Technische Universität München
Lehrstuhl für Kommunikationsnetze

Bordnetze für verteilte heterogene Subsysteme

Dipl.-Ing. (Univ.) Bernd Müller-Rathgeber

Vollständiger Abdruck der von der Fakultät Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr.-Ing. Gerhard Rigoll
Prüfer der Dissertation:	1.	Univ.-Prof. Dr.-Ing. Jörg Eberspächer
	2.	Univ.-Prof. Dr. sc.techn. Andreas Herkersdorf

Die Dissertation wurde am 15.04.2010 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 26.11.2010 angenommen.

Bordnetze für verteilte heterogene Subsysteme

Dipl.-Ing. (Univ.) Bernd Müller-Rathgeber

15. April 2010

Vorwort

Diese Arbeit ist während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Kommunikationsnetze der Technischen Universität München entstanden.

Mein besonderer Dank gebührt Herrn Prof. Dr.-Ing. Jörg Eberspächer, dass er mir die Chance gegeben hat, meine Kreativität bei der Arbeit am Lehrstuhl frei zu entfalten.

Dann danke ich natürlich meinen Eltern, die mir nicht nur das Studium an der TU ermöglicht, sondern mich immer auf meinen Wegen unterstützt haben.

Den Kollegen am Lehrstuhl, welche für ein einmalig freundliches Arbeitsklima sorgten und mit denen man gerne auch außerhalb der Arbeit seine Zeit verbringt, gilt ebenso mein Dank. Speziell erwähnen möchte ich Michael Eichhorn, der mir während der gesamten Projektlaufzeit immer zur Seite stand, sowie Robert Nagel, der als Bürokollege immer ein offenes Ohr für Diskussionen und Fragen hatte und auch die Aufgabe des Lektors übernommen hat.

Mein Dank gilt auch dem Industriepartner im Projekt IT_Motive 2020, der BMW Group Forschung und Technik GmbH, Herrn Prof. Dr.-Ing. habil Raymond Freymann sowie Herrn Steinberg als Abteilungsleiter der ZT-4, die den Zugang zu interessanten Gesprächen und technischen Inhalten ermöglichten. Speziell auch dem Projektleiter Hans-Ulrich Michel für das entgegengebrachte Vertrauen und seine Ruhe in allen Lagen habe ich zu danken.

Und natürlich danke ich Herrn Prof. Dr. sc.techn. Andreas Herkersdorf für die Übernahme des Zweitgutachtens meiner Arbeit.

München, April 2010 Bernd Müller-Rathgeber

Kurzfassung

Die Arbeit beschäftigt sich mit Konzepten für die Vernetzung von Steuergeräten in verteilten Echtzeitsystemen in Fahrzeugen. Es wurde ein Systemdesign für zukünftige Bordnetze durchgeführt und die Ergebnisse wurden verifiziert.

Die ständige Steigerung des Vernetzungsgrades und der notwendigen Datenraten der Funktionen in Fahrzeug-Bordnetzen führt zu einer steigenden Komplexität der vorhandenen Systeme. Die vorgeschlagene neue Kommunikationsarchitektur trägt dazu bei, diese Komplexität zu verringern und dabei die Zuverlässigkeit des Gesamtsystems zu verbessern.

Es wird eine Lösung aufgezeigt unter Verwendung von IEEE 802.3 Ethernet in einer Sterntopologie. Erweiterungen, um in diesem Netz Realzeiteigenschaften sicherzustellen, werden beschrieben. Weiter wurden Methoden und Hilfsmittel entwickelt, um die Grenzen des Realzeitverhaltens nachzuweisen und das Kommunikationsverhalten zu simulieren. Die Lösung ermöglicht inhärent eine Fehlerauflösung bei Ausfall von Ressourcen durch mehrere mögliche Pfade durch das Netz und ein Sicherheits- und Konfigurationsmanagement.

Der begrenzte Platz, der Stromverbrauch und die Kosten machen eine genaue Planung der Komponenten des Kommunikationsnetzes notwendig. Es wurde eine Optimierungsumgebung konzipiert auf der Basis eines Genetischen Algorithmus und unter Verwendung problemangepasster Metriken. Mit diesem Verfahren wurden die benötigten Sternknoten dimensioniert und deren Platzierung, die Kabelverbindungen zwischen den Steuergeräten sowie der Einsatz von Aggregationsnetzen mit CAN-Bussen optimiert.

Um eine effiziente Übertragung der Daten zu gewährleisten, wurde ein neues Kommunikationsprotokoll definiert und implementiert. Der Austausch von Daten über auf unterschiedlichen physikalischen Schichten basierende Subnetze, Zeitstempel und variable Datensicherung sind in diesem Protokoll bereits verankert. Die Aggregation mehrerer Pakete zu einem Rahmen minimiert die durch den Nachrichtenkopf benötigte Datenmenge.

Die entwickelten Lösungen sind auch für andere Bereiche realzeitfähiger Systeme wie Avionik und Industrieautomatisierung anwendbar.

Inhaltsverzeichnis

Abkürzungen

AFDX	Avionics Full Duplex Switched Ethernet
CAN	Controller Area Network
CP	Control Plane
CSMA	Carrier Sense Multiple Access
ECU	Electronic Control Unit
FIBEX	Field Bus Exchange Format
GA	Genetischer Algorithmus
GMII	Gigabit Media Independent Interface
HIL	Hardware in the Loop
HS-CAN	High Speed CAN
IFG	Inter Frame Gap
IP	Internet-Protokoll
LIDAR	Light Detection and Ranging
LIN	Local Interconnect Network
LS-CAN	Low Speed CAN
MAC	Media-Access-Control
MOST	Media Oriented Systems Transport
MP	Management Plane
MTU	Maximum Transmission Unit
OEM	Original Equipment Manufacturer
PEP	Produktentstehungsprozess
POF	Polymer-optische Faser
QoS	Quality of Service
RADAR	Radio Detection and Ranging
SoC	System on a Chip
TP	Transport Plane
TT	Transmission Time
uMAP	universal Multilayer Automotive Protocol

VK Virtuelle Kostenfunktion

VL Virtual Link

WCTT Worst Case Transmission Time

1 Einleitung

Das Moore'sche Gesetz [Moo65], als Faustregel im Bereich der Rechenleistung, prognostiziert einen exponentiellen Zuwachs der Komplexität integrierter Schaltkreise über die Jahre. Die Kosten sind in dieser Betrachtung die maßgeblichen Treiber der Entwicklung.

Es hat sich gezeigt, dass diese Faustregel sich auf viele Bereiche der Elektrotechnik sowie der Informationstechnik anwenden lässt. Weinmann hat die Menge an Software-Code im Bordnetz von Personenwagen untersucht [Wei] und bemerkte eine analoge Entwicklung. Die Menge an Speichern, kumuliert über alle im Fahrzeug verbauten Electronic Control Units (ECUs), hinkt dem Speicher von Personalcomputern um ungefähr sieben Jahre hinterher, entwickelt sich aber auch mit einer Verzehnfachung alle vier Jahre weiter. Betrachtet man die Leistung von Prozessoren in eingebetteten Systemen oder die Datenrate der Feldbusse in Bordnetzen, ergeben sich auch dort analoge Verhältnisse.

Nimmt man eine beliebige Funktion aus einem Fahrzeug und beobachtet diese über die letzten 20 Jahre hinweg, bestätigt sich die Aussage. Bestand die Funktion „Blinken" früher aus einem Lenkstockhebel, zwei Leuchtmitteln und ein wenig Kabel, sind mittlerweile in Serienfahrzeugen mehr wie sechs getrennte Rechner an mehr wie drei unterschiedlichen Kommunikationsnetzen an dem Vorgang beteiligt.

Folgt man nun Moores Regel, haben Bordnetze in 10 Jahren Kommunikationsressourcen und Rechenressourcen, wie sie im heutigen Büroumfeld zu finden sind. Die Komplexität der dort ausgeführten Funktionen steigt in vergleichbarem Maße und damit auch die Verknüpfung der Funktionen untereinander.

1.1 Motivation der Arbeit

Die Idee dieser Arbeit ist es nun zu untersuchen, inwieweit Technologien, welche zum heutigen Zeitpunkt im Büroumfeld und in Rechenzentren eingesetzt werden, Auswirkungen auf zukünftige Entwicklungen im Bereich von Bordnetzen für verteilte heterogene Subsysteme haben werden.

Die Motivation ist, durch Abstraktionsschichten, durch Virtualisierung von Funktionen und ein leistungsfähiges Kommunikationsnetz neue Funktionalitäten, höhere Ausfallsicherheit und deutliche Kosteneinsparungen im Produktentstehungsprozess (PEP) zu erreichen sowie neue Geschäftsfelder zu eröffnen. Der Ansatz ist die räumliche Trennung von Datenquelle, Datenverarbeitung und Datensenke sowie das Aufbrechen der Verknüpfung zu physikalischen Einheiten. Dies impliziert zwei Para-

digmenwechsel: Sensor/Aktor und Kundenfunktion werden räumlich getrennt realisiert und Kundenfunktionen können überall ausgeführt werden. Dies spiegelt den „Cloud"-Gedanken wider, der die Verarbeitungsleistung als virtuelles Gut definiert, das an beliebigen Orten mit beliebigen Ressourcen zur Verfügung steht. Dies ermöglicht nun zahlreiche Erweiterungen, die in klassischen Bordnetzen zum heutigen Zeitpunkt kaum realisierbar wären. Eine unvollständige Aufzählung von Beispielen folgt im nächsten Absatz.

