohne Kosten durch die Anschaffung neuer Hardware getroffen werden können und Prototypen nicht verfügbar sein müssen. Zudem kann wegen der Unterscheidung zwischen der Systemzeit innerhalb des Modells (Modellzeit) und der realen Simulationsdauer (Rechenzeit) eine von der

Echtzeit unabhängige Entwicklung erfolgen. Im Idealfall ist dabei in einer Sekunde Rechenzeit mehr als eine Sekunde Modellzeit berechenbar.

2.2.1 Simulationsmodellierung

Die Simulation umfasst die drei wichtigen Bereiche des Modellentwurfes, der Implementierung und der Systemanalyse, wodurch genaue Kenntnis des zu simulierenden Systems gewonnen wird (Page / Liebert / Heymann u. a. 1991). Der Entwurf eines Simulationsmodells kann unterschiedliche Ziele betreffen. Entweder kann es dazu dienen, bekannte Szenarien nachzuvollziehen, diese zu optimieren oder zukünftige Prognosen über unbekannte Szenarien zu treffen (vgl. Bungartz 2013). In jedem Fall kann das Modell als Abbild die Realität nicht 1:1 nachbilden, weil die Komplexität nicht handhabbar ist. Somit müssen verschiedene Stufen gefunden werden, die es ermöglichen, die wesentliche Aspekte abzubilden. Daher gilt der Satz: *„Modellbildung bedeutet immer Vereinfachung, Zusammenfassung, Weglassen, Abstraktion. Modellbildung ist daher prinzipiell nicht möglich ohne Auswahl und Entscheidungsvorgänge.“* (Bossel 1994)[S. 36]. Um ein korrektes Modell zu entwerfen, kann der komplexe Prozess der Simulationsbildung nicht als integraler Akt (lt. Bungartz 2013) gesehen werden, weil er aus mehreren Schritten besteht. Diese Schritte werden iterativ durchlaufen und geben einen Überblick über die unterschiedlichen Aufgaben, die es bei dem Entwurf und der Abstraktion einer Simulation zu beachten gilt.

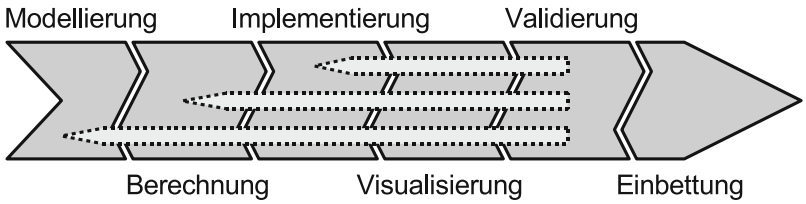

Abbildung 2.8: *Simulationspipeline. (Bungartz 2013)[S. 03]*

Abbildung 2.8 zeigt die „Simulationspipeline“ nach Bungartz (Bungartz 2013), die im Wesentlichen sechs Schritte in den folgenden Bereichen abdeckt:

- *Modellierung:* Eine vereinfachte Beschreibung eines wirklichen Systems, das als Grundlage für die Berechnung beziehungsweise Simulationsmodellierung dienen kann. Diese Beschreibung wird als Modell bezeichnet.
- *Berechnungs- / Simulationsmodellierung:* Aufbereitung des Modells in effiziente Algorithmen. Dazu gehört die Ausarbeitung einer geeigneten Zielarchitektur, die mit unterschiedlichen Komponenten das Verhalten abstrakt beschreibt.

- *Implementierung:* Die entwickelten Algorithmen müssen auf der Zielarchitektur effizient implementiert werden. Dabei bezieht sich die Effizienz einer Simulation in erster Linie auf das Laufzeitverhalten, die Speicherkomplexität und die Parallelisierbarkeit. Zudem ist das Ziel, eine geeignete Architektur nach Softwareentwicklungskriterien zu entwickeln. Mithilfe der Verifizierung kann eine Überprüfung der Implementierung zur Spezifikation stattfinden.
- *Visualisierung/Datenexploration:* Die aufgezeichneten Kennzahlen eines Simulationsdurchlaufes müssen interpretiert werden. Dabei ist es wichtig, die relevantesten Informationen erkennbar zu machen, damit diese Rückschlüsse auf das Verhalten des Modells zulassen.
- *Validierung:* Bei der Validierung werden die Ergebnisse unterschiedlicher Modelle miteinander verglichen. Aufgrund von Abweichungen zwischen den gleichen aufgezeichneten Werten der verschiedenen Modelle kann das Verhalten der einzelnen Modelle, der aufbereiteten Algorithmen beziehungsweise der Implementierung analysiert und validiert werden.
- *Einbettung:* Die Einbettung dient zur Integration einer Simulation in einen Entwicklungs- oder Produktionsprozess. Dabei ist ein hohes Maß an Automatisierung erforderlich, um zuverlässig und schnell unterschiedliche Simulationsdurchläufe zu starten.

Die ersten beiden Aspekte beschreiben die direkte Modellentwicklung und spiegeln die wichtigsten Punkte wider. Häufig wird jedoch die Implementierung als Modellierung genommen, wodurch der Modellierungsaspekt zu kurz gefasst wird. Über die Visualisierungs- und Validierungsphase wird die Auswertung des Modells getrieben und spielt gerade im simulativen Bereich eine entscheidende Rolle. Der Einbettungsprozess wird bei der Simulationsentwicklung häufig nicht hinreichend beachtet. Dabei wird durch ihn die Nutzbarkeit des Modells für Experten der Domäne geöffnet.

2.2.2 Verifikation und Validierung von Simulationsmodellen

Damit Simulationsmodelle einem gewissen Grad der Wirklichkeit entsprechen, ist es notwendig, die Ergebnisse zu bewerten. Dazu muss die Betrachtung sowohl auf den einzelnen Algorithmen des Modells als auch auf der Eignung des Modells im Bezug auf die Aufgabenstellung liegen. Diese beiden Probleme werden als Verifikation und Validierung unterschieden. Die Verifikation kann dabei mit der Frage: „Ist das Modell richtig?“ und die Validierung mit der Frage: „Ist es das richtige Modell?“ verknüpft werden (vgl. Gerberich 2011).

Die *Verifikation* beschreibt, ob sich die Implementierung konform zur Spezifikation des Modells verhält. Die Funktion kann anhand von Funktionstests überprüft werden, die sich an

allgemeinen Softwaretests orientieren. Diese wiederum können als Ergebnisse aus analytischen Modellen hervorgehen. Dieses entspricht einer etablierten Vorgehensweise, wobei die Verifikation der eingesetzten Algorithmen auf Konvergenzbeweisen basiert (vgl. Bungartz 2013).

Ziel der *Validierung* ist es, die Korrektheit des Modells zu prüfen und festzustellen, ob die Anforderungen der Realität entsprechen. Dazu muss bereits eine Umsetzung des Modells erfolgt sein. Anschließend kann auf unterschiedliche Weise das modellierte Systemverhalten auf verschiedene Eingangsgrößen und Parameter mit der Realität verglichen werden. Der klassische Weg setzt sich aus Vergleichen zwischen berechneten Simulationsergebnissen und Ergebnissen, die auf experimentellen Untersuchungen beruhen, zusammen. Durch einen umfangreichen und kostspieligen Aufwand bei der Realisierung von realen Testreihen bietet diese Art der Validierung jedoch häufig keine ausreichende Möglichkeit an. Eine günstigere Alternative existiert in *A-posteriori Beobachtungen*. Hierbei werden simulierte Ergebnisse mit eingetretenen Ergebnissen aus der Realität verglichen. Dieses Vorgehen stellt für Simulationsmodelle von Netzwerkprotokollen jedoch keine Alternative dar. Eine dafür geeignetere Möglichkeit der Validierung existiert, falls die Protokolle auch von anderen Modellen abgebildet werden. Dann können im Idealfall die gleichen Versuchsabläufe hinsichtlich der gesetzten Parameter in den Modellen durchgeführt werden und anschließend ein Gleichnis der Ergebnisse erläutern. Diese Art der Validierung wird *Modellvergleich* genannt (vgl. Bungartz 2013). In jedem Fall ist die Validierung sehr kostspielig, weil theoretisch jeglicher Parameterumfang getestet werden müsste. Abhilfe schafft hier der Test von Grenz- und Standardwerten. Für die Eignung der Modelle lässt sich dann allerdings nicht vollständig nachweisen, ob das Verhalten korrekt ist.

2.2.3 Diskrete, ereignisbasierte Simulation

Bei Simulationen findet die Klassifizierung anhand ihrer Art der Zustandsübergänge statt. Dabei werden *statische* und *dynamische Modelle* unterschieden. Wegen der fehlenden Zustandsübergänge in statischen Simulationsmodellen sind diese Arten der Modelle für die Simulation nicht relevant. Dynamische Modelle hingegen besitzen zeitabhängige Zustandsänderungen (vgl. Strümpel 2003). Bei den dynamischen Modellen wird zwischen *kontinuierlichen* und *diskreten* Modellen unterschieden. Die Zustandsvariablen bei kontinuierlichen Systemen sind mittels Differentialgleichungen modellierbar und lassen sich vor allem in den Bereichen aus der Mechanik und der Elektrotechnik finden. Das Erschließen von valide anerkannten Funktionen ist oftmals schwierig. Im Gegensatz dazu nehmen bei diskreten Modellen die Zustandsvariablen eine Änderung sprunghaft an. Abbildung 2.9 zeigt die Zustände in kontinuierlichen und diskreten Systemen. Bei den diskreten Zeitpunkten der Zustandsänderung handelt es sich um *Ereignisse* beziehungsweise *Events* (vgl. Guérin / Peris 1999).

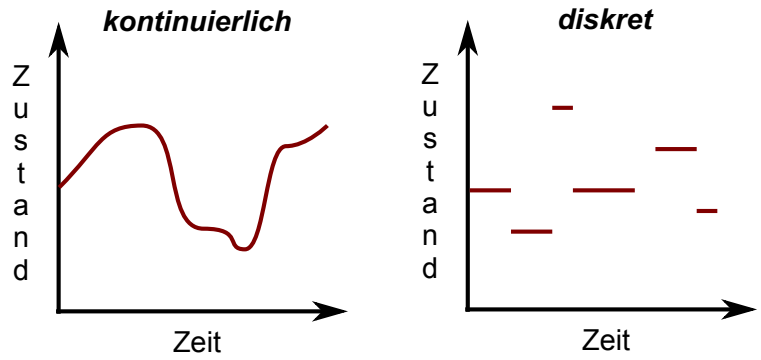

Abbildung 2.9: *Zustandsübergänge in kontinuierlichen und diskreten Systemen*

Diskrete Simulationen lassen sich in zwei Arten einordnen: Bei der zeitorientierten, diskreten Simulation wird der Zustand des Systems anhand der Simulationszeit geändert. Dabei werden mit Hilfe eines Taktes, der ein festes Intervall besitzt, die Zustandsübergänge der Ereignisse ausgelöst. Bei der ereignisorientierten, diskreten Simulation stehen die Ereignisse im Vordergrund, sodass bei dem Eintreten eines Ereignisses die Folgeereignisse und ihre Simulationszeitpunkte generiert werden. Ereignisse können dabei das Anfordern und das Freigeben von Ressourcen sein (vgl. Banks 2010). Abbildung 2.10 zeigt die Einteilung von Simulationsmodellen in die unterschiedlichen Klassen nach Schäfer (Schäfer 2004).

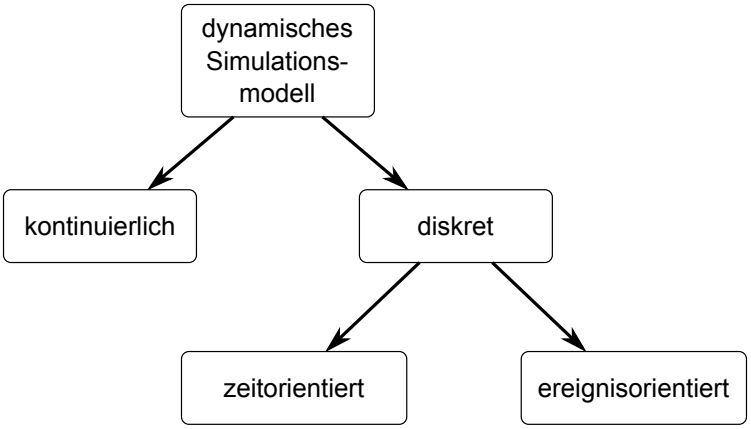

Abbildung 2.10: *„Klassifizierung von Simulationsmodellen nach Zustandsübergängen des zugrunde liegenden konzeptionellen Modells“ nach Schäfer (Schäfer 2004)[S. 46]*

Diskrete, ereignisorientierte Simulation ist durch die eventbasierte Herangehensweise prädestiniert für Transportsysteme, Queueysteme und digitale Netzwerke (vgl. Mouftah / Sturgeon 1990). Daher liegt der weitere Fokus auf dieser Art der Simulation. Im Folgenden wird OMNeT++ als Simulationsumgebung erläutert, die diskrete, ereignisorientierte Simulation unterstützt.

2.2.4 OMNeT++ als Simulationsumgebung

Die *Objective Modular Network Testbed in C++* (OMNeT++ (OMNeT++ Community [b])) ist eine diskrete, ereignisorientierte Simulationsumgebung. Sie wurde für die Modellierung und Simulation von Kommunikationsnetzwerken entwickelt, wird jedoch auch zur Simulation von komplexen, verteilten Systemen wie Hardwarearchitekturen und Warteschlangennetzwerken eingesetzt. Als Eclipse-basierte Entwicklungsumgebung mit einer grafischen Benutzerführung ist sie als Open-Source Projekt für akademische sowie für private Zwecke unter der GNU General Public License (GPL) nutzbar (Varga 2001). Mit OMNEST ist eine kommerziell verwendbare Version verfügbar. Die Programmierung von Modulen findet in C++ statt, ist allerdings auch in anderen Programmiersprachen –wie beispielsweise Java– möglich. Ein Vergleich mit anderen Netzwerksimulatoren ist in Weingartner, Lehn und Wehrle (Weingartner / Lehn / Wehrle 2009) zu finden. Dabei stellt sich heraus, dass neben der Möglichkeit sehr große Netzwerke simulieren zu können, die eigene Modellierungssprache eine bessere Modellierung im Gegensatz zu den anderen Simulatoren ermöglicht.

Die Grundstruktur von OMNeT++ sieht die Unterteilung des Modells in unterschiedliche Module vor. Die aktiven Komponenten werden dabei *Simple Modules* genannt. Jedes Simple Module verfügt über eine Anzahl von *Gates*. Dabei wird meistens zwischen Input- und Outputgates unterschieden. Wenn das Inputgate mit einem Outputgate eines anderen Simple Modules über eine gerichtete *Connection* verbunden wurde, können *Messages* ausgetauscht werden. Ungerichtete Verbindungen werden ebenfalls zugelassen. Die Bezeichnung für Connections mit spezifischen Parametern ist *Channel*. Neben den Simple Modules gibt es sogenannte *Compound Modules*. Diese werden genutzt, um mehrere Simple- und Compound-Modules zu gruppieren. Die Verschachtelungstiefe kann beliebig erweitert werden, sodass die Anzahl der Hierarchieebenen nicht begrenzt ist. Verschachtelte Module können allerdings nur mit Modulen auf der gleichen Ebene verbunden werden. Somit ist es innerhalb eines Compound Modules möglich, Connections nur zwischen zwei Submodules oder zwischen einem Submodule und einem Gate des Compound Modules zu verbinden. Abbildung 2.11 zeigt die mögliche Modellstruktur in OMNeT++. (vgl. Varga / Hornig 2008)

In OMNeT++ wird die Struktur der einzelnen Komponenten über die Network Description (NED) Beschreibungssprache modelliert. Dazu zählen Deklarationen der Simple Modules, der Compound Modules und der Netzwerkdefinitionen. Neben der textuellen Beschreibungssprache ist ein grafischer Editor vorhanden, mit dem es möglich ist, vorhandene Module mit

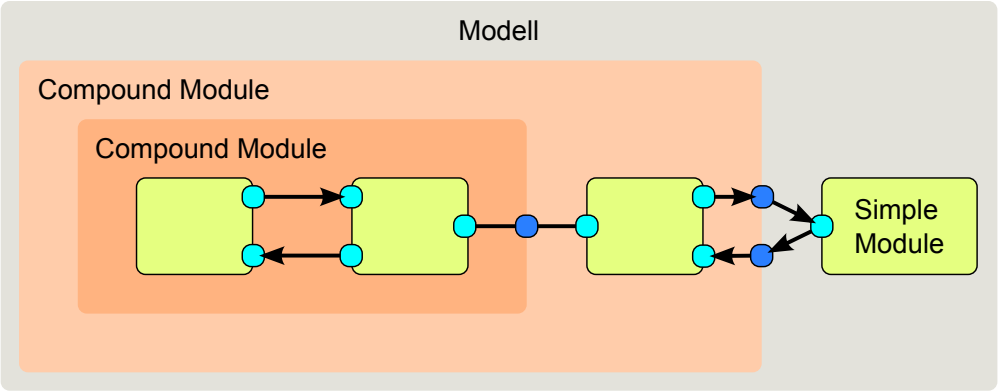

Abbildung 2.11: *Modellstruktur in OMNeT++. Sie besteht aus Simple Modules, die in Compound Modules verschachtelt sein können. Die Verbindungen zwischen den Modulen heißen Connections und werden mit den Gates verbunden.*

anderen Modulen grafisch zu verbinden. Der Wechsel zwischen der grafischen und der textuellen Ansicht ist dabei jederzeit möglich. Damit die Komplexität der Beschreibung nicht mit umfangreichen Modellen skaliert, sind folgende Techniken aus der Objektorientierung in der NED-Sprache vorhanden:

- **Inheritance:** Module und Channel sind von anderen Modulen und Channels ableitbar, wodurch Parameter, Gates, weitere Submodule und Connections hinzugefügt werden können.
- **Interfaces:** Konkrete Module oder Channel können Schnittstellen implementieren, damit gemeinsame Signaturen definiert sind.
- **Packages:** Es existiert eine Packagestruktur, wodurch Modelle aus unterschiedlichen Packages gleich ansprechbar sind.
- **Inner Types:** Innerhalb eines Compount Modules können Channel und Module deklariert werden, wodurch sie dem Namespace verborgen bleiben.
- **Metadata Annotations:** Module, Parameter, Gates und Submodule können mit Annotations versehen werden, wodurch definierte Eigenschaften für andere Tools, die IDE und andere Module erkennbar sind.

Während die Topologie eines Modells in NED-Files steht, wird das Verhalten der einzelnen Komponenten in C++ beschrieben. OMNeT++ stellt dabei benötigte Klassen zur Modellierung mit der *OMNeT++ simulation class library* zur Verfügung. Simple Modules werden mit

der *cSimpleModule* Klasse beschrieben. Durch Vererbung dieser Klasse können Funktionalitäten eigener Module implementiert werden. Der Simulations-Kernel ruft bei Erreichen einer Message in einem Gate eines Simple Modules die *handleMessage*-Methode auf. Anschließend ist eine Verarbeitung der Nachricht möglich.

Spezifische Eigenschaften des Simulationsmodells und Modellparameter werden in OMNeT++ durch Dateien mit der *.ini*-Erweiterung auf textueller Basis beschrieben. Jedes Projekt muss dafür eine *omnetpp.ini* definieren, in der verschiedene Konfigurationen für Szenarien und Simulationsabläufe gesteuert werden.

Um Simulationsresultate aufzuzeichnen und zu analysieren, stellt OMNeT++ verschiedene Möglichkeiten zur Verfügung. Die Aufzeichnung von Kennzahlen erfolgt in der Regel bei der Verarbeitung von Messages und wird in Skalaren oder Vektoren festgehalten. Zur Analyse der Werte kann das in OMNeT++ integrierte Scave Tool dienen, das über input-Files Anforderungen überprüfen kann. Weitere Verarbeitungsmöglichkeiten gibt es über den Export in Datenformate für externe Tools. Somit ist es beispielsweise möglich, mit der freien Programmiersprache R statistische Problemstellungen zu lösen.

2.2.5 Netzwerkprotokollframeworks für OMNeT++

Durch den modularen Aufbau von OMNeT++ wurden zahlreiche Frameworks entwickelt, die einen großen Teil bekannter Protokolle unterstützen. Die Standardprotokolle für den Internetstack (TCP, UDP, IPv4, IPv6, etc.), dem Linklayer (Ethernet, PPP, IEEE 802.11, etc.) und vielen weiteren Protokollen aus dem OSI-Modell sind im INET Framework (OMNeT++ Community [a]) vorhanden. Durch die mögliche Vererbung von Komponenten in OMNeT++ ist eine Erweiterung der Protokolle möglich, so dass vorhandene Elemente um Echtzeit-Eigenschaften und andere Anwendungsbereiche erweiterbar sind. Die Echtzeit-Eigenschaften werden für die Simulation heterogener Fahrzeugnetzwerke benötigt. Mit CoRE4INET (CoRE RG [b]) existiert eine Erweiterung des INET Frameworks, das die Protokollfamilie um echtzeitfähige Protokolle wie TTEthernet (AS6802) und IEEE 802.1 Audio Video Bridging (AVB) ergänzt. Zudem wird mit FiCo4OMNeT an wichtigen Feldbustechnologien für das Fahrzeugnetzwerk wie dem CAN- und dem FlexRay-Bussystem gearbeitet.

2.3 Domänenspezifische Sprachen zur Konfigurationsgenerierung

Domänenspezifische Sprachen (DSL) sind ausdrucksstarke Programmiersprachen, die bestimmte Domänen beschreiben und nur in diesen eingesetzt werden können (vgl. Deursen / Klint / Visser 2000). Durch ein hohes Maß an Abstraktion ist es für Domänenexperten möglich, verständliche Sprachelemente eines bestimmten Anwendungsgebietes zu modellieren, mit denen Applikationen aus der Domäne und deren benötigten Komponenten generiert werden können. Daher finden sie in der modellgetriebenen Softwareentwicklung eine immer größer werdende Bedeutung, weil die DSL mit Sprachkonstrukten aus dem jeweiligen Fachgebiet Spezialisten einen Zugang zur Entwicklung geben kann. Der Sprachumfang einer DSL ist meistens auf ein gestelltes Teilproblem festgelegt und hat somit einen deutlich reduzierten Sprachumfang im Gegensatz zu einer universell einsetzbaren Programmiersprache. Im Abschnitt 2.3.1 werden die Grundlagen domänenspezifischer Sprachen genannt und der Bezug zu der Einbettung in den Simulationsentwicklungsprozess hergestellt. Anschließend wird ein Framework zum Erstellen domänenspezifischer Sprachen erläutert, das eine DSL für die OMNeT++ Simulationsumgebung ermöglicht.

2.3.1 Grundlagen domänenspezifischer Sprachen

Domänenspezifische Sprachen lassen sich in zwei unterschiedliche Formen einteilen. Diese heißen „Interne DSLs“ und „Externe DSLs“. Interne DSLs, auch eingebettete DSLs genannt, basieren auf einer Weiterentwicklung einer vorhandenen Sprache und erweitern diese um Sprachkonstrukte, so dass ein Gefühl einer neuen Sprache entsteht.

„Externe DSLs“ beschreiben neue Sprachen, die von Grund auf neu entwickelt werden. Dazu muss neben einer Syntax, welche die Domäne sinnvoll abgrenzt, ein Parser für die Sprache existieren. Die Syntax beschreibt die Grammatik der Sprache. Dabei muss zwischen der abstrakten und der konkreten Syntax unterschieden werden. Die abstrakte Syntax beschreibt zunächst das Modell der Sprache, das heißt die generische Struktur von Datentypen. Darauf aufbauend können verschiedene konkrete Syntaxen die Grammatik, also die Darstellung der Wörter und Zeichen und somit den Aufbau der Sprache regeln (vgl. Smolka 2008). Diese kann sowohl einer grafischen als auch einer textuellen Darstellung entsprechen. Während der Übersetzungszeit kann mithilfe von Regeln aus der statischen Semantik eine Überprüfung der abstrakten Syntax stattfinden, so dass beispielsweise Parameter gesetzt sein müssen und Referenzen nicht leer sein dürfen. Die dynamische Semantik beschreibt eine Kontrolle zur Laufzeit, wodurch eine Überprüfung von Inhalten gesetzter Werte möglich ist.

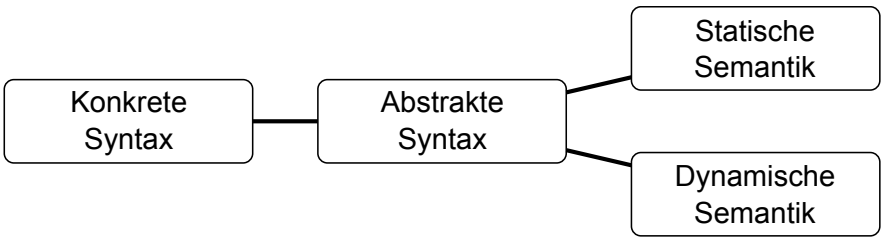

Abbildung 2.12: *Die abstrakte Syntax beschreibt die generische Struktur der Datentypen, während die konkrete Syntax eine dazugehörige Grammatik definiert. Dabei legt die statische Semantik zulässige Ausdrücke für bindungs- und typkonsistente Prozeduren fest, während mit der dynamischen Semantik eine Inhaltsüberprüfung stattfindet. (vgl. Smolka 2008)[S. 241]*

Wie bereits erwähnt, übersetzt der Parser aus der konkreten Syntax in den abstrakten Syntaxbaum, auf dem anschließend Berechnungen und Transformierungen ausgeführt werden können. Das heißt, er übersetzt eine konkrete in eine abstrakte Syntax.

Neben den bisher genannten existieren zwei weitere Aspekte, die für eine DSL relevant sind. Zum einen definieren Korrektheitsregeln, sogenannte *Constraints*, gängige Formalismen zur Beschreibung der abstrakten Syntax. Diese booleschen Ausdrücke geben die Korrektheit des Modells an. Zum anderen traversieren Interpreter den abstrakten Syntaxbaum eines Modells und führen beispielsweise Berechnungen auf diesem durch. Als spezieller Fall kann hier der Codegenerator gesehen werden, der als spezieller Interpreter Quellcode in einer existierenden Programmiersprache ausgibt (vgl. Völter).

Zur Erstellung aussagekräftiger DSLs werden Werkzeuge benötigt, mit denen die verschiedenen Aspekte modelliert werden können. Jedes Werkzeug besitzt dabei unterschiedliche Vor- und Nachteile. Da sich die Simulationsumgebung OMNeT++ in einer Eclipseentwicklungsumgebung befindet und ein durchgehender Entwicklungsprozess angestrebt wird, ist eine Lösung in der selben Entwicklungsumgebung wünschenswert. Im Eclipsebereich hat sich das XText-Framework etabliert, das direkte Erweiterungen in Form von Plugins zur Verfügung stellt. Daher werden im nächsten Kapitel die Möglichkeiten dieses Frameworks erläutert.

2.3.2 Framework zur Entwicklung von Programmiersprachen und DSLs

Zur Entwicklung einer domänenspezifischen Sprache zur Codegenerierung wird an dieser Stelle das Xtext-Framework (Eclipse Public License) beschrieben. Xtext stellt ein professionelles open-source Framework dar, das auf dem Eclipse Modeling Framework (EMF) basiert und zur Entwicklung von Programmiersprachen und domänenspezifischen Sprachen eingesetzt

werden kann (Behrens / Clay / Efftinge u. a.). Dabei stellt es neben einem Parser für die entwickelte Sprache ein Klassenmodell für den typsicheren Abstract Syntax Tree (AST) zur Verfügung. Beide können unabhängig von Eclipse eingesetzt werden. Zusätzlich bietet Xtext eine vollständige Unterstützung der Eclipse-IDE durch einen integrierten Editor für die DSL an. Dadurch sind bekannte Konzepte, wie die Syntaxhervorhebung, Codevervollständigung, Indexierung, Quellcodeformatierung und Rename Refactoring vorhanden, die hohe Qualität mit großer Benutzerfreundlichkeit verbinden.

Zur Implementierung der Sprache stellt Xtext verschiedene Artefakte zur Verfügung. Zur Analyse der Sprache nutzt Xtext einen Scanner, der die Eingabedaten nach den definierten Regeln der entwickelten Sprache in Token zerlegt und für den Parser bereitstellt. Der Parser kümmert sich um die Grammatik der Eingabe und wandelt sie für den AST des Ecore-Modells um. Xtext stellt zudem Klassen zur Verfügung, mit denen Regeln für die Validierung der Sprache, Codeformatierungsregeln für den Editor und Codegeneratoren hinzugefügt werden können.

Erweiterte Backus-Naur-Form

Die konkrete Syntax einer domänenspezifischen Sprache wird in Xtext in der erweiterten Backus-Naur-Form (EBNF 1996) definiert. Diese eignet sich insbesondere zur Beschreibung kontextfreier Grammatiken und somit für die Definition von Regeln, die festlegen, was in der jeweiligen Grammatik erlaubt ist. Dafür werden Terminalsymbole benötigt, aus denen die Sprache bestehen kann und als Basiselemente der Sprache dienen.. Über Regeln wird beschrieben, wie aus mehreren terminalen Symbolen ein nicht-terminales Symbol gebildet werden kann. Dafür findet eine Zuordnung statt, wobei auf der linken Seite ein nicht-terminales Symbol gesetzt wird, während auf der rechten Seite verschiedene Anordnungen terminaler Symbole diesem zugeordnet werden. Wegen der erweiterten BNF können diese nicht nur alternativ gruppiert sein, sondern auch über Klammerung definiert werden. Dadurch ist eine bessere Lesbarkeit und kompaktere Schreibweise möglich, weil angegeben werden kann, wie häufig die geklammerten, terminalen Symbole vorkommen dürfen. (vgl. Garshol 2006)

Feature Diagramme zur DSL-Modellierung

Zur Modellierung von domänenspezifischen Sprachen eignen sich insbesondere Feature Diagramme mit einer hierarchischen Baumstruktur. Diese Art von Diagrammen fasst die Relationen zwischen Features zusammen, die gemeinsame Eigenschaften zwischen den Systemen darstellen. Die Wurzel des Feature-Baums repräsentiert dabei ein Konzept, während die darunterliegenden Knoten die Features des Konzeptes darstellen. Diese können über Kardinalitäten, optionale oder vorausgesetzte Möglichkeiten und Gruppierungen zu verschiedenen

Konzepten verknüpft werden (vgl. Czarnecki / Helsen / Eisenecker 2004). Abbildung 2.13 fasst die Modellierungsmöglichkeiten der Feature-Syntax zusammen.

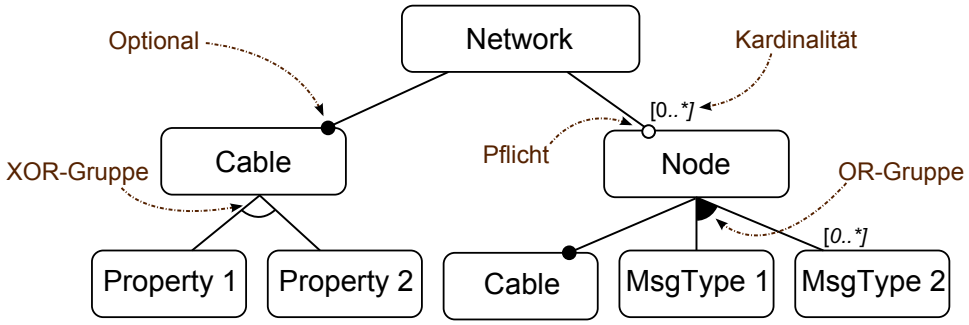

Abbildung 2.13: *Die verschiedenen Auswahlmöglichkeiten der Feature-Syntax*

2.4 Verwandte und vergleichbare Arbeiten

Die derzeitige Aufteilung der Domänen im Fahrzeug rückt mit der Einführung einer Backbonearchitektur immer weiter in den Hintergrund. Während die Domänen zu Beginn noch erhalten bleiben und nur eine Verbindung über ein Backbone erfolgt, läuft der Ersatz durch ein flaches Netz, bei dem sämtliche Geräte das gleiche Protokoll sprechen, immer weiter voran. Als Backbonetechnologie wird sich wahrscheinlich Ethernet mit einer Echtzeiterweiterung wie Audio Video Bridging (AVB) oder Time-Triggered Ethernet (TTEthernet) etablieren (vgl. Tuohy / Glavin / Hughes u. a. 2014), das den zukünftigen Anforderungen hinsichtlich der zeitkritischen Übertragung und der benötigten Bandbreite gewachsen ist. Während zu Beginn noch der Einsatz von Standard Ethernet zwischen den Domänen mit unterschiedlichen Topologien getestet wurde (Lim / Krebs / Volker u. a. 2011), und es sich zeigte, dass Ethernet als Backbone eingesetzt werden kann, jedoch eine Echtzeitkontrolle fehlt, beschäftigt sich die Analyse derzeitig mit der Frage, welche Echtzeit-Ethernet-Erweiterung die geeignetste im Automobil darstellt. Für AVB gibt es im Automotive Bereich vor allem Arbeiten von der BMW-Group, die sich mit der simulationsbasierten Analyse (Lim / Herrscher / Waltl u. a. 2012) in OMNeT++ beschäftigen. Für Time-Triggered Ethernet finden sich mehrere Simulationsstudien in der CoRE-Arbeitsgruppe (CoRE RG [a]) an der Hochschule für Angewandte Wissenschaften Hamburg, die einen Einsatz von TTEthernet in der gleichen Simulationsumgebung nutzen (vgl. Dieumo Kenfack 2010; Kempf 2011; Steinbach 2011; Todorov / Steinbach / Korf u. a. 2013). Zudem existiert für das TTEthernetmodell der CoRE-Gruppe eine Kombination mit einem AVB-Stack (Rumpf / Steinbach / Korf u. a. 2014), der als zusätzliche Nachrichtenklasse genutzt werden kann. Ein Vergleich zwischen AVB und TTEthernet als Backbonetechnologie wurde ebenfalls mit dem CoRE- und dem

BMW-Simulationsmodell durchgeführt (Steinbach / Lim / Korf u. a. 2012) und hat gezeigt, dass beide den zukünftigen Anforderungen gewachsen sind.

Die genannten Arbeiten haben gemeinsam, dass hier eine Betrachtung auf das Backbone gelegt wurde und dabei die heterogene Ansicht über die Gateways, die zusätzliche Latenzen verursachen und kritische Punkte in der Kommunikation darstellen, außer Acht lassen. Für die Simulation heterogener Netzwerke ist es jedoch erforderlich, sowohl geeignete Simulationsmodelle als auch ein geeignetes Datenmodell, das die Ende-zu-Ende Kommunikation zwischen den Geräten ausreichend beschreibt, zu nutzen. Zudem ist ein definiertes Datenmodell notwendig, wenn Ergebnisse zwischen verschiedenen Modellen verglichen werden sollen. In „Development of an Automotive Communication Benchmark"(Mohammad / Al-Holou 2010) wurde der Gedanke eines standardisierten Datenmodells bereits aufgegriffen, kann jedoch zukünftigen Anforderungen mit hohen Bandbreiten nicht gerecht werden. Die grundlegenden Kommunikationsnachrichten lassen sich auf mehrere Klassen mit unterschiedlichen Anforderungen aufteilen (vgl. Rahmani / Hillebrand / Hintermaier u. a. 2007). Allerdings ist auch dies nicht standardisiert, wodurch in anderen Arbeiten die Klassenanforderungen different ausfallen (vgl. Steiner / Bauer / Hall u. a. 2009), beziehungsweise die Klassenanzahl nicht gleich ist.

Wenn die Funktionen, die das Kommunikationsmodell ausreichend beschreiben, jedoch bekannt sind, ist eine Konfiguration des Netzwerkes so weit zu abstrahieren, dass unterschiedliche Softwarearchitekturen unterhalb gebildet werden können. In „Engineering Automotive Software“ (Broy / Krüger / Pretschner u. a. 2007) werden verschiedene Abstraktionsstufen eingeführt, die eine mögliche Beschreibung vorantreiben. Wird eine abstraktere Stufe als die direkte Parametereingabe gewählt, müssen viele Parameter automatisch berechnet und generiert werden.

Für die Wahl der richtigen Topologie im Automobil gibt es verschiedene Ansätze, die genutzt werden. Zum einen können diese nach Kostenaspekten evaluiert werden, wodurch die Minimierung der Kabelkosten, des benötigten Platzes der Verkabelung und der Leistungsaufnahme im Vordergrund steht (vgl. Müller-Rathgeber / Michel 2009). Zum anderen kann eine Bewertung nach Leistungsmetriken durchgeführt werden. Diese Art der Verlässlichkeitsanalyse paketvermittelnder Netzwerke (vgl. Schäfer 2004) ermöglicht die Ermittlung einer Dienstgüte, wodurch auch das Gesamtnetzwerk evaluiert werden kann. Für Standard Ethernet wurde dies bereits für homogene Fahrzeugtopologien durchgeführt (Lim / Weckemann / Herrscher 2011).

Ein gesamtheitlicher Ansatz mittels einer domänenspezifischen Sprache für die Kommunikation im Automobil ist nach derzeitigem Kenntnisstand nicht verfügbar. Durch die Möglichkeiten der Konfigurationsgenerierung für verschiedene Simulationsmodelle und die Erweiterbarkeit auf andere Simulationsumgebungen, können DSLs auch im simulativen Bereich zur Einbindung von Domänenexperten, die selber keine Programmiererfahrung besitzen, durchgeführt werden.

3 Anforderungen zur Evaluierung heterogener Fahrzeugnetzwerke

Um heterogene Fahrzeugnetzwerke analysieren und miteinander vergleichen zu können, umfasst der Evaluierungsprozess verschiedene Aspekte. Durch den immer größer werdenden Bedarf an Kommunikation zwischen den Domänen und dem wachsenden Bereich der Fahrerassistenzsysteme ist es notwendig, die Kommunikationsstrukturen im Automobil zu erarbeiten. Dazu gehören neben den Endgeräten und den Topologien vor allem die Nachrichten, die es ermöglichen, realistische Fahrzeugdesigns zu analysieren. Wurde ein vollständiges Datenmodell erarbeitet, ist die Evaluation verschiedener Topologien möglich. Obwohl das Datenmodell bei jeder Simulation gleich ist, müssen umfangreiche Anpassungen an den unterschiedlichen Konfigurationsdateien bei jeder Simulation stattfinden. Dabei steht vor allem die Heterogenität, also das Zusammenspiel von verschiedenen Bussystemen, die über ein Backbone miteinander verbunden sind, im Vordergrund. Gerade die Anpassung der Struktur des Backbones muss einfach sein und ein Teil der Konfigurationskomplexität verborgen bleiben. Dabei gilt es, statische Parameter, die vor jeder Simulation vom Anwender eingegeben werden können, festzuhalten und dynamische Parameter, die sich aus den Anforderungen für das Datenmodell und der Topologie ergeben, zu analysieren. Anschließend kann eine abstrakte Beschreibung für Netzwerke gefunden werden, die es ermöglicht, benötigte Konfigurationsdateien für Simulationen zu generieren.