Ausfallsicherheit: Muss eine Funktion nicht mehr auf einer dedizierten, fest zugewiesenen ECU ausgeführt werden, erhöht sich die Verfügbarkeit und damit die Ausfallsicherheit der Funktion. Unter der Nebenbedingung, dass die Cloud auch nach Abzug der Ressourcen der fehlerhaften ECU ausreichende Verarbeitungsleistung besitzt, kann die Funktion nahtlos an anderer Stelle weiter ausgeführt werden. Gerade im Hinblick auf die stärkere Verschmelzung der Funktionen untereinander, die zu einer Verlängerung der so genannten Wirkketten führt, hat ein Glied dieser Kette und damit eine Funktion starke Auswirkungen auf das Gesamtsystem. Fällt eine Funktion aus, sind meist viele andere zusätzlich davon betroffen.

Erweiterbarkeit: Eine Cloud kann beliebig vergrößert oder verkleinert werden. Der Lebenszyklus eines Fahrzeuges beträgt zehn und mehr Jahre. Der Innovationszyklus in der Informations- und Kommunikationsindustrie ist jedoch weit kürzer und beträgt nur wenige Jahre. Dies führt dazu, dass Fahrzeuge während ihrer Verwendung früher oder später technisch veraltet sind. Bei einem PEP von fast drei Jahren kann dies bereits zum Start der Produktion der Fall sein. Spielt man den Cloud-Gedanken weiter, kann mit einer Art Plug-and-Play-Methode veraltete ECUs gegen neue Versionen gleicher Bauform, aber mit höheren Ressourcen ausgetauscht werden. Gerade der Bereich Aftermarket gewinnt damit für den Original Equipment Manufacturer (OEM) deutlich an Bedeutung. Die gewonnenen Ressourcen in der Verarbeitungsleistung können dann mit neuen Funktionen im Bereich der Fahrzeugsysteme vom OEM oder durch den Kunden und Dritthersteller selbst gefüllt werden. Als Zukunftsszenario ist auch wie bei Google Android und Apple iPhone ein App-Store mit kostenlosen und kostenpflichtigen Zusatzfunktionen denkbar oder eine Mietmodell, bei dem Funktionalität nur für den wirklich verwendeten Zeitraum bezahlt werden muss.

Flexibilität: Betrachtet man Verarbeitungsleistung nun als frei skalierbares Gut mit einer definierten Abstraktionsebene zu den Anwendungen, schafft das Flexibilität in beide Richtungen. Die Hardware unterhalb der Abstraktionsschicht kann verändert werden, ohne dass die Funktionen oberhalb derselben beeinflusst werden. So können Produkte verschiedener Zulieferer über einen längeren Zeitraum oder sogar gleichzeitig eingesetzt werden. Auch können neue Technologien eingeführt werden ohne die Notwendigkeit der Anpassung aller Funktionen. Dies steigert den Wiederverwendungsgrad der Software. Im Gegenzug kann die gleiche Software in verschiedenen Systemen und damit zum Beispiel unterschiedlichen Baureihen eines OEM eingesetzt werden. Dieser steigende Einsatz von Gleichteilen fördert die Kosten- und Komplexitätsreduktion und die Modularisierung mit einem Hardware-Software-Baukasten.

Ähnliche Probleme wie die Realisierung von Clouds im Internet und im Büroumfeld

treffen auch auf die Ansätze in Bordnetzen zu: Eine definierte Dienstqualität muss sichergestellt werden. Während im Internet ein Überschreiten dieser Grenzen nur monetären Schaden verursacht, können in Bordnetzen katastrophale Ereignisse bis zum Tod von Menschen führen. Um die Ansätze übernehmen zu können, muss deterministisches Verhalten bei der Kommunikation der Daten sowie bei der Verarbeitung in der Cloud ermöglicht und gegenüber staatlichen Überwachungsorganisationen nachgewiesen werden.

Für die Kommunikation in solchen Bordnetzen bedeutet dies, dass über eine physikalische Verbindung mehrere virtuelle Datenverbindungen laufen, ohne dass diese sich gegenseitig stören. Ideal wäre eine vollständige gegenseitige Nichtbeeinflussung, welche aber nur mit hohem Aufwand realisierbar ist. Wichtig ist vielmehr, dass zu jedem Zeitpunkt das Verhalten der Datenverbindung in ihren Grenzen vorhersagbar sein muss. Verwendet werden dabei Virtualisierungstechniken, um den gemeinsamen Kanal für die einzelnen Verbindungen zu partitionieren. Um die Ortsflexibilität zu erreichen, wird eine Schicht benötigt, welche die logische Verbindung zwischen Quelle und Senke der Datenverbindung vom physikalischen Weg mit eventuell notwendigen Zwischenstationen abstrahiert. Nur so ist eine Änderung des Verarbeitungsortes in der Cloud ohne Anpassung der Funktionskette möglich. Die Abstraktionsschicht muss damit ein passendes Adressierungsverfahren für Kommunikationspartner sowie ein Mechanismus zur Wegesuche beinhalten.

1.2 Beitrag der Arbeit

Die Arbeit behandelt den Einsatz von den im Büroumfeld vorherrschenden Techniken und Methoden in Bordnetzen für verteilte heterogene Subsysteme. Untersucht wurde speziell der Einsatz von IEEE 802.3 Ethernet in Punkt-zu-Punkt-Verbindungen über einen Switch als Sternknoten zur Verbindung der einzelnen Geräte sowie die Eignung des Internet-Protokoll (IP) als universelles Netzprotokoll.

Um dies zu erreichen, wurden zuerst generell die in unterschiedlichen Industriebereichen vorhandenen Lösungen zur Kommunikation in eingebetteten Systemen untersucht und bewertet. Dabei wurde neben der Automobilindustrie eine Blick auf die Automatisierungstechnik sowie die Luftfahrtindustrie geworfen. Betrachtet wurden aktuelle Lösungen sowie in der Entwicklung und Forschung befindliche zukünftige Technologien. Um die gestellten qualitativen und quantitativen Anforderungen an die Lösungen zu erhalten, wurden die Bordnetze sowie das Kommunikationsverhalten mehrere Fahrzeuge untersucht und ausgewertet. Durch Abgleich der gewonnenen Erkenntnisse und Extrapolation der Daten wurde eine Referenzarchitektur für die Realisierung eines Bordnetzes mit Zielhorizont dem Jahr 2020 entwickelt.

Die dringlichste Erweiterung beim Einsatz von Ethernet für Kommunikation in Bordnetzen ist die Nachweisbarkeit eines deterministischen Verhaltens. Mit den vorhandenen Methoden des Standes der Technik können zwar Aussagen über die zu erwartende Dienstqualität gegeben, nicht aber garantiert werden. Bei dem Einsatz in Fahrzeu-

gen und bei anderen Einsatzzwecken kann aber bereits der Verlust einer übertragenen Information durch zu große Verzögerung zu katastrophalen Ereignissen führen. In der Arbeit wurde deswegen ein spezielles Design für die Sternknoten entwickelt, die eine deterministische Übertragung ohne Modifikation des IEEE 802.3-Standard-Medienzugriffes ermöglicht. Weiter wurde mit Hilfe analytischer Methoden ein Algorithmus zur Berechnung der größten auftretenden Verzögerung eines jeden Datenpaketes entwickelt. Somit kann die so genannte Worst Case Transmission Time (WCTT) während der Designphase berechnet werden.

Um die Ausfallsicherheit des Gesamtsystems zu erhöhen und die geforderte Flexibilität zu ermöglichen, wurde ein Sicherheits- und Konfigurationsmanagement entwickelt. Das Managementsystem besteht aus verteilten Komponenten, welche an den Sternknoten die Informationen über die verbundenen Kanten generieren. Diese werden über das Kommunikationsnetz zu einer zentralen Instanz übertragen und dort aggregiert sowie ausgewertet. Auf der Basis der Daten wird anschließend das Routing durchgeführt und die Konfiguration der einzelnen Sternknoten ausgeführt. Die Konfiguration besteht aus den Filtern für die Datenströme, der Konfiguration der Warteschlangensysteme sowie dem Setzen der Routen. Durch dieses zentrale Setzen und Überwachen der Routen ist es nun möglich, Datenströme über gewünschte Wege zu leiten. Sind mehr als ein Weg möglich, kann der bessere (was nicht mit dem kürzeren gleichzusetzen ist) gesetzt und bei einer Störung auf den alternativen Pfad umgestellt werden. Mit ähnlichen Mechanismen können neue Geräte in das Netz gebracht werden. In dieser Arbeit entstand die Architektur des Sicherheits- und Konfigurationsmanagements mit der Partitionierung der Funktionen und der Informationsübertragung. Das System wurde implementiert und in einem Laboraufbau sowie einem Prototypen eingesetzt, um die Funktionalität zu evaluieren. Es wurde untersucht, inwieweit ein Optimieren des Routings Verzögerungen im Bordnetz bei der Übertragung der Daten minimieren kann. Dafür wurden mehrere Optimierungsziele definiert und miteinander verglichen.

Betrachtet man Topologien in klassischen Büronetzen, so ist die vorherrschende eine Sterntopologie. Die klassischen Lösungen mit Hilfe von Feldbussen sind, wie der Name bereits vermuten lässt, in Bustopologie angeordnet. Oftmals wird auch der Begriff „Daisy Chain“ dafür verwendet. Diese Änderung im Paradigma stellt hohe Herausforderungen, denn es müssen zusätzliche Sternknoten zum Verbinden der ECUs im Bordnetz verbaut werden. In dieser Arbeit wurden Metriken entwickelt, welche die Aufwände für die zusätzlichen Sternknoten quantifizieren, um auf reale Kosten abbildbare Werte zu erhalten. Diese Werte wurden mit Hilfe eines entwickelten Optimierungsverfahrens auf der Basis eines evolutionären Algorithmus minimiert, indem die Anzahl sowie die Platzierung der Sternknoten und die Verkabelung zwischen diesen und den ECUs optimiert wurde. Diese Methode wurde für klassische Bordnetze entwickelt sowie auf redundante Netze mit zwei kanten- und knotendisjunkten Wegen erweitert. Zur weiteren Kostenoptimierung wurde der Einsatz heterogener Lösungen mit Controller Area Network (CAN) als Aggregationsnetz untersucht. Ergebnis ist ein kostenoptimales Bordnetz bei der Verwendung einer Sterntopologie.

Für den Einsatz in heterogenen Bordnetzen, die als kostengünstige Möglichkeit bei

den Untersuchungen in dieser Arbeit entwickelt wurden, wurde ein spezielles Übertragungsprotokoll definiert, um die Effizienz des Kommunikationsnetzes zu erhöhen. Das universal Multilayer Automotive Protocol (uMAP) befindet sich im OSI-Modell auf Schicht 3-5, also oberhalb der zur Übertragung verwendeten Technologie. uMAP unterstützt, neben anderen, die transparente Kommunikation in Subnetzen mit inhomogenen Protokollen über Gateways, die Aggregation kleinerer Informationseinheiten zu einer Gesamtnachricht sowie das zeitsynchrone Verarbeiten der Nachrichten mit einer Genauigkeit im μ-Sekunden Bereich. Die Verschmelzung mehrerer Schichten ermöglicht ein ressourcenschonenderes Verarbeiten der Daten. So können durch die integrierte Fragmentierung und Zeitsynchronisierung zum Beispiel Audio- und Videodaten ohne Durchlauf weiterer Protokollebenen direkt an verteilte Senken übertragen werden.

Im Rahmen der vorliegenden Arbeit entstand ein Entwurfs- und Planungssystem, welche die in den vorherigen Abschnitten aufgezeigten Methoden in einem Framework vereinen. Durch den Einsatz von Parallel-Programmierung und *General Purpose Computation on Graphics Processing Unit* (GPGPU) wurde ein leistungsstarkes Optimierungssystem geschaffen. Die einzelnen Komponenten sind über XML-basierte Schnittstellen zum Datenaustausch verbunden.

1.3 Struktur der Arbeit

Nach dieser Einleitung in Kapitel 1 beschäftigt sich Kapitel 2 mit dem Stand der Technik und den Ansätzen in der Forschung im Bereich der Bordnetze. Es werden die verwendeten Zeitbegriffe definiert sowie die Anforderungen und Annahmen festgelegt, auf denen die weiteren Arbeiten basieren.

Kapitel 3 behandelt dann die notwendigen Erweiterungen und Methoden zur Berechnung der garantierten Übertragungsverzögerung sowie den Aufbau des Sicherheits- und Konfigurationsmanagements. Die verwendeten Optimierungsmethoden, Metriken sowie die gesetzten Zielfunktionen werden in Kapitel 4 hergeleitet und die Ergebnisse diskutiert.

Das universal Multilayer Automotive Protocol (uMAP) und der Vergleich zum weit verbreiteten Internet-Protokoll (IP) ist Inhalt des Kapitels 5. Die Besonderheiten des Kommunikationsprotokolls und die Felder im Nachrichtenkopf werden erläutert. Eine Referenz der Inhalte der Kopffelder befindet sich in Anhang C.

Die Kombination der entwickelten Methoden zu einem Entwurfs- und Planungssystem wird in Kapitel 6 beschrieben. Dort wird auch auf das Projekt IT_Motive 2020 in Zusammenarbeit mit der BMW Forschung und Technik GmbH und der BMW Group AG im Rahmen der CAR@TUM-Initiative sowie das dort erstellte Prototypenfahrzeug eingegangen.

Die Ergebnisse der Arbeit werden in Kapitel 7 zusammengefasst und es wird ein kurzer Ausblick auf weiter Entwicklungen gegeben.

2 Architektur von Bordnetzen

In diesem Kapitel werden der Begriff Bordnetz im Allgemeinen sowie die Besonderheiten dieser Netze behandelt. Es werden Ansätze aus verschiedenen Industriebereichen betrachtet und eine Lösung für zukünftige Bordnetze in Fahrzeugen wird erarbeitet. In Absatz 2.3 werden die zukünftigen Kommunikationsanforderungen an mögliche Lösungen genauer aufgezeigt. Vor allem die Konvergenz der Netze von spezialisierten Einzellösungen zu homogenen, universell einsetzbaren Systemen bildet den Schwerpunkt. Die zeitlichen Anforderungen, also auf welchen Vorgaben und Annahmen die weitere Beurteilung der Lösungsalternativen basiert, wird in Abschnitt 2.4 definiert.

2.1 Bordnetz

Auf einer bekannten Webseite ist zu finden: „Als Bordnetz wird die Gesamtheit aller elektronisch/elektrischen Komponenten (E/E-Komponenten) in einem Automobil, Flugzeug, Zug oder anderen Fahrzeug bezeichnet. Der Begriff Bordnetz wird allgemein bei fast allen Verkehrsmitteln genutzt.“ (Wikipedia).

Wir betrachten hierbei ausschließlich die Kommunikation zwischen den Steuergeräten, auch Electronic Control Unit (ECU) genannt, und nicht die Stromversorgung oder die ECUs selbst. Das heißt, der Begriff Bordnetz steht im Folgenden für die Kommunikationsnetze zur Verbindung der verteilten Systeme. Dabei kann ein Bordnetz aus mehreren unterschiedlichen, über Gateways verbundenen Kommunikationsnetzen oder einzelnen Kommunikationsleitungen bestehen.

2.2 Stand der Technik

Obwohl diese Arbeit primär im Bereich der Automobile entstanden ist, sind die Lösungen allgemein gültig. Aus diesem Grund wurden von Anfang an Architekturen aus anderen Industriebereichen betrachtet. Von den Anforderungen ähnlich und damit relevant sind, neben Fahrzeugnetzen, Ansätze aus der Avionik, wie die Kommunikationsarchitekturen in Passagierflugzeugen der neusten Generation, sowie aus der Automatisierungstechnik. Gerade im Bereich Zuverlässigkeit und Zeitverhalten stellen diese hohe Anforderungen.

Trotzdem können diese Lösungen nicht direkt auf den Bereich Fahrzeugnetze angewendet werden, da hier die Stückkosten sowie Stromverbrauch und Bauraum eine viel höhere Rolle spielen.

2.2.1 Fahrzeug

Wie oben erwähnt, sind bei Kraftfahrzeugen, und damit sind speziell, aber nicht ausschließlich Personenwagen gemeint, die Stückkosten, also die tatsächlichen Produktionskosten der verwendeten Hardware, maßgeblich für Designentscheidungen. Aus diesem Grund haben sich mehrere, für den jeweiligen Zweck ideale Lösungen durchgesetzt und ein vermischtes, heterogenes System gebildet. Abbildung 2.1 zeigt exemplarisch das Bordnetz eines Luxusklasse-Fahrzeuges mit Vollausstattung.

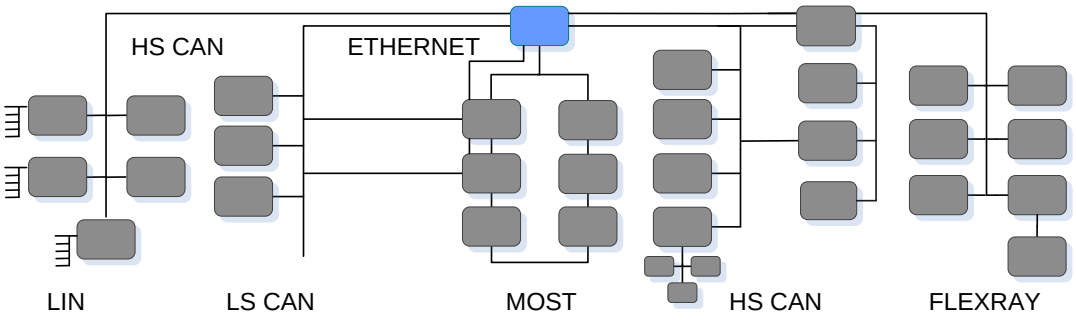


Abbildung 2.1: Fahrzeug-Architektur

Das Netz ist aufgeteilt in mehrere, funktional unterschiedliche Bereiche, die so genannten Domänen. Allgemein durchgesetzt hat sich die Aufteilung nach: Safety, Powertrain, Body und Infotainment.

Domäne	Technologie	Datenrate	Medienzugriff
Body	LS-CAN	100 kbit/s	CSMA/CR
Powertrain	HS-CAN	500 kbit/s	CSMA/CR
Safety	Flexray	5 Mbit/s	Time Triggered
Infotainment	MOST	50 Mbit/s	Circuit Switched
Aggregation	LIN	20 kbit/s	Single Master

Tabelle 2.1: Übersicht Fahrzeugnetze

Die in Tabelle 2.1 gezeigten Zuordnungen sind exemplarisch und zeigen typische verwendete Technologien und Datenraten auf. Jeder Fahrzeughersteller hat eigene interne Vorgaben, die hiervon abweichen. Das Buch von W. Zimmermann, „Bussysteme in der Fahrzeugtechnik“ [ZS06], gibt einen guten Überblick über den aktuellen Stand der Technik.

Der Controller Area Network (CAN) [Bos91] in den zwei Ausführungen *High Speed* und *Low Speed* ist der am meisten verbreitete Bus im Fahrzeug. Von vielen Herstellern gibt es bereits Prozessoren mit CAN-Transceivern als System on a Chip (SoC) für einfachste Integration. Der gemeinsame Zugriff wird über den CAN-Identifier, der gleichzeitig die Priorität der Übertragung festlegt, und eine Arbitrierungsphase geregelt. Flexray [bel02] ist die jüngste Innovation. Es kombiniert Kommunikation in

festen Zeitschlitzen mit einem ereignisbasierten Zugriff am Ende des Kommunikationszyklus. Durch den auf einem Zeitschlitz basierenden Zugriff kann einfach eine garantierte Übertragungsverzögerung festgelegt werden. Zur Übertragung von Audio-Strömen dient der Media Oriented Systems Transport (MOST) [mos04] Bus. Als Ring-Topologie ausgelegt über eine Polymer-optische Faser (POF) können mit 25, 50 und in neuster Entwicklungsstufe 150 Mbit/s Audiokanäle, Daten und Videos übertragen werden. MOST ist ein proprietäres System der MOST Cooperation, was mit hohen Lizenzkosten bei der Verwendung verbunden ist. Um Motoren und Schalter mit einer ECU zu verbinden, zum Beispiel im Bereich der Klimaregelung oder der Sitzverstellung, wird oft Local Interconnect Network (LIN) [lin99] verwendet. Es ist ein serieller Bus, bei dem ein Master, die ECU, mit einem festen Zyklus die Slave-Stationen abfragt oder Daten an diese sendet. LIN wird mehr und mehr von CAN abgelöst, da die Bauteilekosten sich immer weiter annähern. Daneben gab es noch mehrere andere Ansätze, wie Byteflight, D-Bus, IDB-1394 [Vol00], basierend auf Firewire, die nur kurz am Markt zu sehen waren.