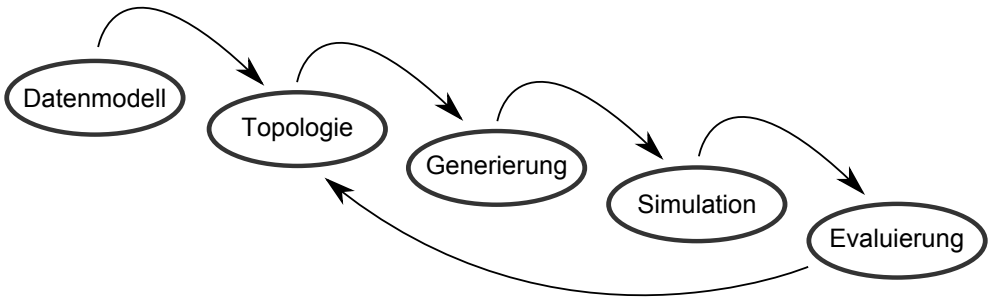

Abbildung 3.1: *Um diverse Topologien zu testen, sind unterschiedliche Teilschritte notwendig. Mit Hilfe eines Datenmodells können verschiedene Topologien erstellt, generiert und simuliert werden.*

Um die Simulation anschließend auszuwerten und analysieren zu können, werden in ihr Metriken, also Kennzahlen, die bestimmte Größen beschreiben, aufgezeichnet. Diese gilt es, in der Evaluierungsphase zu bewerten und in geeigneter Form aufzubereiten, so dass deutlich wird, inwiefern die gewählte Topologie die Anforderungen erfüllt. Abbildung 3.1 zeigt, welche Aufgabenbereiche Anforderungen benötigen.

3.1 Datenmodellierung

Durch die erhöhte Komplexität im Automotive-Kontext unterscheiden sich die Anforderungen an das heutige Automobilnetzwerk zu dem damaligen. Neben zeitkritischen Funktionen, die für die Sicherheit der Insassen harte Echtzeitanforderungen erfüllen müssen, und Funktionen, die verschiedene Dienste für den Komfort bereitstellen, sind zukünftig bandbreitenintensive Anwendungen, die ebenfalls harten Echtzeitanforderungen genügen müssen, relevant. Dabei ist es entscheidend, welche Daten heute und in Zukunft wichtig sind. Dieser Abschnitt erläutert die Anforderungen an ein standardisiertes Datenmodell, das als Grundlage für zukünftige Evaluierungen genommen werden kann. Dazu gehört das Identifizieren und Spezifizieren von Kommunikationsnachrichten, wodurch die Anforderungen an den Aufbau, die Anzahl der Nachrichten und die Abstraktion auf bestimmte Nachrichtenklassen gemeint sind.

3.1.1 Kommunikationsnachrichten als Analysegrundlage

1993 wurde von der Society of Automobile Engineers (SAE) ein Dokument veröffentlicht, das die Anforderungen an ein sicherheitskritisches Fahrzeugnetzwerk mit Punkt-zu-Punkt Verbindungen aufzeigt (Lupini 2004). Es dient als Grundlage für weitere Projekte, die sich mit der Kommunikation im Automobil beschäftigt haben. In dem Dokument sind 53 Signale beschrieben, die zwischen sieben verschiedenen Endgeräten ausgetauscht werden und Anforderungen an jedes Signal spezifiziert. In der folgenden Tabelle 3.1 ist die Spezifikation der Signale dargestellt:

Merkmal:	Descr.	Size	Jitter	Time	Periodic	Deadline	Sender	Receiver
Einheit:	-	bytes	ms	ms	yes/no	ms	-	-

Tabelle 3.1: *Spezifikation von Signalen für Punkt-zu-Punkt Verbindungen nach Tindell / Burns (Tindell / Burns 1994).*

Neben der Beschreibung sind die prägensten Merkmale eines Signals die Größe in Byte und die Periode, mit der das Signal auf den Bus gelegt wird. Daneben gibt es einen Sender und einen Empfänger, der das Signal erhalten soll. Zudem sind die wichtigsten Anforderungen

für zeitkritische Kommunikation mit der Deadline[1] und dem Jitter[2] angegeben. Hierbei ist zu überprüfen, ob diese Spezifikation ausreichend ist, um die Kommunikation im Automobil hinreichend zu beschreiben. Beispielsweise können nicht mehrere Empfänger das gleiche Signal empfangen, weil nur ein Empfänger angegeben werden kann. Da es zukünftig eine große Rolle spielt, eine 1:n Beziehung im Automobil für einzelne Signale zu beschreiben, ist die Spezifikation des SAE-Benchmarks in diesem Punkt nicht mehr ausreichend.

Neben dem SAE-Benchmark wurde 1997 der PSA-Benchmark als Analysegrundlage für Funktionen im Automobil veröffentlicht (vgl. Castelpietra / Song / Simonot-Lion u. a. 2000). Dieser soll das CAN-Netzwerk in einem Peugeot Citroen widerspiegeln. Er besteht aus einem CAN- und einem VAN-Bus, über die insgesamt 19 Nachrichten geschickt werden. Beide Benchmarks können die Komplexität moderner Automobilanwendungen mit der Nachrichtenanzahl und ihrer benötigten Bandbreite nicht bedienen. Eine Zusammenführung der beiden Benchmarks in dem Paper „Development of an Automotive Communication Benchmark“ (Mohammad / Al-Holou 2010) an der Universität in Detroit zeigt, in welche Richtung sich ein Kommunikationsmodell bewegen muss. Hier werden beide Benchmarks in Funktionen aufgeteilt, überprüft und mit modernen Anforderungen erweitert. Anschließend findet eine Validierung der 84 Signale in einer CAN-Bus Timinganalyse statt.

Die erwähnten Benchmarks stellen nur bedingt ein realistisches Fahrzeugdesign dar. Aufgrund der geringen Anzahl von Endsystemen und Signalen ist die Option, ein Fahrzeugbackbone mit heterogenen Ausprägungen zu evaluieren, stark begrenzt. Daher muss sowohl die Anzahl der Endsysteme als auch die Anzahl der Signale beziehungsweise der Nachrichten erhöht werden.

3.1.2 Aufbau einer Kommunikationsmatrix

Die bereits im vorhergehenden Abschnitt 3.1.1 beschriebenen Signale stellen Kommunikationsbeziehungen zwischen einzelnen Endsystemen im Automobil dar. Diese sind sogenannte Sender-Empfänger-Beziehungen, bei denen jede Kommunikationsnachricht die benötigten Informationen von einem Sender zu einem oder mehreren Empfängern für das Netzwerk bereithält. Werden die Nachrichten tabellarisch aufgelistet, spricht man von einer Kommunikationsmatrix (K-Matrix), die alle kommunikationsrelevanten Informationen des Netzwerkes bereit hält (Schäuffele / Zurawka 2013). Ein unterschiedlicher Aufbau einer K-Matrix ist möglich. Tabelle 3.2 zeigt ein Beispiel mit einer Aufteilung nach sendenden Netzknoten.

[1] Jeder Prozess, auch Task genannt, in einem Real-Time System besitzt eine Antwortzeit, die aussagt, wie lange der Task ausgeführt wird. Die maximal erlaubte Antwortzeit wird als Deadline bezeichnet. (Ben-Ari 1990)

[2] Die Varianz der Verzögerung, mit der ein Signal eintrifft. Im Gegensatz zu der Latenz eines Paketes sollte der Jitter möglichst klein sein, so dass die Latenz gleicher Pakete konstant ist. (Kopetz 2004)[S. 09]

Netzknoten	Nachricht	Signal	ABS-Steuergerät	Motorsteuergerät	Getriebesteuergerät	⋰
ABS-Steuergerät	ABS-1	Raddrehzahl vorne links	S		E	
		Raddrehzahl vorne rechts	S		E	
						...
	ABS-2	Raddrehzahl hinten links	S		E	
		Raddrehzahl hinten rechts	S		E	
Motorsteuergerät	MS-1	Fahrpedalwert		S	E	
		Motordrehzahl	E	S	E	
						...
...	...	...	...	...	...	...

Tabelle 3.2: *Kommunikationsmatrix mit einer Sortierung nach Netzknoten. (Schäuffele / Zurawka 2013)[S. 84].*

In der ersten Spalte mit den Netzknoten werden alle Endgeräte des Netzwerkes aufgelistet. In den darauffolgenden Spalten sind die einzelnen Nachrichten mit den beinhaltenden Signalen aufgeführt und die empfangenden Endgeräte in der Spalte der jeweiligen ECU mit einem „E“ gekennzeichnet. Eine Kombination dieser Kommunikationsmatrix mit den Eigenschaften der einzelnen Nachrichten und den Anforderungen, die sich für jede Nachricht ergeben, ist dabei für eine vollständigere Kommunikationsmatrix anzustreben.

Bei zeitkritischen Systemen sind die Anforderungen an Zuverlässigkeit, Sicherheit und Fehlertoleranz sehr hoch. Deshalb müssen Systemarchitekturen für Echtzeitanwendungen konstruktionsbedingt ein vorhersagbares und nachweisbares Verhalten erlauben (Reif 2010). Dieses Verhalten muss in der Kommunikationsmatrix durch Anforderungen an jedes Signal sichtbar gemacht werden und für jeden Empfänger definierbar sein. In erster Linie sind damit die bereits erwähnte Deadline und der Jitter für jedes Signal einzuhalten, die zuvor definiert sein müssen. Gerade bei heterogenen Netzwerken bedarf die Einhaltung der definierten Kennzahlen einer großen Bedeutung, damit kritische Teilbereiche identifiziert werden können.

3.1.3 Klassifizierung von Nachrichten und Endgeräten

Damit die Betrachtung auf der Topologie des Netzwerkes und dem dahinterliegenden Backbone gelegt werden kann, ist es sinnvoll, die Nachrichten in einer einheitlichen und allgemeinen

Darstellung zu spezifizieren. Dadurch kann eine Abstraktion verschiedener Nachrichten auf eine Klasse, welche diese Nachrichten repräsentiert, stattfinden. Eine Schwierigkeit ist die Definition verschiedener Nachrichtenklassen, welche die unterschiedlichen Anforderungen im Automobil abbilden. Dazu gilt es, die Nachrichten zu analysieren und zu evaluieren, damit ähnliche Nachrichten identifiziert und in Klassen gruppiert werden können.

Von der Society of Automotive Engineers wurden typische Anwendungsklassen für Bussysteme definiert. Dabei sind unterschiedliche, funktionale Anforderungen an Bussysteme analysiert und unter Berücksichtigung des Einsatzgebietes, der notwendigen Übertragungsrate sowie der Botschaftslänge in Klassen untergliedert worden (vgl. Reif 2009). Diese unter dem Namen SAE-Klassen bekannten Anforderungsbereiche beschreiben unterschiedliche Merkmale und sind in Tabelle 3.3 mit einem jeweils typischen Bussystem abgebildet.

SAE-Klasse:	**Merkmale**	**Typisches Bussystem**
A	Vernetzung von Aktoren und Sensoren, Geringe Datenraten (ca. 10 kBit/s), Geringe Ausprägung von Fehlererkennungs- und -behebungsmechanismen	LIN-Bus
B	Vernetzung von Steuergeräten (z. B. Komfortbereich), Mittlere Datenraten (ca. 125 kBit/s), Komplexe Mechanismen zur Fehlererkennung und -behebung	CAN-Bus („Low Speed“)
C	Vernetzung von Steuergeräten mit „einfachen“ Echtzeitanforderungen (z.B. Antriebsstrang), Hohe Datenraten (bis zu 1 MBit/s)	CAN-Bus („High Speed“)
D	Multimedia-Anwendungen, Sehr hohe Datenraten (bis zu 10 MBit/s) und Botschaftslängen	MOST-Bus

Tabelle 3.3: *Unterteilung von Bussystemen in SAE-Klassen (Reif 2009)[S. 14].*

In heterogenen Fahrzeugnetzwerken ist ein Ziel, mehrere Bussysteme aus den verschiedenen Klassen über ein leistungsstarkes Backbone miteinander zu verknüpfen. Daher kann das Gesamtnetzwerk nicht mehr einer Klasse zugeordnet werden. Jede Nachricht behält aber ihre Eigenschaften bezüglich einer fest definierten Größe, Periode und Anforderungen, die beispielsweise ihre Übertragungszeit betreffen. Wenn es möglich ist, jeder Nachricht aus der Kommunikationsmatrix einer definierten Klasse zuzuordnen, kann eine differenziertere Ansicht auf das Netzwerk erfolgen. Jedes Endgerät besitzt dann bei jeder Nachrichtenklasse eine bestimmte Anzahl an Nachrichten, die es ermöglicht, ebenfalls das Endgerät zu kategorisieren. Anschließend kann von einem abstrakten Netzwerk mit standardisierten Endgeräten gesprochen werden, wodurch eine Betrachtung auf unterschiedliche Topologien möglich ist. Das nachfolgende Kapitel beschäftigt sich mit der derzeitigen Topologie im Automobil und den Hindernissen, die mit einer Backbonetopologie überwunden werden muss.

3.1.4 Topologien für ein Echtzeit-Ethernet-Backbone

Die Kommunikation zwischen den einzelnen Endgeräten im derzeitigen Automobil ist in verschiedene Domänen eingeteilt. Dabei spielen unterschiedliche Netzwerktechnologien, wie Controller Area Network (CAN), FlexRay und Media Oriented System Transport (MOST) verschiedene Rollen in den Domänen, um verschiedene Anforderungen zu bedienen. Die domänenübergreifende Kommunikation findet meist über ein zentrales Gateway statt, das in dem heterogenen Fahrzeugnetzwerk die verschiedenen Protokolle übersetzt (vgl. Lim / Weckemann / Herrscher 2011). In Abbildung 3.2 ist ein exemplarisches Automobilnetzwerk mit vier Bussystemen und einem zentralen Gateway, das zu Diagnosezwecken dient, dargestellt.

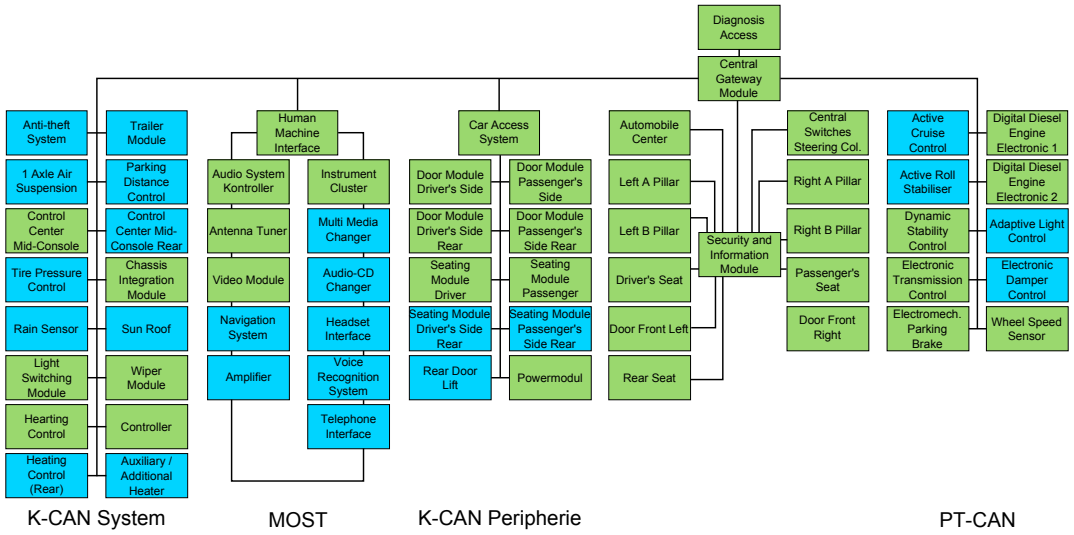

Abbildung 3.2: *Klassische Netzwerktopologie im Automobil mit verschiedenen Bussystemen und zentraler Gatewayarchitektur. Quelle: (Lim / Weckemann / Herrscher 2011)[S. 166]*

Die Karosserie-CANs (K-CAN) sind Low-Speed CAN-Busse, die für die Komfort- und Karosserietechnik zuständig sind. Der Powertrain-CAN (PT-CAN) ist ein High-Speed CAN-Bus und wird als Sicherheits- und Informationsbussystem genutzt. Daneben gibt es noch einen MOST-Bus, an den Endgeräte mit bandbreitenintensiven Anwendungen wie Multimediadaten angeschlossen sind. Das Konzept mit einem zentralen Gateway und dieser Aufteilung der Bussysteme kann ebenfalls bei diversen Fahrzeugen wie dem Mercedes-Benz SL gefunden werden (vgl. Drossos / Lindbüchl / Grohmann u. a. 2013)[S. 211]). Die Anzahl der Endgeräte, die Domänen und die dafür eingesetzten Bussysteme variieren durch die unterschiedlichen Konfigurationsmöglichkeiten eines Fahrzeugs sehr stark. Daher ist ein Ziel, verschiedene To-

pologien zu einem Referenzmodell zusammenzutragen, damit ein Grundgerüst von Varianten abgedeckt ist.

Ein Problem, das bei dem hier dargestellten zentralen Ansatz auftritt, ist, dass der Ausfall des Knotenpunktes ein Versagen der gesamten domänenübergreifenden Kommunikation nach sich zieht. Zudem wird die Anzahl von Endgeräten im Automobil immer größer und neue Technologien werden eingeführt, die sowohl bandbreitenintensiven als auch zeitkritischen Anforderungen genügen müssen. Aus diesem Grund kann eine mögliche zukünftige Lösung in einem Backbone-Bussystem liegen, das mit dezentralen Gateways unterschiedliche Ansätze bietet (vgl. Reif 2009)[S. 34]. Eine entscheidende Frage ist, welche Topologie für das Backbone eine geeignete Struktur aufweist, sodass die Nachrichten dem Quality of Service (QoS) des Netzwerkes entsprechen.

In dem Paper „Automotive Network Planning - a genetic approach“ (Müller-Rathgeber / Michel 2009) wurde der Frage nach einem Topologieansatz für ein ethernetbasiertes Netzwerk einer zukünftigen Kommunikationsarchitektur im Automobil nachgegangen. Dabei wurde das Netzwerk auf Kostenaspekte wie Kabelkosten, benötigter Platz der Verkabelung und Leistungsaufnahme untersucht und nach der besten Lösung evaluiert. Abbildung 3.3 zeigt die beste Lösung mit 76 ECUs, 22 Switches und 98 Verbindungen. Die erzeugte Topologie

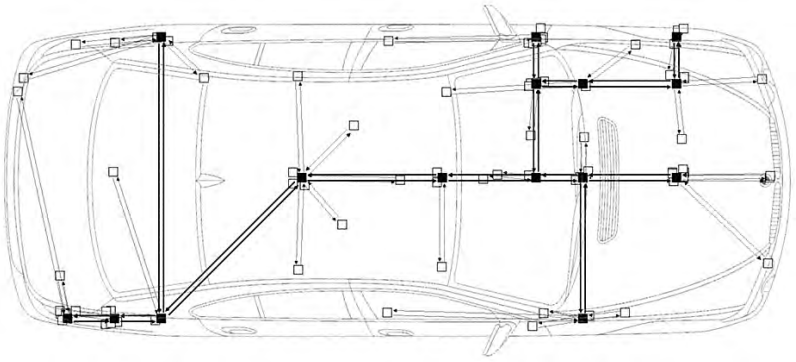

Abbildung 3.3: *Physikalischer Graph. Die schwarzen Linien stellen die Verbindungen des Backbones dar, die grauen die zwischen den Endgeräten und den Switches. Quelle: (Müller-Rathgeber / Michel 2009)*

zeigt, wie ein Backbone eines Fahrzeugnetzwerkes im Bezug zu den Kosten aussehen kann. Das Problem, das sich mit einer großen Anzahl an Switches ergibt, ist die Konfigurierung der zeitkritischen Kommunikation. Je mehr Switches eingesetzt werden, desto größer wird die Latenz der einzelnen Nachrichten. Daher müssen komplizierte Netzwerktopologien aus den Grundformen der Netzwerktopologie (vgl. 2.1.3) aufgebaut sein (Schäuffele / Zurawka 2013) [S. 81].

3.2 Simulation heterogener Netzwerke

Zur Analyse heterogener Fahrzeugnetzwerke bedarf es neben einer umfangreichen Datenbasis mehrerer Komponenten, welche die einzelnen Modelle der Bussysteme abbilden. Die Simulation soll dabei den richtigen Detailgrad erreichen, wodurch die real existierenden Bussysteme auf möglichst genaue Modelle abstrahiert werden müssen, ohne sich in Details zu verlieren. Durch ein zu wenig abstraktes Modell steigt die Anzahl der Parameter zu sehr, wodurch die Simulation unnötig verkompliziert wird und die Simulationsgeschwindigkeit erheblich beeinflusst werden kann. Wird das Modell jedoch zu ungenau beschrieben, ist eine realistische Auswertung nicht mehr möglich, weil das Simulationsergebnis unter Umständen unbrauchbar ist und die Realität nicht hinreichend abbildet.

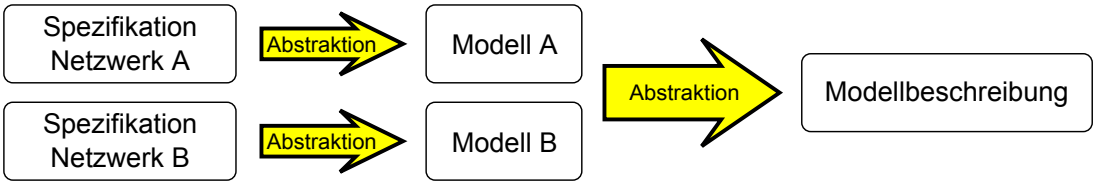

Abbildung 3.4: *Abstraktionsstufen verschiedener Netzwerkspezifikationen auf Modelle. Jedes Modell ist einzeln ansprechbar. Durch eine gesamtheitliche Modellbeschreibung können verschiedene Modelle zusammengefasst werden.*

Bei heterogenen Netzwerken ist darauf zu achten, dass jedes Subnetzwerk für sich den richtigen Grad der Abstraktion erreicht. Bei der Abweichung eines Modells von der Spezifikation ist die Auswertung des gesamten Netzwerkes nicht mehr möglich, und falsche Rückschlüsse auf die reale Umsetzung des Problems könnten gezogen werden. Aufgrund der gesteigerten Komplexität des Netzwerkes durch die Verbindung verschiedener Modelle ist es notwendig, eine abstraktere Form der Darstellung für das Gesamtnetzwerk zu finden. Dadurch ist es möglich, die Komplexität wieder auf ein geeignetes Maß zu setzen, während die Komplexität der einzelnen Modelle nicht beeinträchtigt wird. Abbildung 3.4 zeigt die Abstraktionsstufen, die eine Betrachtung heterogener Modelle erlaubt. Aus den direkten Spezifikationen werden Modelle gebildet, die in einer weiteren Abstraktionsstufe als ein Modell beschrieben werden. Darüber hinaus soll es möglich sein, aus einer Modellbeschreibung die benötigten Konfigurationen für die Komponenten der Modelle zu generieren und so die Funktionalität der einzelnen Modelle zu nutzen. Aus den genannten Gründen werden im Folgenden die benötigten Komponenten für die Simulation heterogener Systeme erläutert und beschrieben, wie das Zusammenspiel aus Simulationserstellung und Auswertung auf eine abstrakte Weise erfolgen kann.

3.2.1 Benötigte Simulationskomponenten für heterogene Netzwerke

Für die Simulation heterogener Netzwerke sind mehrere Komponenten zu modellieren und in einer Simulationsplattform umzusetzen. Wie aus der klassischen Topologie 3.2 ersichtlich ist, werden mehrere Bussysteme mit einer Vielzahl von Endgeräten benötigt. Zudem sind Gateways notwendig, die zwischen den Bussystemen übersetzen, so dass eine domänenübergreifende Kommunikation möglich ist. Die Bussysteme müssen als unabhängige Komponenten flexibel, konfigurierbar und in verschiedenen Ausbaustufen (Skalierungen) ausführbar sein. Zudem müssen die protokollspezifischen Aspekte aus Kapitel 2.1.1 den Modellierungsprozess unterstützen. Um die Heterogenität zu simulieren, bedarf es – wie bereits erwähnt – Gateways, die zwischen unterschiedlichen Protokollen übersetzen. Dabei kann der zentralisierte Ansatz gewählt werden, wobei jedes Gateway die Spezifikation sämtlicher zu übertragender Protokolle beherrschen muss oder ein dezentraler Ansatz, bei dem ein Gateway zwischen einem leistungsstarken Backbone und mehreren Bussen eines Typs übersetzt.

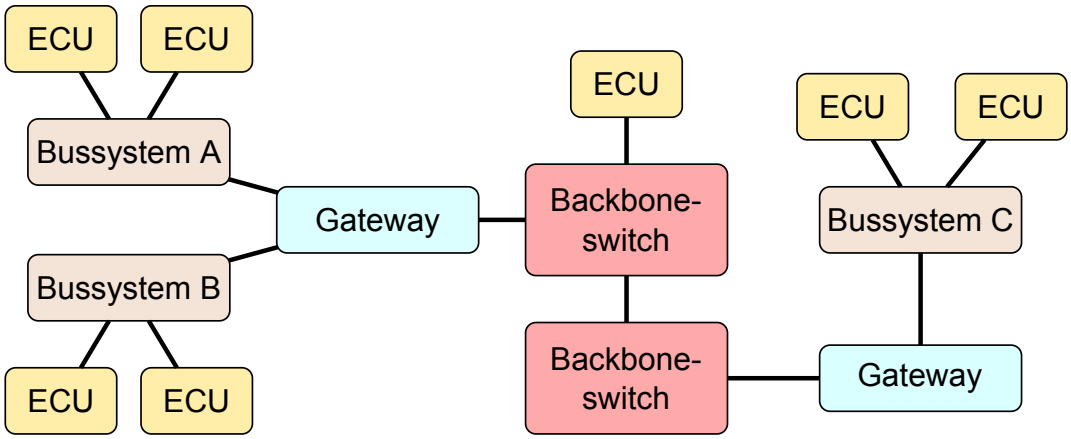

Abbildung 3.5: *In der Simulation werden Komponenten für Endgeräte, Bussysteme, Gateways und das Backbone benötigt.*

Wegen der erhöhten Komplexität und der hohen Anforderungen an die domänenübergreifende Kommunikation durch mehr Endgeräte und Nachrichten, die an mehr Teilnehmer adressiert sind, ist es notwendig, den dezentralen Ansatz weiter zu verfolgen. Dafür muss das Backbone eine ausreichende Bandbreite zur Verfügung stellen, um den gesteigerten Erwartungen gerecht zu werden und als zeitkritische Anforderung Echtzeitfähigkeit bereithalten. Aus den in Kapitel 2.1.2 beschriebenen Gründen eignet sich Ethernet im Bereich des Automotive Kontextes. Erweiterungen wie Time-Triggered Ethernet oder AVB geben Ethernet die benötigte Echtzeitfähigkeit.

Als Module in der Simulation werden somit Komponenten des Backbones, der Gateways und der Endgeräte, die als Lastgeneratoren den Traffic erzeugen, benötigt. Um den heterogenen Ansatz zu verfolgen, muss neben dem Backbone ein Bussystem umgesetzt sein. Als dominante Bustechnologie für den Fahrzeugstrang und als Verbindung zwischen den Bordcomputern (vgl. Fröberg / Sandström / Norström u. a. 2003) ist der CAN-Bus die erste Wahl.

3.2.2 Analysemethoden zur Evaluierung

Damit Netzwerke analysiert werden können, wurden mehrere Methoden entwickelt, die sich für verschiedene Netzwerktypen eignen und eine unterschiedliche Komplexitätsgüte haben. Dabei muss bei jeder eingesetzten Methode, wie in Abbildung 3.1 bereits gezeigt, zuerst ein Datenmodell und eine Topologie erstellt oder generiert worden sein. Anschließend können mit analytischen Methoden oder simulationsbasierten Analysen Kennzahlen gewonnen werden, die Rückschlüsse auf das Verhalten zulassen.

Analytische Methoden

Analytische Methoden eignen sich bei heterogenen Netzwerken für die Analyse einzelner Busse oder Nachrichtenklassen. In dem Paper „Network Calculus for the Validation of Automotive Ethernet in-Vehicle Network Configurations“ (Manderscheid / Langer 2011), das an dem Fraunhofer Institut 2011 veröffentlicht wurde, wird eine Worst-Case Analyse für ethernetbasierte Fahrzeugnetzwerke vorgestellt. Bei einer Worst-Case Analyse senden alle Endsysteme zu einem bestimmten Zeitpunkt gleichzeitig ein Paket, so dass größtmögliche Warteschlangen in den Switches entstehen und einige Signale maximal delayed werden. Durch diese pessimistische Ansicht wird die maximale Verzögerung des Netzwerkes ermittelt. Der große Vorteil besteht in der deterministischen, schnellen Berechnung, die sich auf unterschiedliche Netzwerke erweitern lässt. Aufgrund des gewählten Ansatzes aus Georges / Divoux / Rondeau (Georges / Divoux / Rondeau 2005) ist es möglich, beliebige homogene Topologien und zeitkritische Anwendungen auszuwerten.

Bei heterogenen Netzwerken ist es schwierig, domänenübergreifend analytische Methoden zu entwickeln. Zu jedem Teilnetz müssten unterschiedliche Berechnungen herangezogen werden, und eine domänenübergreifende Analyse spiegelt meist nicht die Realität wider. Bei Time-Triggered Ethernet, das drei verschiedene Nachrichtenklassen unterscheidet, lassen sich die einzelnen Klassen für sich ebenfalls gut mit einer analytischen Methode berechnen, allerdings müssten die drei unterschiedlichen einzelnen Modelle zu einem kombiniert werden, wodurch keine lineare Komplexität erreicht werden würde.

Simulationsbasierte Analyse

Simulationsbasierte Analysen eignen sich besser für das Zusammenspiel verschiedener Modelle. Jedes Modell stellt dabei ein in sich geschlossenes, konnektivitätsbasiertes Modell dar und kann über Schnittstellen und Zwischenmodule mit differenten Modellen kommunizieren. Jede Komponente eines Modells ist dabei imstande, Metriken zu erheben und somit Auskunft über den Zustand des Netzwerkes zu geben. Bei Simulationen sind unterschiedliche Auswertungszeitpunkte denkbar. Zum einen kann die Analyse online erfolgen, was eine Rückgabe der Simulation während der Laufzeit bedeutet. Das können überlaufende Buffer, verlorene Pakete oder andere Anforderungen, die vorher definiert, aber nicht eingehalten wurden, sein. Zum anderen ist es möglich, einige beziehungsweise alle erwünschten Metriken aufzuzeichnen und, nachdem die Simulation beendet wurde, auszuwerten.

Bei heterogenen Systemen ist es erforderlich, die Berechnung des Verhaltens nach der Simulationszeit durchzuführen. Durch lange Laufzeiten, die sich bei umfangreichen Netzwerken ergeben und mehrere Stunden dauern können, ist eine Auswertung während der Simulation nicht erstrebenswert.

Um einzelne Systeme zu testen, hat sich für die Entwicklung von Steuergeräten in der Automobilindustrie das Mittel der Restbussimulation verbreitet (vgl. Riegraf / Behh / Kraus 2007). Hierbei werden nicht alle Geräte als verteiltes System simuliert, sondern nur neu entwickelte Geräte werden als reelle Teilnehmer in einer simulierten Umgebung getestet. Dabei ist es in erster Linie erforderlich, dass die Verbindung zwischen dem Restbussimulator und dem reellen Teilnehmer korrekte Daten bereithält, weil dieses als System under Test (SuT) getestet werden soll.

Für die Auswertung neuer Netzwerktopologien mit verschiedenen Bussystemen sind Restbussimulation nicht optimal geeignet. Durch die Konfiguration, die für sämtliche Teilnehmer bei einer Topologieänderung angepasst werden muss, ist eine rein simulative Lösung erstrebenswert, bei der die Verkehrslast schnell überprüft werden kann.

3.2.3 Metriken für die simulationsbasierte Analyse

In Kapitel 3.2.2 wurden verschiedene Varianten zur Netzwerkanalyse vorgestellt. Dabei muss bei jeder Methode festgelegt werden, welche Kenngrößen, auch Metriken genannt, festzuhalten sind, damit eine hinreichende Analyse möglich ist. Jede Metrik beschreibt dabei eine Funktion, einen Systemzustand des Netzwerkes oder einer Netzwerkkomponente und ordnet dieser eine reelle, meist positive Zahl zu (vgl. Schäfer 2004). Jede Metrik kann dabei einer Kategorie zugeteilt werden, die als Maß für ihre Bewertung steht. Bei Steinbach (Steinbach 2010) sind Metriken in folgende Klassen unterteilt:

- *Ökonomische Metriken* dienen zur Messung wirtschaftlicher Größen, wie den Anschaffungs-, Installations- oder Unterhaltungskosten.
- *Verlässlichkeits-Metriken* stellen die Zuverlässigkeit und somit die fehlerfreie Funktion eines Systems dar.
- *Leistungs-Metriken* bewerten die Güte und Leistungsfähigkeit eines Systems.

Während sich ökonomische Metriken zur Analyse der Rentabilität und nicht als Ergebnis einer Netzwerksimulation eignen, können sich Verlässlichkeits- und Leistungs-Metriken als Kennwerte aus der Simulation ergeben. Damit diese Arten von Metriken aufgezeichnet werden können, ist es wichtig, passende Referenzpunkte zu wählen. Dabei gilt es, zwei unterschiedliche Ansichten zu differenzieren. Metriken, die innerhalb eines Systems zu messen sind und Informationen über den Netzwerk-Zustand aus den Zwischenknoten gewinnen, nennen sich System-Metriken. Dazu zählt beispielsweise die Paketverlustrate, die auf der Transportdienst-Ebene Auskunft für nicht übertragene Pakete auf Teilstrecken geben kann. Demgegenüber stehen Metriken, die aus der Sicht des Anwenders formuliert und Benutzer-Metriken genannt werden. Diese stellen eine Ende-zu-Ende Sicht dar und geben direktes Feedback über den Gesamtzustand des Systems. Eine sehr wichtige Metrik, die Auskunft über die Dienstgüte des Systems gibt, ist die Ende-zu-Ende Latenz. Durch sie erfährt der Benutzer unmittelbar, ob das System die gestellten Anforderungen an die Datenflüsse erfüllt, weil die Auswirkungen auf der Anwendungsebene spürbar sind.

Bei heterogenen Netzwerken spielen sowohl die Metriken innerhalb von Subnetzwerken als auch die Metriken in Ende-zu-Ende Beziehungen eine entscheidende Rolle. Während der gesamtheitliche Überblick für den Benutzer Probleme des Netzwerkes darstellt, ist eine ausführlichere Betrachtung der Probleme in den Subnetzwerken durch die System-Metriken gegeben. Eine Anforderung an die Simulation ist es somit, neben den üblichen Quality-of-Service Parametern, wie der Latenzzeit, dem Jitter, der Paketverlustrate und der genutzten Bandbreite (vgl. Tanenbaum 2003), erweiterte Metriken zu definieren und sie in der Simulation verfügbar zu machen. Dabei liegt der große Vorteil der simulativen Messung in der Verfügbarkeit jeglicher Information, ohne dass die Simulation als solches beeinflusst werden muss.

3.2.4 Simulationskonfiguration und Generierung

Simulationen von Netzwerken zeichnen sich dadurch aus, dass aufgrund vieler Einstellungsparameter realistische Modelle gezeigt werden. Ein häufiges Problem stellt dabei die nicht vorhandene benutzerfreundliche Konfiguration dar, wodurch ein schneller Weg zur Evaluation neuer Topologien nicht gegeben ist. Durch unterschiedliche Busmodelle wird ein Generierungsprozess benötigt, der dafür die notwendigen Konfigurationsdateien aus einem abstrakten, einfachen Modell erstellt. Hierdurch wird gewährleistet, dass unabhängig von dem

gewählten Simulationsmodell die Topologie und die Datenflüsse beschrieben werden können. Wie bereits im Abstraktionsmodell 3.4 gezeigt, bedarf es also einer abstrakten Modellbeschreibung, die für jedes Bussystemmodell die Simulationskonfiguration generiert.

Als Codegeneratoren, welche die benötigten Konfigurationsdateien erstellen sollen, werden hier zwei Möglichkeiten vorgestellt. Bei dem ersten Ansatz kann die Modellierung eines Netzwerkes mit Hilfe einer GUI erfolgen, in der jedes Element per Drag and Drop hinzugefügt werden kann. Dazu gehören neben den Endgeräten und Switches als Knoten die Verbindungen zwischen den Knoten. Dieser Vorteil der Visualisierung erzeugt sofort einen Überblick über die genutzte Topologie, muss jedoch Parameter und Nachrichten der einzelnen Knoten in Untermenüs verstecken. Gerade bei verschiedenen Bussystemen ist die Übersichtlichkeit irgendwann nicht mehr gegeben, weil das Suchen bestimmter Eigenschaften meist nicht intuitiv erfolgt und den Anwender gerade zu Beginn überfordert.

Der zweite Ansatz beschäftigt sich mit der textuellen Beschreibung eines Netzwerkes mit Hilfe einer domänenspezifischen Sprache (DSL), in der die Endgeräte, Bussysteme und Nachrichten abbildbar sind. Bei einer DSL werden dem Anwender an jeder Stelle die Möglichkeiten der Domäne angezeigt, so dass eine einheitliche Beschreibung entsteht, die für Domänenspezialisten ohne zusätzliches Wissen bedienbar ist. Für den Einsatz einer domänenspezifischen Sprache zur Generierung von Konfigurationsdateien sprechen zahlreiche Vorteile (vgl. Voelter / Benz / Dietrich u. a. 2013), die hier im Folgenden erläutert werden:

Ein großer Vorteil besteht in der gesteigerten Effizienz bei der Nutzung einer domänenspezifischen Sprache. Wurde die DSL erstellt und bildet die gewünschte Domäne auf einem hohen Abstraktionslevel ab, wird durch die kompaktere Schreibweise sowohl die Beschreibung der Komplexität vereinfacht als auch das Verständnis über die Semantik, also die Aussage des Inhaltes, erhöht. Durch die Einbettung in eine bekannte, integrierte Entwicklungsumgebung (IDE), die toolbasierte Unterstützung für Autovervollständigung, Autokorrektur und Templates ermöglicht, ist eine schnelle Anpassung und Fehleranalyse gegeben.