Nachteil dieser Entwicklung ist die Notwendigkeit, Daten in mehreren Bussystemen parallel verfügbar zu machen. Denn moderne Assistenzsysteme benötigen viele Daten von anderen Steuergeräten, um ein Fahrzeugzustandsmodell aufzubauen und damit optimal zu reagieren. In Abbildung 2.1 ist ein zentrales Gateway zu sehen, das diese Aufgaben übernimmt, sowie einige ECUs, die an mehreren Netzen parallel angeschlossen sind. Dies erhöht natürlich die Systemkomplexität und verringert die Ausfallsicherheit durch das Gateway als „Singe Point of Failure“.

Es gibt eindeutige Trends und Bemühungen, Ethernet in das Fahrzeug zu bringen und damit weitere spezielle Lösungen zu ersetzen. Erste Ethernet-Verbindungen werden bereits in Serienfahrzeugen verwendet, meist zum Aufladen neuer Firmware oder im Bereich Infotainment. Besondere Erwähnung sollten hier die Arbeiten um Kopetz mit TTA [KB03], TTP und TTE [KAGS05, AKGS06] finden. Untersuchungen für den Einsatz im Bereich Automotive wurden unter anderem von Hedenetz et al. [Hed98, RSBH98] durchgeführt. „Time-Triggered Ethernet“ (TTE) ist ein Ansatz, basierend auf einer unmodifizierten IEEE 802.3 Ethernet physikalischen Schicht ein realzeittaugliches Netz aufzubauen. Verwendet wird dabei eine redundante Stern-Topologie [AKG+06] mit einem auf FPGAs realisierten Switch [SGAK06]. Der Nachteil der Verwendung von festen Zeitschlitzen bei der Kommunikation ist in Kapitel 3.1.2 dargelegt.

2.2.2 Flugzeug

Als weiterer Industriezweig wurde die Luftfahrt und speziell die Personenbeförderung betrachtet. Mit den neuen Modellen Airbus A380 und Boeing 777 kamen auch neue Ansätze von Bordnetzen zum Einsatz. Herausgehoben werden muss Avionics Full Duplex Switched Ethernet (AFDX) als auf Ethernet basierendes System mit Erweiterungen für sicherheitskritische und zeitkritische Anwendungen. Die unter dem ARINC-Standard 664 [ari05] zusammengefassten Erweiterungen von Ethernet umfas-

sen neben anderem Quality of Service (QoS) und Source Routing.

Das 100 Mbit/s schnelle AFDX kann über POF oder Kupferkabel verwendet werden und besitzt eine Stern-Topologie. Es werden Virtual Links (VLs) zwischen den Endsystemen durch das Netz auf vordefinierten Pfaden gelegt. Jeder VL besitzt eine garantierte Datenkapazität. Mit Hilfe des *Bandwidth Allocation Gap*, eines für jeden VL definierten Mindestabstandes zwischen zwei Datenpaketen, wird der Zugriff auf das gemeinsame Medium beschränkt. Die Hauptaufgabe der Zeiteinteilung übernimmt dabei die Netzverbindung in den Endsystemen, in den Switchen wird zusätzlich eine Zeitkontrolle durchgeführt.

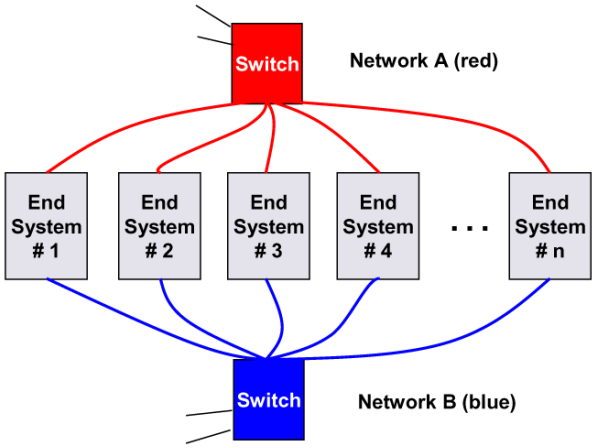


Abbildung 2.2: AFDX-Architektur

Zur Erhöhung der Verfügbarkeit des Bordnetzes wird 1:x-Resilience, also ein so genanntes Schattennetz verwendet. Dies bedeutet, es gibt x Netze parallel und die Daten werden über alle Systeme gleichzeitig übertragen. AFDX ist für zwei parallele Anschlüsse spezifiziert, die Netze werden meist *red* und *blue* genannt, wie in Abbildung 2.2 beispielhaft zu sehen. Bei hochverfügbaren Systemen, wie sie unter anderem in Kampfjets zu finden sind, werden bis zu fünf Netze verbaut. Ein Grund ist, dass bei einer dualen Redundanz, also dem Vorhandensein zweier empfangener Werte, zwar ein Fehler detektiert, nicht aber korrigiert werden kann. Dort werden „2-aus-3“- oder „4-aus-5“-Entscheidungen verwendet.

In Flugzeugen wird also stark auf die Verringerung der Systemkomplexität und die Maximierung der Systemverfügbarkeit Wert gelegt. Dies wirkt sich direkt auf die Stückkosten des Bordnetzes aus, die weit über denen der Automobilbranche liegen.

2.2.3 Automatisierungstechnik

Weitere verwandte Anforderung findet man in der Automatisierungstechnik, das heißt in verteilten Regelungen von Produktionsanlagen. Hauptsächlich im Markt befinden sich Bussysteme diverser Hersteller, welche signifikante Ähnlichkeiten zu denen in Fahrzeugen und in der Luftfahrt aufweisen. Und auch in diesem Bereich

ist ein starker Trend in Richtung Ethernet als Nachfolger dieser Bussysteme sichtbar. Aus diesem Grund wurden die Eigenschaften und besonderen Merkmale von Ethernet/IP, EtherCAT, Powerlink, PROFINET, SERCOS3 sowie weitere untersucht.

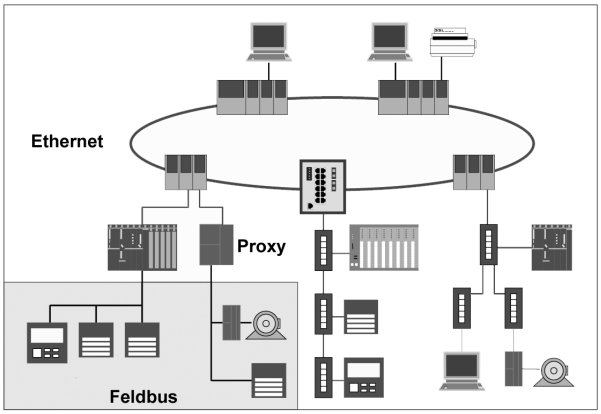


Abbildung 2.3: PROFINET-Architektur

Das Produkt PROFINET von *Siemens Automation and Drives* [FSN04] zeigt sehr flexible Architektureigenschaften auf, wie in einer Beispielarchitektur in Abbildung 2.3 zu sehen ist. Durch diese Flexibilität ist die Zukunftstauglichkeit von PROFINET größer und der Einsatz in unterschiedlichsten Szenarien denkbar. Über einen redundanten Ring werden Regelungen ausfallsicher miteinander verbunden, die wiederum mit Stichleitungen Geräte über PROFINET ansprechen oder über Gateways auf klassischen Bussysteme zugreifen. PROFINET selbst ist als Multi-Sterntopologie mit speziellen Koppelpunkten ähnlich einem Ethernet-Switch ausgelegt. Dieser separiert den Verkehr und ermöglicht damit unabhängig von den Regelungsaufgaben die Datenkommunikation über die gemeinsamen Leitungen. In den Arbeiten von Ferrari et al. [FFMT04, FFV05] wurde die Leistung von PROFINET im Hinblick auf die Einhaltung von zeitlichen Grenzwerten untersucht. Sie zeigen, dass eine auf Ethernet basierende Lösung für die hier gesetzten Anforderungen grundsätzlich möglich ist.

Nachteil der Lösungen aus der Automatisierungstechnik ist, dass sie überwiegend für den einmaligen, festen Einbau in großen Hallen entwickelt worden sind. Dennoch kann in späteren Kapiteln auf Konzepte dieser Produkte zurückgegriffen werden, um eine Lösung für die hier gestellten Anforderungen zu entwickeln.

2.3 Heterogene Kommunikationsanforderungen

Im Fahrzeuge-Bordnetz der Zukunft herrschen jedoch signifikant andere Anforderungen als für die in Kapitel 2.2 gezeigten Lösungen. Vor allem die stark heterogenen Kommunikationsverbindungen weisen unterschiedliche Eigenschaften auf, die ein Design der optimalen Kommunikationslösung schwierig machen. Einerseits sind die Regelkreise selbst sehr divergent, da Anwendungsbereiche wie die Temperaturregelung des Innenraums oder die Steuerung der elektronischen Dämpfer der Achse

andere Anforderungen besitzen. Auf diese wird im folgendem Kapitel 2.4.2 näher eingegangen.

Andererseits ist eine vollständige Medien- und Infotainment-Architektur in Fahrzeugen der Luxusklasse zu finden. Gerade diese steigende Anzahl an Videoverbindungen und Audio-Quellen und -Senken machen einen erheblichen Teil der Verkabelung aus. Während Audio bereits durch Technologien wie MOST auf Datennetze zwischen den ECU transportiert wird, sind fast alle Videoübertragungen als Punkt-zu-Punkt-Verbindungen realisiert. Die Anzahl von vier Displays sowie DVD, DVB, Navigation, Nachtsicht und Einparkhilfen mit fünf und mehr Kameras als möglichen Contenterzeugern bereits in aktuellen Modellen verdeutlichen die Herausforderungen für zukünftige Ansätze. Will man eine freie x-nach-y-Zuweisung der Inhalte gewähren, muss man eine aufwendige, zentrale Videomatrix verbauen, was nur in hochpreisigen Modellen möglich ist.

Die Daten dieser Kameras müssen nicht mehr ausschließlich dem Menschen, also dem Fahrer oder den Passagieren, zugänglich gemacht werden, auch moderne Fahrerassistenzsysteme basieren mehr und mehr auf den Daten komplexer Sensoren. Dies können einerseits Kameras sein, um zum Beispiel die Fahrspur oder die Sitzbelegung zu erkennen, andererseits Rohdaten von Radio Detection and Ranging (RADAR)- und Light Detection and Ranging (LIDAR)-Sensoren. RADAR und LIDAR können vom Standpunkt der Datenkommunikation als Graustufen- oder Farbvideos betrachtet und deswegen mit Lösungen dieser Klasse abgedeckt werden.

Die in dieser Arbeit gezeigte Lösung eines Bordnetzes für verteilte Systeme ermöglicht eine uneingeschränkte Übertragung von Audio- und Videodaten mit allen im Office- und Home-Bereich verfügbaren, auf IEEE 802.3 Ethernet aufbauenden Mechanismen. Es wird hier nicht näher auf die Wahl des geeigneten Übertragungsprotokolls eingegangen, dies ist Schwerpunkt anderer Arbeiten [Rah09].

2.4 Zeitverhalten

Um Regelkreise und deren Anforderungen an die Kommunikationsinfrastruktur zu definieren, wird das zeitliche Verhalten derselben verwendet. Dafür werden in Kapitel 2.4.1 die weiter verwendeten Begriffe eingeführt und die der Designentscheidung zugrunde liegenden Daten in Kapitel 2.4.2 erläutert.

2.4.1 Definition Zeitbegriff

In diesem Kapitel werden die grundlegenden Begriffe zur Einordnung der Zeiten, die bei der Übertragung der Daten auftreten, definiert. Die Übersicht ist in Abbildung 2.4 dargestellt.

Das Übertragen von Informationen a innerhalb eines Regelkreises beginnt zu einem beliebigen Ereignis im Zeitpunkt t_a. Von dort ausgehend werden alle Begriffe defi-

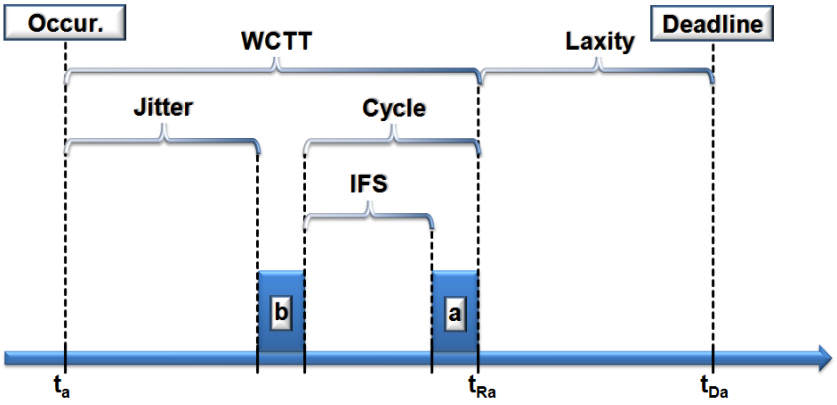


Abbildung 2.4: Definition Zeitbegriffe

niert, und sie haben die Eigenschaft $t_x \geq t_a$. Zum Zeitpunkt t_{Ra} wurde das Paket erfolgreich empfangen. Der Abstand zu t_a ist die Übertragungszeit oder Transmission Time (TT). Ein wichtiger Indikator ist die maximale Übertragungszeit, Worst Case Transmission Time (WCTT). Diese gibt die maximale, unter allen Umständen eingehaltene TT an, die nachweislich nicht überschritten wird. Dies ist wichtig, um die korrekte Funktionalität der Regelkreise zu sichern. In rein ereignisgesteuerten Kommunikationssystemen wie CAN kann keine obere Schranke und damit keine WCTT angegeben werden. Auch in IEEE 802.3 Ethernet ist dies ohne Erweiterungen nicht möglich.

Der Zeitpunkt t_{Da} repräsentiert die Übertragungsgrenze, die Deadline der Information. Sie ist der spätmöglichste Zeitpunkt, bei dem die Nachricht am Zielknoten vorliegen muss. Die Deadline ergibt sich nicht aus dem Kommunikationsverhalten, sondern ist ein von der Funktion abhängiger, im Designprozess festgelegter Wert, wie zum Beispiel die Anforderungen der Regelkreise zur Einhaltung des Stabilitätskriteriums. Die Differenz zwischen dem tatsächlichen Eintreffen und der Deadline stellt eine Zeitreserve dar. Im englischen Sprachgebrauch wird diese Zeitreserve Laxity genannt. Dieser Begriff wird auch im Folgenden verwendet. Für ein gültiges System muss die Laxity für alle auftretenden Fälle $t_{Da} - t_{Ra} \geq 0$ sein. Die Laxity ist somit bei Betrachtung zeitkritischer Systeme neben der WCTT die wichtigste Kenngröße.

Zur Quantifizierung der Last wird die Zykluszeit in Verbindung mit der Datengröße verwendet. Die Zykluszeit, *Cycle*, ist der Abstand zwischen zwei Sendeanforderungen der gleichen Information. Bei ereignisbasierter Übertragung repräsentiert Cycle den geringsten Abstand zwischen zwei Sendeanforderungen.

Betrachtet man generell den Abstand zwischen zwei Paketen, so wird dieser in der Literatur *Inter Frame Space* (IFS) oder alternativ *Inter Frame Gap* (IFG) benannt. Dieser Zeitraum berechnet sich aus dem Abstand zwischen Ende der Aussendung eines Paketes bis zum Beginn der Aussendung des nächsten. Meist aus Gründen der Synchronisierung des Medienzugriffs wird ein minimaler IFG im Protokoll festgelegt.

Die Definition des *Jitter* wird nicht eindeutig und klar in allen Bereichen der Kommunikationssysteme durchgehalten. In dieser Arbeit wird der Begriff Jitter für die

Verteilung der Dauer der Übertragung verwendet. Der Jitter beinhaltet also erwartete und unerwartete Verzögerungen bei Durchleitung eines Paketes.

Die Begriffe Paket, Rahmen, Frame und Nachrichten werden in dieser Arbeit gleichwertig eingesetzt und stehen für einen über das Kommunikationsnetz übertragenen Datenblock inklusive Nachrichtenkopf.

2.4.2 Anforderungen

Wichtig für den Architekturdesignprozess ist das Wissen über die Anforderungen an die zu findende Lösung. Zu diesem Zwecke wurden aktuelle Bordnetze aus Kraftfahrzeugen untersucht und ausgewertet. Die Daten stammen direkt vom Hersteller und repräsentieren eine so genannte 120%-Konfiguration, also die Summe aller möglichen ECUs, welche aber nicht alle gleichzeitig verbaut werden, da sie teils die gleiche Funktion für unterschiedliche Sonderausstattungen bereitstellen. Dabei wurden die unterschiedlichen Signale auf allen Busnetzen, soweit spezifiziert, zusammengenommen. Die Ergebnisse dieser Auswertung sind in den nachfolgenden zwei Diagrammen visualisiert.

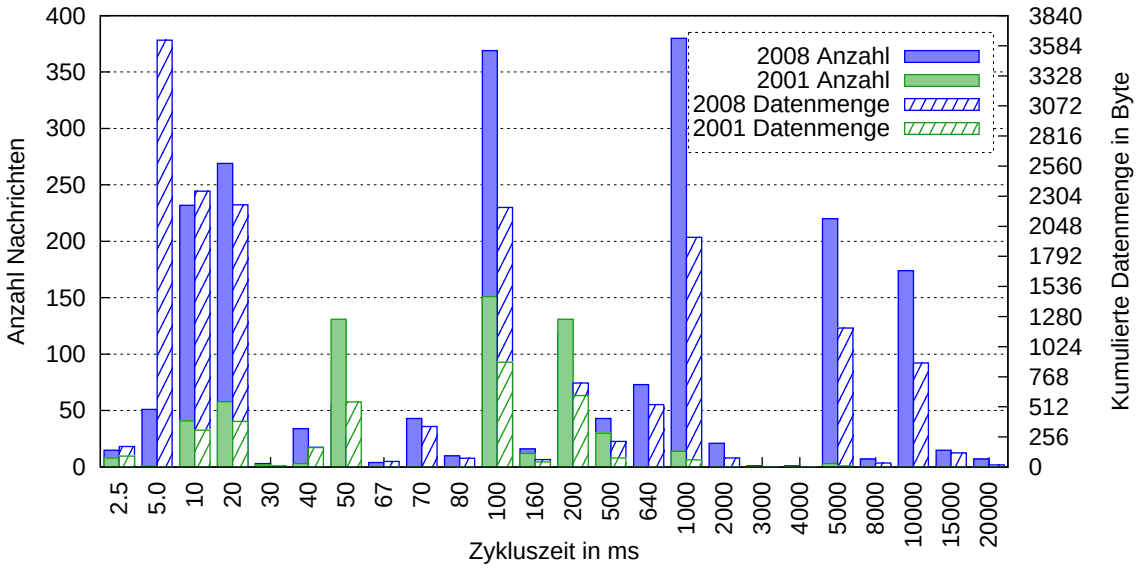


Abbildung 2.5: Verteilung Zykluszeiten

Abbildung 2.5 stellt die Anzahl der Nachrichten und die kumulierte Datenmenge für bestimmte Zykluszeiten dar. Dabei wurden einerseits Daten eines im Jahr 2008 auf den Markt gebrachten Fahrzeugs untersucht (blaue Balken) sowie eines der gleichen Klasse aus dem Jahr 2001 (grüne Balken). Die Auswertung ergab eine Vervielfachung des Kommunikationsaufwandes über die Jahre, die mit einem steigenden Funktionsumfang und der Komplexität der neuen Funktionalität einhergeht. Außerdem tritt durch die leistungsfähigeren Kommunikationslösungen eine Linksverschiebung des Mittelwertes zu geringeren Zykluszeiten und damit höheren Datenraten ein.