Die Qualität, die mit einer DSL erreicht werden kann, hängt von dem Abstraktionsgrad der Beschreibung ab. Wenn es weniger Freiheiten bei der Beschreibung aufgrund der Entfernung unwichtiger Parameter gibt, kann die Fehlerrate gesenkt und durch generierten, strukturierten Code ein qualitativ höherwertiges Produkt geschaffen werden.

Eine eigenständige DSL-Definition ermöglicht eine optimale Anpassung der Sprachgrammatik an die Konzepte der Domäne im Fahrzeug. Durch das strikte Domänenkonzept ist es möglich, Verifikationsregeln zu definieren, die direkte Rückgabe zur erstellten Konfiguration geben. Somit können fehlende Referenzen und Parameter, die außerhalb des definierbaren Bereichs liegen, frühzeitig erkannt werden. Die semantische Syntax kann zudem zur effizienten Validation genutzt werden, weil die domänenspezifischen Charakteristika direkt sichtbar sind.

Eine gut designte domänenspezifische Sprache ist frei von spezifischen Implementationen. Das bedeutet im simulativen Bereich, dass sich die Konfigurationsdateien oder die Simulationsplattform ändern können, während die Sprache keiner Änderung bedarf. Nur der Generator muss entsprechend angepasst werden.

Als Nachteile können der verbundene Aufwand zum Designen, zur Implementierung und zur Instandhaltung gesehen werden. Gerade bei der Designentscheidung ist es schwierig, das richtige Gleichgewicht in der Sprachabstraktion zu finden (vgl. Deursen / Klint / Visser 2000). Dazu kommt, dass Benutzer in die Sprache eingewiesen werden müssen, weil sie häufig nicht intuitiv genug entworfen wurde.

Damit der Generierungsprozess auch für weitere Modelle eingesetzt werden kann, ist eine Kompatibilität zu einem etablierten marktüblichen Datenmodell wünschenswert. Für CAN-Bussysteme wurde das proprietäre CANdb Format (vgl. Vector Informatik) von Vector Informatik für die Kommunikationsbeschreibung entwickelt. Es ermöglicht eine Beschreibung der Eigenschaften von CAN-Netzwerken, der Endgeräte, Botschaften und der Signale. Wegen der fehlenden Unterstützung mehrerer Bussysteme wurde CANdb durch das offene „Fieldbus Data Exchange Format“ (FIBEX) abgelöst, das von der Association for Standardization of Automation and Measuring Systems (ASAM 2013) entwickelt und spezifiziert wurde. Als XML-basiertes Datenformat, das sämtliche relevanten Bussysteme im Automotive Kontext und Ethernet unterstützt, ist es als Datenformat für heterogene Netzwerke geeignet, in denen jedes einzelne Bussystem für sich beschrieben werden kann. Bei Bartols (Bartols 2014) wird eine Erweiterung der FIBEX-Spezifikation für Echtzeit-Ethernet Netzwerke, die durch Time-Triggered Ethernet repräsentiert werden, beschrieben. Damit ist es möglich, die für Time-Triggered Ethernet spezifizierten, charakteristischen Merkmale in FIBEX festzuhalten.

Für Time-Triggered Netzwerke ist ein Schedule notwendig, der für jede Nachricht festlegt, wann sie innerhalb des Zyklus gesendet werden kann. Dieser wird statisch festgelegt, unterscheidet sich aber bei jeder Topologieänderung. Wegen der großen Anzahl der Nachrichten in einem Automobilnetzwerk kann ein Schedule, der die Zeitanforderungen jeglicher Nachrichten beachtet, nicht ohne Optimierungsalgorithmen gelöst werden. Da es unterschiedliche Ansätze für solche Algorithmen gibt und sich diese Arbeit mit den Simulationsanforderungen beschäftigt, soll eine Berechnung des Schedules an einer externen Stelle stattfinden und über das FIBEX Format ausgetauscht werden. Für die Evaluierungsphase ist es zudem notwendig, neben der Simulationskonfiguration Anforderungen für Metriken zu definieren, die bei der Auswertung Grenzwerte darstellen. Damit ist es möglich, für die Datenflüsse direkt bei der Planung des Netzwerkes Ziele zu definieren.

4 Konzept und Architektur

In diesem Kapitel werden die Konzepte, die sich hinter der Analyse heterogener Fahrzeugnetzwerke verbergen und eine Vorgehensweise für die zukünftige Evaluierung vorgestellt. Im ersten Schritt wird ein Datendesign erläutert, das ein realistisches Datenmodell hinreichend genau abbildet und die Komplexität zukünftiger Anforderungen widerspiegelt. Anschließend findet eine Typisierung der Daten statt, wodurch eine unabhängigere Sichtweise auf das Netzwerk getroffen werden kann. Dieses soll als späteres Anwendungsbeispiel zur Analyse des Evaluierungsprozesses dienen. Im nächsten Schritt werden die notwendigen Komponenten der Simulationsumgebung charakterisiert. Dazu gehören vor allem Lastgeneratoren, die unterschiedliche Verkehrsmodelle in der Simulation zulassen. Außerdem wird ein Konfigurierungsvorgang vorgestellt, der eine abstrakte Beschreibung komplexer Netzwerke zulässt. Der letzte Schritt beschreibt die Analyse heterogener Fahrzeugnetzwerke in der Simulation. Hierbei werden verschiedene Metriken gezeigt und mögliche Messmethoden erläutert.

4.1 Konzept zur vereinheitlichten Analyse

Die Relevanz eines vereinheitlichten Konzeptes zur Analyse von Fahrzeugnetzwerken in Bezug auf eine gemeinsame Datenbasis wird immer relevanter. Die verschiedenen Simulationsstudien nutzen zur Evaluierung entweder neue Analysedaten oder greifen einen fremden Ansatz auf, wodurch die Vergleichbarkeit nur mit diesem einem zulässig ist. Benchmarks, wie der PSA- und SAE-Benchmark, greifen den Gedanken der vereinheitlichten Analyse zwar auf, sind allerdings heutige Funktionen nicht angepasst. Daher wird im folgenden Kapitel das Konzept zur Entwicklung eines einheitlichen Standards beschrieben.

4.1.1 Ausarbeitung eines Datenmodells

In der Anforderungsanalyse eines Datenmodells für simulationsbasierte Fahrzeugnetzwerke in Kapitel 3.1.2 wurde der Bedarf einer Kommunikationsmatrix, die den Datenfluss eines Netzwerkes beschreibt, erkennbar gemacht. Die Kommunikationsmatrix muss sämtliche Sender-Empfänger-Beziehungen darstellen, damit alle kommunikationsrelevanten Informationen vorhanden sind. Eine mögliche Matrix kann aus folgenden Feldern bestehen:

Beschreibung: Dieses Feld steht optional für den Nutzer zur Verfügung, damit die Nachrichten später einer Aufgabe zugeordnet werden können.

Größe (Byte): Die Größe des Payloads einer Nachricht.

Periode (ms): Die Zeit zwischen zwei aufeinanderfolgenden Nachrichten.

Ereignisgesteuertes Auftreten (ms): Wird die Zeit nicht als Periode angegeben, ist stattdessen ein ereignisgesteuertes Auftreten anzuzeigen. Die genannte Zeit kann dann entweder als periodisch oder als nicht relevant eingestuft werden. Beides hat unterschiedliche Auswirkung auf die Auslastung des Netzwerkes. Eine Worst-Case Interpretation als Periode ist in der Simulation sinnvoll.

Priorität: Die Relevanz einer Nachricht kann über die Priorität abgebildet werden. Dabei ist entweder eine absolute oder eine relative Priorität zu wählen. Letztere ermöglicht eine spätere Berechnung der absoluten Priorität und kann bei der Erstellung eines Datenmodells sinnvoll sein.

Latenz (ms): Die Latenz beschreibt die maximal zulässige Verzögerung, die eine Nachricht bei Durchquerung des Netzwerkes erreichen darf und ist als Anforderung zu verstehen.

Jitter (ms): Der Jitter ist ebenfalls als Anforderung zu sehen und bezieht sich auf den maximalen Jitter der Nachrichten. Bei einem niedrigen Jitter muss die Latenz der Nachrichten die gleiche Größenordnung besitzen.

Klassifizierung: Mit der Klassifizierung wird eine Einordnung der Nachricht anhand der Größe, Periode, Latenz und Jitter in eine Klasse vorgenommen und kann eine Betrachtung des Netzwerkes vereinfachen. Der nachfolgende Abschnitt 4.1.2 beschreibt das Konzept hinter der Klassifizierung.

Domänenzugehörigkeit: Optionaler Parameter, der die spätere Zuordnung einer Nachricht zu einer Domäne erleichtert. Dabei kann eine Backbonezugehörigkeit ebenfalls als Domäne angegeben werden.

Sender: Der Sender beschreibt das Endgerät, über das die Nachricht verschickt wird.

Empfänger: In der Kommunikationsmatrix können die Empfänger entweder über eine Liste oder ein Feld für jeden Empfänger angegeben werden.

Vereinfachte Formen, bei denen beispielsweise die Anforderungen weggelassen werden, sind möglich, jedoch nicht erstrebenswert. Nur mit einer vollständigen Matrix kann zudem eine Abbildung der Domänen in einem heterogenen Fahrzeugnetzwerk erfolgen. Dabei ist eine Aufteilung derzeitig in fünf Domänen üblich. Diese unterteilen sich in nicht sicherheitskritische und sicherheitsrelevante Domänen auf. Während zu den erstgenannten Multimedia- und Komfortdaten gehören, teilen sich die sicherheitsrelevanten in die Domänen der Passagiersicherheit, Antriebs/Fahrwerk und Wartung auf (Broy / Krüger / Pretschner u. a. 2007).

4.1.2 Klassifizierung der Kommunikationsnachrichten

Wie in Kapitel 3.1.3 beschrieben, wurden Bussysteme anhand ihrer spezifischen Merkmale und ihrer Domäne, in der sie eingesetzt werden können, in verschiedene Klassen eingeteilt. Bei einer heterogenen Ansicht ist die Zuordnung eines Gesamtnetzwerkes in eine bestimmte Klasse nicht mehr möglich. Viele Nachrichten im Automobil besitzen allerdings ähnliche Eigenschaften und eine gleichartige Struktur in den charakteristischen Merkmalen wie der Paketgröße, der benötigten Bandbreite, der Ende-zu-Ende Verzögerung und deren maximaler Varianz. Bei der Analyse dieser Eigenschaften können wie bei den Bussystemen Klassen gefunden werden. Anhand der Nachrichtenanzahl in der jeweiligen Klasse kann eine Einordnung des gesamten Netzwerkes erfolgen.

In dem Paper „A Novel Network Architecture for In-Vehicle Audio and Video Communication“ (Rahmani / Hillebrand / Hintermaier u. a. 2007) werden die Anforderungen an vier Nachrichtenklassen gestellt, die sich im Automobil finden lassen. Jede Klasse wird anhand der typischen Paketgröße, der Fehlerrate und der Cycle-Time, also der Zeit, in der eine Antwort das System wieder erreichen kann, eingeordnet. Die erste Klasse folgt der Bezeichnung *Hard real-time*:

- Cycle time: 2-20 ms
- Paketgröße: 2-32 Byte
- Fehlerrate: extrem niedrig

Diese Werte wurden anhand der CAN und FlexRay Nachrichten in einem Fahrzeug analysiert und sind allen sicherheitskritischen Applikationen und Kontrollsystemen zuzuordnen, die hohe Anforderungen an die Dienstgüte des Netzwerkes stellen. Diese zeichnen sich mit kleinen Paketgrößen und einer geringen Übertragungsverzögerung aus, bei dem keine Fehler in der Übertragung tolerierbar sind.

Die zweite Klasse folgt der Bezeichnung *Soft real-time*:

- Cycle time: 10-100 ms
- Paketgröße: 2-32 Byte
- Fehlerrate: extrem niedrig

Zur Analyse der Nachrichten wurden derzeitige Soft-Realtime-Implementationen analysiert. Der Unterschied zur vorigen Klasse besteht in einer zulässig erhöhten Übertragungsverzögerung, die aber einer gewissen Qualitätsgüte unterliegt. Auch hier sind es meist kleine Paketgrößen, die Kontrollnachrichten des Fahrzeugs beschreiben.

Die dritte Klasse ist *Multimedia*:

- Transmission Time: 5-20 ms (Buffering auf der Empfängerseite)
- Paketgröße: 200-1500 Byte
- Fehlerrate: sehr niedrig

Zur Klasse Multimedia gehören alle Streamingsysteme, die im Fahrzeug relevant sind. Dazu sollen sowohl Audio- und Videoübertragungen zu Entertainmentzwecken als auch sicherheitskritische Fahrerassistenzsysteme mit Kamerasystemen zählen. Durch das Streamverhalten wird die Übertragungszeit nun als Transmission Time bezeichnet und sehr gering eingestuft. Der große Unterschied zur ersten Klasse besteht in den Paketgrößen, die wesentlich mehr Daten zu transportieren haben. Zudem ist die Fehlerrate nicht mehr als extrem niedrig einzustufen, sondern ein wenig geringer.

Die vierte und letzte Klasse heißt *Best-effort*. Hierbei handelt es sich um keine sicherheitsrelevanten Daten für unmittelbares Feedback. Daher wird die Dienstgüte nicht weiter beschrieben. Das heißt, dass es keine weiteren Anforderungen an die Fehlerrate und die Übertragungsverzögerung gibt.

In dem neueren Paper „Design and Realization of an IP-based In-car Network Architecture“ (Steffen / Bogenberger / Hillebrand u. a. 2010) werden die Nachrichten ebenfalls in vier Kategorien unterschieden. Die erste Klasse *Real-time control data* beschreibt wie in dem vorangegangenen Paper die systemkritischen Kontrolldaten, die hohe Anforderungen an das Ende-zu-Ende Delay mit bis zu 2.5 ms besitzen. Die zweite Kategorie unterscheidet sich jedoch. Hierbei handelt es sich nicht um die Soft-real time Kontrolldaten, die ein Ende-zu-Ende Delay von 50 ms verlangen, sondern um *Real-time audio and video data*. Dazu zählen alle sicherheitsrelevanten Kamerasysteme für Fahrerassistenzfunktionen und zeitkritische Anwendungen wie Voice over IP. Dabei wird mit sehr großen Paketen und einem maximalen Ende-zu-Ende Delay von 33 ms eine hohe Anforderung an das Netzwerk gestellt. Die dritte Klasse findet auch hier die Bezeichnung *Multimedia*, unterscheidet sich jedoch in den Zeitanforderungen, weil die sicherheitskritischen Anwendungen in Klasse 2 verschoben wurden, und es sich nun um klassische Bufferinganwendungen handelt. Die letzte Klasse beschreibt auch hier den *Best-effort* Datenverkehr.

Aus den Anforderungen, die sich aus dem in Kapitel 4.1.1 ergeben, müssen Nachrichten ausschlaggebende Kriterien aufweisen. Dazu gehören die Periode einer Nachricht, die benötigte Bandbreite, die Latenz und die Varianz der Latenz. Diese Eigenschaften werden jedoch in den genannten Quellen nur teilweise erläutert. In der erstgenannten wurden drei Kriterien aufgeführt, die jedoch zum Teil eine unterschiedliche Bezeichnung aufweisen. Daher soll nun eine geeignete einheitliche Einteilung in Kategorien erfolgen.

Bei der Analyse der Daten aus dem vorigen Kapitel fällt auf, dass sich zwei Gruppen von Daten beschreiben lassen. Die einen benötigen eine Bandbreite von unter zwei kB/s und

beanspruchen somit nur einen sehr kleinen Teil der vorhandenen Bandbreite, während die anderen Nachrichten auf eine erheblich größere Bandbreite im Netzwerk angewiesen sind. Dabei verursachen die Kameras für die Fahrerassistenzfunktionen mit über 3200 kB/s beziehungsweise 25 Mbps mit einem unkomprimierten Datenfluss das höchste Datenvolumen.

Klasse	1	2	3	4
Periode (ms)	10	1000	10	25
Bandbreite (kB/s)	4000	2	2	4000
Latenz (ms)	10	10	50	100
Jitter (ms)	< 1	< 1	< 5	-

Abbildung 4.1: *Die Nachrichten lassen sich in vier verschiedene Klassen einteilen: Klasse 1: Regelungsdaten Fahrerassistenzfunktionen. Klasse 2: Sicherheitskritische Regelprozesse. Klasse 3: Ein-/Ausschaltvorgänge. Klasse 4: Pufferungsfähige Daten.*

Der zweite Faktor stellt die Periode einer Nachricht dar. Diese gibt an, wie häufig eine Nachricht versendet werden soll. Da es sich bei der Analyse eines Netzwerkes um Worst-Case Szenarien handelt, werden Daten, die sporadisch auftreten, ebenfalls periodisch gesendet. Hierdurch werden höhere Anforderungen an das Netzwerk erreicht. Mit der Periode und der Bandbreite ist eine Berechnung der Paketgröße möglich.

Der dritte Faktor, der sich erheblich auf die Einordnung einer Nachricht auswirkt, ist die Latenz. In den Quellen wurden zum Teil die Spezifikationen aus analysierten Implementationen entworfen. Allerdings sollte sich die Implementation an der Spezifikation orientieren. Daher sollte der Grad der Dienstgüte anhand der benötigten Funktionen festgelegt werden. Für den sicherheitskritischen Verkehr darf die Verzögerung maximal 10 ms betragen. Die Ein- und Ausschaltvorgänge, die eine unmittelbare Reaktion auf ein Ereignis erwarten, fallen in die Kategorie der weichen Echtzeit und werden mit einer Verzögerung von 50 ms angegeben. Für pufferungsfähige Daten ist die Verzögerung nicht so ausschlaggebend. Diese erhalten jedoch ebenfalls eine maximale Verzögerung von 100 ms, um das Ziel eines nicht zu überlastenden, heterogenen Netzwerkes zu verdeutlichen.

Tabelle 4.1 zeigt Nachrichten aus dem Datenmodell, die sich in den jeweiligen Klassen befinden. In der höchsten Klasse liegen demnach die Kameras für die Fahrerassistenzfunktionen,

	Size	Period	Bandwidth	Latency	Jitter	Sender	Receiver
Kl. 1:	1500 Byte	0.46 ms	3184.44 kB/s	10	1	Kamera	HU
Kl. 2:	16 Byte	10 ms	1.56 kB/s	10	1	ECM	Getriebe
Kl. 3:	5 Byte	100 ms	0.05 kB/s	50	5	Sensor	HU
Kl. 4:	100 Byte	4 ms	24.4 kB/s	100	10	Telefon	HU

Tabelle 4.1: *Beispiele von Nachrichteneinteilungen auf die vier verschiedenen Klassen.*

die mit 1500 Byte und einer Periode von 0.46 ms eine Last von 25 Mbps erzeugen. Die maximale Latenz soll bei einer Nachricht 10 ms mit einem maximalen Jitter von 1 ms betragen. In der zweiten Klasse ist eine Nachricht von dem Engine control module (ECM) an das Getriebe. Sie wird alle 10 ms geschickt und besitzt nur einen Payload von 16 Byte. Die Ende-zu-Ende Verzögerung ist genau wie bei der Klasse 1. In der dritten Klasse befindet sich eine Nachricht von einem Sensor zu der Head-up-Unit (HU). Diese wird sehr selten ausgelöst und hat einen sehr geringen Payload. Die zeitlichen Anforderungen bewegen sich mit 50 ms Latenz im Soft real-time Bereich. In die letzte Klasse wird eine Nachricht mit 100 Byte eingeordnet. Da es sich um einen Audiostream handelt, besitzt sie keine sicherheitskritischen Anforderungen und kann als Best-Effort-Traffic übertragen werden.

Nachfolgend werden den Klassen Namen zugeordnet, die herausstellen sollen, um was für Nachrichtentypen es sich in der Gruppe handelt:

- **1. Regelungsdaten Fahrerassistenzfunktionen (Hard real-time):** Fahrerassistenzfunktionen wie Kamerasysteme mit zeitkritischer Anforderung und hoher Bandbreite.
- **2. Sicherheitskritische Steuerdaten (Hard real-time):** Steuerdaten und Kommunikation für die Sicherheitselektronik mit zeitkritischer Anforderung und einer geringen benötigten Bandbreite.
- **3. Ein- und Ausschaltvorgänge (Soft real-time):** Nach der Auslösung wird ein direktes Feedback erwartet.
- **4. Pufferungsfähige Daten (Best-effort):** Bei diesen Daten ist die Latenz und die Paketverlustrate vernachlässigbar.

Mit dieser Aufteilung kann aus einem Datenmodell eine abstrakte Nachricht aus einer Klasse für jede konkrete Nachricht gefunden werden.

Für jedes Netzwerk gibt es anschließend eine Nachrichtenanzahl in jeder Klasse, so dass die konkreten Nachrichten nicht mehr benötigt werden. Hieraus lassen sich Abstraktionsstufen für Endgeräte ableiten, die im folgenden Kapitel erläutert werden.

4.1.3 Abstraktionsstufen von Endgeräten

Endgeräte lassen sich genau wie einzelne Nachrichten kategorisieren. Durch die Einteilung in spezielle Klassen ist es möglich, skalierbare Lösungen zu finden, die tatsächliche Eigenschaften der Daten abstrahiert darstellen. Eine skalierbare Variante zu finden ist dabei wichtig, um zukünftige Modellierungsarten von Netzwerken mit abstrakten Designalternativen zu testen.

Stufe 1 - Abstrahierung der Namen

Die erste Stufe beschäftigt sich mit der Vereinheitlichung der Endgerätenamen. Wegen der Benennung der Endgeräte mit ECU-1 bis ECU-n sind Topologieänderungen, die nicht in Verbindung zu den Daten stehen, durchführbar. Aufgrund der Sender-Empfänger-Eigenschaften jeder Nachricht geht dabei nur die Domänenzugehörigkeit des Endgerätes, nicht jedoch der Datenfluss verloren.

Stufe 2 - Klassifizierte Nachrichten

Die zweite Stufe besteht darin, alle vorhandenen Nachrichten des Netzwerkes mit klassifizierten Nachrichten aus dem Kapitel 4.1.2 zu ersetzen. Über die K-Matrix eines Datenmodells, die definiert, welche Nachrichten in einer 1:n Beziehung zwischen den Endgeräten verschickt werden, wird jeder Nachricht eine abstrakte hinzugefügt. Anschließend besitzt jede ECU nur noch Nachrichten aus den Klassen, die mit erhöhten Spezifikationen zukünftige Anforderungen besser bedienen können. Zudem ist es nun möglich, die Endgeräte mit klassifizierten Nachrichten einer eigenen Klasse zuzuordnen. Dabei fällt ein Endgerät immer in die Klasse mit den Nachrichten der höchsten Priorität.

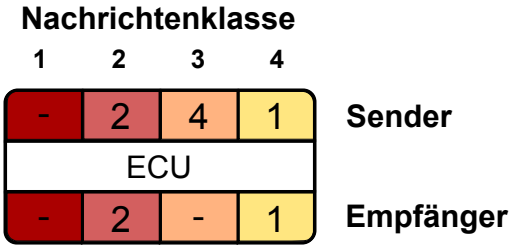

Abbildung 4.2: *Nachrichtenzahlen in den Klassen eines Endgerätes mit Sender- und Empfängerseite. Hier wird ein Endgerät der Klasse 2 abgebildet.*

Die Prioritäten der Nachrichtenklassen gestalten sich wie folgt: Klasse 1 besitzt die höchste Bandbreite, wobei die Übertragung der Daten mit einer sehr geringen, konstanten Latenz wichtig ist. Klasse 2 zeichnet sich durch die ebenfalls sehr geringe, konstante Latenz aus. Allerdings weisen die sicherheitskritischen Steuerdaten meist eine kleine Paketgröße auf. In der dritten Klasse entsprechen die Latenzanforderungen nur noch weichen Echtzeitanforderungen, während für die Latenz bei der letzten, vierten Klasse Anforderungen aus dem Best-Effort Bereich genügen. Besitzt ein Endgerät Nachrichten aus der zweiten und dritten Klasse, findet eine Einordnung in die Endgeräteklasse 2 statt.

Stufe 3 - Klassifizierte Endgeräte

Bei der dritten Stufe ist die Gleichstellung aller Endgeräte das Ziel. Jede ECU wird anschließend aus einem standardisierten Endgerät repräsentiert, wodurch die Skalierbarkeit eines Fahrzeugnetzwerkes durch fest ermittelte und definierte Werte gegeben ist. Die Einordnung folgt dabei den Nachrichtentypen der gesendeten und empfangenen Nachrichten.

Zur Generierung einer Standard-ECU können unterschiedliche Ansätze verfolgt werden. Um den maximalen Datendurchsatz zu erhöhen, kann die maximale Anzahl der Nachrichten gewählt werden. Dabei erhalten alle Endgeräte die Anzahl der Nachrichten von der ECU mit den meisten Nachrichten in der jeweiligen Nachrichtenklasse. Anschließend liegen Endgeräte ohne Spitzen vor. Das gesamte Netzwerk muss imstande sein, die Raten zu übertragen. Somit stellt dieses einen Worst-Case-Szenario Ansatz dar.

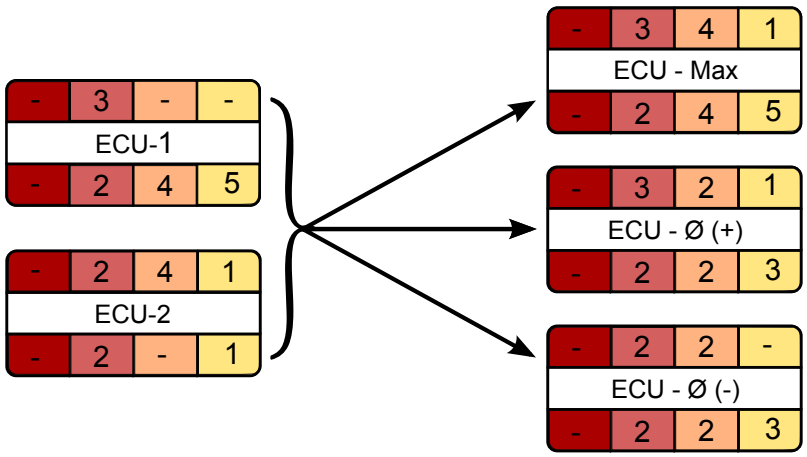

Abbildung 4.3: *Einteilung von Endgeräten der Klasse 2 in eine Standard-ECU. Die drei Ansätze sind: Maximale Nachrichtenzahl, aufgerundeter und abgerundeter Durchschnitt.*

Ein Ansatz, bei dem ebenfalls keine Lastspitzen mehr verursacht werden und eine gleichmäßigere Verteilung der Last auf die Geräte stattfindet, ist über einen Durchschnittsansatz lösbar. Jeder ECU einer Klasse werden dabei wieder die gleiche Anzahl an Nachrichten für die einzelnen Nachrichtenklassen zugeordnet, so dass im Idealfall genauso viele Nachrichten wie im ursprünglichen Netzwerk vorhanden sind. Bei einer ungeraden Anzahl innerhalb einer Nachrichtenklasse muss diese entweder aufgerundet oder abgerundet werden. In jedem Fall verlieren einige ECUs ihre Nachrichten, und andere gewinnen neue hinzu. Abbildung 4.3 fasst die Klassifizierung zusammen.

Das Problem beider Ansätzen ist, dass die Nachrichten ihre definierten Empfänger verlieren. Durch die Verschiebung der Nachrichten ist nicht mehr festgelegt, woher die Nachrichten ursprünglich stammen. Daher wird in dieser Arbeit der Fokus auf die ersten beiden Stufen gelegt, ohne einen Algorithmus zu nennen, der die Nachrichten an die Empfänger verteilt.

4.1.4 Topologievariationen eines Echtzeit-Ethernet-Backbones

Nachdem im letzten Kapitel eine Klasseneinteilung der Nachrichten vorgenommen wurde, werden an dieser Stelle mögliche Topologien vorgestellt, die als Varianten im Automobil mit einem Ethernetbackbone in Frage kommen. Dazu muss die klassische, hierarchische Domänenstruktur aus Effizienz- und Kostengründen verändert werden. Im Anforderungskapitel 3.1.4 wurde dazu bereits eine klassische Netzwerktopologie mit einem zentralen Gateway behandelt und geklärt, warum dieses Modell zukünftigen Anforderungen nicht genügt. Viele Endgeräte sitzen an Bussystemen, die durch das Ethernetbackbone ersetzt werden können. Das heißt, dass einige oder alle Endgeräte des Bussystems direkt an das Backbone angeschlossen werden. Bei der Frage nach einer geeigneten Backbonetopologie gibt es verschiedene Ansätze, die verfolgt werden können. Dazu zählen der kostenoptimierte-, die performanceoptimierte-, die zentrale-, die Daisy-chain- und der hierarchische Topologieansatz. Als Switches im Backbone der Zukunft werden zur Verdeutlichung Time-Triggered Ethernetswitches (TTE-Switch) und CAN-Busse, welche die verschiedenen Domänen abbilden, genutzt.

Zentralisierte Topologie

Eine zentralisierte Topologie eignet sich zur Messung der direkten Links der einzelnen Endgeräte. Jedes Endgerät wird dabei direkt und Bussysteme über Gateways an einen Knoten verbunden. Durch die entstandene Sterntopologie ist eine Prüfung der Linkauslastung möglich und kann –in einer Simulation evaluiert– wichtige Erkenntnisse zu den Datenübertragungsraten der Systeme liefern. Aufgrund der Anforderung, die Datenübertragung von mehr als 70 Geräten zu steuern, eignet sich die Topologie jedoch nicht für den realen Einsatz im Fahrzeug.

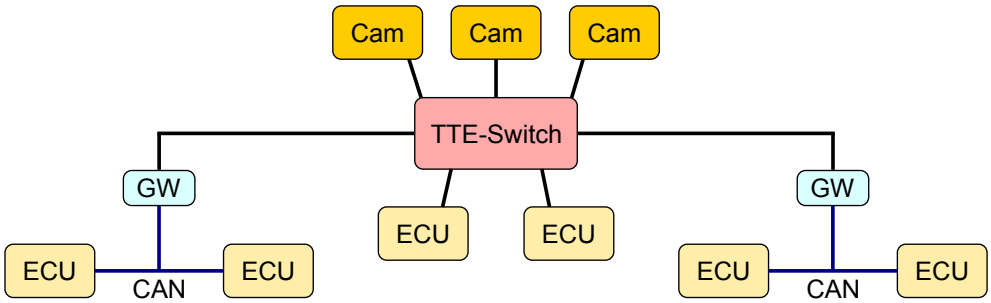

Abbildung 4.4: *Heterogenes Netzwerk mit zentraler Stern-Topologie.*

Kostenoptimierte Topologie

Bei der kostenoptimierten Topologie wird das Ziel durch geringe Kabelkosten, geringe Leistungsaufnahmen und wenig benötigten Verkabelungsplatz definiert (vgl. Müller-Rathgeber / Michel 2009). Das Ziel der Leistungsaufnahme kann dabei durch wenige Knotenpunkte realisiert werden. Bei Lim / Krebs / Volker u.a. (Lim / Krebs / Volker u. a. 2011) wird eine Double-Star Topologie erläutert, in der zwei Switches die Endgeräte nach hoher Bandbreite und Daten, die eine Echtzeitübertragung benötigen, unterteilen. In zukünftigen Netzwerken, die eine hohe Bandbreite bei Echtzeitübertragungen erfordern, kann diese Unterteilung nicht erfolgen. Allerdings ist eine kostenoptimierte Topologie mit der Aufteilung auf wenig Switches für ein Backbone durchaus sinnvoll. Somit können bisherige Bussysteme über Gateways direkt an den nächsten Switch angeschlossen werden und zukünftig hinzugefügte Endgeräte den nächstliegenden Switch erreichen. Abbildung 4.5 zeigt eine vereinfachte Double-star Topologie mit Gateways zu den CAN-Bussystemen und Kameras, die direkt an das Backbone angeschlossen sind. Durch das Hinzufügen weiterer Switches kann das System gut skalieren und an benötigten Stellen entlastet werden.

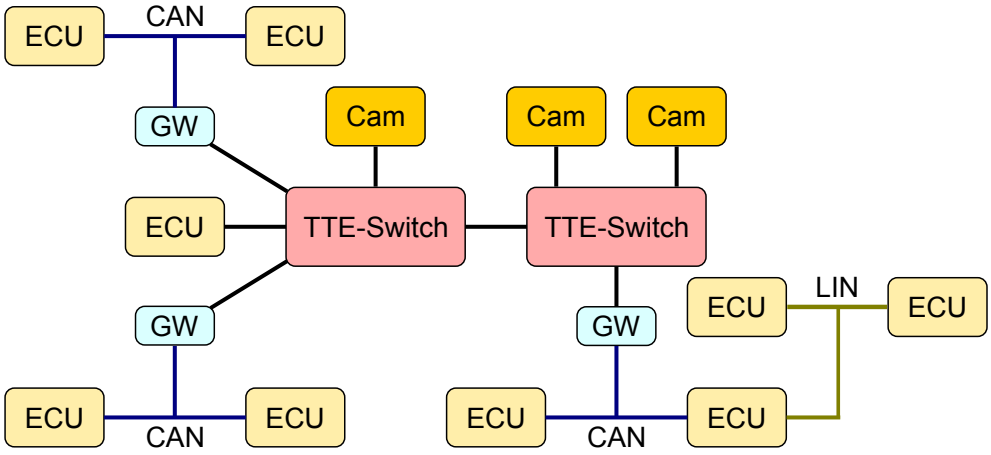

Abbildung 4.5: *Heterogenes Netzwerk mit Double-star als kostenoptimierte Topologie. Bisherige Bussysteme werden über Gateways an nahe liegende Switches verbunden.*

Bei der kostenoptimierten Topologie muss das Netzwerk bezüglich der Dienstgüte beurteilt werden, weil die Last der einzelnen Switches besonders groß ist. Zudem wird das Netzwerk einer konstant hohen benötigten Bandbreite auf der Verbindung zwischen den beiden Switches ausgesetzt, weil hinzugefügte Endgeräte die Verbindung zusätzlich belasten, und es nicht auf verschiedene Teilstrecken ausgelagert werden kann.

Performanceoptimierte Topologie

Bei einer performanceoptimierten Topologie wird das Netzwerk bezüglich der Dienstgüte beurteilt und versucht, die einzelnen Services so in dem Netzwerk zu verteilen, dass sie sich nicht gegenseitig beeinträchtigen. Dazu wird anhand durchsatz- und latenzanalysierender Methoden entschieden, ob es sich um eine geeignete Struktur handelt. Die Anzahl der Switches und Länge der Verbindungen ist bei diesem Ansatz vernachlässigbar. Wichtiger ist es, einzelne Endgeräte zu bewerten, die einen großen Einfluss auf den Datendurchsatz haben und diese über eine minimale Anzahl an Switches kommunizieren zu lassen. Mit Hilfe eines vermaschten Netzes können zudem Verbindungen entlastet werden.

Daisy-chain Topologie

Eine Topologie, mit der größere Strecken im Fahrzeug zurückgelegt werden können und die sich mit kleineren Switchdesigns auseinandersetzt, ist die Daisy-chain-Topologie. Dabei

bilden im Sonderfall Drei-Port-Switches eine Kette und verbinden jeweils ein Endgerät oder Gateway zu einem anderen Bussystem. Diese Topologie erlaubt das Zurücklegen größerer Entfernungen und wird häufig zur Aufteilung von Endgeräten auf mehrere Switches genutzt. Bei dem Domänendesign im Fahrzeug kann beispielsweise jedes Bussystem über ein Gateway einem Switch zugeordnet werden.

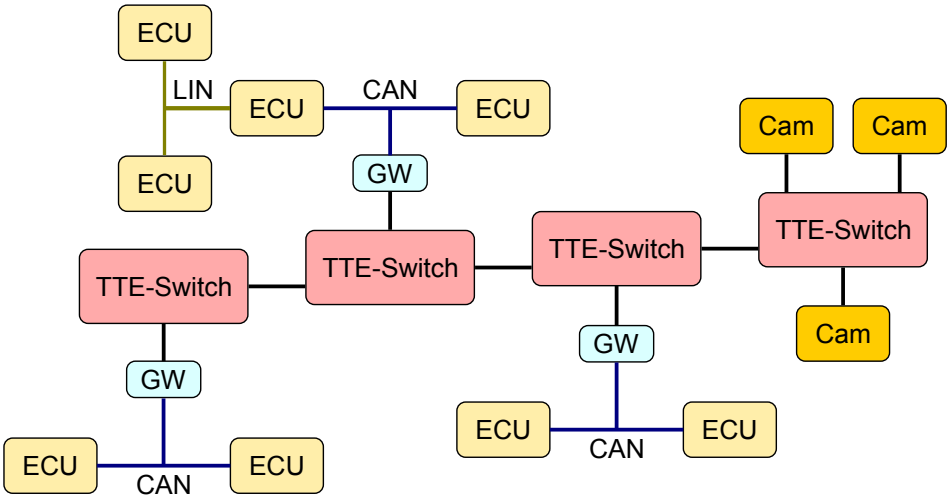

Abbildung 4.6: *Heterogenes Netzwerk mit Daisy-chain Topologie. An jedem Switch können je nach Topologiewunsch eine Domäne oder lokale Endgeräte angeschlossen sein.*

Das Problem einer Daisy-chain besteht aus der erhöhten Last für die Switches mit mehreren Verbindungen. Jeder Switch, der zwischen zwei anderen Switches liegt, muss die Kommunikationsnachrichten weiterleiten, die zwischen zwei nicht direkt verbundenen Endgeräten ausgetauscht werden. Dadurch entstehen höhere Warteschlangen. Daten, die keinen Echtzeitschedule besitzen, erhalten eine erhöhte Verzögerung.