Betrachtet man die Verteilung der Zykluszeiten unter dem Wissen des physikalischen Verhaltens eines Fahrzeuges [MW04], repräsentieren Nachrichten im Bereich von $2{,}5-5$ ms tendenziell die Vertikaldynamik, von $5-10$ ms die Querdynamik und von $10-40$ ms die Längsdynamik. Nachrichten mit höheren Zykluszeiten betreffen überwiegend die Komfortfunktionen beziehungsweise dienen zu Diagnosezwecken. In Zukunft zu erwarten ist somit neben einer allgemeinen Steigerung der Nachrichtenmenge eine höhere Ausprägung im Bereich von $1-2{,}5$ ms, in dem Regelkreise der Vertikaldynamik von lokalen auf verteilte Ansätze erweitert werden, wobei Zykluszeiten von < 1 ms physikalisch nicht sinnvoll sind und aus diesem Grund nicht betrachtet werden. Auch wird eine überproportionale Erhöhung der Rahmen mit Zyklen von $30-40$ ms erwartet, die Daten von videobasierten Systemen übertragen, welche mit einer Frequenz von 25 Hz und 30 Hz Einzelbilder generieren.

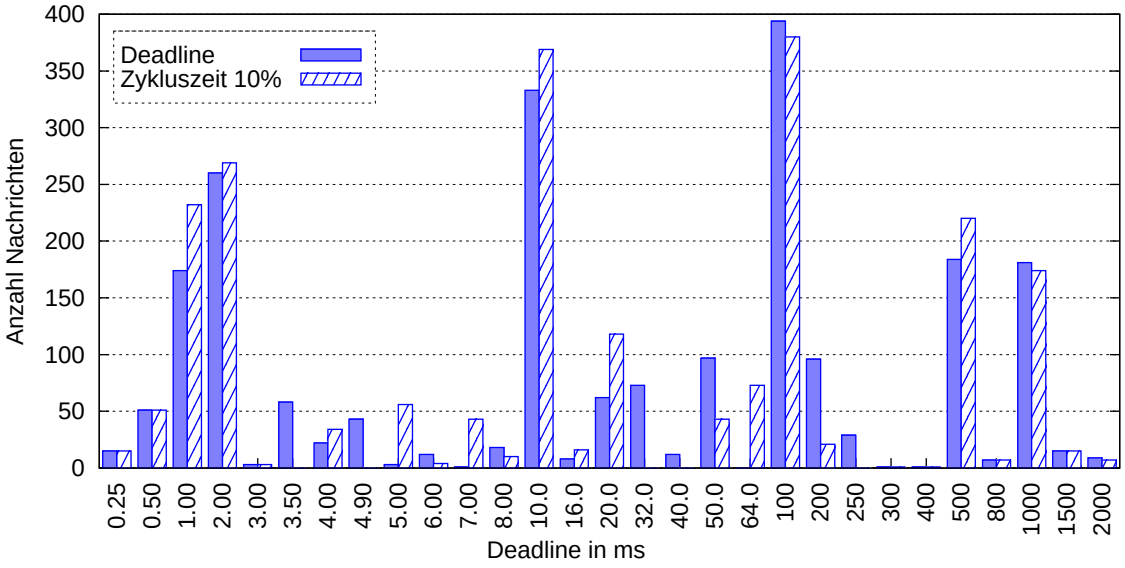


Abbildung 2.6: Verteilung Übermittlungsgrenzen

Diagramm 2.6 zeigt eine Verteilung der Übermittlungsgrenzen, auch „Deadline“ genannt, des blauen Fahrzeuges. Auch diese Daten stammen aus der Spezifikation eines Fahrzeugherstellers zur Überprüfung des Kommunikationsverhaltens. Zu jeder Deadline-Klasse sind zwei Balken angegeben, einerseits die ausgefüllten Balken, welche die realen Werte der Spezifikation wiedergeben, und als zweite Information die Anzahl der Nachrichten bei Verwendung einer generellen Deadline von 10% des Zyklus (nicht gefüllte Balken).

Aus der Abbildung ist zu erkennen, dass die generelle Annahme einer Übermittlungsverzögerung von 10% der Zykluszeit einer Nachricht den spezifizierten Werten sehr nahe kommt. Diese Information kann damit in Verbindung mit den vorherigen Aussagen zur Extrapolation der erwarteten Anforderungen an zukünftige Bordnetze in Fahrzeugen verwendet werden.

2.5 Systemdesign eines zukünftigen Bordnetzes

Im Rahmen des Projektes IT_Motive 2020 wurden anhand einer Anforderungsanalyse zukünftige Trends aufgezeigt, die Einfluss auf das Design eines Bordnetzes im Fahrzeug haben werden. Basierend auf diesen Informationen wurde ein Referenzdesign für ein zukünftiges Bordnetz entwickelt [Cha10]. Diese Architektur ist innerhalb des Projektes entstanden und dient in der vorliegenden Arbeit nur als Grundlage der weiteren Ansätze.

Das Design des zukünftigen Bordnetzes basiert auf drei Säulen: Homogenisierung, Zentralisierung und Virtualisierung. Mit der Homogenisierung wird die Komplexität des Gesamtsystems verringert. Es wird vorgeschlagen, weniger unterschiedliche Lösungen für Kommunikationsnetze und ECUs zu verwenden, damit die gleichen Teile über mehrere Produkte hinweg die Kosten verringern und die Qualität erhöhen.

Bei der Zentralisierung wird die Verknüpfung zwischen Sensor und Aktor, zwischen Verarbeitung und Verarbeitungshardware aufgebrochen. Aktuell verwendet jede Funktion meist eine eigene ECU in der Nähe der Sensorik und Aktorik. Diese Funktionen sollen nun in Zukunft auf einige vom Hersteller bereitgestellte, zentrale ECUs verschoben werden. Es bleibt nur noch so viel Intelligenz an den Sensoren bzw. Aktoren übrig, um die dort generierten Daten vorläufig zu verarbeiten und über ein Kommunikationsnetz zu übertragen. Die Kundenfunktionalität liegt in den zentralen ECUs unabhängig von der dort verwendeten Hardware.

Für diese Abstraktion von der verwendeten Hardware, sei es innerhalb der ECU oder bei der Kommunikation, muss Virtualisierung eingesetzt werden. Gerade die bei der Virtualisierung nach oben hin stabileren Schnittstellen erleichtern die Weiterverwendung der bereits entwickelten Lösungen für andere Modelle. Auch ermöglicht es die Virtualisierung, mehrere unterschiedliche Funktionen einer Hardwarekomponente zuzuweisen, aber gleichzeitig angeforderte Ressourcen zu garantieren.

Der Einsatz von Homogenisierung, Zentralisierung und Virtualisierung eröffnet nun den Einsatz von Rekonfigurierung. Durch die örtliche Trennung von Datengenerierung, -konsumierung und die Verarbeitung der Daten ist die Wahl des Ausführungsortes nicht mehr streng festgelegt. Fasst man diesen Gedanken weiter, ermöglicht dies, bei einem Ausfall von Ressourcen die Verarbeitung auf eine andere ECU zu verschieben, wenn die notwendigen Kommunikationsressourcen auch dort vorhanden sind. Mit Einsatz dieser Methode kann eine höhere Robustheit des Gesamtsystems erreicht werden.

2.6 Zusammenfassung

In Bordnetzen in Fahrzeugen sowie in anderen Industriebereichen gibt es viele Lösungen für Kommunikationsnetze. Meist wurden sie speziell auf eine bestimmte Funktionalität hin entwickelt und werden genau für diese eingesetzt. Es gibt kaum einen Austausch der Lösungen zwischen den Industriebereichen, weil die überwiegende

Anzahl der Lösungen schrittweise aus unterschiedlichen Ausgangsbedingungen entstanden sind. Dies führt oft zur Koexistenz mehrere unterschiedlicher Kommunikationssysteme in einem Produkt, wo über Gateways oder durch parallele Verbindungen Informationen zwischen den unterschiedlichen Systemen ausgetauscht werden.

Als neuer Trend in allen diesen hier erwähnten Industriebereichen ist der Einsatz von Ethernet zur Ablösung der klassischen Feldbusse auszumachen. Dieser Wandel beruht vor allem auf dem stark steigenden Datenratenbedarf neuer Systeme, wo klassische Lösungen im Bereich < 10 Mbit/s langsam an ihre Grenzen stoßen. Untersuchungen an Netzen aktueller Fahrzeuge bestätigen den Trend zum stetig steigenden Kommunikationsaufwand. Anstatt nun von Grund auf eine neue Kommunikationstechnik zu entwickeln, wird die im Umfeld des Büros und in lokalen Netzen vorherrschende Lösung Ethernet den neuen Anforderungen angepasst.

Der Vorteil besteht in der Möglichkeit der Wiederverwendung vorhandener Lösungen in Form von Hardware wie Kabel, Stecker, Dosen, Messgeräte bis hin zu Siliziumprodukten wie Media-Access-Control (MAC)- und Transceiver-Bausteinen. Auch Intellectual Property, also vorhandenes Wissen und Design-Bausteine, können teilweise ohne Modifikation übernommen werden, so zum Beispiel entsprechende Ethernet-MAC-Blöcke für FPGA oder ASIC. In diesem Rahmen wird oft von „components-off-the-shelf“ (COTS) gesprochen. Dadurch wird eine starke Auswirkung auf die Entwicklungs- und Produktionspreise erwartet.

Ziel einer neuen Architektur für Bordnetze für verteilte heterogene Subsysteme ist es, die Komplexität sowie die Kosten zu verringern bei gleichzeitiger Beibehaltung der Robustheit. Der Ansatz besteht darin, durch Zentralisierung und Homogenisierung unter Zuhilfenahme von Virtualisierung dieses Ziel zu erreichen.

Infolge der gesetzten Randbedingungen gibt es keine fertige Lösung, die direkt im automativen Bereich eingesetzt werden kann. Aus diesem Grund werden in den weiteren Kapiteln notwendige Erweiterungen entwickelt sowie Einschränkungen dargestellt, die bei der Verwendung eines auf Ethernet basierenden Kommunikationsnetzes zu beachten sind.

3 Zuverlässiger Datentransport

Durch die Homogenisierung der Kommunikation sowie die Zentralisierung der Datenverarbeitung entstehen neue Herausforderungen, die klassische Büronetze auf Ethernetbasis nicht abdecken können.

Sie besteht auf der einen Seite in der deterministischen Übertragung von Daten. Ethernet, als klassisch paketvermittelndes System, ermöglicht direkt keine Angabe über die maximale Verzögerung bei der Übertragung über mehrere Knoten. Gerade bei der Planung der Regelkreise verteilter Systeme ist diese unvorhersagbare, dynamische Komponente eine große Herausforderung. Für Realzeitnachweise muss eine obere Schranke eingerechnet und durch das System die Einhaltung dieser Schranke sichergestellt werden. Im Gegensatz dazu kann ein frühzeitiges Übermitteln der Daten mit geringen Mitteln durch Einsatz von Puffern ausgeglichen werden, sobald die maximale Puffergröße berechenbar ist.

Ein weiterer Punkt ist die Verfügbarkeit des Gesamtsystems und der einzelnen Funktionen. Zentralisiert man Funktionalität, verändert sich im Gegenzug auch die Fehlertoleranz. Es sind zwar damit in der Verarbeitungskette weniger Komponenten beteiligt, was aus Sicht einer einzelnen Funktion die Verfügbarkeit erhöhen müsste. Jedoch sind die entsprechenden Komponenten komplexer, was wiederum die Ausfallrate erhöht. Aus der Perspektive des Gesamtsystems führt der Ausfall einer Komponente zum Versagen vieler Einzelfunktionen und verringert so die Gesamtverfügbarkeit enorm. Insofern müsste, um diesen Effekt auszugleichen, die verwendete Hardware trotz gestiegener Komplexität weniger oft ausfallen, was nur durch teuren Einsatz von weiteren Bausteinen möglich ist.

Wenn neue Knoten in das Netz kommen oder Funktionalität von einem auf einen anderen Knoten verschoben wird, ändern sich die Routen sowie die Belegung der Kanten. Zukünftige Kommunikationsarchitekturen müssen ermöglichen, dass während des Lebenszyklus des Produktes, also nach Fertigstellung im Werk, neue Funktionen, Knoten und damit Routen dem System hinzugefügt werden können. Dabei dürfen sie die vorhandene Funktionalität nicht beeinträchtigen.

Dieses Kapitel beschäftigt sich mit Erweiterungen des Netzes um Mechanismen, die trotz Beibehaltung des IEEE 802.3 -Standards für Ethernet das Einhalten der Anforderungen für den Einsatz bei sicherheitskritischer Realzeitkommunikation ermöglicht. Speziell für einen späteren Zertifizierungsvorgang wurde auf analytische Greifbarkeit der Kommunikationsprozesse Wert gelegt. Weiter wurde ein Sicherheits- und Konfigurationsmanagement entwickelt, welches das Bordnetz dynamisch konfigurieren und bei Ausfall von Kommunikationsressourcen diesen Fehler durch Ändern der Pfade auflösen kann.

3.1 Deterministische Übertragung

3.1.1 Definition und Ziele

Ziel ist es, eine obere Schranke der benötigten Zeit zur Übertragung eines bestimmten Paketes analytisch berechenbar zu machen. Laut Definition aus Kapitel 2.4.1 handelt es sich dabei um die Worst Case Transmission Time (WCTT). Dieses Ziel kann nicht ohne Modifikation der Hardware erreicht werden.

Eine feste obere Schranke der Verzögerung wird benötigt, um eine Absicherung der Regelkreise durch Realzeitnachweise zu ermöglichen. Dies ist eine unverzichtbare Grundlage, um eine Zertifizierung des Produktes durch öffentliche Stellen wie TÜV, LBA, FAA, DOD und weitere zu ermöglichen.

3.1.2 Medienzugriff

Grundsätzlich können die Medienzugriffsprotokolle in zwei große Klassen eingeteilt werden: ereignisgesteuerte und zeitgesteuerte. Ausgehend von einem Sendewunsch, wird ein Datenpaket erstellt und in einer Warteschlange, einem Puffer, gespeichert. Beim ereignisgesteuerten Medienzugriff wird das Paket nun zum gewünschten Sendezeitpunkt, meist umgehend nach Eintreffen in der Warteschlange, gesendet, sobald das Medium verfügbar ist. Das Medium kann zum Beispiel durch eine laufende Übertragung oder anderweitige Blockierung, wie das obligatorische Inter Frame Gap (IFG), gesperrt sein. Die Belegung des Kanals erfolgt also nach Eintreten eines definierten Ereignisses, hier des Sendewunsches. Die Arbeiten um Loeser und Haertig [LH04, LHD04] beschäftigen sich mit dem Einsatz eines ereignisgesteuerten Ethernet für realzeittaugliche Kommunikation und zeigen einige interessante Ansätze, auch wenn sie sich überwiegend auf den Bereich Automatisierungstechnik konzentrieren.

Beim zeitgesteuerten Medienzugriff erfolgt die Aussendung des im Puffer befindlichen Datenpaketes zu einem vorher festgesetzten, globalen Zeitpunkt. Das Ereignis „Sendewunsch“ und das tatsächliche Übertragen der Daten sind zeitlich entkoppelt. Die Arbeiten von Albert [Alb04] vergleichen die unterschiedlichen Medienzugriffe im Hinblick auf verteilte Systeme, die Arbeiten von Kopetz [Kop91, Kop93] sind grundlegend in diesem Gebiet.

Beim Medienzugriff werden in dieser Arbeit grundsätzlich nur kollisionsfreie Methoden betrachtet. Das kaum vorhersagbare Verhalten bei Kollisionen erschwert die analytische Betrachtung und führt zu großen Überreservierungen und einem damit verbundenen Effizienzverlust.

Diagramm 3.1 zeigt Ergebnisse einer Simulation des Referenznetzes (siehe Kapitel A.2) mit unterschiedlichem Medienzugriff. Dabei werden Pakete zu beliebigen gleichverteilten Zeitpunkten innerhalb ihrer Zykluszeit versendet. So wird die durchschnittliche Datenrate über einen längeren Zeitraum gemittelt konstant gehalten, die Sendezeitpunkte jedoch gleichverteilt. Die Ende-zu-Ende-Verzögerung aller Pakete wurden

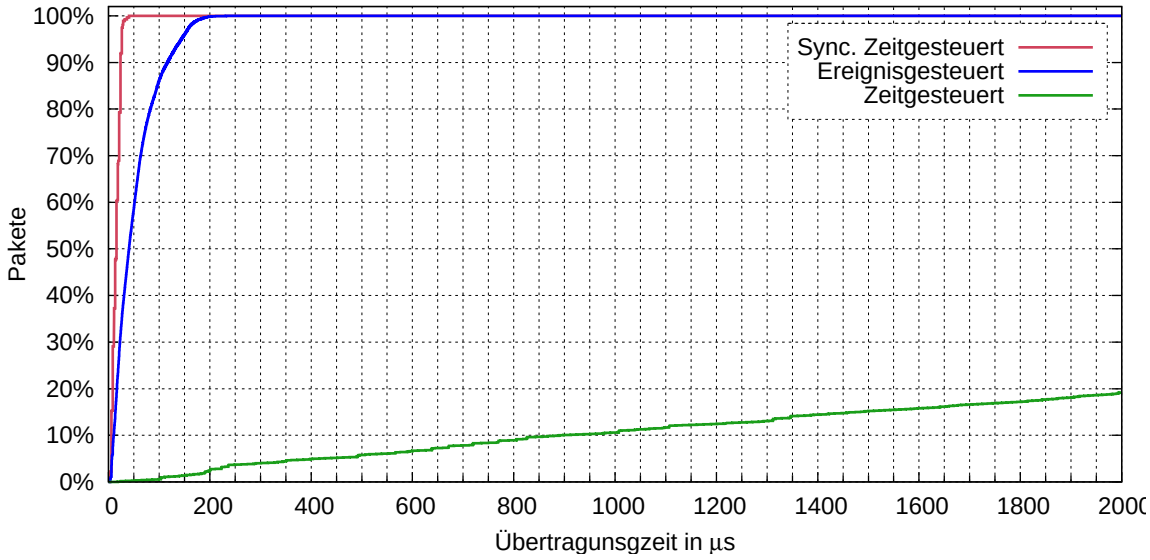


Abbildung 3.1: Übertragungszeit beim unterschiedlichem Medienzugriff

ausgewertet und in einer Verteilungsfunktion dargestellt. Beim zeitgesteuerten Zugriff, der grünen Linie, ergibt sich erwartungsgemäß eine Gerade. Bei gleichverteiltem Sendewunsch und festem Sendezeitpunkt ergibt sich in dem betrachteten Zeitraum eine näherungsweise gleichverteilte Verzögerung. Die Zeitschlitze zum Senden der Daten wurden dabei global festgelegt, das heißt, zu einem bestimmten Zeitpunkt wurde das Netz exklusiv für das Paket reserviert. Jedem Paket steht in seiner Zykluszeit ein Zeitschlitz zur Verfügung. Betrachtet man die Verteilung der Zykluszeiten aus Abbildung 2.5 sowie die Tatsache, das ein Paket im Schnitt die halbe Zyklenlänge warten muss, ist die durchschnittliche Verzögerung damit sehr hoch. Auch der Jitter der Übermittlung ist entsprechend, wenn man diesen vom Sendewunsch und nicht vom Start der Übertragung aus berechnet.