Hierarchische Topologie

Die hierarchische Topologie ist ähnlich dem derzeitigen Aufbau des Designs nach den Domänen. Dabei wird für jede Domäne ein Switch bereitgestellt, an den sowohl der ehemalige Bus als auch die Endgeräte, die direkt an den Switch ausgelagert werden, angeschlossen sind. Die Kommunikation läuft anschließend über eine oder mehrere Hierarchieebenen ab und weist eine baumartige Struktur auf. Gerade am Anfang der Evaluierung neuer Backbonesysteme werden die ursprünglichen Domänen nicht verändert, sondern es wird sich an dem bisher bekannten Verhalten orientiert, das aus einem zentralisierten Gateway bestand. Durch das

Hinzufügen mehrerer Gateways und Switches für jedes Bussystem entsteht ein skalierbarer Ansatz, bei dem Endgeräte direkt einer Domäne zugeordnet werden können. Eine sukzessive Herangehensweise mit der Überführung von den Endgeräten zu dem Backbone ist möglich. Dieses Vorgehen unterstützt den zukünftigen Wechsel eines heterogenen zu einem flachen Netzwerk mit nur einem Bussystem.

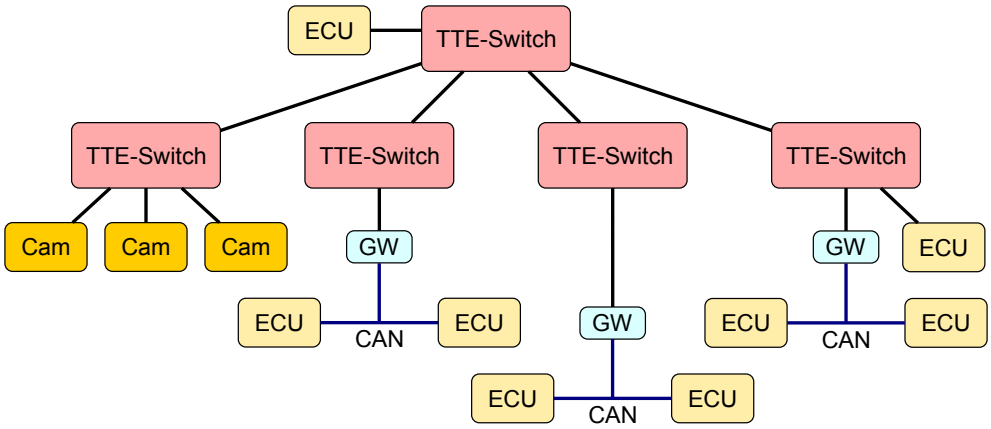

Abbildung 4.7: *Heterogenes Netzwerk mit hierarchischer Topologie nach Domänen.*

Abbildung 4.7 zeigt vier unterschiedliche Domänen, für die jeweils ein Switch hinzugefügt wurde. Zwischen den verschiedenen Domänen wird über einen zentralen Knotenpunkt kommuniziert, wobei die Protokollübersetzung durch den Backboneansatz auf mehrere Gateways aufgeteilt wurde.

4.2 Architektur und Komponenten

4.2.1 Simulationskomponenten des Modells

Dieses Kapitel befasst sich mit den Komponenten, die für eine Analyse von heterogenen Fahrzeugnetzwerken in der Simulation relevant sind. Die Anforderungsanalyse aus Kapitel 3.2.1 hat gezeigt, dass dafür derzeitig eingesetzte Bussysteme, ein Backbone, Endgeräte und Gateways, die zwischen den Bussystemen übersetzen, benötigt werden. Aufgrund der derzeitigen Topologie im Automobil hat sich zudem herausgestellt, dass es eine Einteilung in Quell- und Zielbussystem gibt, die über das Backbone miteinander verbunden werden. Jeder Nachricht muss dementsprechend die Information, wie sie sich in dem jeweiligen Subnetz verhalten soll, zugeteilt sein. Um diese Heterogenität abzubilden, bedarf es mindestens eines modellierten Bussystems und eines Backbonemodells. Mit dem im Fahrzeug am häufigsten

eingesetzten Bussystem, dem CAN-Bus, liegt die Verwendung eines CAN-Busmodells als Quell- und Zielbussystem nahe. Als Backbonesystem wird das Time-Triggered Ethernetprotokoll genutzt, das zukünftigen Anforderungen der verschiedenen Domänen genügt und von dem ein Modell in der Simulationsumgebung OMNeT++ existiert (vgl. 2.4). Nachfolgend werden dementsprechend die Konzepte der TTEthernetkomponenten, des CAN-Busses, der Lastgeneratoren und dem TTE-CAN-Gateway erläutert.

TTEthernetkomponenten

Das Time-Triggered Ethernetmodell für OMNeT++ ist aus mehreren Arbeiten entstanden, in denen die einzelnen Nachrichtenklassen modelliert und validiert wurden. Durch die zahlreichen Vorarbeiten ist es möglich, Simulationsstudien mit einem echtzeitfähigen Ethernetbackbone durchzuführen. Die Aufteilung der Komponenten für Time-Triggered-Nachrichten im Endsystem wurde von Hermand Dieumo in Teilen seines Masterstudienganges realisiert (vgl. Dieumo Kenfack 2010). Der TTE-Switch wurde von Till Steinbach während der Anfertigung seiner Masterarbeit erstellt (vgl. Steinbach 2011). Beide Komponenten wurden anschließend mit der Rate-Constrained-Traffic-Klasse erweitert, die gemäß der ARINC-664 Spezifikation Nachrichten verschickt, weiterleitet und empfängt (vgl. Kempf 2011). Für das Time-Triggered Ethernetprotokoll ist die Uhrensynchronisation der einzelnen Systeme unerlässlich. Wegen der auftretenden Clock-drift der einzelnen Systeme weicht die Zeit über einen gewissen Zeitraum ab. In der Simulation fand zu diesem Zeitpunkt die Synchronisierung zu einer globalen Zeit, die innerhalb der Simulationsumgebung abrufbar ist, statt. Erst mit der Analyse und der Implementierung von Lazar Todorov wurden die notwendigen Konzepte zur Synchronisation gemäß der AS6802 Spezifikation (Todorov / Steinbach / Korf u. a. 2013) und damit die Modellierung des TTEthernet-Backbones abgeschlossen. Aufgrund der Notwendigkeit auch andere Protokolle wie AVB in das Modell mit einfließen zu lassen und somit Backbonealternativen zu erschließen, wurden einige Veränderungen vorgenommen, sodass die Weiterentwicklung des Modells eine Aufteilung der einzelnen Komponenten eines TTE-Backbones im Gegensatz zu den in den einzelnen Arbeiten beschriebenen Modellen erforderte. Anhand der Veränderung in Abbildung 4.8 wird verdeutlicht, wie das Modell mit zusätzlichen Protokollen (hier blau dargestellt) erweiterbar ist.

Das Modell besteht aus dem Standard-Ethernet-Protokollstack, der um die Time-Triggered Funktionalität erweitert wurde. Für jede der drei Nachrichtenklassen von TTEthernet und jedem davon unabhängigen Protokoll existieren mehrere Buffer für die Nachrichten mit unterschiedlichen Virtual Links. Diese sind in Buffer für eintreffende Nachrichten und für solche, die versendet werden sollen, unterteilt. Eintreffende Nachrichten werden dabei in dem *inControl*-Modul bewertet und in einem Buffer eingeordnet. Eine Applikation kann diese Nachricht dann aus dem Buffer abholen und weiter verarbeiten. Um Nachrichten zu versenden, wird

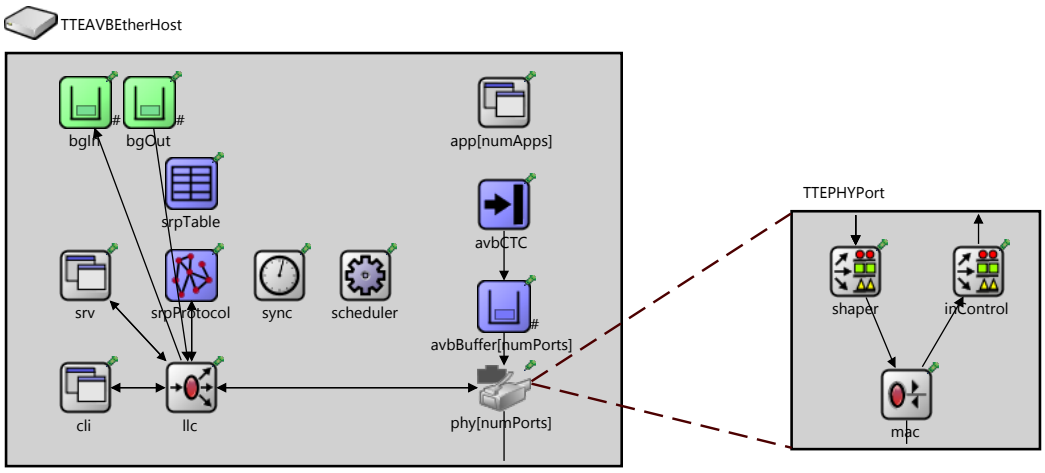

Abbildung 4.8: *Modellierung des TTEthernetHosts und eines physikalischen Ports, der die Verteilung der Nachrichten in die einzelnen Buffer übernimmt.*

diese von einer Applikation erstellt und in einen Sendebuffer gelegt. Der Shaper regelt anschließend, aus welchem Buffer eine Nachricht als nächstes gesendet werden soll und leitet diese an den physikalischen Port weiter. Die Funktionsweise auf der Switchseite funktioniert ähnlich wie auf der Endgeräteseite, nur dass die Nachrichten nicht von Trafficapps erzeugt und empfangen, sondern direkt nach dem Zwischenspeichern in einem Puffer weitergeleitet werden. Genauere Aspekte der Modellierung sind aus den genannten Quellen entnehmbar.

CAN-Bus

Das CAN-Busmodell basiert auf der Arbeit „Ein Framework zu einer OMNeT++ basierten Simulation von CAN-Netzwerken auf der Sicherungsschicht“ von Jonas Engler (Engler 2013). Das Ziel einer CAN-Bussimulation ist die Modellierung der Bus-Arbitrierung, die Abbildung der verschiedenen Nachrichtenarten, deren Versendung und grundlegende Konfigurationsmöglichkeiten des Bussystems. Das Bussystem als solches kann als einfacher Knoten modelliert werden, der jede eintreffende Nachricht an alle Verbindungen weiterleitet. Wichtig für das Simulieren heterogener Netzwerke ist in erster Linie die Übertragung der Data- und der Remoteframes sowie die Parametrisierung der Busgeschwindigkeit. Abbildung 4.9 zeigt den Knoten des Modells mit einem Port, der sich in empfangende und sendende Komponenten aufteilt. Auch in einem Endgerät des CAN-Modells wird die Lastgenerierung über Trafficapps geregelt.

Die genaue Modellierung kann dem Framework (CoRE-Arbeitsgruppe [b]) entnommen werden.

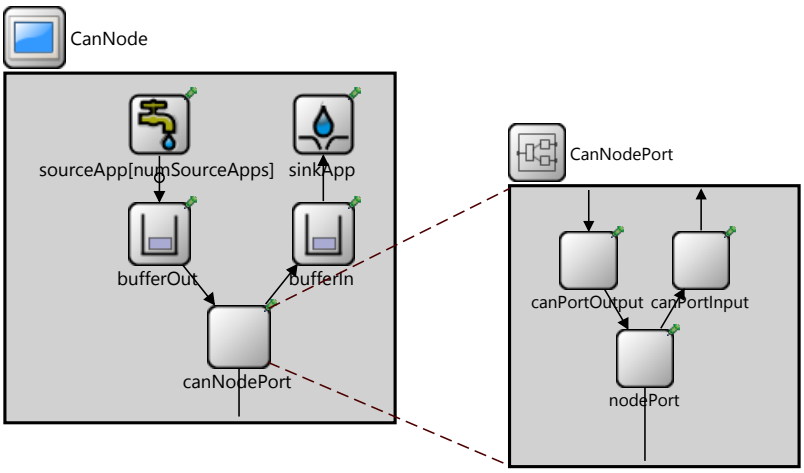

Abbildung 4.9: *Modellierung eines CAN-Knotens mit seinem physikalischen Port.*

Lastgeneratoren

Zur Generierung des Datenverkehrs in paketvermittelnden Netzwerken eignen sich im Besonderen Lastgeneratoren. Sie ermöglichen die Erzeugung hoher Datenraten zwischen zwei Netzknoten oder in einem Subnetz. Jeder Lastgenerator stellt dabei Untersuchungen realer Anwendungen durch Hintergrundlast zur Verfügung und kann mit verschiedenen Verkehrsmixen Experimente in gekoppelten Netzen unterstützen (vgl. Kolesnikov / Wolfinger / Kulas 2009). In heterogenen Netzwerken, die verteilte Systeme mit mehreren Endgeräten darstellen, ist die Abbildung eines Endgerätes auf einen oder mehrere Lastgeneratoren möglich. Jede Verkehrsart benötigt dabei unterschiedliche, charakteristische Verkehrsflüsse, die in der CoRE-Arbeitsgruppe schon von Hermand Dieumo Kenfack (vgl. Dieumo Kenfack 2010) für TTEthernet durchgeführt wurden. Viele Daten im Automobil werden konstant mit gleichbleibendem Intervall versendet, wodurch die Lastgenerierung ebenso konstant erfolgen muss. Nachfolgend werden stochastische Modelle zur Lastverteilung eingeleitet, die sich für die Simulation heterogener Netzwerke eignen und vor allem den event-basierten Verkehr betreffen.

Poissonartige Verkehrsmodellierung: Verkehrsmodellierungen, die auf der Poissonverteilung basieren, stellen die ältesten Techniken zur Lastgenerierung von Sprachverkehr in Telefonnetzen dar. Nach Schäfer (Schäfer 2004) wird unter dem Begriff *Poisson-Modell* ein Verkehrsmodell verstanden, das auf zwei Annahmen basiert:

1. Die Anzahl der eintreffenden Verbindungswünsche in einem vermittelnden Netzknoten ist durch einen Poisson-Prozess beschreibbar.

2. Die Darstellung der Verbindungsdauern ist durch exponentialverteilte Zufallsvariablen oder Verteilungstypen ohne heavy-tailed[3] Eigenschaft möglich.

Hierdurch wird es machbar, eintreffende Ereignisse innerhalb eines Intervalls zu modellieren. Es wird daher in Szenarien mit vielen unabhängigen Lastgeneratoren genutzt.

Markow-Modelle: Markow Modelle basieren auf der Markow-Eigenschaft, die besagt, dass ein zukünftiger Zustand X_{n+1} nur von seinem vorherigen Zustand X_n abhängig ist, nicht jedoch von den weiter zurückliegenden Zuständen $X_1, X_2, \ldots, X_{n-1}$ (vgl. Gaede 1977). Dieses Verhalten ermöglicht die Modellierung einer zeitlichen Abhängigkeit. Anders als in den Poisson-Modellen kann damit auf das Eintreffen von Nachrichten reagiert werden, wodurch beispielsweise die Generierung weiterer Nachrichten beeinflusst wird.

Verkehrsmodelle mit konstanter Bandbreite: In der Simulation ist es notwendig, neben zufallsbedingten Verkehrsströmen konstante Datenraten zu generieren. Diese zeichnen sich durch eine über die Zeit hinweg konstante Bandbreite aus und dienen zur Grundauslastung des Netzwerkes durch signalbasierten Verkehr . Nach Schäfer (Schäfer 2004)[S. 69] ergeben sich für die zu generierende Datenrate $v[kbps]$ für Datenverkehrsmodelle folgende Parameter:

- *Paketgröße* s_p: Angabe der Paketgröße in Bit. Dabei wird die gleiche Größe aller Pakete angenommen.
- *Paketrate* n: Anzahl der zu generierenden Pakete pro Zeiteinheit. Die Basiseinheit wird als 1 Sekunde angenommen, wodurch sich die Größe von n mit $\frac{1}{s}$ bestimmt.
- *Paket-Zwischenankunftszeit* δ_p: Zeitspanne zwischen dem Versenden aufeinander folgender Pakete.

Der Zusammenhang wird dabei mit der Formel 4.1 beschrieben.

$$v = \frac{n \cdot s_p}{1000} = \frac{s_p}{\delta_p} \tag{4.1}$$

Für die Lastgenerierung in Fahrzeugnetzwerken werden in erster Linie Verkehrsmodelle mit konstanter Bandbreite benötigt. Viele Nachrichten von Sensoren und Aktoren werden in festen Intervallen definiert und können somit gleichmäßig gesendet werden. Für die Erzeugung von Kameradaten können ebenfalls die Verkehrsmodelle mit konstanter Bandbreite genutzt werden. Durch die Definition von Paketen pro Zeiteinheit und der kontinuierlichen Übertragung wird ein konstanter Stream simuliert. Für die Simulation von Entertainmentfunktionen

[3] Heavy-tailed-Verteilungen beschreiben eine endlastige Wahrscheinlichkeitsverteilung mit einer unendlichen Varianz. Eine Verteilungsfunktion $F(x), x \geq 0$, besitzt die heavy-tailed Eigenschaft, falls es ein $0 < \alpha < 2$ und ein $c > 0$ gibt, so dass $\lim_{x \to \infty} F(x) = c \cdot x^{-\alpha}$ gilt (Crovella / Taqqu / Bestavros 1998).

ist der Einsatz poissonartiger Verkehrsmodelle sinnvoll. Diese generieren auf Anfrage zufallsbasierten Verkehr innerhalb eines Intervalls und eignen sich für die Analyse des Gesamtsystems bei plötzlich eintretender, hoher benötigter Bandbreite.

Gateways

Das Gatewaymodell wurde im Rahmen der Arbeit „Simulationsmodell eines Multi-Bus Realtime Ethernet Gateways“ von Sebastian Müller (Müller 2014) modelliert und implementiert. Es ermöglicht eine Transformierung von CAN Nachrichten in die drei Nachrichtenklassen von Time-Triggered Ethernet und in entgegengesetzter Richtung. Für das Routing wird eine entsprechende XML-Datei benötigt, in der für jede Nachricht neben dem Quellbussystem mit entsprechenden Konfigurationsparametern, die Nachrichtenklasse des Backbones und die Zielbussysteme genannt sind. Abbildung 4.10 zeigt das Gatewaymodell. Die CAN-Nachrichten werden über den BusConnector mit dem Port verbunden, während das „iocom“-Modul die Verbindung zur Ethernetseite herstellt und beide Nachrichtentypen an die „route“-Komponenten weiterleitet.

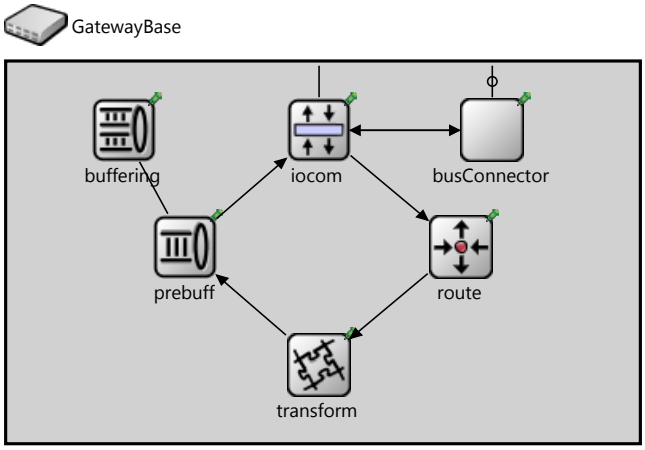

Abbildung 4.10: *Gatewaymodell mit Komponenten zur Transformierung von Nachrichten.*

In dem „route“-Modul wird die ID und Nachrichtenklasse der Nachricht mit den Routingeinträgen des XMLs verglichen. Ist ein Eintrag vorhanden, kann das Transformmodul sich um die Umwandlung der Nachricht kümmern, bevor sie an den entsprechenden Port weitergeleitet wird. Die genaue Modellierung ist auch an dieser Stelle der genannten Arbeit zu entnehmen.

4.2.2 Konzept zur Generierung der Simulationskonfiguration

Zur Integration einer Simulation in einen Entwicklungs- oder Produktionsprozess ist es notwendig, die Konfiguration der Simulationskomponenten so einfach wie möglich zu gestalten und Experten der einzelnen Domänen zugänglich zu machen. Mit Hilfe einer domänenspezifischen Sprache kann dieses Ziel erreicht werden. Bevor die DSL modelliert werden kann, bedarf es jedoch eines einheitlichen Modells, mit dem es möglich ist, die benötigten Konfigurationsdateien zu erstellen und statische Simulationsparameter wie den Nachrichtenschedule der Time-Triggered-Nachrichten zu berechnen. Somit ist ein Modell, das Kenntnis über die Topologie und den spezifischen Konfigurationsparametern der einzelnen Systeme aufweist und zwischen der domänenspezifischen Sprache und dem Simulationsmodell übersetzt, erforderlich. Abbildung 4.11 zeigt das Zusammenspiel der einzelnen Modelle.

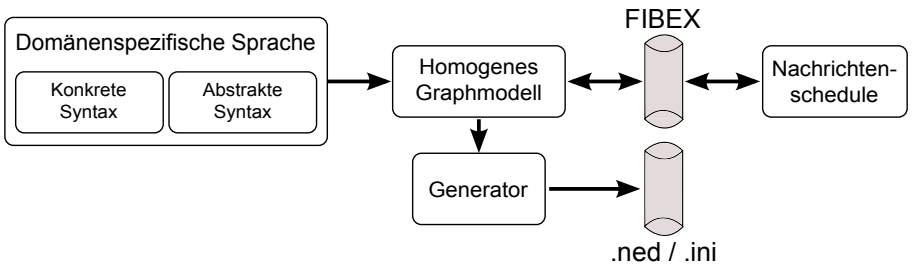

Abbildung 4.11: *Mit der domänenspezifischen Sprache wird ein Graphmodell beschrieben. Dieses kann zudem FIBEX-Dateien laden und mit Generatoren Simulationskonfigurationen generieren.*

Die Aufgabe der domänenspezifischen Sprache ist die Modellierung des Netzwerkes durch einen Domänenexperten. Näheres dazu wird in Kapitel 5.1 erläutert. Wurde ein Szenario definiert, kann daraus anschließend das Modell zur Generierung der Simulationskonfiguration mit Informationen angereichert werden. Eine Darstellung des Modells als Graph, in dem jeder Knoten des Netzwerkes mit einem anderen Knoten verbunden werden kann, ist sinnvoll. Bussysteme, Switches, Gateways und Endgeräte werden dabei als *„HardwareNodes“* miteinander verbunden, wodurch ein homogenes Modell mit differenten Ausprägungen in den Knoten entsteht. Mithilfe des intensiven Gebrauchs der Vererbung und der damit verbundenen Vergleichbarkeit kann jeder Knoten eine String-Repräsentation für die Konfiguration unterschiedlicher Simulationsmodelle bereitstellen. Für jedes Simulationsmodell gibt es demnach ein Repräsentations-Interface, das die Knotenklassen um die benötigten Methoden erweitert. Das Netzwerkmodell kann anschließend dem Generator die Knotenobjekte übergeben, die implementierten Methoden aufrufen und somit die gewünschten Konfigurationen erzeugen. In einem alternativen Modell kann ein Generator für jedes Objekt die Stringrepräsentation erzeugen. Durch diese Herangehensweise braucht die eigentliche Knotenklasse

keine Information über die Stringrepräsentation der eigenen Objekte, wodurch die Kapselung zwischen Generator und Knotenobjekt weiter getrennt wird. Allerdings leidet darunter die Übersichtlichkeit bezüglich der implementierten Methoden der einzelnen Klassen. Mit der Interfacelösung ist es möglich, den Generator genereller zu modellieren, wodurch er nicht bei jeder neu zu unterstützenden Simulation verändert werden muss.

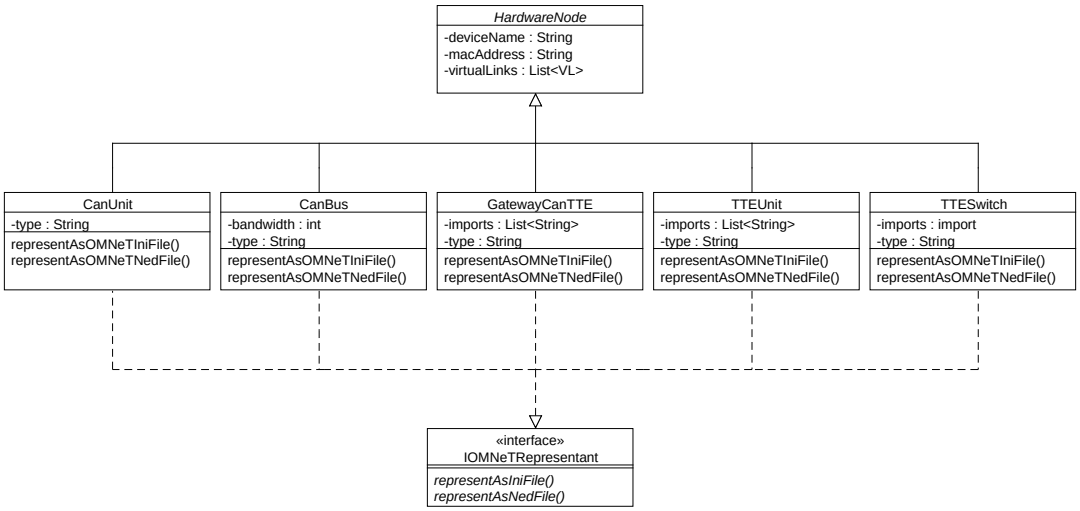

Abbildung 4.12: *Mit dem HardwareNode wird ein Graphmodell beschrieben, das die einzelnen Komponenten abbildet. Über Interfaces wird die Repräsentation für unterschiedliche Simulationsmodelle kenntlich gemacht.*

Als Beispiel sei hier die OMNeT++ Simulationsumgebung genannt. Für die Erzeugung der Konfiguration benötigt jeder Knoten eine *.ini*-Datei mit Informationen über die zu sendenden und weiterzuleitenden Nachrichten und eine *.ned* Repräsentation, mit der die Topologieeigenschaften und Objekte innerhalb des Simulationsmodells beschrieben werden. Wegen der Modellierung neuer Knoten für andere Protokolle ist das bestehende Modell mit neuen Knoten erweiterbar. Abbildung 4.12 zeigt ein UML-Beispiel mit einem Netzwerk, das CAN und TTE unterstützt und über den *IOMNeTRepresentant* eine Schnittstelle nach außen hin anbietet.

Bei der domänenspezifischen Sprache handelt es sich um eine Beschreibung der Topologie, der Nachrichten und der Anforderungen. Für Time-Triggered Ethernet, das auf einem TDMA-Ansatz basiert, müssen allerdings Zeitfenster für die Nachrichten vorausberechnet werden. Durch die große Anzahl der Nachrichten in großen Netzwerken ist dieses nur über Algorithmen effizient zu lösen. Die Berechnung kann entweder direkt auf dem Graphmodell oder extern und somit außerhalb des Modells durchgeführt werden. Für eine externe Be-

rechnung eignet sich das FIBEX-Datenformat als Austausch zwischen Modellen. Dafür wird die bisherige Konfiguration ohne die Zeitfenster aber mit den Anforderungen in eine FIBEX spezifizierte XML-Datei geschrieben. Anschließend können die Fenster mit verschiedenen Schedulingvarianten berechnet werden. Ein Framework zur Berechnung von Schedulingvarianten für TTE wird in Kamieth / Steinbach / Korf u.a. (Kamieth / Steinbach / Korf u. a. 2014) vorgestellt. Ist das FIBEX nun vollständig, kann es wieder in das Modell eingelesen werden und entweder ein komplettes Netzwerk beschreiben oder das vorhandene Netzwerk um die Nachrichtenfenster ergänzen.

Validierung der Simulationskonfiguration

Damit überprüft werden kann, ob sich das System für die abstrakte Darstellung von Netzwerken eignet, soll eine Validierung der Simulationskonfiguration stattfinden. Da bereits für diverse Bussysteme und auch Time-Triggered Ethernet eine FIBEX-Spezifizierung existiert und Modelle in der Simulationsumgebung umgesetzt wurden, eignet sich ein Vergleich der Modelle zur Validierung. Dazu muss es möglich sein, mit der Toolchain gleiche Ausgaben wie bei einer manuellen Konfiguration in der Simulation zu erzielen. Ist dies der Fall und Grenzfälle sowie Standardwerte sind getestet, ist das Verhalten der Toolchain korrekt und die Einbettung in den Entwicklungsprozess abgeschlossen.

Abstraktion der Simulationskomponenten

In Kapitel 4.2.1 wurde die Architektur der Simulationskomponenten bereits erwähnt. Diese stellt die modellspezifische Beschreibung dar. Zur Entwicklung einer generellen Beschreibungssprache wird hier das Konzept einer abstrakteren Ansicht erläutert. Dazu gehören die benötigten Topologieinformationen der unterschiedlichen Netzwerke, also die Endgeräte, Switches, Gateways und Verbindungen zwischen diesen und die Nachrichtenparameter, die ausreichen, um Nachrichten zu beschreiben. Für die Kommunikation über heterogene Netzwerke, also über die Routergrenzen hinweg, müssen zudem Routinginformationen über die unterschiedlichen Netzwerke vorhanden sein. Viele Parameter können in der Simulation als Standardwerte interpretiert werden, wodurch die Modellierung dieser Parameter nur eine optionale Anforderung widerspiegelt. Das Ziel ist es vielmehr, große Netzwerke effizient zu erstellen und in der generierten Konfiguration Feinjustierungen vorzunehmen.

Topologieinformationen: Die Topologie eines Netzwerkes kann gut mit einem Feature-Diagramm dargestellt werden (vgl. 2.3.2). Das Konzept wird dabei als Network betitelt, worunter alle Ausprägungen verstanden werden. Weil ein Netzwerk aus Knoten und Kanten besteht, sind diese dem Netzwerk als Features hinzuzufügen. Das kann durch verschiedene Varianten erfolgen. Zum einen können generelle Features für die Endgeräte, Ethernetswitches

und Bussysteme erstellt werden. Dadurch wird auf dieser Ebene noch nicht zwischen den verschiedenen Bussystemen unterschieden, und die Unterteilung kann über XOR-Gruppierungen unterhalb der generellen Features erfolgen. Zum anderen können Features schon an dieser Stelle einem Unterscheidungsmerkmal unterzogen werden. Dadurch sind spezialisierte Verhaltensweisen möglich, und das domänenspezifische Design wird für den Anwender ersichtlicher.

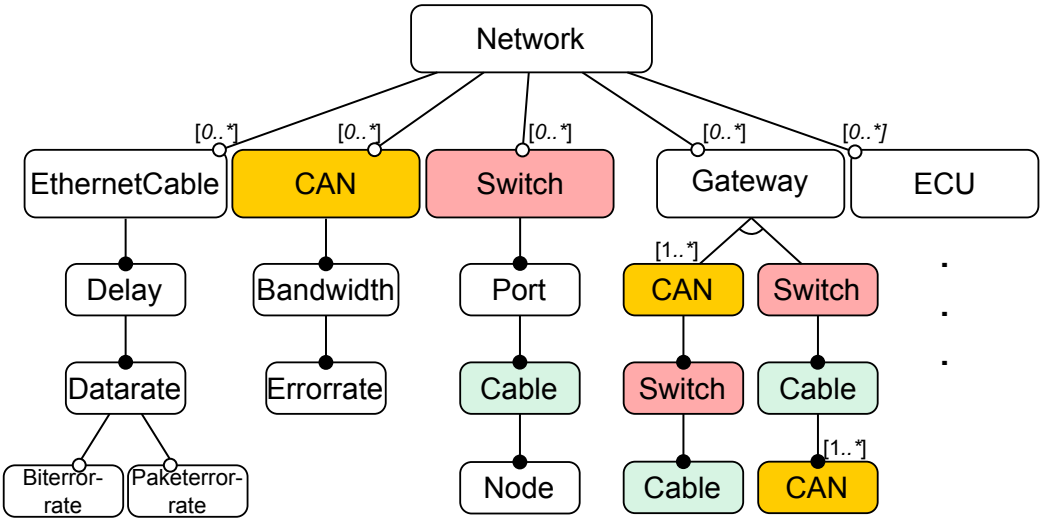

Abbildung 4.13: *Das Konzept eines Networkmodells basiert auf Topologiefeatures mit unterschiedlichen Parametern.*

Abbildung 4.13 zeigt einen Ansatz, der beide Ausprägungen vereint. Zur Modellierung der Netzwerktopologie wird ein spezialisierter Ansatz gewählt, der Kabel, Bussysteme und Backbone unterscheidet. Wird ein neues Bussystem ergänzt, ist eine Anpassung des Modells auf dieser Ebene erforderlich, indem ein neues Feature hinzugefügt wird. Kabel und Bussysteme besitzen Parameter, die über das Delay, die Datenrate und die Fehlerrate bei der Übertragung entscheiden. Bei Switches können Kabel, aber keine Bussysteme referenziert und mit einem Port und einem Knoten verbunden werden. Dieser Knoten wird durch einen Switch, ein Gateway oder eine ECU dargestellt. Zur Modellierung der Gateways und Endgeräte wird ein genereller Ansatz gewählt. Gateways können Bussysteme und Switches mit den benötigten Kabeln referenzieren. Zudem werden gatewayspezifische Parameter, wie die Holduptime, angegeben. Endgeräte werden mit der Referenz auf ein Bussystem oder einen Switch mit einem Kabel beschrieben. Da es sich bei den Endsystemen um die Lastgeneratoren handelt, werden sie in Abbildung 4.14 mit unterschiedlichen Nachrichtentypen separat dargestellt.

Die Spezifizierung über die Verbindung eines Bussystems hat den großen Vorteil, eine Ende-zu-Ende Definition der Nachrichten zu modellieren. Demnach sind sowohl die sendenden als

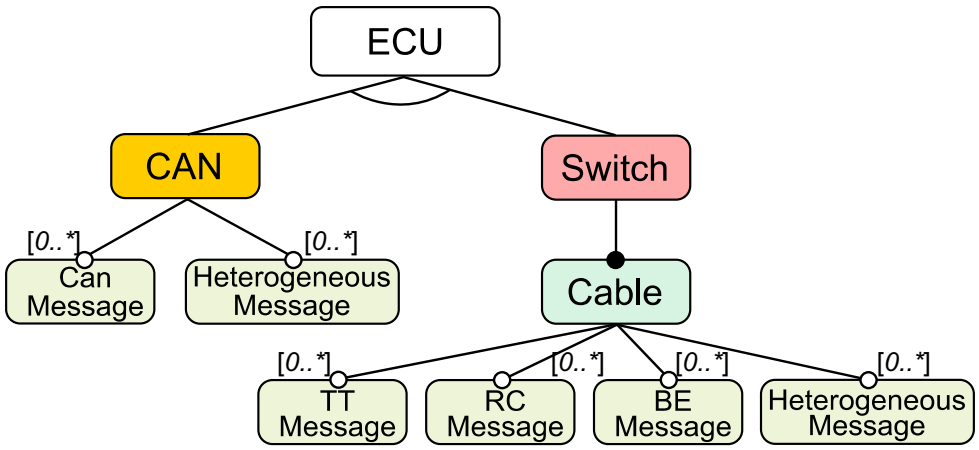

Abbildung 4.14: *Ein Endgerät wird anhand seiner Verbindung identifiziert. Anschließend können Nachrichten für das jeweilige Netzwerk versendet werden.*

auch die empfangenden Endgeräte als ECU-Referenzen zu betrachten. Bei Time-Triggered-Nachrichten sind alle drei Nachrichtenklassen zu modellieren, während bei den Bussystemen die jeweils typischen Nachrichten abzubilden sind.

Modellierung der Nachrichtentypen: Alle Nachrichten werden als Ende-zu-Ende Beziehung mit einem Sender und einem oder mehreren Empfängern dargestellt. Die Empfänger können zudem noch Anforderungen wie maximale Latenz oder maximaler Jitter bekommen, um damit den Scheduler für die Nachrichten zu beeinflussen. Zusätzlich besitzen die Nachrichten eine ID und einen Payload.

Jedes Bussystem benötigt weitere Eigenschaften für die Nachrichten, damit das Netzwerk modelliert werden kann. Die Nachrichten des CAN-Bus beispielsweise haben zusätzlich den FrameType, der erkenntlich macht, ob es sich um ein Remote- oder DataFrame handelt. Bei den Nachrichten für Time-Triggered Ethernet sind für RC-Nachrichten noch zusätzlich die Priorität innerhalb der RC-Klasse anzugeben.

Für heterogene Netzwerke, in denen die Nachrichten über mehrere Bussysteme hinweg von einem zu mehreren Sendern geschickt werden sollen, reichen die oben beschriebenen Nachrichtenfeatures nicht mehr aus. Hierzu ist es sinnvoll, jedes Netzwerk, das eine Nachricht durchläuft, um zu den Empfängern zu gelangen, über eine Liste anzugeben. Dabei wird sowohl die ID als auch der Type benötigt, damit gerade bei einem TTEthernetbackbone die richtige Nachrichtenklasse gewählt wird. Abbildung 4.15 zeigt die Features heterogener Nachrichten.

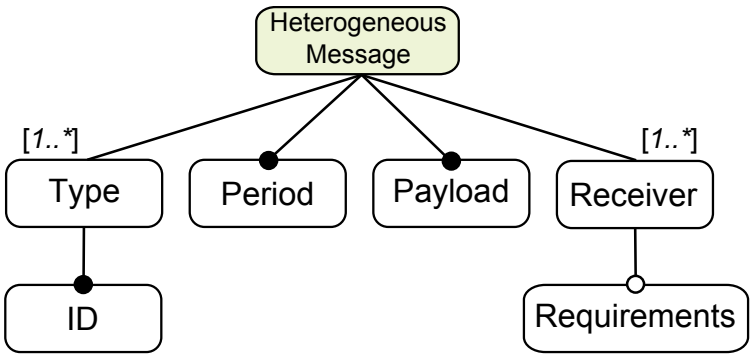

Abbildung 4.15: *Heterogene Nachrichten zeichnen sich durch mehrere Typen und IDs aus. Für jedes Subnetz wird jeweils eine Definition benötigt.*

Auch bei den Nachrichten ist eine Beschreibung mit Types innerhalb der generellen Nachricht nicht besser. Die Modellierung kann mit spezielleren Nachrichten, deren Namen direkt an das jeweilige Bussystem angelehnt sind und die gängigen Bezeichnungen dieser nutzen, intuitiver erfolgen. Daher ist wie in Abbildung 4.14 dargestellt, neben der generellen heterogenen Nachricht ebenfalls eine homogene Nachricht für jeden Typ zu realisieren.

4.2.3 Modell zur Konfigurationsgenerierung

Damit das Netzwerk vollständig im Fibex abgebildet werden kann, wird an dieser Stelle der Schritt von der abstrakten Beschreibung des Netzwerkes hin zu einer vollständigen Beschreibung im homogenen Netzwerk (vgl. 4.11) erläutert. Während sich hinter der Beschreibungssprache noch kein Graphmodell verbirgt, ist das Netzwerkmodell ein Graph. Somit muss für jedes Endgerät, jedes Gateway und jeden Switch ein Knoten im Graph erstellt werden. Dieser *HardwareNode* wurde bereits in der Abbildung 4.11 gezeigt. Über die Verbindungen, die zwischen den Knoten in der DSL beschrieben werden, ist eine Übernahme der Topologie direkt möglich. Zudem sind die Parameter den einzelnen Geräten direkt zuzuordnen, und Standardparameter, die in der DSL für den Netzwerkdesigner nicht relevant und solche, die als optionale Parameter angegeben sind, können gesetzt werden.

Während die Topologie 1:1 zu übertragen ist, muss eine Konfiguration der Nachrichten auf jedem Gerät zwischen dem angegebenen Sender und den Empfängern erfolgen. Anhand des Graphmodells und den damit verbundenen Algorithmen zur Pfadfindung kann nach unterschiedlichen Gesichtspunkten der kürzeste Pfad gefunden werden, und auch redundante Topologien sind nutzbar. Nun stellt sich jedoch die Frage, wie aus den vorgegebenen Werten der Beschreibungssprache unterschiedliche Konfigurationen der einzelnen Geräte erstellt werden können. Bei Bussystemen –wie dem CAN-Bus– stellt dieses kein Problem dar. Jede Nachricht

hat eine Periode, mit der sie verschickt wird, eine ID, und auch sonst sind alle möglichen Informationen vorhanden. Bei einem Backbone, das differente Nachrichtenklassen zulässt, ist dieses weitaus problematischer. Bei Time-Triggered-Nachrichten wird der Schedule extern berechnet und anhand dieses entschieden, wann die Nachrichten bei dem Sender und bei den Empfängern eintreffen. Somit ist nur der Pfad über die verschiedenen Switches erforderlich. Bei Rate-constrained Nachrichten muss zusätzlich zum Pfad die Bandwidth-Allocation-Gap in jedem Switch gesetzt werden. Diese hat sowohl einen Wert für den sendenden als auch für den empfangenden Port. Wenn also die Nachricht pünktlich zur Periode im Lastgenerator erstellt wird, ist eine kleinere BAG beim Weiterleiten erforderlich. Dies ist notwendig, um Staus durch verzögerte Pakete entgegenzuwirken und keine Pakete entfernen zu müssen. Die Differenz zwischen Sendeperiode und BAG ist als Konfigurationsparameter einstellbar.

Bei der heterogenen Kommunikation über mehrere Bussysteme werden Gateways benötigt, die zwischen den Protokollen übersetzen. Durch die Angabe einer heterogenen Nachricht mit ebenfalls einem Sender und mehreren Empfängern können im Graphmodell alle Bussysteme gleichbehandelt und der Pfad über Gateways gefunden werden. Aufgrund der Angabe eines Types und der ID in der Nachricht für jedes zu durchlaufende Bussystem ist eine Zuordnung möglich. Abbildung 4.16 zeigt dabei das Problem, das bei dieser Zuordnung entstehen kann.

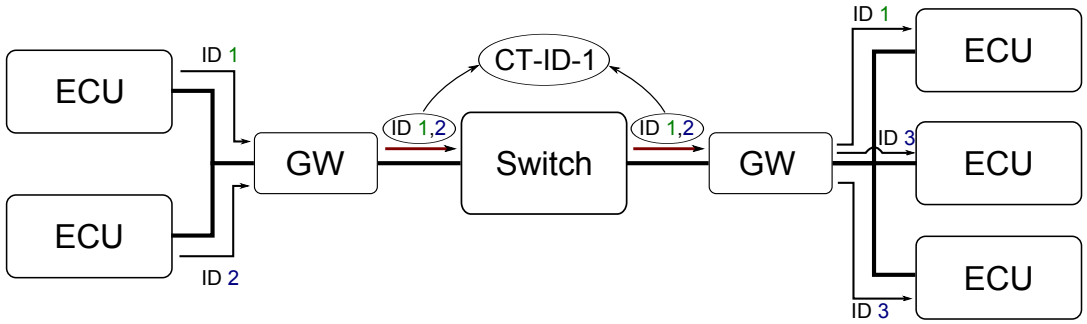

Abbildung 4.16: *Designstrategie für die Beschreibungsform von Gateways. In jedem Subnetz besitzen die Nachrichten unterschiedliche Eigenschaften wie beispielsweise ihre ID.*

Die zwei auf der linken Seite befindlichen Endgeräte senden zwei heterogene Nachrichten über ein Quell- zu einem Zielbussystem. Dazwischen existiert ein Ethernetbackbone, das über Gateways verbunden ist. Die erste Nachricht wird mit den Types CAN, TT und CAN bezeichnet, wobei die IDs in jedem Subnetz “1“ sein sollen. Die zweite Nachricht trägt die gleichen Types für die Subnetze, wobei sich die IDs mit “2“, “1“ und “3“ aber unterscheiden. Nur im Backbone werden beide Nachrichten auf die gleiche ID gemappt. Dafür ist in dem Gateway eine Holduptime notwendig, die auf Nachrichten wartet und diese als gemeinsames

Paket weiterleitet. Diese muss für jede Zusammenfassung von Nachrichten einzeln berechnet werden und kann sich aus den Anforderungen der einzelnen Nachrichten und deren Perioden zusammensetzen. Das Problem besteht nur bei der Angabe von Ende-zu-Ende Nachrichten. Die Angabe für die Nachrichten für jedes Subnetz ist ein anderer Ansatz, bei dem keine Routingtabelle für das Netzwerk existiert, sondern die Routinginformation in jedem Gateway direkt gesetzt wird. Dafür müsste das Gateway als Endgerät fungieren, das Nachrichten sowohl senden als auch empfangen kann. Dieser Ansatz stellt allerdings in komplexen Netzwerken keine wirkliche Alternative zum erstgenannten dar, weil er sich nicht nahtlos in den Prozess von der einfachen Erstellung komplexer Netzwerke bis zu ihrer Evaluation einbetten lässt.

4.3 Konzept zur Evaluation heterogener Fahrzeugnetzwerke

Zur Netzwerkanalyse sind bestimmte Metriken zu definieren, die eine Aussage über die Güte des Netzwerkes zulassen und weitere Informationen über das Netzwerk bereitstellen. Wie im Anforderungskapitel 3.2.3 beschrieben, lassen sich die Metriken dazu in drei Klassen einteilen, wobei die Verlässlichkeits- und insbesondere die Leistungsmetriken eine genaue Betrachtung des Netzwerkes ermöglichen. Daher werden in diesem Kapitel die Metriken definiert, eingeordnet und Aufzeichnungsansätze gezeigt. Anschließend werden Auswertungsmöglichkeiten im Hinblick auf einen generierten Auswertungsreport erläutert.

4.3.1 Metriken als Analysegrundlage

Das Wort Metrik entstammt dem Griechischen und bedeutet „Messen“. In der Netzwerktechnik werden Metriken (Kennzahlen) insbesondere für die Güte der Verbindungen zwischen dem Sender und den Empfängern von Nachrichten genutzt. Dabei sind in Netzwerken in erster Linie die Metriken für die Dienstgüte bekannt. Diese lassen aus Benutzersicht eine Bewertung zu, weil auf der Anwenderebene nicht eingehaltene Werte direkt bemerkbar sind. Allerdings existieren weitere Metriken, die den Systemzustand festhalten und Informationen zu den einzelnen Netzknoten bereithalten. Diese, unter dem Namen Systemmetriken bekannten Metriken, können für ein besseres Topologiedesign eingesetzt werden, da sie die Auslastung der einzelnen Knoten überwachen.

Quality-of-Service Parameter

Die Qualitätsanforderungen an Kommunikationsdienste werden in IP-Netzen hauptsächlich durch die drei Metriken Latenzzeit, Jitter und Paketverlustrate beschrieben. Nachfolgend werden diese Metriken erläutert, definiert und die Formel für ihre Berechnung angegeben:

Latenzzeit: Die Latenz beschreibt die Verzögerung von Botschaften. Damit ist die Zeit gemeint, die eine Botschaft von dem Beginn ihrer Sendung bis zum tatsächlichen Empfang an einem Zielpunkt benötigt (Reif 2009). In Automobilnetzwerken ist eine niedrige und konstante Latenz entscheidend für die fehlerfreie Funktionsweise. Die Latenz wird dabei in die minimale, maximale und durchschnittliche Latenz unterteilt. Die minimale Latenz zeigt die kleinste Verzögerung an, während die maximale Latenz die Verzögerung des Paketes mit der maximalen Laufzeit angibt. Bei der durchschnittlichen Latenz ist ein Intervall anzugeben, in dem die Laufzeitenverzögerung aller Pakete gemittelt wird (vgl. Sarr / Guérin-Lassous 2007). Die Formel, nach der sich die Latenz in der Simulation berechnet ist $T_{Receive} - T_{Send}$, da die Sende- und Empfangszeitpunkte jederzeit vorhanden sind. Die Auswertung von Teilpfaden ist sinnvoll. Dazu muss jedes Paket in allen durchlaufenen Knoten aufgezeichnet werden.

Jitter: Der Jitter beschreibt die Variabilität der Latenz von Paketen in Netzwerken (vgl. Mansour / Patt-Shamir 1998). Das bedeutet, dass in idealen Netzwerken die Latenz stets konstant ist und gleiche Nachrichten immer die gleiche Verzögerung aufweisen. Time-Triggered Ethernet besitzt mit der TT-Nachrichtenklasse einen TDMA-Ansatz, der eine Verzögerung von unter $1\mu s$ erlaubt. Andere Nachrichten werden allerdings in Switches und Endsystemen durch Queueing-Algorithmen unterschiedlich behandelt. Eine Möglichkeit den Jitter zu reduzieren, besteht in dem Einsatz von Jitterbuffern. Allerdings steigt dadurch die durchschnittliche Latenz, weil zusätzliche Buffer die Nachrichten zurückhalten (vgl. Verma / Zhang / Ferrari 1991). Für die Berechnung werden die Laufzeiten zweier Pakete in Bezug zueinander gesetzt. Der Bezug kann entweder zwischen zwei aufeinanderfolgenden Paketen oder in Bezug zu den Paketen mit der höchsten, beziehungsweise niedrigsten Latenz gesetzt werden. Die Formel nach der sich der Jitter berechnet ist: $|(T_{Receive_1} - T_{Send_1}) - (T_{Receive_2} - T_{Send_2})|$.

Paketverlustrate: Mit der Paketverlustrate (PLR) wird die Übertragungsqualität des Bussystems aufgezeigt. Durch das Verhältnis von gesendeten zu übertragenen Datenpaketen kann eine Einordnung auf verschiedenen Ebenen erfolgen. In der Simulation können Paketverluste durch Übertragungs- und Konfigurationsfehler, die beispielsweise zu überlaufenden Buffern führen, deutlich werden. Bei Time-Triggered-Netzwerken sollten bei TT- und RC-Nachrichten nur Übertragungsfehler auftreten und nur bei Best-Effort Nachrichten auch Bufferüberläufe vorkommen. Allerdings kann die fehlerhafte Synchronisierung der Systeme oder falsch definierte Zeitfenster die Kommunikation mit TT-Nachrichten beeinflussen. Bei Rate-Constrained Nachrichten kann eine falsch definierte BAG dazu führen, dass es entweder zum Stau beim Versenden oder zu nicht empfangbaren Nachrichten kommt. Grundsätzlich gibt die PLR immer ein Verhältnis zwischen den verlorenen zu den gesendeten Paketen an, also mit $PLR = \frac{P_lost}{P_send}$ und kann somit an bestimmten Stellen eine Sättigung des Netzwerkes signalisieren. Passende Aufzeichnungspunkte in der Simulation sind Buffer, Links und Anwendungen.

Systemmetriken

Systemmetriken sind für den Anwender nicht direkt durch Auswirkungen spürbar. Sie dienen daher zur Analyse eines Netzwerkes und können über die gewählten Topologien, Routing-Algorithmen und Konfigurationen der einzelnen Systeme Auskünfte geben. Allerdings lassen sie keine Aussagen über die Dienstgüte zu und beeinflussen mit schlechten Werten vielmehr die QoS Parameter. Die Messung der folgenden Metriken beruht in erster Linie auf Zeitintervallen. Diese bestimmen den Zeitraum, den die gemessene Kennzahl rückwirkend beschreibt. Daher ist eine Angabe häufig im Verhältnis zum maximalen Wert in Prozent anzugeben, durch den eine Einordnung unabhängig von der Intervalldauer erfolgen kann.

Kapazität: Mit der Kapazität wird der Payload, der maximal auf einzelnen Links oder einem Pfad in einem Netzwerk erreicht werden kann, beschrieben (vgl. Prasad / Murray / Dovrolis u. a. 2003). Dafür werden die Pakete abzüglich ihres Headers des gleichen Layers im Bezug zu der Paketgröße auf dem darunterliegenden Layer betrachtet. Mit der Messung eines Intervalls kann demnach entschieden werden, ob es sich vorwiegend um die Übertragung kleiner oder großer Pakete handelt. In IP-Netzen spielt die Kapazität für den Layer 3 (L_3) eine große Rolle. Die Kapazität dieses Layers (C_{L_3}) setzt sich aus dem Payload und den Headern (H_{L_2}), die auf Layer 2 hinzugefügt werden, zusammen. Somit bewegt sich die Kapazität zwischen einem minimalen Paket bei $\frac{46Byte}{84Byte} = 54.76\%$ und einem maximalen Paket bei $\frac{1500Byte}{1538Byte} = 97.53\%$. Der Overhead durch die 38 Byte bei Fast Ethernet setzt sich hierbei aus 18 Byte für den Ethernetheader, 8 Byte für die Frame-Preamble und dem Interframe Gap mit 12 Byte zusammen.

Für die Berechnung der Kapazität ist die Übertragungszeit (Δ_{L3}) erforderlich und benötigt die Paketgröße L_{L_n} in Bytes.

$$\Delta_{L3} = \frac{L_{L_n} + H_{L_{n-1}}}{C_{L_{n-1}}} \tag{4.2}$$

Für ein bestimmtes Paket kann die Berechnung der Kapazität mit folgender Formel erfolgen:

$$C_{L_n} = \frac{L_{L_n}}{\Delta_{L_n}} = \frac{L_{L_n}}{\frac{L_{Ln}+H_{L_{n-1}}}{C_{L_{n-1}}}} = C_{L_{n-1}} \frac{1}{1 + \frac{H_{L_{n-1}}}{L_{Ln}}} \tag{4.3}$$

Die Kapazität eines Ende-zu-Ende Pfades ergibt sich aus der minimalen Kapazität der Links, die auf dem Pfad liegen. In Formel 4.4 wird die Berechnung der Kapazität eines Pfades dargestellt. Dabei beschreibt K die Anzahl aller Links des Pfades.

$$C_{path} = \min_{i=1,\ldots,K}(C_i) \tag{4.4}$$

Der Link mit der geringsten Kapazität wird *Narrow Link* genannt (vgl. Prasad / Murray / Dovrolis u. a. 2003).

Verfügbare Bandbreite: Die verfügbare Bandbreite zeigt die Bandbreite an, die für weitere Anwendungen auf einem bestimmten Link zur Verfügung stehen. Dabei ist eine Angabe in Prozent oder in Mb/s sinnvoll. Die Metrik richtet sich nach der reellen Bandbreite, die einen Unterschied zur definierten darstellt. Die reelle Bandbreite kann aufgrund physikalischer Gegebenheiten beeinflusst sein und dadurch deutlich abweichen. Die verfügbare Bandbreite misst die Zeit, in der die Leitung während des anzugebenden Intervalls sendet. Bei einem Intervall von 10 Sekunden und einer Sendedauer von 7 Sekunden beträgt die verfügbare Bandbreite 70 % für den gewählten Zeitraum. Wird eine Angabe in Mb/s gewünscht, kann dieser Wert mit der maximal möglichen reellen Bandbreite multipliziert werden. Ein Link, der eine gegebene Bandbreite von 100 Mb/s und eine reelle Bandbreite von 90 aufweist, besitzt bei dem oben genannten Beispiel eine verfügbare Bandbreite von $0.7 \cdot 90Mb/s = 63Mb/s$. Jeder Pfad, den eine Nachricht durchläuft, um vom Sender zu den Empfängern zu gelangen, besitzt einen *Tight Link*. Dieser ist der Link mit der kleinsten verfügbaren Bandbreite des gesamten Pfades (Hu / Member / Steenkiste u. a. 2003). Die verfügbare Bandbreite des Pfades wird dann mit der des Tight Links gleichgesetzt.

Linkauslastung: Die Linkauslastung zeigt im Gegensatz zur verfügbaren Bandbreite die Auslastung eines Links im Verhältnis zur maximalen Bandbreite, die innerhalb eines Intervalls übertragen werden kann, an. Das Intervall wird typischerweise mit einer Sekunde angegeben. Während eine Angabe der Linkauslastung entweder in Prozent oder Mb/s erfolgen kann, wird hier ähnlich wie bei der verfügbaren Bandbreite gemessen. Werden während der Intervalldauer nur die Hälfte der Zeit Daten übertragen, beträgt die Auslastung 50%. Der große Vorteil in der Wahl von kleinen Intervallen ist die bessere Erkennung auftretender Spitzen, weil nur die Übertragung weniger Pakete möglich ist und jedes Paket eine größere Auswirkung auf die Auslastung aufweist.

Bei Time-Triggered Ethernet ist die Aufteilung der Linkauslastung auf die unterschiedlichen Nachrichtenklassen und deren Virtual Links interessant. Dafür muss auf jedem Link die Aufteilung der Linkauslastung auf die einzelnen Virtual Links und den Best-Effort-Traffic erfolgen. Anschließend kann der Traffic der Virtual Links in Bezug zu der gesamten Auslastung des Links gesetzt werden.

Buffergröße: Mit der Metrik Buffergröße wird die Anzahl der in den Switches und Endgeräten zwischengespeicherten Frames beschrieben. Befinden sich viele Nachrichten gleichzeitig in einem Buffer, steigt die Latenz für jedes ankommende Paket erheblich. Da es nicht möglich ist, unbegrenzt Nachrichten im Buffer abzulegen, müssen ab einem bestimmten Punkt neu eintreffende Pakete verworfen werden. Bei der Nutzung eines Ende-zu-Ende Transportprotokolls, wie dem Transmission Control Protokoll (TCP), resultiert jeder Paketverlust, der ein erneutes Senden zur Folge hat, in einer

niedrigeren Dienstgüte des Pfades (vgl. Guérin / Peris 1999). Somit beeinflusst die Buffergröße direkt andere QoS-Parameter. In der Simulation ist darüber hinaus das Testen mit unendlich großen Buffern denkbar, wodurch die maximalen Füllstände der Buffer ersichtlich werden.

Für eine möglichst geringe Buffergröße ist die Nutzung verschiedener Buffer-Management- und Paket-Scheduling-Algorithmen üblich, wodurch die Priorität von Datenübertragungen und der Durchsatz des Netzwerkes erhöht werden kann (vgl. Fang / Yen / Pan u. a. 2010).

Bei Time-Triggered Ethernet ist eine individuelle Messung für jeden Virtual Link eines physikalischen Links durchführbar, weil die Buffer für unterschiedliche Nachrichtenklassen und Critical-Traffic-Ids existieren.

4.3.2 Aufzeichnung und Auswertung der Metriken

Die Metriken lassen sich auf verschiedene Weisen in der Simulation messen. Der einfachste Weg stellt die direkte Aufzeichnung eines Wertes dar, mit dem spätere Berechnungen durchgeführt werden können. Viele Metriken lassen sich aus den üblichen ableiten, allerdings kann diese Berechnung nicht zur Laufzeit erfolgen. Abgeleitete Metriken lassen sich oftmals nur über ein vordefiniertes Zeitintervall angeben. Dieses erfordert eine kontinuierliche Aufzeichnung, bei der jeder aufgezeichnete Punkt für zukünftige Aufzeichnungswerte relevant ist. Beispielsweise soll die Linkauslastung bei einem Zeitintervall von einer Sekunde zu jedem Zeitpunkt die Linkauslastung der letzten Sekunde in Prozent angeben. Dazu ist ein Aufsummieren aller Werte innerhalb des gesetzten Intervalls erforderlich.

Für das Problem der kontinuierlichen Aufzeichnung werden hier zwei Ansätze beschrieben, die unterschiedliche Laufzeiten besitzen. Beide setzen das Aufzeichnen aller Werte in einem zusätzlichen Buffer mit einem Zeitstempel voraus.

Bei dem ersten Ansatz wird das Event zur Berechnung der Metrik beim Eintreffen eines Paketes ausgelöst. Trifft ein Paket ein, werden zuerst die Pakete von hinten nach vorne in dem Zwischenspeicher bezüglich der Zeitpunkte überprüft. Befindet sich ein Paket nicht mehr zurückliegend im Intervall, ist es zu entfernen. Daraufhin muss genau eine Intervalllänge später ein Wert mit den restlichen Werten im Buffer für den Ausgabevektor berechnet werden. Diese Vorgehensweise wird so lange wiederholt, bis das erste Paket innerhalb des letzten Intervalls liegt. Anschließend kann das aktuelle Paket in den Zwischenspeicher aufgenommen werden, so dass sich der aktuelle Ausgabewert aus den Werten des Buffers zusammensetzt. Liegen aufeinanderfolgende Werte mehr als eine Intervalllänge auseinander, müssen Zwischenpunkte mit 0 Werten hinzugefügt werden, die genau auf den Intervallgrenzen liegen. Abbildung 4.17 fasst die verschiedenen Probleme zusammen.

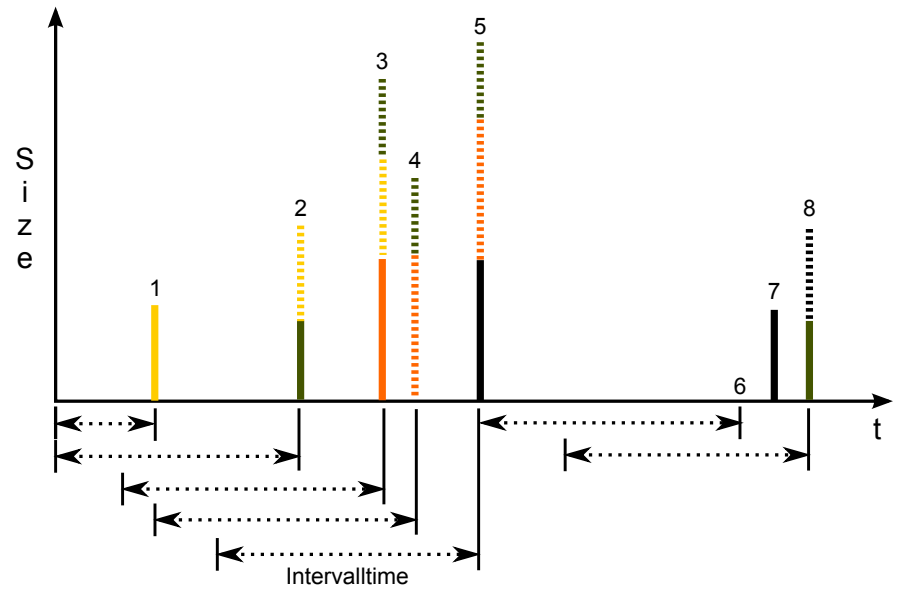

Abbildung 4.17: *Bei dem präzisen Aufzeichnungsmodus wird für jede Änderung über die letzte Intervalldauer ein Wert für die Metrik berechnet.*

Es werden für die sechs aufgezeichneten Werte insgesamt acht Ausgabewerte für die Metrik erzeugt. Dabei wird der vierte Ausgabewert durch das Verlassen des Wertes mit der Zahl 1 aus dem Puffer ausgelöst. Ausgabewert 6 hat ein Größe von 0, weil über eine Dauer von einer Intervalllänge kein Wert in dem Buffer eingetroffen ist. Der sechste Wert mit der Größe 0 wird allerdings erst beim Eintreffen des siebten Wertes zu dem Ausgabevektor hinzugefügt, weil dieser rückwirkend die Entfernung von Nachrichten triggert.

Der zweite Ansatz beruht auf der Berechnung zu vorbestimmten Zeitpunkten. Dazu werden wieder alle Werte gebuffert. Überschreitet der aktuelle Aufzeichnungszeitpunkt die Dauer des letzten Aufzeichnungszeitpunktes zuzüglich einer Intervalllänge, werden alle Werte, die vor dem letzten Aufzeichnungspunkt liegen, entfernt und die Werte im Buffer aufsummiert. Anschließend kann der Ausgabewert im Ausgabevektor aufgenommen werden. Abbildung 4.18 bildet den zweiten Ansatz ab.

In diesem Fall werden für dasselbe Beispiel nur drei Werte aufgezeichnet und zwar genau im Abstand des definierten Intervalls.

Der Unterschied der beiden Ansätze liegt in der Häufigkeit der Berechnung und dem damit verbundenen höheren Rechenaufwand. Der erste Ansatz ergibt eine präzisere Darstellung und zeigt sämtliche Schwankungen an. Dafür werden aber bei dem oben angegebenen Beispiel acht Berechnungen benötigt. Der zweite Ansatz verbirgt Spitzen, weil die Berechnung nur

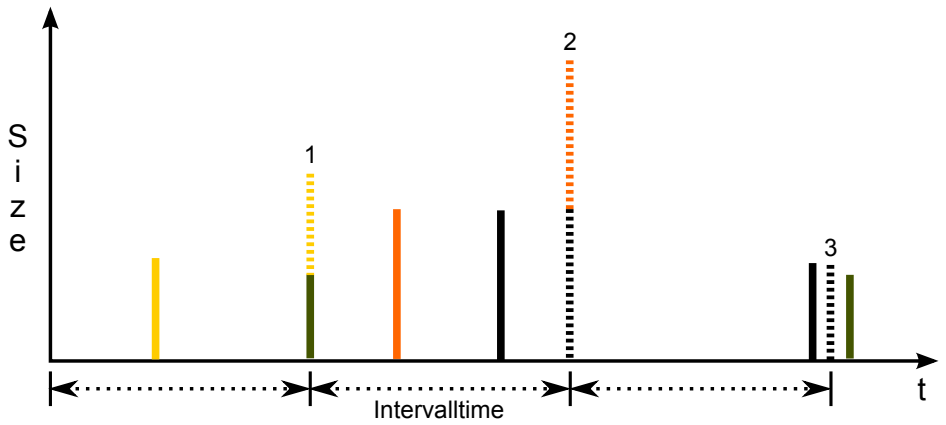

Abbildung 4.18: *Beim performanten Aufzeichnungsmodus wird zu jedem Intervallzeitpunkt ein Wert berechnet. Das ergibt drei Aufzeichnungspunkte bei drei Intervalllängen.*

einmal pro Intervall stattfindet und liegt mit drei Berechnungen deutlich unter Ansatz 1. Inwieweit sich die beiden Ansätze in der Simulation auswirken und ob die genauere Darstellung den hohen Rechenaufwand rechtfertigt, gilt es zu prüfen.

Die Auswertung der aufgezeichneten Werte kann entweder mit einem generierten Report, in dem gewünschte Werte aufbereitet und aufgelistet werden oder mit Hilfe des Auswertungstools, das in der Simulationsumgebung vorhanden ist, erfolgen. Letzteres ermöglicht über Filterangaben die Auswahl und Überprüfung der Anforderungen, die an das Netzwerk gestellt wurden, wodurch umfangreiche Analysen möglich sind. Daher stellt es die flexiblere Variante dar, wenn neue Metriken erhoben werden und neue Simulationsmodelle unterstützt werden sollen.

5 Umsetzung einer Lösung zur Generierung, Simulation und Evaluation

In diesem Kapitel wird die Umsetzung des Konzeptes zur Simulation von heterogenen Netzwerken erläutert. Der erste Abschnitt (5.1) befasst sich mit der Implementierung einer domänenspezifischen Sprache, die zur Generierung der Simulationskonfiguration dient. Dazu gehört die Entwicklung der konkreten Syntax und die Konfigurationsgenerierung eines Simulationsmodells. Im darauffolgenden Abschnitt 5.2 wird die Aufzeichnung und die Evaluierung der Metriken mit den unterschiedlichen Modi beschrieben.

5.1 Domänenspezifische Sprache zur Generierung von Fahrzeugnetzwerken

Während in Kapitel 4.2.2 ein Konzept für die Abstraktion der Simulationskomponenten erläutert wurde, wird in diesem Kapitel die Umsetzung einer konkreten Sprache sowohl für die Topologie als auch für die Nachrichtenmodellierung dargestellt. Das zuvor erläuterte Konzept soll dazu als abstrakter Syntaxbaum dienen und für die in den Feature Diagrammen gezeigten Module eine intuitive Eingabe ermöglichen. Die Eingabe kann anhand einer konkreten Syntax, die mithilfe des Xtext-Frameworks in einer EBNF-Grammatik (2.3.2) erstellt wird, erfolgen. Diese Sprache soll anschließend NedGenerator (NedG) heißen und kann als Plugin in einer Eclipse-Entwicklungsumgebung eingesetzt werden. Die nach der konkreten Syntax erstellten Konfigurationen werden in nedG-Dateien gespeichert und vom NedGenerator verarbeitet.

5.1.1 NedG-File Format

Eine nedG-Datei besitzt neben dem Netzwerknamen, der das Konfigurationsmodell beschreibt, ein oder mehrere Netzwerke, die entweder direkt in der Sprache angegeben oder über ein FIBEX eingelesen werden. Bei der direkten Angabe eines Netzwerkes ist eine Unterteilung in verschiedene Abschnitte vorgesehen. Diese sind Netzwerkparameter, Generationsparameter, Kabel und Bussysteme, Switches und Endgeräte. Netzwerkparameter betreffen dabei direkte Eigenschaften des Netzwerkes. Bei Time-Triggered Ethernet ist beispielsweise

in erster Linie die Länge eines Zyklus anzugeben. Mit den Generationsparametern werden Eigenschaften bei der Generierung beeinflusst. Dazu gehört neben der möglichen Ignoranz einzelner Nachrichtenklassen auch die Auswahl gewünschter Scheduling-Algorithmen für die Time-Triggered-Nachrichten. Die wichtigen Eigenschaften eines Netzwerkes werden anschließend angegeben. Mit der Definition von Kabeltypen, Switches und ECUs ist eine homogene Topologie aufzubauen. Die Switches und Endgeräte bekommen dabei die Information des zu nutzenden Kabels und des zu verbindenden Gerätes. Bei dem Switch ist zudem eine genaue Definition des Switchports vorgesehen, damit im generierten Netzwerk eine leichtere Zuordnung erfolgen kann. Der Ausschnitt 5.1 zeigt ein homogenes Netzwerk mit einem Kabel und zwei Switches, die untereinander und jeweils mit einem Endgerät verbunden sind.

```
Networkname exampleNoMsg
Fibex fibex.xml
Network {
      Netzwerkparameter {
            Cycle 10 ms
      }
      Generationsparameter {
            NoRC, NoBE, ...
      }
      EthernetCable c1 {
            Delay 100 us Datarate 100 Mb/s
                  Packeterrorrate 5 % Biterrorrate 0 %
      }
      Switch s1 {
            Port 1 to Switch s2 Cable c1
            Port 2 to Ecu ecu1 Cable c1
      }
      Switch s2 {
            Port 1 to Switch s1 Cable c1
            Port 2 to Ecu ecu2 Cable c1
      }
      Ecu ecu1 {
            Connect s1 Cable c1
      }
      Ecu ecu2 {
            Connect s2 Cable c1
      }
}
```

Listing 5.1: *Gewünschtes Codebeispiel für ein homogenes Netzwerk mit zwei Switches und zwei Endgeräten*

Hinzufügen von Bussystemen

Damit aus dem Netzwerk ein heterogenes werden kann, muss es möglich sein, Bussysteme definieren zu können und diese mit den Endgeräten zu verbinden. Listing 5.2 zeigt, wie das Netzwerk mit einem CAN-Bus und einer ECU, bei der die Identifizierung als CAN-Endgerät anhand der Verbindung fällt, erweitert wird. Zudem wird der CAN über ein Gateway mit einem Switch verbunden. Das Hinzufügen weiterer Bussysteme ist nach diesem Schema vorgesehen.

```
1  CanBus can {
2        Bandwidth 125000 bit/s
3        Packeterrorrate 10%
4  }
5  Ecu canEcu1 {
6        Canbusconnect can
7  }
8  Ecu canEcu2 {
9        Canbusconnect can
10 }
11 Gateway gw{
12       Canbusconnect can Switchconnect s1 Cable c1
13 }
```

Listing 5.2: *Die homogene Beschreibung kann über Gateways mit anderen Bussystemen gekoppelt werden. Über das Gateway werden die zwei Can-Knoten mit dem Switch verbunden.*

Die Parameterliste des hier dargestellten CAN-Busses umfasst sowohl vorausgesetzte Parameter wie die *Bandwidth* als auch optionale wie die *Packeterrorrate*. Den Endgeräten wird anstelle des *Connects* zu einem Switch die Aufgabe mit *Canbusconnect* zugeordnet.

Netzwerktraffic mit unterschiedlichen Nachrichtenklassen

Das Abbilden von Nachrichten muss für einzelne Bussysteme und das Ethernetbackbone möglich sein. Für Time-Triggered Ethernet-Endgeräte sind die Nachrichtentypen innerhalb des TTEthernet-Netzwerkes, Time-Triggered, Rate-Constrained und Best-Effort. Nachrichten des Time-Triggered- und Rate-Constrained-Typs werden über das Virtual Link Konzept beschrieben. Jeder Virtual Link besitzt demnach eine Critical Traffic ID, die einmalig im Netzwerk vorhanden sein darf. Zudem wird der Nachrichtenpayload und die Periode, also die Dauer zwischen erneutem Senden über den VL, angegeben. Für jeden Empfänger ist die Definition von Anforderungen wie die maximale Ende-zu-Ende-Latenz möglich. Während dies bei TT-Nachrichten für das Scheduling benötigt wird, stellt es bei RC eine Möglichkeit zur

direkten Evaluation in der Simulation dar. Rate-Constrained Nachrichten besitzen zusätzlich noch eine Priorität, die für Nachrichten innerhalb der Klasse gilt. Listing 5.3 zeigt die konkrete Syntax der beiden Nachrichtentypen.

```
TTVL {
      ID 1 Payload 20 byte Period 200 us
      Receiver {
            Rec ecu2 max latency 10 ms
      }
}
RCVL {
      ID 2 Payload 1000 byte Period 1 ms Priority 2
      Receiver {
            Rec ecu2 max latency 50 ms
            Rec ecu3
      }
}
```

Listing 5.3: *Konkrete Syntax für Time-Triggered- und Rate-Constrained Nachrichten mit der Darstellung als Virtual Links*

Nachrichtenpakete für Standardethernetverkehr sind über die Best-Effort-Traffic-Klasse abzubilden. Da es sich hierbei um zufällig auftretende Verkehrsdaten handelt, und der Ethernetverkehr nach einer gewünschten Zeit eintritt, sind weitere Parameter als bei den anderen beiden Klassen modellierbar. Demnach wird eine Startzeit vorausgesetzt und ein Intervall, in dem die Pakete erneut auftreten. Dieses Intervall erstreckt sich über einen Bereich, in dem es zwischen verschiedenen Werten schwankt. Bei den Nachrichten ist die Angabe eines Responsepaketes möglich, das eine Antwort von der Empfängerseite mit der gewünschten Paketgröße zulässt. In dem folgenden Listing 5.4 wird die Syntax des Best-Effort-Traffics gezeigt.

```
Best-Effort-Traffic {
      StartTime 1 s Interval 200 us to 300 us Payload 120 byte
         ResponsePacket 100 byte
      Receiver {
            Rec ecu1
      }
}
```

Listing 5.4: *Konkrete Syntax für Best-Effort-Background-Traffic*

Nachrichten von Bussystemen sehen ähnlich aus, wie die bereits gezeigten. Sie besitzen eine ID zur Identifikation, eine Periode, die das Intervall angibt und den Nachrichtenpayload. Beim CAN-Bus muss zudem noch der Nachrichtentyp genannt werden. Als Receiver der

Nachrichten sind alle Endgeräte des gleichen Bussystems zugelassen. Listing 5.5 zeigt die dazugehörige Syntax.

```
CAN-Messages {
      ID 15 Period 50 ms Payload 8 byte DataFrame
            Receiver {
                  Rec aws
            }
}
```

Listing 5.5: *Konkrete Syntax der CAN-Bus Nachrichten*

Für Nachrichten, die über Domänengrenzen hinausgehen und dabei verschiedene Subnetze durchlaufen, wird eine andere Darstellung der oben genannten Nachrichtentypen gewählt. Über sogenannte *Heterogeneous Messages* ist die Modellierung möglich. Diese benötigt wie alle Nachrichten eine Periode als Intervall und einen definierten Payload. Anschließend werden über das Schlüsselwort *Subnetworks* die Parameter der einzelnen zu durchlaufenden Netzwerke –wie die ID und Prioritäten– angegeben. Im Fahrzeugnetzwerk ist dabei meistens die Aufteilung in Quell- und Zielbussystem, die über ein Backbone oder Gateway miteinander verbunden sind, festgelegt. Das Listing 5.6 zeigt die Syntax für domänenübergreifende Kommunikation.

```
Heterogeneous Message {
      Period 4 ms Payload 8 byte
            Subnetworks {
                  CANid 5
                  RCid 10 Priority 1
                  CANid 3
            }
            Receiver {
                  Rec ecu2 max latency 50 ms
                  Rec ecu3
                  Rec ecu6
            }
}
```

Listing 5.6: *Heterogene Nachrichten zeichnen sich durch die Subnetzwerke mit unterschiedlichen Parametern aus.*

5.1.2 Validationsregeln

Zur Implementierung einer domänenspezifischen Sprache gehört die Definition von Validationsregeln, mit denen überprüft werden kann, ob nur valide Operationen vorhanden sind.

Diese unterstützen den Domänenexperten bei der Modellierung eines Fahrzeugnetzwerkes und überprüfen sowohl statisch als auch dynamisch den abstrakten Syntaxbaum.

Für Parameter sind die Bereichsgrenzen einzuhalten. Bei Nichteinhaltung werden dem Anwender entweder über *Warnings* oder *Errors* signalisiert, dass der Wert nicht der Spezifikation entspricht. Mit der Warning wird die Netzwerkkonfiguration der Warning entsprechend generiert. Bei der Auslösung eines Errors wird der Generierungsprozess unterbrochen, und ein anderer Wert ist zu wählen. Nachfolgend werden die Regeln für die Knoten aufgezählt, die notwendig sind, damit keine Fehler im Prozess auftreten:

Gerätenamen: Für die Namen der Switches, Gateways und Endgeräte sind unterschiedliche Namen zu wählen.

Verbindungen: Verbindungen zwischen Geräten müssen auf beiden Seiten gesetzt sein. Wird nur eine Seite verbunden, ist dieses zu signalisieren. Bei Switches darf ein Port nur einmal genutzt werden.

Wegen der größeren Anzahl an Parametern von Nachrichten gibt es dementsprechend mehr Validationsregeln, die den Netzwerkdesigner unterstützen und die regelkonforme Modellierung überprüfen:

Größe des Nachrichtenpayloads: Jede Nachricht eines bestimmten Typs darf eine maximale Payloadgröße nicht überschreiten und führt zu einem Fehler. Bei Ethernetframes wird zudem der Payload zu einer minimalen Größe von 46 Byte erweitert. Dieses Verhalten ist als Warning zu sehen. Bei heterogenen Nachrichten darf der Payload nicht größer als der kleinste, maximale Payload der Subnetze sein.

NachrichtenIDs: Für Time-Triggered- und Rate-Constrained-Nachrichten sind unterschiedliche Critical-Traffic IDs zu wählen. Innerhalb von Bussystemen muss darauf hingewiesen werden, dass zwei Nachrichten die gleiche ID aufweisen.

Empfänger: Der Sender ist nicht als Empfänger von Nachrichten vorgesehen. Zudem muss der Empfänger im gleichen Subnetz vorhanden sein. Bei heterogenen Nachrichten ist eine Kontrolle der Subnetze durchzuführen, bei der sowohl die IDs als auch die Angabe des Nachrichtentyps zu überprüfen sind.

5.1.3 Interpretation der Sprache

Nachdem ein Netzwerk gemäß der Validationsregeln in einer NedG-Datei gespeichert und in den abstrakten Syntaxbaum überführt wurde, muss der AST interpretiert werden. Dieses erfolgt innerhalb der von Xtext zur Verfügung gestellten Generatorklasse. Die Sprachelemente der Geräte, Verbindungen und Nachrichten werden der Reihe nach in das Graphmodell überführt (5.1). Bei den Geräten werden die Switches, Endgeräte, Gateways und Busse dem

Graph als Knoten ohne jegliche Verbindungen nacheinander hinzugefügt. Anschließend sind die Verbindungen zwischen den einzelnen Knoten herzustellen. Dieses gestaltet sich von der Mitte nach außen hin, wodurch zuerst die Verbindungen des Backbones mit den Switchverbindungen und den Endgeräten an der Reihe sind. Danach können die Bussysteme mit ihren Knoten über die Gateways angegliedert werden.

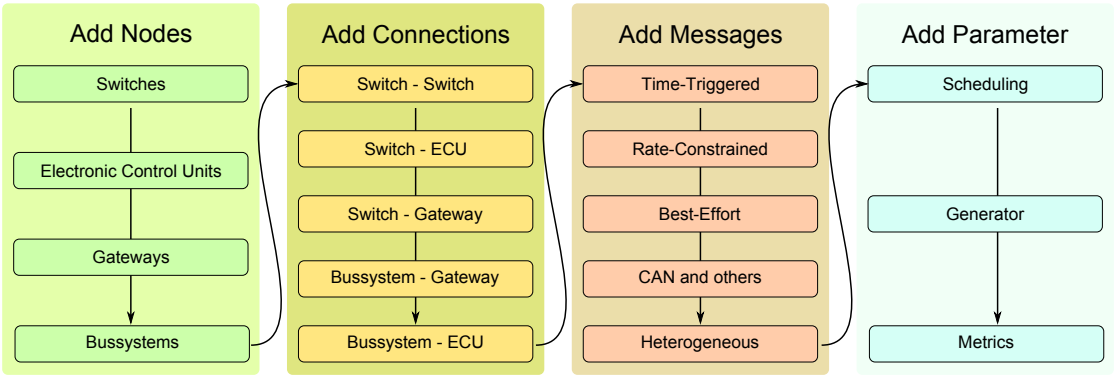

Abbildung 5.1: *Überführung der Sprachelemente in das Graphmodell. Der Reihe nach werden Knoten, Verbindungen, Nachrichten und Parameter hinzugefügt.*

Da bei den Virtual Links eines TTEthernet-Netzwerkes jedes Gerät zwischen dem Sender und den Empfängern offline konfiguriert werden muss, ist das Hinzufügen der Ende-zu-Ende beschriebenen Nachrichten nicht analog zu den Knoten und Kanten. Allerdings lassen sich auf dem vollständigen Graph Wegfindungsalgorithmen anwenden. Mithilfe des Bellman-Ford- oder Dijkstra-Algorithmus (vgl. Redmer 2012)[S. 09] ist eine Berechnung der kürzesten Pfade von dem sendenden Knoten zu den empfangenden Knoten realisierbar. Anschließend sind bei jedem Switch die Nachrichten und die Ports, auf welchen diese empfangen und gesendet werden, zu setzen. Bei Rate-Constrained ist zudem die BAG einer Nachricht zu wählen, die sich an der Periode orientiert. Die BAG zum Senden eines Paketes muss kleiner als die Periode sein, damit es zum Abbau der Nachrichten in den Switches kommen kann. Wird eine RC-Nachricht in einem Switch delayed, kann es vorkommen, dass zwei Nachrichten mit der gleichen CTid vorhanden sind. Daher muss die Senderate im Switch ein wenig schneller konfiguriert sein, da sich die Nachrichten sonst aufstauen. Auf der empfangenden Seite muss die BAG noch kleiner als auf der sendenden Seite sein. Aufgrund der abweichenden Uhren der Systeme oder bei ungleichen Größen aufeinanderfolgender Nachrichten kann bei gleicher BAG-Angabe die Nachricht fälschlicherweise verworfen werden, falls die BAG auf dem empfangenden System noch nicht wieder bereit ist. Abbildung 5.2 zeigt das erforderliche Verhalten.

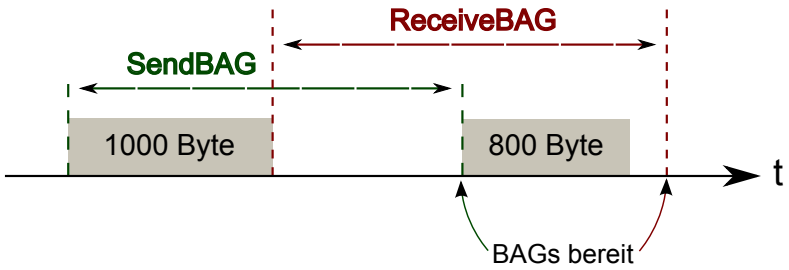

Abbildung 5.2: *Bei Gleichsetzung der Sende- und EmpfangsBAG ist das Senden aufeinanderfolgender Pakete mit unterschiedlicher Paketgröße nicht möglich.*

Die Time-Triggered-Nachrichten werden zwar in jedem Switch genau wie die Rate-Constrained Nachrichten hinzugefügt, jedoch fehlen die Angaben der Empfangs- und Sendefenster. Diese werden zu einem späteren Zeitpunkt im externen Scheduler berechnet.

Für die Best-Effort und die Nachrichten der Bussysteme ist eine direkte Übernahme in das Quell- und Zielendgerät möglich. Diesen werden für das Senden von Nachrichten *Traffic-Apps* des jeweiligen Nachrichtentyps und zum Empfang von Nachrichten *Traffic-Sinks* hinzugefügt. Die Lastgeneratoren enthalten die Merkmale der Nachrichten wie Periode und Payload.

Bei heterogenen Nachrichten sind die Nachrichten über die einzelnen Subnetzwerke und den dazwischenliegenden Gateways zu routen. Durch die Angabe der einzelnen IDs und weiteren spezifischen Eigenschaften in den Subnetzen können die Nachrichten wie mehrere homogene Nachrichten zwischen den Gateways hinzugefügt werden. Mit dem Mapping von eintreffenden auf zu verschickende Nachrichten in jedem Gateway wird eine Zuordnung hergestellt. Zusätzlich wird ein Eintrag für das RoutingXML vorbereitet. Dieses wird später in der Simulation für die Konfiguration der Gateways benötigt.

Für die Berechnung der Time-Triggered Kommunikation, die sowohl die direkten TT-Virtual Links als auch die TT-VLs der heterogenen Nachrichten betrifft, wird nach dem Hinzufügen aller Nachrichten ein FIBEX mit den Anforderungen, Topologieinformationen und dem Nachrichtenverhalten gemäß der FIBEX-Spezifikation erstellt. Anschließend wird der Scheduler aufgerufen, der das FIBEX einliest und den Schedule für die Fenster der Time-Triggered Nachrichten berechnet. Danach wird die Information zurück in das FIBEX geschrieben und in den einzelnen Knoten des Netzwerkes hinzugefügt.

5.1.4 Generieren von Konfigurationsdateien

Der Hauptgenerator ist dafür zuständig, mehrere Generatoren aufzurufen, die das Modell für unterschiedliche Simulationsumgebungen und standardisierte Formate aufbereiten und

ausgeben können. Für die OMNeT++ Simulationsumgebung gibt es demnach eine *OMNeTFilesGenerator*-Klasse, in der die spezifische Konfigurationssyntax von OMNeT++ zu finden ist. Für die Endgeräte, Gateways, Switches und Bussysteme und das Gesamtnetzwerk findet eine Aufteilung in verschiedene NED-Dateien und dazugehörige „.ini“-Dateien statt. Dadurch ist eine Übersichtlichkeit in der Simulationsumgebung gegeben und kleinere Anpassungen sind leicht vorzunehmen.

Zuerst werden die Dateien für das Gesamtnetzwerk erstellt. In der NED-Datei des Netzwerkes befinden sich neben den Geräten als Submodule die Definitionen der Kommunikationskanäle und die Verbindung der Geräte untereinander. Die dazugehörige „omnetpp.ini“ enthält Konfigurationsparameter und solche, die das Verhalten des gesamten Netzwerkes betreffen. Anschließend werden die einzelnen Geräte generiert. In der NED-Datei sind die benötigten Buffer der unterschiedlichen Nachrichtenklassen zu definieren, während in der „ini“-Datei die Schedules der TT-, die Konfiguration der RC- und der weiteren Nachrichtenklassen stehen. Zusätzlich werden die „TrafficApps“ und „TrafficSinks“ mit ihrem Verhalten beschrieben.

5.2 Simulation und Evaluation

Um in der Simulation die in Kapitel 4.3.1 beschriebenen Metriken aufzuzeichnen, bieten sich die *Resultrecorder*- und *Resultfilter*-Interfaces der OMNeT++ Simulationsumgebung an. Diese sind ab der Version 4.2rc1 (OMNeT++ Community [c]) verfügbar und wurden für eine einheitliche Möglichkeit zur Aufzeichnung von Simulationsdaten hinzugefügt. Über ein Publish-subscribe System können in jedem Modul vorhandene Filter und Recorder registriert werden, die Signale aus der Simulation abgreifen und vor der Aufzeichnung bearbeiten. Die Resultfilter bieten eine Möglichkeit, die Darstellung der aufzuzeichnenden Werte zu wählen. Durch das Hintereinanderschalten mehrerer Filter ist eine Vielzahl von Möglichkeiten abbildbar. Mit dem Resultrecorder wird anschließend die Bearbeitung der festgehaltenen Werte definiert. Durch die von der API zur Verfügung gestellten Recorder ist es beispielsweise schon möglich, die Werte zu addieren, zu subtrahieren oder zu zählen. Der folgende Code dient zur Verdeutlichung des Zusammenspiels von Filter und Recorder:

```
1 @signal[rxPk](type=EtherFrame);
2 @statistic[receivedPackets](source="packetBytes(rxPk)"; record=sum");
```

Mit der *@signal*-Annotation wird das Modul für den Empfang des „rxPk“ Signals eingetragen. Anschließend können verschiedene Statistiken gebildet werden. In dem Beispiel zeigt der Resultfilter „packetBytes“, dass die Ausgabe der Paketgröße in Bytes erfolgt. Mit der Auswahl des Resultrecorders „sum“ wird ein Zusammenfassen der aufgezeichneten Werte in einer Summe erreicht.

5.2.1 Aufzeichnung der Metriken

Das Aufzeichnen des Wertes einer Metrik wird in der Simulation durch das Eintreffen oder Verlassen eines Paketes in einer Komponente ausgelöst. Für jede Metrik gibt es demnach unterschiedliche Aufzeichnungspunkte, wobei nicht jeder eindeutig zu ermitteln ist, beziehungsweise keine weitere Information für die Evaluation bietet. Die Aufzeichnung der Metriken in Fahrzeugnetzen kann sowohl beim Sende-/Empfangsmodul eines Gerätes als auch bei den Buffern, in denen die Nachrichten vorgehalten werden, stattfinden. Somit kann eine Auswertung des zeitkritischen Übertragungsverhaltens und der Auslastung des Netzwerkes erfolgen. Tabelle 5.2.1 zeigt die aus dem Konzept (4.3.1) stammenden Metriken mit ihren wichtigsten Eigenschaften.

Metrik	Einheit	MAC	Buffer	Modi	Signale
Latenz	Sekunde	x	x		rxPk/txPk
Jitter	Sekunde	x	x		rxPk/txPk
Paketverlustrate	Anz. Pakete	x		Intervall (1/2)	rxPk
Kapazität	Prozent	x	x	Intervall (1/2)	rxPk
Verfüg. Bandbreite	Prozent	x		Intervall (1/2)	rxPk
Linkauslastung	Prozent	x		Intervall (1/2)	rxPk
Buffergröße	Anz. Pakete		x		rxPk

Tabelle 5.1: *Überblick über die relevanten Metriken in Automobilnetzwerken und ihren Aufzeichnungspunkten in der Simulation. (vgl. Kempf 2014)*

Die Tabelle zeigt neben der Einheit, in der die Metrik angegeben wird, die beiden Aufzeichnungspunkte MAC und Buffer. Als Modi werden die beiden im Konzept beschriebenen Aufzeichnungsmodi bezeichnet, die eine präzise oder schnelle Aufzeichnung erlauben. In der letzten Spalte sind die auslösenden Signale zu sehen, die sich entweder auf den Empfang der Pakete (rxPk) oder auf das Verschicken eines Paketes (txPk) beziehen.

Wenn ein Paket auf einem Link eintrifft, wird bis auf die Buffergröße zu jeder Metrik ein Wert aufgezeichnet. Dies könnte beim Verlassen ebenfalls der Fall sein, jedoch ist dieses bei den Metriken, die sich auf eine Auslastung beziehen, nicht nötig, weil die Gegenstelle die gleiche Auslastung besitzt. Daher ist diese Ansicht optional und wird hier nicht weiter aufgeführt. Für die Latenz und den Jitter kann durch den Aufzeichnungspunkt eines gesendeten Paketes der Verlauf des Paketes genau verfolgt werden.

Der Aufzeichnungspunkt im Buffer ist nur für die Latenz, den Jitter, die Kapazität und die Buffergröße vorgesehen, weil nur bei diesen ein Interesse für die Aufteilung des Verkehrs auf die Nachrichtenklassen besteht. Die anderen drei Metriken werden zur Analyse der Verbindung genutzt.

Tendenziell können die Aufzeichnungsmodi über eine Intervallzeit bei allen Metriken gemessen werden. Aufgrund der Werte, die jedes Paket als Latenz, dem Jitter und der Buffergröße liefert, ist eine Messung über ein Intervall nicht erforderlich und würde zudem noch Spitzen in der Latenz und dem Jitter marginalisieren.

5.2.2 Ausgabe und Evaluierung der Metriken

OMNeT++ erzeugt für die Kennzahlen, die während der Simulation aufgenommen werden, Ausgabevektoren, die nach der Simulation analysierbar sind. Zur Auswertung eignet sich das in OMNeT++ integrierte Scave Tool, das Vergleiche zwischen den Simulationsdurchläufen durch Aufbereitung der Werte ermöglicht. Jeder Wert kann demnach gefiltert ausgegeben werden, wodurch ein Vergleich einer Metrik über das gesamte Netzwerk erfolgen kann. Die Auswertung kann anschließend mit den reinen Zahlenwerten oder dem aufbereiteten Graph innerhalb des Tools vorgenommen werden. Angesichts der unterschiedlichen Aufzeichnungsmodi muss geklärt werden, ob sich auch der performantere Modus zur Auswertung eignet, und wie die Auswertung der einzelnen Metriken erfolgt.

Vergleich der Modi für Intervallaufzeichnungen

Um zu zeigen, inwiefern sich die beiden Aufzeichnungsmodi voneinander unterscheiden und ob auch eine Auswertung mit dem performanten Modus möglich ist, soll an dieser Stelle ein Vergleich des präzisen Aufzeichnungsmodus und des performanten Modus anhand der Linkauslastung gezeigt werden. Die Intervallzeit beträgt 20 ms, und es wird eine erhöhte Last zwischen zwei Switches erzeugt. Für den präzisen Modus bedeutet dies, dass bei dem Eintreffen eines Paketes auf dem Link und beim Herausfallen eines Paketes aus dem Intervall ein Durchschnittswert über die letzten 20 ms für die Linkauslastung erzeugt wird. Bei dem performanten Modus wird nur zu der vollen Intervallzeit, also alle 20 ms, ein Wert aufgezeichnet. Abbildung 5.3 zeigt die Werte beider Aufzeichnungsmodi vom Start der Simulation bis zu einem Zeitpunkt von 60 ms.

Die Werte des präzisen Modus steigen zu Beginn an und erzeugen eine Linkauslastung bis zu einem Maximum von 85 %. Anschließend alterniert der Wert zwischen einer Auslastung von 80 % und 85 %, wodurch das Herausfallen eines Wertes und das Ankommen neuer Pakete sichtbar wird. In den 60 ms zeichnet der andere Modus genau drei Werte auf. Zu jedem Zeitpunkt zeigt der Wert die gleiche Auslastung wie der Wert des präzisen Modus an. Allerdings wird das alternierende Verhalten nicht dargestellt.

Der performante Modus gibt Werte, die über eine gewisse Dauer gleich bleiben, sehr genau an. Daher eignet er sich für die Aufzeichnung in Fahrzeugnetzwerken, bei dem ein Großteil der Kommunikation wiederkehrend ist. Bei der Wahl kleiner Intervalle können zudem auch

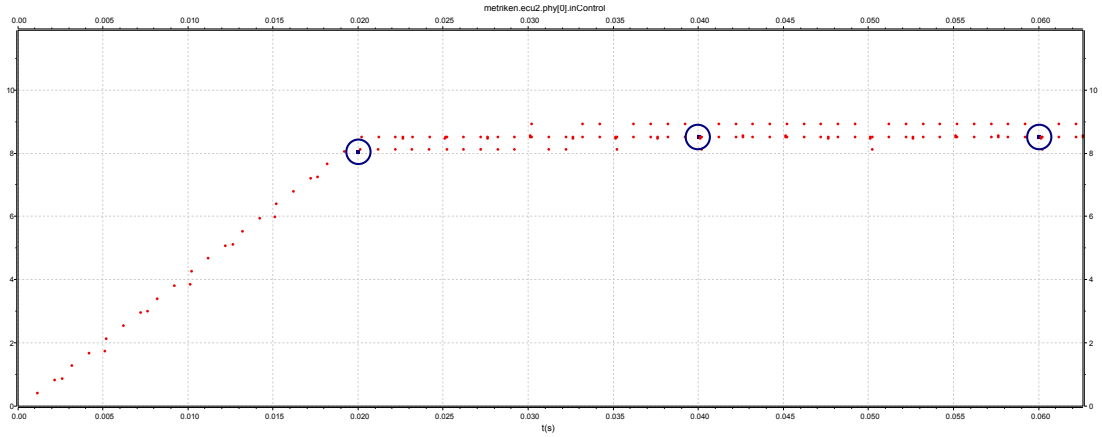

Abbildung 5.3: *Der präzise und der performante Aufzeichnungsmodus im Vergleich. Während beim erstgenannten jedes eintreffende Paket ein Wert erzeugt, trifft dies beim performanten Modus nur zu jedem vollen Zeitintervall zu.*

Schwankungen relativ gut erkannt werden, wobei er immer noch weniger Berechnungen als der präzise Modus benötigt. Eine Evaluation ist somit durch beide Modi möglich.

6 Validierung des Generierungsprozesses

Um die Eignung der domänenspezifischen Sprache auf die Erstellung von Netzwerken und insbesondere von heterogenen Netzwerken zu prüfen, werden die Simulationsergebnisse von generierten Netzwerken mit der gewünschten Eingabe verglichen. Aufgrund der bereits erfolgten Validierungen und Verifikationen der genutzten Simulationsmodelle in anderen Arbeiten (vgl. 4.2.1) ist bei erfolgreicher Validierung der Sprache ein Einsatz für eine effiziente, simulationsbasierte Analyse möglich. Nachfolgend werden erstellte *nedG*-Dateien die verschiedenen Simulationsmodelle ansprechen und überprüfen, ob sich das gewünschte Nachrichtenverhalten in ausgewählten Metriken wiederfinden lässt. Dafür ist in erster Linie nur die Auswertung der Anzahl der übertragenen Pakete erforderlich, weil damit eine eindeutige Prüfung des erstellten Verhaltens stattfindet.

6.1 Konfiguration eines Time-Triggered Ethernet Backbones

Um die Konfigurationsgenerierung eines Time-Triggered Ethernet Backbones zu testen, soll ein Netzwerk mit drei Endgeräten, die über zwei Switches miteinander verbunden sind, erstellt werden. Die gebildete nedG-Datei kann dem Anhang (A.1) entnommen werden. In der gewünschten Topologie sind die Endgeräte „e1“ und „e2“ an den Switch „s1“ anzuschließen, während der Switch „s2“ mit dem ersten Switch und dem letzten Endgerät „e3“ verbunden wird. Für die Lastgenerierung wird eine Nachricht jeder Time-Triggered-Klasse gewählt, wodurch drei Nachrichten benötigt werden. Die beiden Endgeräte des ersten Switches senden diese Nachrichten an das Endgerät „e3“, wodurch eine Übertragung über die Verbindung zwischen den beiden Switches stattfindet. Die Time-Triggered Nachricht soll eine Periode von $500\mu s$ und einen Payload von $100Byte$ besitzen und wird von „e1“ nach „e3“ gesendet. Somit erzeugt sie auf den Links eine Auslastung von:

$$
\begin{aligned}
Auslastung_{TT} &= \frac{\ddot{U}bertrageneDatenmenge}{MaximaleDatenmenge} && (6.1)\\
&= \frac{(Payload + Header) \cdot \frac{Datenrate}{Periode}}{MaximaleDatenmenge} && (6.2)\\
&= \frac{(100B + 18B) \cdot 8 \cdot \frac{100Mbit/s}{500\mu s}}{100Mbit} \approx \underline{\underline{1.8\%}} && (6.3)
\end{aligned}
$$

Für die Rate-Constrained Nachricht wird vom Sender „e2“ eine Nachricht mit einer verdoppelten Senderate, also einer Periode von $250\mu s$ nach „e3“ geschickt. Diese Nachricht erzeugt somit eine Auslastung von:

$$Auslastung_{RC} = \frac{(100B + 18B) \cdot 8 \cdot \frac{100Mbit/s}{250\mu s}}{100Mbit} \approx \underline{\underline{3.77\%}} \quad (6.4)$$

Als letztes wird nach einer Sekunde eine Best-Effort Nachricht mit einer einheitlichen Größe von $200Byte$ von „e1“ nach „e3“ geschickt. Die Periode, mit der die Nachricht gesendet wird, soll dabei zwischen $100\mu s$ und $300\mu s$ schwanken, das eine nicht konstante Datenübertragungsrate zur Folge hat. Die Links werden somit nach einer Sekunde mit weiteren $5,8\%$ bis $17,4\%$ belastet. Zusammengefasst müsste die theoretische maximale Linkauslastung zwischen den Switches und auf dem Link „e3“ wie folgt aussehen:

$$LinkAuslastung_{e3} = Auslastung_{TT} + Auslastung_{RC} + Max(Auslastung_{BE}) \quad (6.5)$$
$$\approx 23.1\% \quad (6.6)$$

Abbildung 6.1 zeigt das generierte Netzwerk mit der graphischen Ansicht der „.ned“-Dateien in OMNeT++. Die parametrisierten Nachrichten der Klassen TT, RC, und BE wurden als Sender-Receiver-Beziehungen der Abbildung hinzugefügt.

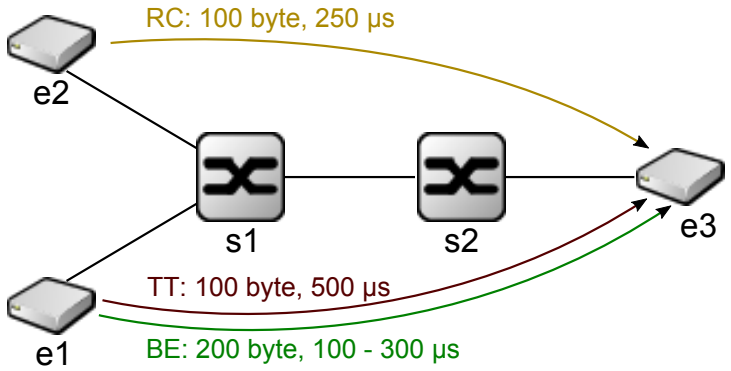

Abbildung 6.1: *Ein Time-Triggered Ethernet Netzwerk mit drei Endgeräten und zwei Switches. Eine Nachricht jeder Nachrichtenklasse wird an das Endgerät „e3“ gesendet.*

Die Ausführung der Simulation beträgt zwei Sekunden, wobei nach einer Sekunde die Best-Effort Nachrichten zu den von Anfang an gesendeten RC und TT Nachrichten hinzukommen. Während dieser Zeit wird für alle Links die Auslastung aufgezeichnet und anschließend mit den berechneten Werten verglichen. Das Aufzeichnungsintervall beträgt 50 ms.

Der Graph 6.2 zeigt die Linkauslastung der empfangenden Ports von „s1“, „s2“ und „e3“. Der Port, der mit „e2“ verbunden ist, empfängt über die gesamte Simulationsdauer die Rate-Constrained Nachrichten. Dieses Verhalten ist in dem Graph an der roten Linie deutlich zu erkennen. Mit einem Wert, der bei ungefähr 3.7 % liegt, spiegelt er den berechneten Wert aus der Formel 6.4 wider.

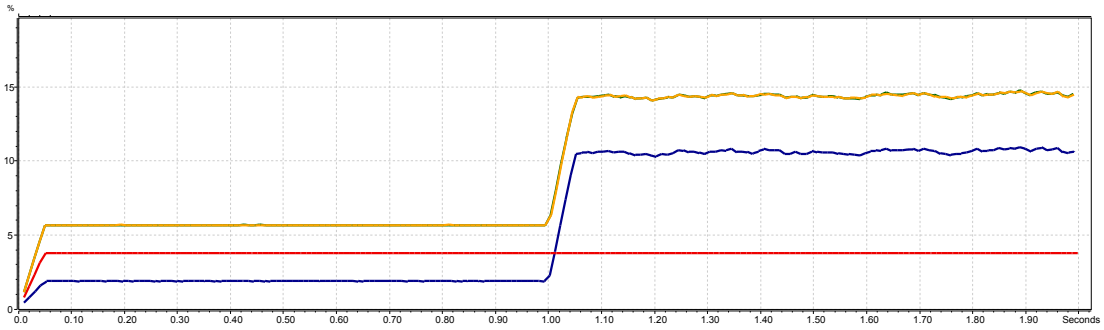

Abbildung 6.2: *Die Auslastung der empfangenden Links:*
Rot: Link von „s1“ zu „e2“ - RC
Blau: Link von „s1“ zu „e1“ - TT ; ab 1s: TT, BE
Gelb: Link von „s2“ zu „e3“ - TT, RC ; ab 1s: TT, RC, BE

Der andere Port des Switches empfängt die Best-Effort- und Time-Triggered Nachrichten. Dieser ist in dem Graph blau dargestellt und hat eine Auslastung wie in der Berechnung 6.3 von 1.8 %. Nach der ersten Sekunde ist deutlich zu erkennen, wie die Auslastung steigt, weil die Best-Effort Nachrichten ebenfalls empfangen werden.

Die gelbe Linie zeigt sowohl die Auslastung des eintreffenden Ports von „s2“ als auch vom Endgerät „e3“. Hier ist die höchste Auslastung. Zuerst wird sie ausschließlich durch die Nachrichten der Virtual Links mit $1.8\% + 3.7\% = 5,5\%$ beeinflusst. Anschließend kommt noch der Best-Effort Verkehr dazu. Dieser pendelt um die 14 % Marke, obwohl die theoretische maximale Auslastung vorher mit 23.1 % (Formel 6.6) berechnet wurde. Da sich die Auslastung durch den BE-Traffic aber zwischen $5,8\%$ und $17,4\%$ befindet, wird durch die zufallsbasierte Verteilung der Pakete das Niveau des Mittelwerts erreicht. Dieser Wert wurde über ein Intervall von 50 ms gebildet.

Das gezeigte Beispiel erzeugt aus einer „nedG“-Datei mit 52 Zeilen ein FIBEX-konformes XML mit 367 Zeilen und die benötigte Konfiguration für die Modelle der OMNeT++ Simulationsumgebung mit 355 Zeilen, die in 12 Dateien aufgeteilt sind. Zudem ist es weniger fehleranfällig und erzeugt einen funktionierenden Schedule für Time-Triggered Nachrichten, wodurch der Generierungsprozess für die Entwicklung eines Time-Triggered Ethernetnetzwerkes effizient genutzt werden kann.

6.2 Konfiguration eines Bussystems

Für die Validierung der Generierung eines CAN-Bussystems soll überprüft werden, ob es möglich ist einen Bus mit vier Endgeräten zu modellieren und Dataframes zwischen den Endgeräten zu verschicken. Die Endgeräte werden direkt mit einem Bus verbunden und zwei Nachrichten erstellt. Die gebildete nedG-Datei ist im Anhang(A.2) zu finden. Die erste Nachricht mit der ID 1 wird von „e3“ zu den Endgeräten „e1“ und „e4“ versendet. Die zweite Nachricht mit der niedriger priorisierten ID 2 schickt „e1“ an „e2“ und „e3“. Somit ist zu erwarten, dass durch das Bussystem bedingt jedes Endgerät alle Nachrichten empfängt, jedoch „e1“ und „e4“ die Nachricht mit der ID 1 verarbeiten, während „e2“ und „e4“ die Nachricht mit der ID 2 erwarten. Abbildung 6.3 fasst das generierte Netzwerk mit der graphischen Ansicht der „.ned“-Dateien zusammen.

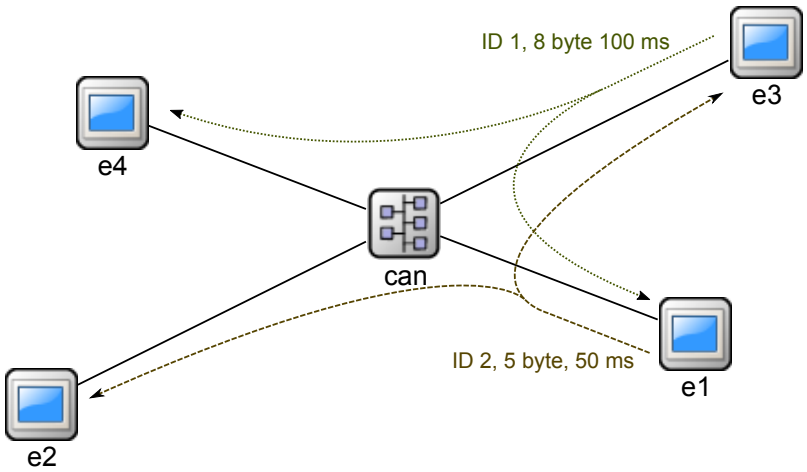

Abbildung 6.3: *Vier Endgeräte werden an einem CAN angeschlossen. Das Endgerät „e1“ sendet genau wie das Endgerät „e3“ eine Nachricht an zwei Empfänger.*

Die Nachricht mit dem Identifier 1 wird alle $100ms$ mit $8Byte$ Payload versendet, während die Nachricht mit der ID 2 doppelt so häufig, nämlich alle 50 ms, mit 5 Byte Payload das Endgerät verlässt.

Bei einer Simulationsdauer von zwei Sekunden müssen demnach alle Endgeräte 60 Nachrichten empfangen. Diese teilen sich in 20 Nachrichten von Endgerät „e3“ und 40 Nachrichten von „e1“ kommend auf. Nach der Simulation soll eine Überprüfung dieses Verhaltens in den aufgezeichneten Vektoren die erfolgreiche Generierung validieren.

Wie bei der CAN-Modellierung in Abschnitt 4.2.1 teilt sich ein CAN-Knoten in den Port und die Buffer auf. In der Simulation wird an diesen Punkten der Wert der empfangenen und gesendeten Pakete aufgezeichnet. Tabelle 6.1 zeigt die Aufteilung der Pakete:

	e1	**e2**	**e3**	**e4**
Port Tx:	20	–	40	–
Port Rx:	60	60	60	60
Buffer Rx:	40	20	20	40

Tabelle 6.1: *Aufteilung der Pakete in den einzelnen Modulen der Endgeräte.*

Für die Ports aller Endgeräte beträgt der Wert der empfangenen Frames wie erfordert 60 Nachrichten. Die Werte der gesendeten Pakete auf den Ports der Endgeräte „e1“ und „e2“ ergeben die erforderten 20 und 40. In den Buffern befinden sich die Werte für die Nachrichten, die in dem jeweiligen Endgerät erwartet wurden. Diese stimmen mit dem konfigurierten Netzwerk der domänenspezifischen Sprache überein. Somit ist es möglich, ein Bussystem in der DSL zu generieren. Im nächsten Schritt kann die Evaluierung der domänenübergreifenden Kommunikation zur Evaluierung heterogener Netzwerke durchgeführt werden.

6.3 Konfiguration einer heterogenen Lösung

Zur Validierung der Generierung einer heterogenen Lösung müssen die Switches des Time-Triggered Ethernets über Gateways mit den Bussystemen verbunden werden. Anschließend können Endgeräte mit heterogenen Nachrichten die Geräten aus anderen Domänen über die Gateways erreichen. Die erstellte nedG-Datei ist im Anhang (A.3) zu finden. Abbildung 6.4 zeigt die generierte Topologie mit sieben Endgeräten, wovon fünf an den drei Bussystemen und zwei Endgeräte direkt an dem Ethernetbackbone angeschlossen sind. Für den Test des Nachrichtenaustauschs zwischen den Systemen wurden vier Nachrichten, die von einer ECU eines Bussystems an ECUs des gleichen Subnetzes, des Backbones und an Endgeräte aus anderen Subnetzen geschickt wurden, hinzugefügt. Jedes Bussystem wird über ein Gateway mit einem Switch verbunden.

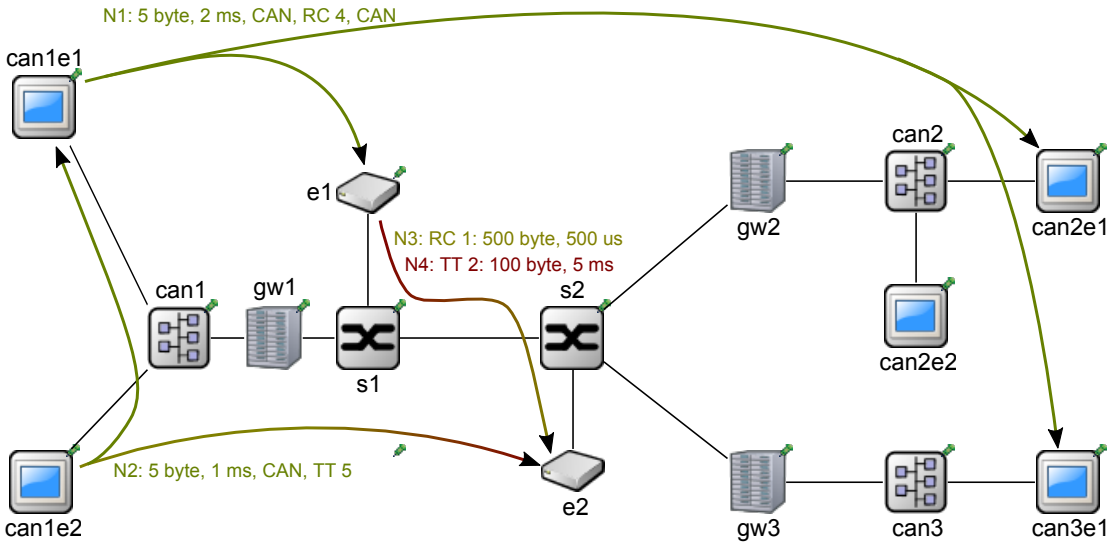

Abbildung 6.4: *Ein heterogenes Netzwerk mit drei Bussystemen, die über ein Backbone miteinander kommunizieren können. Die sieben Endgeräte tauschen dabei vier Nachrichten miteinander aus. Die heterogenen Nachrichten besitzen neben dem Payload und der Periode noch die verschiedenen Nachrichtenklassen.*

Die Nachrichten teilen sich wie folgt auf: Die erste Nachricht wird von dem Endgerät „can1e1“ an drei Empfänger gesendet, die sich alle in unterschiedlichen Subnetzen befinden und wird als Rate-Constrained Nachricht mit der ID 4 im Backbone weitergeleitet. Im Switch „s2“ muss sie an zwei Bussysteme weitergeleitet werden, weil sich die Empfänger „can2e1“ und „can3e1“ in unterschiedlichen Subnetzen befinden. Die zweite Nachricht hat zum Ziel sowohl eine Time-Triggered Nachricht im Backbone zu übertragen als auch die

Möglichkeit, Empfänger im eigenen Subnetz zu adressieren. Sie wird von „can1e2“ an die Endgeräte „can1e1“, das am gleichen CAN angeschlossen ist und „e2“, das sich als TTE-Unit am Backbone befindet, geschickt. Um die Auswirkung der heterogenen Nachrichten auf das Netzwerk zu testen und das Backbone höher zu belasten, werden zwischen den Switches Rate-Constrained und Time-Triggered Nachrichten, die vom Endsystem „e1“ zum Endgerät „e2“ geschickt werden, hinzugefügt.

Um die Generierung der Nachrichten zu überprüfen, wird die Simulation für eine Sekunde ausgeführt und die empfangenen Nachrichten der einzelnen Systeme mit den berechneten Werten verglichen. Zudem soll mit einer Latenzmessung in einem Gateway der Einfluss der domänenübergreifenden Kommunikation deutlich gemacht werden. Die folgende Tabelle 6.2 zeigt die aufgezeichneten Werte der Simulation von ausgewählten Systemen und aus welchen Nachrichten sie sich zusammensetzen.

	Nachrichten	**Anzahl der Pakete**
can1e1:	N2	1000
gw1:	N1 + N2	500 + 1000 = 1500
s1:	N1 + N2 + N3 + N4	500 + 1000 + 2000 + 200 = 3700
s2:	N1 + N2 + N3 + N4	500 + 1000 + 2000 + 200 = 3700
e1:	N1	500
e2:	N2 + N3 + N4	1000 + 2000 + 200 = 3200
gw2:	N1	500
can2e1:	N1	500
gw3:	N1	500
can3e1:	N1	500

Tabelle 6.2: *Empfangene Nachrichten der einzelnen Systeme in dem heterogenen Netzwerk mit der Anzahl der Pakete pro Nachricht.*

Die aufgezeichneten Werte der Simulation stimmen mit der Nachrichtenverteilung überein. Demnach übertragen die Backboneswitches 3700 Pakete in einer Sekunde, die sich aus allen vier Nachrichten zusammensetzen. Die rechte Seite des Netzwerkes empfängt nur die Nachricht N1, wodurch alle Gateways und die Endgeräte der CANs „can2“ und „can3“ die 500 Pakete erhalten. Eine genaue Betrachtung liegt auf dem Endgerät „e2“, das sowohl Nachrichten aus einem anderen Subnetz als auch Rate-Constrained und Time-Triggered Nachrichten des Backbones empfängt und zudem die größte Anzahl an Paketen aller Endsysteme mit 3200 empfängt.

Der Graph 6.5 zeigt die Latenzmessung der drei Nachrichten mit den CTids 1 (N3),2 (N4) und 5 (N2). Die geringste Latenz tragen die Pakete der Rate-Constrained Nachricht (N4). Diese wird durch das direkte Weiterleiten in den Switches, wenn kein höher priorisiertes Time-Triggered Paket übertragen wird, erreicht. Zu Beginn steigt die Latenz einiger Pakete jedoch

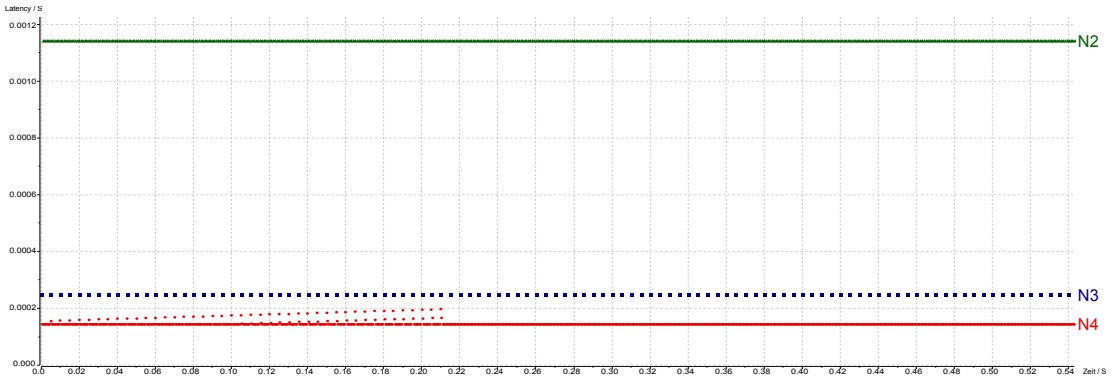

Abbildung 6.5: *Latenzverhalten in einem heterogenen Netzwerk anhand des Endgerätes e2, das drei Nachrichten empfängt. N3 (TT) und N4 (RC) sind reine Backbonenachrichten. N2 wird über ein Gateway gesendet.*

an, was an dem Schedule der Time-Triggered Nachrichten liegt. Die Pakete der Nachrichten N3 und N2 sind hingegen konstant, weil sie der Time-Triggered Nachrichtenklasse angehören und gescheduledte Übertragungsfenster besitzen. Die Pakete der Nachricht „N3“, die nur innerhalb des Backbones übertragen werden, haben eine wesentlich geringere Latenz als die Pakete, die von dem CAN „can1“ ankommen und durch die Übersetzung in dem Gateway „gw1“ zusätzlich verzögert werden. Die hohe Latenz lässt darauf schließen, dass der Schedule zu Beginn des Zyklus ein Fenster vorsieht, während das Paket kurz danach eintrifft und somit immer um einen Zyklus verschoben versendet wird.

Die Validierung hat gezeigt, dass sich die domänenspezifische Sprache zur Generierung komplexer, heterogener Netzwerke eignet und sich diese fehlerfrei, effizient und somit erfolgreich erstellen lassen.

7 Fallstudie zur Untersuchung heterogener Netztopologien

Nachdem im vorigen Kapitel dargelegt wurde, dass sich die domänenspezifische Sprache zur Erstellung eines heterogenen Netzwerkes eignet, soll eine Fallstudie zeigen, wie ein komplexes Beispiel mit einer großen Anzahl an Endgeräten und Nachrichten aussehen kann. Dazu wird ein Referenzdesign erstellt, das sich an dem heute üblichen Design nach Domänen orientiert, jedoch über ein Backbone verfügt, das bandbreitenintensive Anwendungen zulässt. Mit einer anschließenden Topologieveränderung wird demonstriert, wie sich der Einbettungsprozess zur Generierung, Simulation und Evaluation einsetzen lässt.
Das Datenmodell umfasst eine Kommunikationsmatrix mit 92 Nachrichten, die von 30 Endgeräten insgesamt 204 mal empfangen werden. Somit findet eine Duplizierung in den Switches und der mehrfache Empfang einer Nachricht an verschiedenen Endgeräten eines Bussystems statt. Da es sich bei vielen Nachrichten um sicherheitsrelevante Steuerdaten handelt, werden 40 der 92 Nachrichten über Time-Triggered Virtual Links im Backbone übertragen. Die restlichen werden mit Bandwidth Allocation Gaps als Rate-Constrained Traffic behandelt. Eine Ausnahme stellt eine Best-Effort Nachricht dar, die mit erhöhtem Traffic und mit einer Schwankung in der Übertragungsrate das Netzwerk nach einer gewissen Zeit beeinflussen soll.

7.1 Hierarchisches Netzwerk mit abstrahierten Namen

Abbildung 7.1 zeigt ein hierarchisches Design nach Domänen. Dafür wurde für jedes Bussystem ein Switch hinzugefügt, der auf der einen Seite das Bussystem über ein Gateway und auf der anderen Seite das Backbone verbindet. In der Mitte existiert ein zentraler Punkt als Backboneswitch, an dem sternförmig die anderen Switches angeschlossen sind.

Die Endgeräte gehören an die ursprünglichen Busse der Domänen, so lange keine hohe Bandbreite benötigt wird. Direkt mit dem Backbone sind Endgeräte mit bandbreitenintensiven Anwendungen verbunden. Damit das Netzwerk eine hohe Auslastung erreicht und um zukünftige Anforderungen zu zeigen, werden dem Backbone Kameras mit unterschiedlicher Positionierung im Fahrzeug hinzugefügt. Durch die Verteilung der Switches bekommt jeder Switch einer Domäne eine Kamera zugeteilt. Diese senden alle gleichzeitig und durchgehend

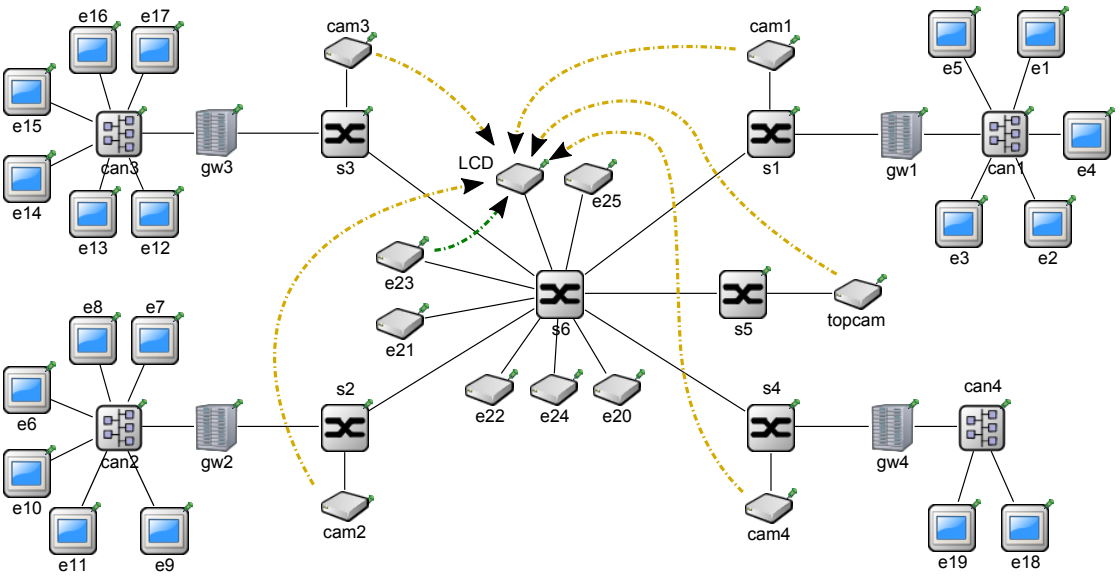

Abbildung 7.1: *Ein hierarchisches Design nach Domänen mit fünf hinzugefügten Kameras und eingezeichneten Rate-Constrained- (Gelb) und Best-Effort- (Grün) Nachrichten.*

an den LCD, damit der Nutzer ohne Verzögerung zwischen den Kameras wechseln kann. Für die Verbindungen innerhalb des Backbones gilt eine Übertragungsgeschwindigkeit von $100Mb/s$ pro Link. Um das Netzwerk nicht zu überlasten, ist es erforderlich, dass die hinzugefügten Kameras zusammen eine maximale Auslastung von 75 % auf dem Link von „s6“ zu dem LCD verursachen. Weil es sich um fünf Kameras handelt, steht somit eine maximale Bandbreite von $\frac{75\% \cdot 100Mb/s}{5} = 15Mb/s$ für jede Kamera zur Verfügung. Das ergibt bei einem Payload von $1500Byte$ eine maximale Paketanzahl pro Sekunde von:

$$\textit{Pakete pro Sekunde} = \frac{\textit{Maximale Bandbreite}}{\textit{Paketgröße in Bit}} \tag{7.1}$$

$$= \frac{15 \cdot 1024 \cdot 1024 \frac{bit}{s}}{1538Byte \cdot 8} \approx 1310 \tag{7.2}$$

Dies resultiert in einer Nachrichtenperiode von $\frac{1000}{1310} \approx 0.76ms$. Die Kameradaten werden dabei als Rate-Constrained-Nachrichten übertragen.

Für das Scheduling wird ein einfacher Algorithmus genutzt, der zu Beginn eines Zyklus die Fenster der Time-Triggered Nachrichten hintereinander berechnet und anschließend Platz

für andere Nachrichtenklassen lässt. Dieses Verhalten sollte sich gerade bei mehreren Time-Triggered-Virtual Links, die über den gleichen physikalischen Link laufen, deutlich in den Rate-Constrained Daten widerspiegeln.

7.2 Simulationsergebnisse

Nachdem das Netzwerk mit der domänenspezifischen Sprache generiert wurde, wird die Simulation für eine Echtzeit-Sekunde ausgeführt. Nach $100ms$ wird der Best-Effort Verkehr von „e23“ ebenfalls zum LCD geschickt. Dieser besitzt einen Payload von $800Byte$ und wird zwischen $1300\mu s$ und $700\mu s$ gesendet. Dies entspricht einer zusätzlichen Bandbreite von $5 - 9Mb/s$.

Abbildung 7.2 zeigt die Linkauslastung unterschiedlicher physikalischer Links an. Dafür wurde der Link mit der höchsten Auslastung des Netzwerkes ausgewählt. Im Vergleich dazu steht ein Switchlink mit den Daten einer Kamera und des Links, über den die Best-Effort Nachrichten versendet werden.

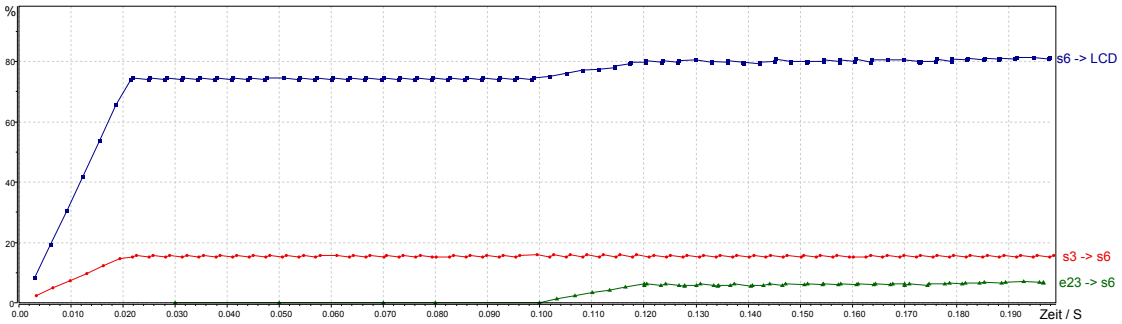

Abbildung 7.2: *Die Linkauslastung für drei ausgewählte Links in der hierarchischen Netztopologie. Deutlich ist der Anstieg nach* $100ms$ *durch den Best-Effort Traffic zu erkennen. Die Auslastung des LCD-Links steigt auf über 80 %.*

Deutlich ist zu sehen, wie nach $100ms$ die Linkauslastung von „e23“ nach „s6“ ansteigt und die Auslastung anschließend zwischen $6Mb/s$ und $8Mb/s$ schwankt. Dieses Verhalten ist aufgrund der Intervallaufzeichnung, die in diesem Fall $20ms$ beträgt, zu erklären. Das verdeutlicht auch die zu Beginn steigenden Kurven. Die verursachte Datenübertragung des Best-Effort Traffics schlägt sich mit kleiner Verspätung auf die Verbindung zwischen dem Switch und dem LCD nieder. Es ist deutlich zu sehen, dass die Linkauslastung auf dieser Verbindung wie vorausberechnet 75% beträgt, bis sie auf über 80% ansteigt. Doch wie verhält es sich mit der Latenz der Nachrichten, die für die Kameradaten übertragen werden? Abbildung 7.3 zeigt die Latenz aller im LCD eintreffenden Pakete.

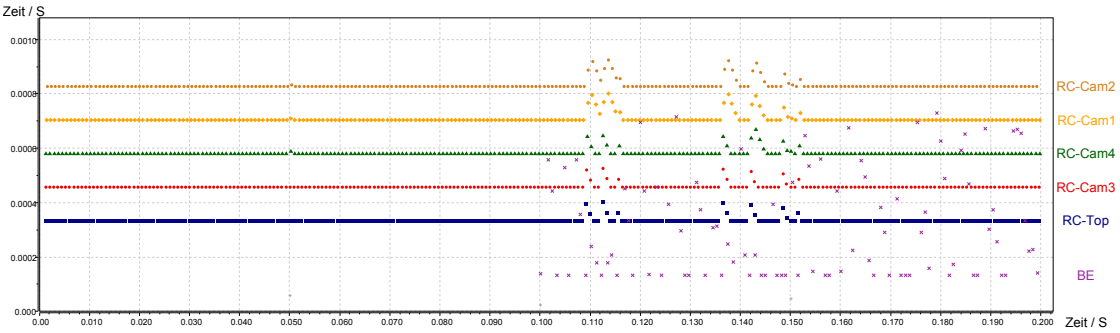

Abbildung 7.3: *Die Latenzen der empfangenen Pakete im LCD mit Aufteilung nach Kameras und Best-Effort Traffic. Deutlich ist ein leichtes Jittern der Latenz mit Ankunft der BE-Nachrichten erkennbar.*

Die Abstände zwischen den Paketen der einzelnen Kameras besitzen die Dauer der übertragenen Pakete. Die Übertragung über einen Zwischenknoten ist durch die zweifache Länge einer Paketübertragung der Topkamera sichtbar. Das Eintreffen der Nachrichten von den verschiedenen Kameras ist durch das Time-Triggered Scheduling beeinflusst. Während die Topkamera die Verbindung zum Switch „s6“ nicht mit anderen Endgeräten teilen muss und somit am Anfang eines Zyklus direkt den RC-Traffic senden kann, werden die anderen Kameras durch die Verbindung zwischen ihren Switches der Domäne und dem zentralen Switch verzögert. Durch das frühere Eintreffen der Nachrichten im Switch „s6“ kann eine Übertragung an den LCD zuerst bei diesen stattfinden. Abbildung 7.4 zeigt, wie sich die Nachrichten der Kamera 3 in den einzelnen Knoten bezüglich der Latenz verhalten. Die Aufzeichnungspunkte der Pakete befinden sich demnach in „s3“, „s6“ und im Empfängersteuergerät.

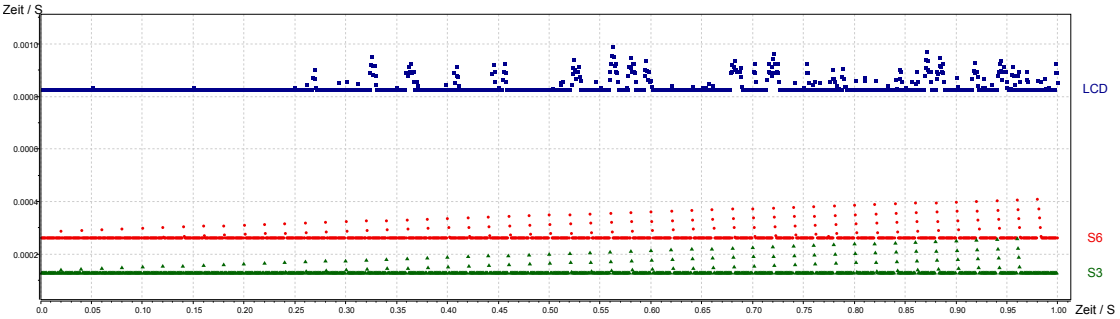

Abbildung 7.4: *Die Latenzen der Pakete von Kamera 3, die über „s3“ und „s6 zum LCD gelangen. Eine Verzögerung einzelner Pakete ist durch das TT-Scheduling erkennbar.*

Auffällig sind die Latenzen der Pakete in den beiden Switches. Diese werden durch die Fenster der Time-Triggered Nachrichten beeinflusst. Dabei stellt sich jedoch die Frage, wie die sprunghafte Zunahme in einem festen Intervall zustande kommt. Wie bereits erwähnt, wurde ein Scheduler gewählt, der die Fenster der Time-Triggered Nachrichten hintereinander berechnet. Somit gibt es innerhalb des Zyklus Zeiträume, in denen kein Paket einer anderen Nachrichtenklasse gesendet werden kann. Zu diesem Zeitpunkt treffen allerdings weitere Pakete ein, so dass sich die Latenzen dementsprechend erhöhen. Aufgrund des Gegeneinanderlaufens der Kamerapakete mit den unterschiedlichen Schedules des TT-Traffics und anderen Rate-Constrained Nachrichten verschiebt sich die Anzahl der Pakete, die während der blockierten Verbindung eintreffen. Ein Abbau der Latenz ist durch kleinere Bandwidth Allocation Gaps beim Senden in den Switches erreicht worden. Kurz vor Erreichen der Simulationsdauer von einer Sekunde fällt zudem in Switch „s3“ auf, dass das Aufstauen der Pakete nicht mehr vorkommt. Somit wurde an dieser Stelle der Zeitpunkt erreicht, an dem das Weiterleiten der Pakete nicht mehr mit den reservierten Fenstern kollidiert.

Abbildung 7.5 zeigt, wie sich die Time-Triggered Nachrichten in dem Netzwerk verhalten.

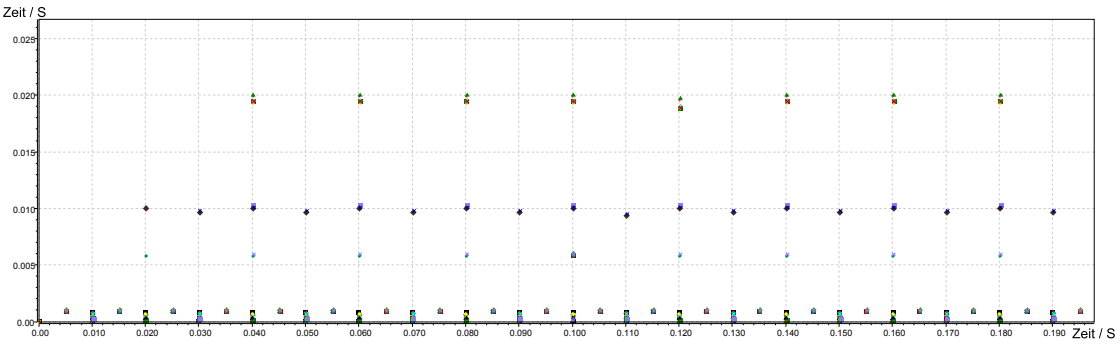

Abbildung 7.5: *Jede Nachricht, die als Time-Triggered in Switch „s1“ an „s6"gesendet wird, besitzt eine konstante Latenz. Die hohe Latenz von* $20ms$ *kommt durch Verzögerungen am Gateway zustande.*

Dazu werden die Latenzen der Pakete aller Time-Triggered Virtual Links, die von „s1“ nach „s6“ gesendet werden, evaluiert. Zuerst ist anzumerken, dass alle Nachrichten, die zu einem Virtual Link gehören, eine fast konstante Latenz aufweisen. Dies ist durch die vorbestimmten Zeitfenster beeinflusst. Allerdings lassen sind die Latenzen in zwei Klassen aufteilen. Bei der ersten Klasse treffen die Nachrichten in dem Gateway rechtzeitig an. Somit wird ein Versenden im vorgesehenen Zeitfenster ermöglicht. Nachrichten der zweiten Klasse treffen zu spät im Gateway ein, wodurch ein Weiterleiten erst in einem späteren Zeitfenster erfolgt und somit genau um eine Nachrichtenperiode verzögert wird. Aufgrund des gleichzeitigen Erstellens der Nachrichten zu Beginn und der anschließenden Übertragung über den CAN-Bus wird der Zeitpunkt überschritten. Die Pakete mit der geringsten Latenz zeigen zudem, wie gering

die Verzögerung zwischen Erstellzeitpunkt im Endgerät des Busses und dem Weiterleiten im Switch ist. Durch größere Offsets zwischen den Erstellzeitpunkten und den Nachrichtenfenstern lassen sich diese umgehen. Abbildung 7.6 zeigt, wie sich die Latenzen in einem neu generierten Netzwerk mit größeren Offsets verändern.

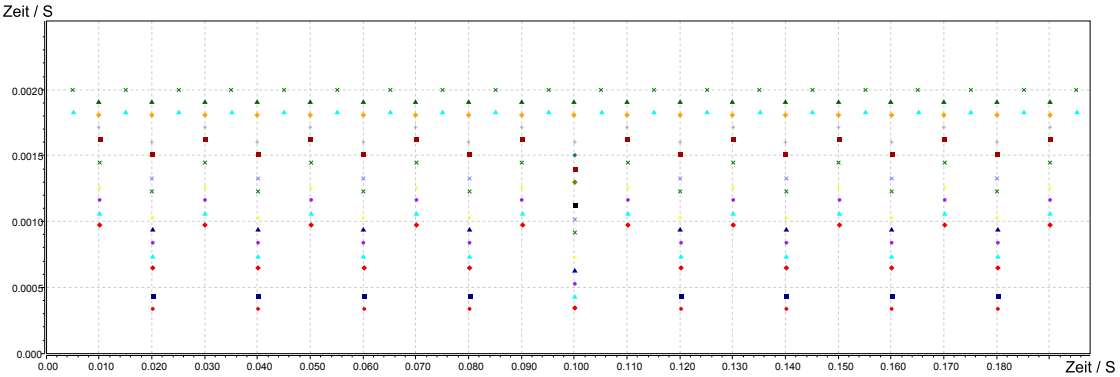

Abbildung 7.6: *Nach der Veränderung des Offsets zwischen Nachrichtenerstellung und dem Fenster zur Weiterleitung in dem Switch ist ein Senden aller Nachrichten im vorgesehenen Zyklus möglich.*

7.3 Vergleich mit einer weiteren Topologie

Eine andere Topologie des Netzwerkes kann in einer unterschiedlichen Switchverteilung liegen. Eine Idee ist dabei, die videoübertragenden Endgeräte einer Domäne zuzuordnen, damit die Latenz nicht von anderen Geräten beeinflusst wird. Jede Domäne hängt demnach an einem Switch und wird in einer Daisy-Chain-Topologie miteinander verkoppelt. Somit bilden die verschiedenen Bussysteme, die kameradatenübertragenden und die nicht zugeordneten Endgeräte jeweils eine Domäne. Abbildung 7.7 zeigt, wie diese Topologie aussehen kann.

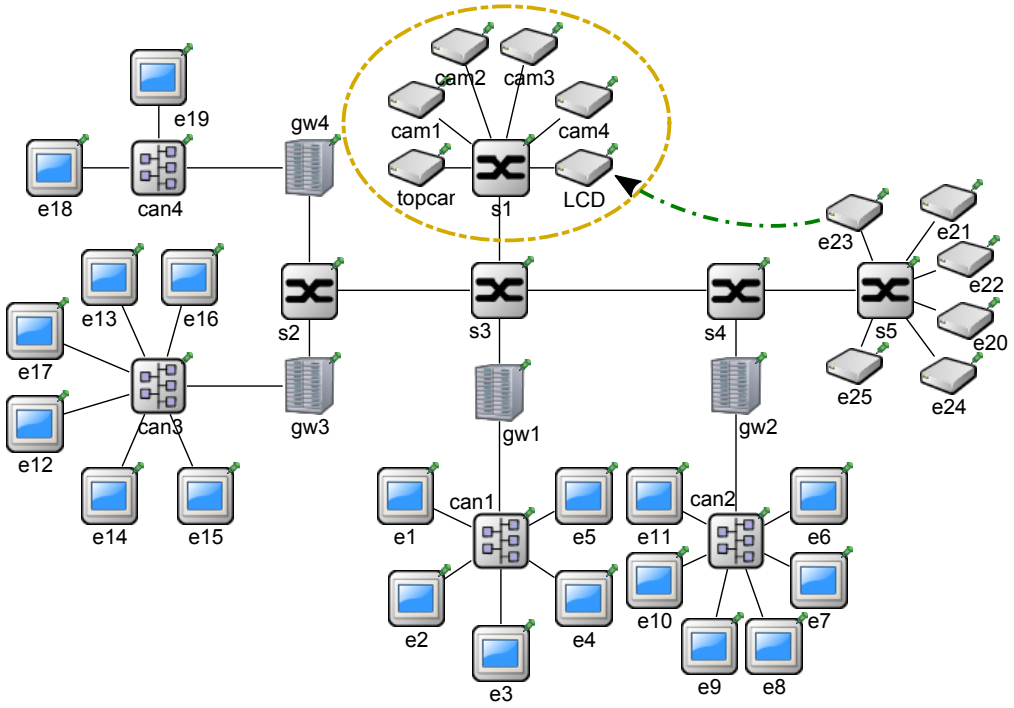

Abbildung 7.7: *Eine weitere Domäne wird mit bandbreitenintensiven Kameradaten ausgegliedert und die restlichen Domänen über eine Daisy-Chain miteinander verbunden.*

Die Nachrichten verhalten sich wie bei dem vorigen Beispiel, wobei wieder alle Kameras ihre Daten und das Endgerät „e23“ die Best-Effort Daten nach $100ms$ an das LCD-Endgerät schicken. Dabei ist die Linkauslastung der einzelnen Endgeräte und Gateways des Backbones wie im vorigen Beispiel und bedarf keiner weiteren Analyse. Dies zeigt jedoch, dass die Nachrichten bei den Empfängern wie gewünscht eintreffen. Die Linkauslastung der Switches verändert sich, wobei allerdings ein Latenzvergleich zwischen den Time-Triggered Nachrichten auf einem Gateway liegen soll. Dazu wurde das Gateway mit den meisten TT-Virtual

Links ausgewählt und in beiden Simulationsdurchläufen aufgezeichnet. Abbildung 7.8 zeigt die Latenzen sowohl in dem hierarchischen als auch in der Daisy-Chain-Topologie an.

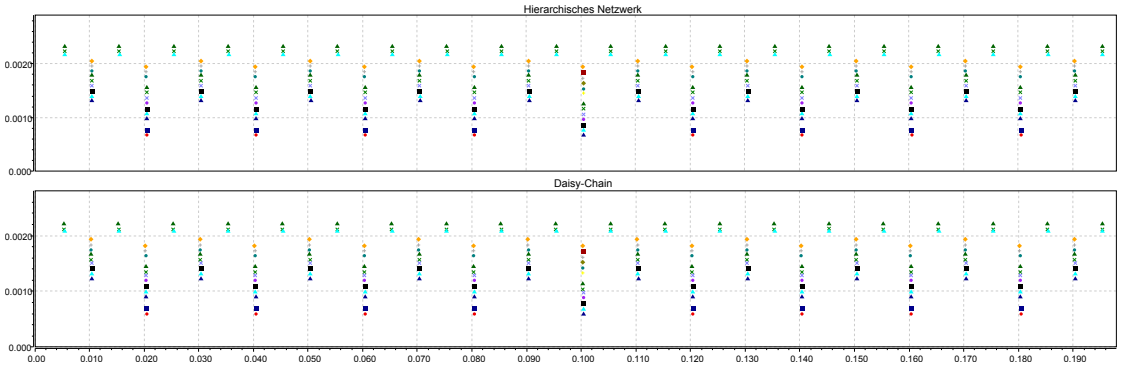

Abbildung 7.8: *Der Latenzvergleich der Time-Triggered Nachrichten in dem Gateway „gw2“ zeigt, dass die gleichen Nachrichten mit einer minimalen Latenzabweichung zugunsten der Daisy-Chain bei beiden Topologien sehr ähnlich übertragen werden.*

Die Latenz der Time-Triggered Nachrichten ist bei beiden Topologien als gleich anzusehen, wobei der gleiche Hopcount der Nachrichten eine erhebliche Rolle spielt. In der Daisy-Chain Topologie sind die Latenzen im Schnitt geringer, was auf anders gescheduledte Fenster rückschließen lässt.

7.4 Bewertung der domänenspezifischen Sprache

Die domänenspezifische Sprache eignet sich insbesondere für den Vergleich unterschiedlicher heterogener Topologien, weil mit der einmaligen Eingabe heterogener Nachrichten die Topologie beliebig verändert werden kann. Dabei wird für jedes Endgerät die passende Konfiguration für die Simulation erstellt, so dass Simulationen direkt ausgeführt und evaluiert werden können. Die Auswertung der benötigten Bandbreite und der Latenz haben gezeigt, wie ein Analysevorgang heterogener Netzwerke aussehen kann, wobei die in dieser Arbeit erwähnten, weiteren Metriken zusätzliche Betrachtungen zulassen. Mit der Einbindung des externen FIBEX-Formats können zudem verschiedene Schedulingvarianten mit einbezogen werden und eine manuelle Anpassung des Schedules erfolgen. Durch die einfache Sprache, die den Anwender auf Fehler durch Validationsregeln hinweist, ist somit der Einbettungsprozess einer simulativen Auswertung abgeschlossen.

8 Zusammenfassung, Fazit und Ausblick

Mit diesem Kapitel schließt die simulationsbasierte Analyse heterogener Fahrzeugnetzwerke ab. Dazu wird nachfolgend die Arbeit zusammengefasst, ein Fazit gezogen und ein Ausblick auf zukünftige Arbeiten gegeben.

8.1 Zusammenfassung der Arbeit

Die Kommunikation im Fahrzeug unterliegt derzeitig einem Wandel. Während im klassischen Automobilnetz die Steuergeräte innerhalb einer Domäne miteinander kommunizieren, wird der Bedarf nach domänenübergreifender Kommunikation, bei der Steuergeräte, die zu unterschiedlichen Bussystemen gehören und Nachrichten miteinander austauschen, immer größer. Durch zeitkritische und vor allem bandbreitenintensive Kommunikation reichen die derzeitig eingesetzten Bussysteme, die meist über ein zentrales Gateway kommunizieren nicht mehr aus. Daher eignet sich ein Backboneansatz, bei dem Steuergeräte mit bandbreitenintensiven Anwendungen und deterministischer Nachrichtenübertragung direkt angeschlossen werden und bisherige Domänen teilweise wegfallen können.
Mit Ethernet steht eine kostengünstige und etablierte Technologie zur Verfügung, zu der zahlreiche Echtzeit-Erweiterungen –wie Time-Triggered Ethernet– existieren. Ziel ist somit eine Verbindung von Endgeräten und Bussystemen über ein TTEthernet-Backbone, wodurch eine heterogene Kommunikationsstruktur entsteht. Durch die deterministische Übertragung ist eine Entkopplung von Funktionen und ihrer physikalischen Position möglich. Allerdings wird eine Evaluierung unterschiedlicher Topologien notwendig. Durch simulativ aufgezeichnete Bewertungsmetriken, kann diese Evaluierung der heterogenen Netzwerke erfolgen. Aufgrund komplexer Simulationsmodelle und der Anforderung, das Netzwerk effizient zu erstellen, ist ein Einbettungsprozess der Simulation erforderlich.
Im Abschnitt Hintergrund (Kapitel 2) werden Grundlagen für die simulationsbasierte Analyse erläutert. Um überhaupt Simulationsstudien im Bereich der Fahrzeugkommunikation durchführen zu können, werden im ersten Schritt die relevanten Kommunikationssysteme mit ihren charakteristischen Ausprägungen und der Übergang zu Time-Triggered Ethernet als deterministisches Fahrzeugbackbone beschrieben. Zudem wird ein Überblick von üblichen Netzwerktopologien gegeben, die für das Backbone infrage kommen. Anschließend werden Simulationsgrundlagen für heterogene Fahrzeugnetzwerke vorgestellt, die eine Modellierung

ermöglichen. Zur Konfigurationsgenerierung für das erstellte Modell und der Einbindung von Domänenexperten in den Evaluierungsprozess eignen sich insbesondere domänenspezifische Sprachen.
In Kapitel 3 wird auf die Anforderungen zur Evaluierung heterogener Fahrzeugnetzwerke eingegangen. Diese setzen sich aus einem Datenmodell, einer dazu passenden Topologie und den Schritten der Generierung, Simulation und Evaluierung zusammen. Dabei wird die Notwendigkeit einer Datenmodellierung, die das Ziel einer Vergleichbarkeit von Simulationsergebnissen hat, verdeutlicht. Darauf aufbauend stehen die Simulation, die benötigten Modellierungsschritte und vor allem die Konfigurationsgenerierung zur effizienten Evaluierung heterogener Netzwerke im Vordergrund.
Mit dem Bezug auf das vorangegangene Kapitel wird ein Konzept (Kapitel 4) zur Entwicklung eines Datenmodells gegeben. Mit einer konkreten Einteilung von Nachrichten in fahrzeugrelevante Klassen und der Möglichkeit, ebenfalls Endgeräte in Klassen zu abstrahieren, wird eine generische Lösung eines Datenmodells vorgestellt, das zukünftige Anforderungen widerspiegelt und Erweiterungen bezüglich nicht vorhergesehener Anforderungen zulässt. Mit der Ausarbeitung heterogener Netzwerktopologien werden Alternativen zur domänentypischen Struktur erläutert, die in unterschiedlichen Szenarien Einsatz finden können. Damit eine Ausführung in der Simulation möglich ist, sind Architekturen unterschiedlicher Implementierungen von benötigten Kommunikationssystemen dargestellt worden. Dabei wurde gezeigt, dass zur Evaluierung heterogener Netzwerke Quellbussystem, Backbone, Gateways und das Zielbussystem modelliert sein müssen. Für den einheitlichen Evaluierungsprozess stand die Entwicklung eines Konzepts zur Generierung der Simulationskonfiguration im weiteren Fokus. Über den Prozess sollte es möglich sein, verschiedene Simulationsmodelle einheitlich, mithilfe einer domänenspezifischen Sprache zu beschreiben und den Entwurf eines Netzwerkes in einer Ende-zu-Ende-Beschreibung für den Nutzer zu vereinfachen. Mit der Evaluierung heterogener Fahrzeugnetzwerke durch geeignete Metriken sowie deren Aufzeichnung und Auswertung in der Simulation wurde dieses Kapitel abgeschlossen.
Der weitere Fokus lag in der konkreten Umsetzung einer domänenspezifischen Sprache zur Konfigurationsgenerierung (Kapitel 5). Diese eignet sich insbesondere für die zur Verfügung stehenden Simulationsmodelle, weil die entwickelte Sprache in der gleichen Entwicklungsumgebung eingesetzt werden kann und somit ein nahtloser Übergang zwischen der Generierung und der Ausführung eines gewünschten Szenarios liegt. Die Aufzeichnung und Auswertung der Metriken in der gewählten Simulationsumgebung dient dem Abschluss des gewählten Evaluierungsprozesses, so dass die Generierung, Simulation und Evaluierung heterogener Fahrzeugnetzwerke abgeschlossen ist.
Das Kapitel 6 zeigt, dass sich die entwickelte domänenspezifische Sprache zur Generierung für eine ausgewählte Simulationsumgebung eignet. Dazu wurden bestimmte Nachrichtenverhalten mit homogenen und heterogenen Netzwerken in der Sprache modelliert und überprüft, ob die Auswertung ausgewählter Metriken das gewünschte Verhalten widerspiegeln.
Mit dem abschließenden Fallbeispiel (Kapitel 7) wird der gesamte Evaluierungsprozess über alle drei Stufen mit einem entwickelten Datenmodell durchlaufen. Durch die Komplexität des

Fallbeispiels, der einfachen Konfigurierung, der möglichen Veränderung der Topologie und der damit verbundenen, erneuten Generierung wurde gezeigt, dass sich die entwickelte Sprache zur Generierung komplexer, heterogener Netzwerke eignet. Mit der Auswertung ausgewählter Metriken wurde zudem die Analyse überprüft und die erstellten Szenarien erfolgreich bewertet.

8.2 Fazit zur Analyse heterogener Netzwerke

Mit der steigenden Komplexität der Bordelektronik und der erforderlichen domänenübergreifenden Kommunikation in Fahrzeugnetzwerken sind effiziente Wege für die Analyse und Evaluierung neuer Netzwerktopologien notwendig.

Mit dieser Arbeit wurden die wichtigen Teilbereiche der Simulation, Evaluierung und vor allem der Konfigurationsgenerierung betrachtet und Konzepte zur Einbettung in einen Entwicklungsprozess erstellt. Dabei wurde mittels einer domänenspezifischen Sprache ein neuer Weg der Generierung für Kommunikationsnetzwerke eingeschlagen, der den Nutzer mit einer abstrakten Ende-zu-Ende Beschreibung des Netzwerkes unterstützt. Mit validierten Simulationsmodellen für verschiedene Kommunikationssysteme ist es mit der Sprache möglich, heterogene Netzwerke zu erstellen, zu simulieren und anschließend über definierte Metriken zu evaluieren. Mit der Unterstützung einer Echtzeit-Ethernet-Variante als möglichem Kommunikationssystem im Automobilnetzwerk ist die Evaluierung eines zukünftigen Backboneansatzes, der die Domänen miteinander verbindet, gegeben. Dieser Ansatz wurde in einer Fallstudie aufgegriffen und heterogene Nachrichtenübertragungen mit zukünftigen Anforderungen gemischt. In dieser wurde die einfach zu ändernde Topologie bei einmaliger Eingabe der Nachrichten deutlich gemacht und kann dementsprechend für Topologiebetrachtungen bei Fahrzeugnetzen der Zukunft exemplarisch angesehen werden.

Mit den erzielten Ergebnissen wird deutlich, wie komplex die Konfiguration heterogener, zeitkritischer Systeme ist. Somit leistet diese Arbeit einen Beitrag für die vielfältigen Aufgaben, die heute an das Bordnetz gestellt werden.

8.3 Ausblick auf zukünftige Arbeiten

Die simulationsbasierte Analyse heterogener Netzwerke ist mit dieser Arbeit nicht abgeschlossen. Obwohl ein Evaluierungsprozess mit Generierung, Simulation und Evaluierung vorgestellt wurde, sind zukünftige Arbeiten in jedem Bereich nötig, damit ein ganzheitliches Modell entstehen kann. In der Simulation ist die Umsetzung weiterer Kommunikationsmodelle und der Anbindung an das Backbone über Gateways wünschenswert. Dadurch müsste sowohl die domänenspezifische Sprache um weitere spezifische Sprachelemente als auch die Generierung

erweitert werden. Bei erfolgreicher Unterstützung würde bei gleichbleibender Komplexität der Nachrichtenbeschreibung die Komplexität des erstellten Netzwerkes ansteigen und dadurch eine einfache Beschreibung den heterogenen Ansatz befürworten.

Eine Spracherweiterung kann durch zusätzliche, optionale Parameter erfolgen. Diese ermöglichen eine speziellere Beschreibung vorhandener Simulationsmodelle, wobei die abstrakte Beschreibung für unterschiedliche Simulationsumgebungen jedoch verloren gehen kann. Eine weitere Variante in dieser Richtung kann ein ganzheitlich modellierbarer Ansatz sein, bei dem die Knoten und deren Verhaltensweisen direkt in der Sprache modelliert werden und eine spätere Anpassung der Sprache bei neu zu unterstützenden Kommunikationssystemen nicht notwendig ist.

Hinsichtlich der Auswertung kann die Generierung eines Reports für eine übersichtliche Darstellung eines Simulationsdurchlaufes sinnvoll sein. Dadurch ist eine algorithmische Bewertung möglich, bei der Werte der aufgezeichneten Kennzahlen analysiert und Probleme vor der Überschreitung angegebener Grenzwerte erkannt werden. Durch die umfangreichen Auswertungsmöglichkeiten der genutzten Simulationsumgebung ist dies jedoch nicht zwingend erforderlich.

A Validierungsbeispiele der entwickelten DSL

```
Network {
      Cycle 10 ms
      Ticklength 100 ns
      Generationsparameter {
            Difference between RC Send and RC Bag 20 us
      }
      EthernetCable c {
            Delay 100 us Datarate 100 Mb/s
      }
      Switch s1 {
            Port 1 to Ecu e1 Cable c
            Port 2 to Ecu e2 Cable c
            Port 3 to Switch s2 Cable c
      }
      Switch s2 {
            Port 1 to Switch s1 Cable c
            Port 2 to Ecu e3 Cable c
      }
      Ecu e1 {
            Connect s1 Cable c
            TTVL {
                  ID 1 Payload 100 byte Period 500 us
                        Receiver {
                              Rec e3
                        }
            }
            Best-Effort-Traffic {
                  StartTime 1 s Interval 100 us to 300 us Payload
                     200 byte
                        Receiver {
                              Rec e3
                        }
            }
      }
      Ecu e2 {
```

```
35              Connect s1 Cable c
36              RCVL {
37                    ID 2 Payload 100 byte BAG 250 us Priority 1
38                         Receiver {
39                               Rec e3
40                         }
41              }
42        }
43        Ecu e3 {
44              Connect s2 Cable c
45        }
46  }
```

Listing A.1: *Time-Triggered Ethernet Netzwerk*

```
Networkname ValidateCAN
Network {
	Cycle 10 ms
	Ticklength 100 ns
	Generationsparameter {
		Difference between RC Send and RC Bag 30 us
	}
	CanBus can {
		Bandwidth 125000 bit/s Errors false
	}
	Ecu e1 {
		CanNode can
		CAN-Messages {
			ID 1 Period 100 ms Payload 8 byte DataFrame
				Receiver {
					Rec e2
					Rec e3
				}
		}
	}
	Ecu e2 {
		CanNode can
	}
	Ecu e3 {
		CanNode can
			CAN-Messages {
			ID 2 Period 100 ms Payload 8 byte DataFrame
				Receiver {
					Rec e2
					Rec e3
				}
		}
	}
	Ecu e4 {
		CanNode can
	}
}
```

Listing A.2: *CAN-Netzwerk*

```
Networkname ValidateHeterogen
Network { Cycle 10 ms
      Ticklength 100 ns
      Generationsparameter {

            Difference between RC Send and RC Bag 30 us
      }
      EthernetCable c {
            Delay 100 us Datarate 100 Mb/s
      }
      CanBus can1 {
            Bandwidth 125000 bit/s Errors false
      }
      CanBus can2 {
            Bandwidth 125000 bit/s Errors false
      }
      Switch s1 {
            Port 1 to Switch s2 Cable c
            Port 2 to Ecu e1 Cable c
            Port 3 to Gateway gw1 Cable c
      }
      Switch s2 {
            Port 1 to Switch s1 Cable c
            Port 2 to Ecu e2 Cable c
            Port 3 to Gateway gw2 Cable c
            Port 4 to Gateway gw3 Cable c
      }
      Ecu e2 {
            Connect s2 Cable c

      }
      Ecu e1 {
            Connect s1 Cable c
            RCVL {
                  ID 1 Payload 500 byte BAG 500 us Priority 2
                        Receiver {
                              Rec e2
                        }
            }
            TTVL {
                  ID 2 Payload 100 byte Period 5000 us
                        Receiver {
                              Rec e2
                        }
            }
```

```
46        }
47        Ecu can1e1 {
48              CanNode can1
49              Heterogeneous Message {
50                          Period 2 ms Payload 5 byte CANid 3 RCid 4
                                CANid 1
51                             Receiver {
52                                    Rec can2e1
53                                    Rec can3e1
54                                    Rec e1
55                             }
56                    }
57              }
58        Ecu can1e2 {
59              CanNode can1
60              Heterogeneous Message {
61                          Period 1 ms Payload 5 byte CANid 1 TTid 5
62                             Receiver {
63                                    Rec can1e1
64                                    Rec e2
65                             }
66                    }
67              }
68        Ecu can2e1 {
69              CanNode can2
70        }
71        Ecu can2e2 {
72              CanNode can2
73        }
74        Gateway gw1 {
75              Canbusconnect can1 Switchconnect s1 Cable c
76        }
77        Gateway gw2 {
78              Canbusconnect can2 Switchconnect s2 Cable c
79        }
80        CanBus can3 {
81              Bandwidth 125000 bit/s Errors false
82        }
83        Gateway gw3 {
84              Canbusconnect can3 Switchconnect s2 Cable c
85        }
86  }
```

Listing A.3: *Heterogenes Netzwerk mit CAN-Bussen, Time-Triggered Ethernet Backbone und Gateways*

Literatur

Aeronautical Radio Incorporated 2009 Aeronautical Radio Incorporated: *Aircraft Data Network, Part 7, Avionics Full-Duplex Switched Ethernet Network*. ARINC Report 664P7-1. ARINC, 2009 – Standard.

ASAM 2013 Association for Standardisation of Automation and Measuring Systems: *Data Model for ECU Network Systems (Field Bus Data Exchange Format)*. 4.1.0. ASAM e.V., 2013-05 – Specification.

Banks 2010 Banks, Jerry: *Discrete-event System Simulation*. Prentice Hall, 2010. – ISBN: 9780136062127.

Bartols 2010 Bartols, Florian: „Leistungsmessung von Time-Triggered Ethernet Komponenten unter harten Echtzeitbedingungen mithilfe modifizierter Linux-Treiber“. Bachelorthesis. Bachelorthesis. Hamburg : HAW Hamburg, 2010-07.

Bartols 2014 Bartols, Florian: „Echtzeit Ethernet Restbussimulation: Frühzeitiges Modellbasiertes Testen in Fahrzeugnetzwerken der nächsten Generation“. Masterthesis. Hamburg : HAW Hamburg, 2014-04.

Behrens / Clay / Efftinge u. a. Behrens, Heiko ; Clay, Michael ; Efftinge, Sven u. a.: *Xtext User Guide*. URL: `http://www.eclipse.org/Xtext/documentation/1_0_1/xtext.pdf` Abruf: 2014-08-13.

Ben-Ari 1990 Ben-Ari, Mordechai: *Principles of Concurrent and Distributed Programming*. Upper Saddle River, NJ, USA : Prentice-Hall, Inc., 1990. – ISBN: 0-13-711821-X.

Bob Pickles 2006 Bob Pickles: *Avionics Full Duplex Switched Ethernet (AFDX)*. SBS Technologies, 2006 – Techn. Ber. URL: `http://www.sierrasales.com/pdfs/AFDX_Overview.pdf` Abruf: 2011-05-09.

Bossel 1994 Bossel, H.: *Modellbildung und Simulation: Konzepte, Verfahren und Modelle zum Verhalten dynamischer Systeme ; ein Lehr- und Arbeitsbuch*. (Informatik & Computer). Vieweg, 1994. – ISBN: 9783528152420.

Broy / Krüger / Pretschner u. a. 2007 Broy, Manfred ; Krüger, Ingolf h. ; Pretschner, Alexander u. a.: Engineering Automotive Software. In: *Proceedings of the IEEE* 95 (2007-02), Nr. 2, S. 356–373. – ISSN: 0018-9219. – DOI: 10.1109/JPROC.2006.888386.

Bungartz 2013 BUNGARTZ, Hans-Joachim: *Modellbildung und Simulation: Eine anwendungsorientierte Einführung*. Springer Berlin Heidelbergl, 2013. – ISBN: 978-3-642-37655-9.

Castelpietra / Song / Simonot-Lion u. a. 2000 CASTELPIETRA, P. ; SONG, Y-Q ; SIMONOT-LION, F. u. a.: „Performance evaluation of a multiple networked in-vehicle embedded architecture“. In: *Factory Communication Systems, 2000. Proceedings. 2000 IEEE International Workshop on*. 2000, S. 187–194. – DOI: 10.1109/WFCS.2000.882549.

Christ / Büttner 2012 CHRIST, Andre ; BÜTTNER, Jörg: *Detailed Passenger Airbag Modelling for Early Stage Events*. 2012. URL: http://www.dynamore.de/de/download/papers/ls-dyna-forum-2012/documents/passive-3-1.

Czarnecki / Helsen / Eisenecker 2004 CZARNECKI, Krzysztof ; HELSEN, Simon ; EISENECKER, Ulrich: „Staged configuration using feature models“. In: *Software Product Lines: Third International Conference, SPLC 2004*. Springer-Verlag, 2004, S. 266–283.

Combs COMBS, Gerald: *Wireshark*. URL: http://www.wireshark.org/ Abruf: 2014-07-16.

CoRE RG CoRE RG: *Communication over Real-time Ethernet*. Hamburg . URL: http://core.informatik.haw-hamburg.de.

CoRE RG CoRE RG: *TTE for INET*. URL: http://tte4inet.realmv6.org.

CoRE-Arbeitsgruppe CoRE-ARBEITSGRUPPE: *Communication over Real-time Ethernet*. Hamburg . URL: http://core.informatik.haw-hamburg.de Abruf: 2014-08-18.

CoRE-Arbeitsgruppe CoRE-ARBEITSGRUPPE: *FiCo4OMNeT: Fieldbus Communication for OMNeT++*. URL: http://core4inet.realmv6.org/wiki/FiCo4OMNeT_Background Abruf: 2014-09-05.

Crovella / Taqqu / Bestavros 1998 CROVELLA, Mark E. ; TAQQU, Murad S. ; BESTAVROS, Azer: „Heavy-Tailed Probability Distributions in the World Wide Web“. In: *In A Practical Guide To Heavy Tails, chapter 1*. Chapman und Hall, 1998, S. 3–26.

Dieumo Kenfack 2010 DIEUMO KENFACK, Hermand: „Erzeugung von charakteristischen Last-Profilen zur simulationsgestützten Analyse von TTEthernet“. Anwendungen 1 Ausarbeitung. 2010-08.

Deursen / Klint / Visser 2000 DEURSEN, Arie van ; KLINT, Paul ; VISSER, Joost: Domain-specific Languages: An Annotated Bibliography. In: *SIGPLAN Not.* 35 (2000-06), Nr. 6, S. 26–36. – ISSN: 0362-1340. – DOI: 10.1145/352029.352035. URL: http://doi.acm.org/10.1145/352029.352035.

Dohmke 2002 DOHMKE, Thomas: *Bussysteme im Automobil: CAN, FlexRay und MOST*. Technische Universität Berlin, DaimlerChrysler AG, 2002 – Techn. Ber.

Drossos / Lindbüchl / Grohmann u. a. 2013 Drossos, Andreas ; Lindbüchl, Roland ; Grohmann, Dieter u. a.: Vernetzung und Komfortelektronik. German. In: ERNSTBERGER, Uwe (Hrsg.) ; WEISSINGER, Jürgen (Hrsg.) ; FRANK, Jürgen (Hrsg.): *Mercedes-Benz SL*. (ATZ / MTZ-Typenbuch). Springer Fachmedien Wiesbaden, 2013. – ISBN: 978-3-658-00799-7, S. 210–220. – DOI: 10.1007/978-3-658-00800-0_24. URL: http://dx.doi.org/10.1007/978-3-658-00800-0_24.

EBNF 1996 EBNF: *ISO/IEC 14977:1996(E) First edition – Information technology – Syntactic metalanguage – Extended BNF*. ISO/IEC, 1996 – Techn. Ber. URL: http://standards.iso.org/ittf/PubliclyAvailableStandards/s026153%5C_ISO%5C_IEC%5C_14977%5C_1996(E).zip.

Eclipse Public License ECLIPSE PUBLIC LICENSE: *Xtext*. URL: http://www.xtext.org/ Abruf: 2014-08-13.

Engler 2013 ENGLER, Jonas: „Ein Framework zu einer OMNeT++ basierten Simulation von CAN-Netzwerken auf der Sicherungsschicht“. Bachelorthesis. Bachelorthesis. Hamburg : HAW Hamburg, 2013-05.

Fang / Yen / Pan u. a. 2010 FANG, Yechang ; YEN, Kang ; PAN, Deng u. a.: Buffer Management Algorithm Design and Implementation Based on Network Processors. In: *CoRR* abs/1005.0905 (2010). URL: http://dblp.uni-trier.de/db/journals/corr/corr1005.html#abs-1005-0905.

FlexRay Consortium FLEXRAY CONSORTIUM: *FlexRay*. Stuttgart . URL: http://flexray.com/ Abruf: 2011-02-07.

Fröberg / Sandström / Norström u. a. 2003 FRÖBERG, Joakim ; SANDSTRÖM, Kristian ; NORSTRÖM, Christer u. a.: *A Comparative Case Study of Distributed Network Architectures for Different Automotive Applications*. ISSN 1404-3041 ISRN MDH-MRTC-69/2003-1-SE. 2003-02 – Techn. Ber. URL: http://www.es.mdh.se/publications/405-.

Gaede 1977 GAEDE, Karl-Walter: *Zuverlässigkeit, mathematische Modelle*. München : Hanser, 1977. – ISBN: 3-446-12370-9.

Garshol 2006 GARSHOL, Lars Marius: *BNF and EBNF: What are they and how do they work?* 2006. URL: www.garshol.priv.no/download/text/bnf.html.

Georges / Divoux / Rondeau 2005 GEORGES, Jean-Philippe ; DIVOUX, Thierry ; RONDEAU, Eric: „Strict Priority Versus Weighted Fair Queueing in Switched Ethernet Networks for Time Critical Applications“. In: *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 2 - Volume 03*. (IPDPS '05). Washington, DC, USA : IEEE Computer

Society, 2005, S. 141–. – ISBN: 0-7695-2312-9. – DOI: 10.1109/IPDPS.2005.413. URL: http://dx.doi.org/10.1109/IPDPS.2005.413.

GE Fanuc Intelligent Platforms 2009 GE FANUC INTELLIGENT PLATFORMS: *TTEthernet - A Powerful Network Solution for Advanced Integrated Systems*. GFT-751. Charlottesville , 2009-08. URL: http://www.ge-ip.com/library/detail/12014 Abruf: 2011-01-18.

Gerberich 2011 GERBERICH, Thorsten: Modellverifikation und -validierung. German. In: *Lean oder MES in der Automobilzulieferindustrie*. Gabler, 2011. – ISBN: 978-3-8349-2929-7, S. 359–390. – DOI: 10.1007/978-3-8349-6754-1_8. URL: http://dx.doi.org/10.1007/978-3-8349-6754-1_8.

Guérin / Peris 1999 GUÉRIN, Roch ; PERIS, Vinod: Quality-of-service in packet networks: basic mechanisms and directions. In: *Computer Networks* 31 (3 1999-02), S. 169–189. – ISSN: 1389-1286. – DOI: 10.1016/S0169-7552(98)00261-X.

Hank / Suermann / Müller 2012 Hank, Peter ; Suermann, Thomas ; Müller, Steffen: Automotive Ethernet, a Holistic Approach for a Next Generation In-Vehicle Networking Standard. English. In: MEYER, Gereon (Hrsg.): *Advanced Microsystems for Automotive Applications 2012*. Springer Berlin Heidelberg, 2012. – ISBN: 978-3-642-29672-7, S. 79–89. – DOI: 10.1007/978-3-642-29673-4_8. URL: http://dx.doi.org/10.1007/978-3-642-29673-4_8.

Hu / Member / Steenkiste u. a. 2003 HU, Ningning ; MEMBER, Student ; STEENKISTE, Peter u. a.: Evaluation and Characterization of Available Bandwidth Probing Techniques. In: *IEEE Journal on Selected Areas in Communications* 21 (2003), S. 879–894.

Institute of Electrical and Electronics Engineers 2005 INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS: *IEEE 802.3: LAN/MAN CSMA/CD Access Method*. IEEE 802.3-2005. IEEE, 2005 – Standard.

Jasperneite 2005 JASPERNEITE, Jürgen: *Echtzeit-Ethernet im Überblick*. 2005.

Joerg Rech 2007 JOERG RECH: *10 Gigabit pro Sekunde über Kupfer*. 2007. URL: http://www.heise.de/netze/artikel/10-Gigabit-pro-Sekunde-ueber-Kupfer-221773.html Abruf: 2014-07-12.

Jasperneite / Watson 2008 JASPERNEITE, Jürgen ; WATSON, Kym: Bestimmung von oberen Zeitschranken in Ethernet-Netzwerken. In: *Automatisierungstechnische Praxis* (2008).

Kamieth / Steinbach / Korf u. a. 2014 KAMIETH, Jan ; STEINBACH, Till ; KORF, Franz u. a.: „Design of TDMA-based In-Car Networks: Applying Multiprocessor Scheduling Strategies on Time-triggered Switched Ethernet Communication“. In: *19th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2014)*. Barcelona : IEEE Press, 2014. Im Erscheinen.

Kempf 2011 KEMPF, Fabian: „Simulation von AFDX-Netzwerken basierend auf Rate-Constrained Traffic für Time-Triggered Ethernet in OMNeT++ “. Bachelorthesis. Hamburg , 2011-08.

Kempf 2014 KEMPF, Fabian: *Metriken und deren simulationsbasierte Umsetzung*. Bericht. 2014-02.

Kopetz 2004 KOPETZ, Hermann: *Real-time Systems: Design Principles for Distributed Embedded Applications*. 8. Aufl. (The Kluwer international series in engineering and computer science: real-time systems). Boston : Kluwer Academic, 2004. – ISBN: 0-7923-9894-7.

Kolesnikov / Wolfinger / Kulas 2009 Kolesnikov, Andry W. ; Wolfinger, Bernd E. ; Kulas, Martin: „UniLoG - Ein System zur verteilten Lastgenerierung in Netzen.“ In: *Echtzeit*. Hrsg. von Wolfgang A. HALANG ; Peter HOLLECZEK. (Informatik Aktuell). Springer, 2009, S. 11–20. – ISBN: 978-3-642-04782-4. URL: http://dblp.uni-trier.de/db/conf/pearl/echtzeit2009.html#KolesnikovWK09.

Lim / Krebs / Volker u. a. 2011 LIM, Hyung-Taek ; KREBS, Benjamin ; VOLKER, Lars u. a.: „Performance evaluation of the inter-domain communication in a switched Ethernet based in-car network“. In: *Local Computer Networks (LCN), 2011 IEEE 36th Conference on*. 2011-10, S. 101–108. – DOI: 10.1109/LCN.2011.6115156.

Lim / Herrscher / Waltl u. a. 2012 LIM, Hyung-Taek ; HERRSCHER, Daniel ; WALTL, Martin Johannes u. a.: „Performance analysis of the IEEE 802.1 ethernet audio/video bridging standard“. In: *Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques*. Desenzano del Garda, Italy : ACM-DL, 2012-03, S. 27–36. – ISBN: 978-1-4503-1510-4.

LIN-Administration LIN-ADMINISTRATION: *Local Interconnect Network*. URL: http://www.lin-subbus.org/ Abruf: 2011-01-06.

Law / Kelton 1991 LAW, Averill M. ; KELTON, W. David: *Simulation modeling & analysis*. 2. Aufl. (McGraw-Hill series in industrial engineering and management science). New York : McGraw-Hill, 1991. – ISBN: 0-07-036698-5.

Lupini 2004 LUPINI, C.A.: *Vehicle Multiplex Communication: Serial Data Networking Applied to Vehicular Engineering*. (R: Society of Automotive Engineers). SAE International, 2004. – ISBN: 9780768012187.

Lim / Weckemann / Herrscher 2011 Lim, Hyung-Taek ; Weckemann, Kay ; Herrscher, Daniel: „Performance Study of an In-Car Switched Ethernet Network without Prioritization“. In: *Communication Technologies for Vehicles*. Hrsg. von Thomas STRANG ; Andreas FESTAG ; Alexey VINEL u. a. Bd. 6596. Desenzano del Garda, Italy : Springer Berlin Heidelberg, 2011-03, S. 165–175.

– DOI: 10.1007/978-3-642-19786-4_15. URL: http://dx.doi.org/10.1007/978-3-642-19786-4_15.

Mohammad / Al-Holou 2010 MOHAMMAD, Utayba ; AL-HOLOU, Nizar: „Development of An Automotive Communication Benchmark“. In: *Canadian Journal on Electrical and Electronics Engineering*. Bd. 1. (5). 2010-8, S. 99–115.

Mall 2009 MALL, Rajib: *Real-Time Systems: Theory and Practice*. 1. Aufl. Upper Saddle River, NJ, USA : Prentice Hall Press, 2009. – ISBN: 978-8131700693.

Manderscheid / Langer 2011 MANDERSCHEID, Martin ; LANGER, Falk: „Network Calculus for the Validation of Automotive Ethernet In-vehicle Network Configurations“. In: *Cyber-Enabled Distributed Computing and Knowledge Discovery*. 2011, S. 206–211. – ISBN: 978-0-7695-4557-8.

Müller-Rathgeber / Michel 2009 MÜLLER-RATHGEBER, Bernd ; MICHEL, Hans Ulrich: „Automotive network planning - A genetic approach“. In: *Intelligent Vehicles Symposium, 2009 IEEE*. 2009-06, S. 1088–1092. – DOI: 10.1109/IVS.2009.5164433.

MOST Cooperation MOST COOPERATION: *Media Oriented Systems Transport*. URL: http://www.mostcooperation.com/ Abruf: 2011-01-06.

Mansour / Patt-Shamir 1998 MANSOUR, Yishay ; PATT-SHAMIR, Boaz: Jitter Control in QoS Networks. In: *IEEE/ACM Transactions on Networking* 9 (1998), S. 492–502.

Marscholik / Subke 2007 MARSCHOLIK, Christoph ; SUBKE, Peter: *Datenkommunikation im Automobil: Grundlagen, Bussysteme, Protokolle und Anwendungen*. (Hüthig Praxis). Heidelberg : Hüthig, 2007. – ISBN: 3-7785-2969-2.

Marscholik / Subke 2011 MARSCHOLIK, Christoph ; SUBKE, Peter: *Datenkommunikation im Automobil: Grundlagen, Bussysteme, Protokolle und Anwendungen*. 2. Auflage. Vde-Verlag, 2011-02. – ISBN: 3800732750.

Mouftah / Sturgeon 1990 MOUFTAH, Hussein T. ; STURGEON, Rene P.: Distributed discrete event simulation for communication networks. In: *Selected Areas in Communications, IEEE Journal on* 8 (1990-12), Nr. 9, S. 1723–1734. – ISSN: 0733-8716. – DOI: 10.1109/49.62858.

Müller 2014 MÜLLER, Sebastian: „Simulationsmodell eines Multi-Bus Realtime Ethernet Gateways“. Bachelorthesis. Hamburg , 2014-10.

OMNeT++ Community OMNET++ COMMUNITY: *INET Framework for OMNeT++ 4.0*. URL: http://inet.omnetpp.org/.

OMNeT++ Community OMNET++ COMMUNITY: *OMNeT++ 4.2.2*. URL: http://www.omnetpp.org.

OMNeT++ Community OMNeT++ Community: *OMNeT++ 4.2rc1*. URL: http://www.omnetpp.org.

Page / Liebert / Heymann u. a. 1991 Page, B. ; Liebert, H. ; Heymann, A. u. a.: *Diskrete Simulation: Eine Einführung mit Modula-2*. (Springer-Lehrbuch). Springer Berlin Heidelberg, 1991. – ISBN: 978-3-642-76862-0.

Prasad / Murray / Dovrolis u. a. 2003 Prasad, R. S. ; Murray, M. ; Dovrolis, C. u. a.: Bandwidth Estimation: Metrics, Measurement Techniques, and Tools. In: *IEEE Network* 17 (2003), S. 27–35.

Rahmani / Hillebrand / Hintermaier u. a. 2007 Rahmani, Mehrnoush ; Hillebrand, Joachim ; Hintermaier, Wolfgang u. a.: „A Novel Network Architecture for In-Vehicle Audio and Video Communication“. In: *Broadband Convergence Networks, 2007. BcN '07. 2nd IEEE/IFIP International Workshop on*. 2007-05, S. 1–12. – DOI: 10.1109/BCN.2007.372741.

Riegraf / Behh / Kraus 2007 Riegraf, Thomas ; Behh, Siegfried ; Kraus, Stefan: Effizientes Testen in der Automobilelektronik - Von der Simulation bis zur Diagnose. In: *ATZ - Automobiltechnische Zeitschrift* 109 (2007-08), S. 648–655.

Real Time Systems Group (RTS) Real Time Systems Group (RTS): *TTEthernet*. URL: http://ti.tuwien.ac.at/rts Abruf: 2010-12-10.

Redmer 2012 Redmer, Andreas: *Effiziente Datenanalyse in Netzwerkgraphen: Durch User Defined Functions in PostgreSQL*. Diplomica Verlag, 2012. – ISBN: 9783842881617.

Reif 2009 Reif, Konrad: *Automobilelektronik*. Wiesbaden : Vieweg und Teubner, 2009. – ISBN: 978-3-8348-0446-4.

Reif 2010 Reif, Konrad: *Batterien, Bordnetze und Vernetzung*. (Bosch Fachinformation Automobil). Vieweg + Teubner, 2010. – ISBN: 9783834897138. URL: http://books.google.de/books?id=8NCldt82AoQC.

Robert Bosch GmbH Robert Bosch GmbH: *Controller Area Network*. URL: http://www.semiconductors.bosch.de/ Abruf: 2011-02-03.

Rumpf / Steinbach / Korf u. a. 2014 Rumpf, Soeren ; Steinbach, Till ; Korf, Franz u. a.: Software Stacks for Mixed-critical Applications: Consolidating IEEE 802.1 AVB and Time-triggered Ethernet in Next-generation Automotive Electronics. In: (2014). URL: http://inet.cpt.haw-hamburg.de/papers/rsks-ssmac-14a.pdf.

Saad 2003 Saad, Alexandre: „Das Automobil als Anwendungsgebiet der Informatik - ein Auto ohne Informatik, geht das?“ In: *INFOS*. Hrsg. von Peter Hubwieser. Bd. 32. (LNI). GI, 2003, S. 37–40. – ISBN: 3-88579-361-X. URL: http://dblp.uni-trier.de/db/conf/schule/schule2003.html#Saad03.

Schäfer 2004 SCHÄFER, Andreas: „Verlässlichkeitsanalyse paketvermittelnder Netzwerke mittels diskreter ereignisorientierter Simulation“. Diss. Karlsruhe : Universität Fridericiana zu Karlsruhe, 2004-07.

Sarr / Guérin-Lassous 2007 SARR, Cheikh ; GUÉRIN-LASSOUS, Isabelle: *Estimating Average End-to-End Delays in IEEE 802.11 Multihop Wireless Networks*. English RR-6259. INRIA, 2007, S. 24 – Research Report. URL: http://hal.inria.fr/inria-00166017.

Steinbach / Korf / Schmidt 2010 STEINBACH, Till ; KORF, Franz ; SCHMIDT, Thomas C.: „Comparing time-triggered Ethernet with FlexRay: An evaluation of competing approaches to real-time for in-vehicle networks“. In: *Factory Communication Systems (WFCS), 2010 8th IEEE International Workshop on*. 2010-05, S. 199–202. – DOI: 10.1109/WFCS.2010.5548606.

Smolka 2008 SMOLKA, Gert: *Programmierung - eine Einführung in die Informatik mit Standard ML*. Oldenbourg, 2008. – ISBN: 9783486586015. URL: http://books.google.de/books?id=ADT9zOiHf24C.

Streichert / Traub 2012 STREICHERT, T. ; TRAUB, M.: *Elektrik/Elektronik-Architekturen im Kraftfahrzeug: Modellierung und Bewertung von Echtzeitsystemen*. (VDI-Buch). Springer, 2012. – ISBN: 9783642254772. URL: http://books.google.de/books?id=6-cZKnObltgC.

Steiner / Bauer / Hall u. a. 2009 STEINER, W. ; BAUER, G. ; HALL, B. u. a.: „TTEthernet Dataflow Concept“. In: *8th IEEE International Symposium on Network Computing and Applications, 2009. NCA 2009.* 2009-07, S. 319–322. – DOI: 10.1109/NCA.2009.28.

Steffen / Bogenberger / Hillebrand u. a. 2010 STEFFEN, Rainer ; BOGENBERGER, Richard ; HILLEBRAND, Joachim u. a.: „Design and Realization of an IP-based In-car Network Architecture“. In: 2010-5. – DOI: 10.4108/ICST.ISVCS2008.3543.

Steinbach / Lim / Korf u. a. 2012 STEINBACH, Till ; LIM, Hyung-Taek ; KORF, Franz u. a.: „Tomorrow's In-Car Interconnect? A Competitive Evaluation of IEEE 802.1 AVB and Time-Triggered Ethernet (AS6802)“. In: *2012 IEEE Vehicular Technology Conference (VTC Fall)*. Piscataway, New Jersey : IEEE Press, 2012-09.

Steiner 2008 STEINER, Wilfried: *TTEthernet Specification*. TTTech Computertechnik AG. 2008-11. URL: http://www.tttech.com.

Steinbach 2010 STEINBACH, Till: „Realtime-Ethernet für automotive Anwendungen: Metriken und deren simulationsbasierte Evaluierung am Beispiel von TTEthernet“. Bericht. 2010-02. URL: http://papers.till-steinbach.de/s-reaam-10.pdf.

Steinbach 2011 STEINBACH, Till: „Echtzeit-Ethernet für Anwendungen im Automobil: Metriken und deren simulationsbasierte Evaluierung am Beispiel von TTEthernet“. Masterthesis. Hamburg : Hochschule für Angewandte Wissenschaften Hamburg, 2011-02.

Strümpel 2003 STRÜMPEL, Frauke: *Simulation zeitdiskreter Modelle mit Referenznetzen*. 2003. URL: http://www.informatik.uni-hamburg.de/TGI/mitarbeiter/koehler/stud/struemp-da.pdf.

Schäuffele / Zurawka 2013 SCHÄUFFELE, Jörg ; ZURAWKA, Thomas: *Automotive Software Engineering*. Wiesbaden : Vieweg und Teubner, 2013. – ISBN: 978-3-8348-2469-1. – DOI: 10.1007/978-3-8348-2469-1.

Tanenbaum 2003 TANENBAUM, Andrew S.: *Computer-Netzwerke - 4., aktualisierte Auflage*. Pearson Studium, 2003. – ISBN: 3-8273-7046-9.

Tindell / Burns 1994 TINDELL, Ken ; BURNS, Alan: *Guaranteed Message Latencies for Distributed Safety-Critical Hard Real-Time Control Networks*. 1994.

Todorov / Steinbach / Korf u. a. 2013 TODOROV, Lazar T. ; STEINBACH, Till ; KORF, Franz u. a.: „Evaluating Requirements of High Precision Time Synchronisation Protocols using Simulation“. In: *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques*. Cannes, France : ACM-DL, 2013-03, S. 307–313. – ISBN: 978-1-4503-2464-9.

Tanenbaum / Steen 2008 TANENBAUM, Andrew S. ; STEEN, Maarten van: *Verteilte Systeme, Prinzipien und Paradigmen*. 2., aktualisierte Aufl. Pearson Studium, 2008. – ISBN: 3-8273-7293-3. Andrew S. Tanenbaum ; Maarten van Steen.

TTTech Computertechnik AG TTTECH COMPUTERTECHNIK AG. Wien . URL: http://www.tttech.com Abruf: 2014-03-24.

TU Wien 1997 TU WIEN: *The TTP Protocols*. 1997. URL: http://www.vmars.tuwien.ac.at/projects/ttp/ttpmain.html.

Tuohy / Glavin / Hughes u. a. 2014 TUOHY, Shane ; GLAVIN, Martin ; HUGHES, Ciaran u. a.: Intra-Vehicle Networks: A Review. In: *Intelligent Transportation Systems, IEEE Transactions on* PP (2014), Nr. 99, S. 1–12. – ISSN: 1524-9050. – DOI: 10.1109/TITS.2014.2320605.

Varga 2001 VARGA, András: „The OMNET++ discrete event simulation system“. In: *Proceedings of the European Simulation Multiconference*. Prague : SCS – European Publishing House, 2001-06, S. 319–324.

Vector Informatik VECTOR INFORMATIK: *DBC Communication Database for CAN*. URL: http://www.vector.com/vi_canoe_de.html Abruf: 2014-08-07.

Varga / Hornig 2008 VARGA, András ; HORNIG, Rudolf: „An Overview of the OMNeT++ Simulation Environment“. In: *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*. (Simutools '08). Marseille, France : ICST (Institute for Computer Sciences, Social-Informatics und Telecommunications Engineering), 2008, 60:1–60:10. – ISBN: 978-963-9799-20-2. URL: http://dl.acm.org/citation.cfm?id=1416222.1416290.

Voelter / Benz / Dietrich u. a. 2013 VOELTER, Markus ; BENZ, Sebastian ; DIETRICH, Christian u. a.: *DSL Engineering - Designing, Implementing and Using Domain-Specific Languages.* dslbook.org, 2013, S. 1–558. – ISBN: 978-1-4812-1858-0.

Völter VÖLTER, Markus: *Werkzeuge zur Erstellung und Verarbeitung von DSLs*. URL: http://www.voelter.de/data/articles/ToolsFuerDSLs.pdf Abruf: 2014-08-31.

Verma / Zhang / Ferrari 1991 VERMA, Dinesh V. ; ZHANG, Hui ; FERRARI, Domenico: „Delay jitter control for real-time communication in a packet switching network“. In: *Communications Software, 1991, 'Communications for Distributed Applications and Systems', Proceedings of TRICOMM '91., IEEE Conference on*. 1991-04, S. 35–43. – DOI: 10.1109/TRICOM.1991.152873.

Winner / Hakuli / Wolf 2009 WINNER, Hermann ; HAKULI, Stephan ; WOLF, Gabriele: *Handbuch Fahrerassistenzsysteme - Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort : mit 550 Abbildungen und 45 Tabellen*. 2009. Aufl. Berlin Heidelberg New York : Springer-Verlag, 2009. – ISBN: 978-3-834-80287-3.

Weingartner / Lehn / Wehrle 2009 WEINGARTNER, Elias ; LEHN, Hendrik vom ; WEHRLE, Klaus: „A Performance Comparison of Recent Network Simulators“. In: *Communications, 2009. ICC '09. IEEE International Conference on*. 2009-06, S. 1–5. – DOI: 10.1109/ICC.2009.5198657.

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §22(4) bzw.§24(4) ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 26. September 2014 Fabian Kempf