Ein völlig anderes Bild ergibt sich jedoch, wenn man den in abgeschlossenen Bordnetzen häufig eingesetzten synchronisierten zeitgesteuerten Medienzugriff verwendet. Beim synchronisierten Medienzugriff wird die Verarbeitung der Daten und damit die Generierung neuer Pakete mit den Zeitschlitzen des Kommunikationsnetzes abgeglichen. Die Verzögerung der Pakete ist damit im Idealfall nur noch von der Pfadlänge durch die Verarbeitungsverzögerung in den Knoten abhängig. Dies ist deutlich an dem treppenstufenartigen Verlauf der roten Kurve erkennbar. Die maximale Verzögerung, also der Punkt, an dem 100% der Pakete übertragen wurden, ist < 50 µs Ende-zu-Ende-Verzögerung. Eine vollständige Synchronisierung zwischen Verarbeitung und Kommunikation ist jedoch sehr aufwendig. Bereits kleine Modifikationen der Berechnungen kann die Verzögerung auf einer Electronic Control Unit (ECU) erhöhen und damit den Sendezeitpunkt des Ergebnisses verschieben. Alle Algorithmen, die wiederrum dieses Ergebnis als Eingangswert haben, verzögern sich somit ebenfalls. Dies führt unter Umständen zu einer Reorganisation aller Zeitschlitze. Gerade dieses Verteilen der Zeitschlitze ist auf Grund der vielen verschiedenen Herstellern

der einzelnen ECUs nicht nur organisatorisch aufwändig, es ist auch nicht optimal lösbar. Das Problem der optimalen Zeitschlitzverteilung ist NP-äquivalent [GJ+79].

Die Kurve für den ereignisgesteuerten Medienzugriff liegt in der Mitte der vorherigen beiden zeitgesteuerten Zugriffe. Die Pakete werden schnellstmöglich nach Sendewunsch übertragen, und durch die nicht exklusive Reservierung des gesamten Netzes zu diesem Zeitpunkt kann es an Sternknoten zu Verzögerungen durch andere Pakete kommen. Da diese Quasi-Kollision in den Warteschlangen nur mit einer gewissen Wahrscheinlichkeit auftritt, sind keine Stufen im blauen Graph ersichtlich. Trotz dieser Kollisionen liegt die in der Simulation maximal auftretende Ende-zu-Ende-Verzögerung mit ≈ 330 µs weiter unter der des asynchronen Zugriffs.

Da in zukünftigen Bordnetzen nicht davon ausgegangen werden kann, dass der Funktionsumfang während des gesamten Lebenszyklus gleich bleibt, muss mit sich ändernden Kommunikationszugriffen gerechnet werden. Dies betrifft einmal die übertragene Datenmenge, andererseits die involvierten Knoten. Auch der hohe Anteil an zugelieferten Komponenten, welche nicht direkt beim Original Equipment Manufacturer (OEM) gefertigt werden, erschwert die Vorhersagbarkeit der Kommunikationsbedürfnisse. Aus diesem Grund führt ein sehr starres System, wie es durch synchronisierte Zeitschlitze festgelegt ist, zu hohen Aufwänden während der Entwicklung und bei einer Erweiterung des Systems.

Als Lösung wird deswegen die Verwendung eines ereignisgesteuerten Medienzugriffes vorgeschlagen. Die notwendigen Anpassungen, um mit dieser Art des Zugriffs die festgelegten Ziele (Kapitel 3.1.1) zu erreichen, sind in den nächsten Kapiteln erläutert. Auch wird näher auf die Leistungsfähigkeit dieser Lösung, speziell im Hinblick auf die Übertragung von Videodaten, eingegangen.

3.1.3 Erweiterung für echtzeittaugliches Ethernet

Im Standard IEEE 802.3 für Ethernet, 1976 von Robert Metcalfe erfunden [MB76], sind mehrere mögliche Medienzugriffsverfahren spezifiziert. Für den Halbduplex-Betrieb, also Senden und Empfangen über dieselbe Leitung, wurde „Carrier Sense Multiple Access with Collision Detection (CSMA/CD)“ spezifiziert. Wie der Name bereits sagt, wird die Leitung überwacht und bei freier Leitung der Frame gesendet. Falls zwei Geräte gleichzeitig senden, wird dies als Kollision erkannt. Um dies über weite Entfernungen im Netz zu ermöglichen, ist ein minimale Framelänge von 64 Byte beziehungsweise 520 Byte bei Gigabit festgelegt.

Diese Form des Medienzugriffs kommt in heutigen Netzen kaum noch zum Einsatz. Sie wurde abgelöst durch den Fullduplex-Betrieb in Kombination mit der Verwendung eines Sternknoten, dem so genannten „Switch“, zum Verbinden der Geräte. Damit existiert im eigentlichen Sinn kein gemeinsames Medium mehr. Jeder Rechner beziehungsweise jeder Anschluss am Sternknoten kann exklusiv auf die verbundene Leitung zugreifen. Kollisionen werden aufgelöst, in dem Empfangs- und Sendewarteschlangen Frames bei blockierten Ressourcen zwischenspeichern. Dieser Einsatz von Warteschlangen ermöglicht nun eine Prädizierung des Verhalten bei Übertragung von

Frames. Dies dient als Grundlage für die in dieser Arbeit untersuchte Übertragungszeitanalyse.

Erfahrungen mit Triple-Play, also der Übertragung von Daten, Video und Audio über eine gemeinsame Leitung, zeigen, dass ohne Modifikationen keine garantierte Qualität sichergestellt werden kann. Ziel ist es, den Standard IEEE 802.3 für Ethernet nicht anzutasten, um die Kompatibilität mit vorhandenen Produkten zu wahren. Aus diesem Grund wurden nur Veränderungen in den Sternknoten durchgeführt, da diese auf Grund der besonderen Einsatzorte in den überwiegenden Fällen speziell entwickelt und hergestellt werden müssen, so dass Anpassungen leichter durchgeführt werden können. Das Blockschaltbild des angepassten, echtzeittauglichen Sternknotens ist in Abbildung 3.2 dargestellt.

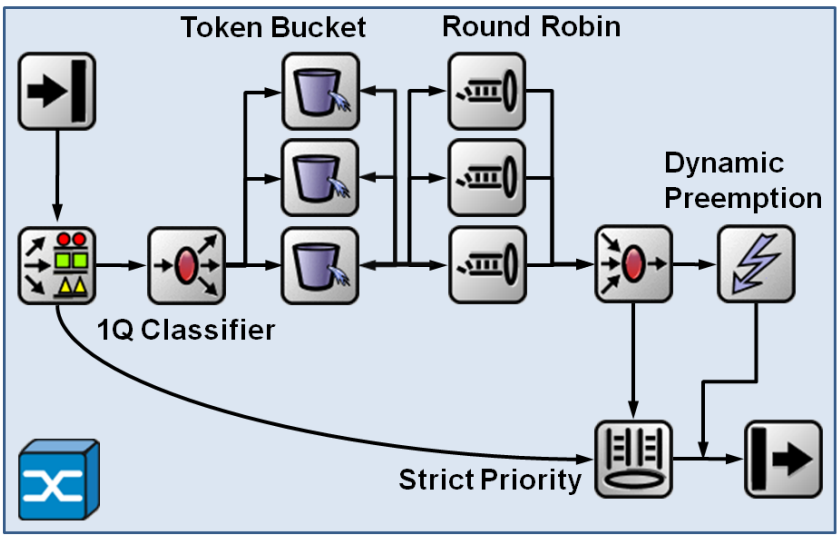


Abbildung 3.2: Blockschaltbild Ethernet Switch

Die Verarbeitung der eingehenden Rahmen beginnt mit einer gemeinsamen Warteschlange vor dem Prozessor. Dieser kann $1 \ldots n$ Rahmen parallel verarbeiten. Im Prozessor wird mit Hilfe der Zieladresse und der Routingtabelle ein Ausgangsport festgelegt. Auf der Basis des IEEE 802.1q-Tag für die Übertragungsqualität wird die Zielwarteschlange des gefundenen Ausgangsports bestimmt. An dieser Stelle wird der zeitkritische Verkehr, also die Ströme, zu denen eine WCTT berechnet werden muss, vom übrigen separiert. Der nicht kritische Verkehr, im Falles des Referenznetzes aus Kapitel A.2 alle Videoverbindungen, werden in *Strict-Priority*-Warteschlangen mit einer Priorität $< max$ eingeordnet. Die Festlegung der Anzahl der Warteschlangen n, wobei $n \geq 2$, ist nicht festgelegt und vom konkreten Einsatz des Bordnetzes abhängig. Die Warteschlange mit der höchsten Priorität ist für den zeitkritischen Verkehr reserviert. An diesem Block ist ein *Dynamic-Preemption*-Mechanismus angeschlossen, der Frames niedriger Priorität unterbrechen kann, um den Kanal für andere Rahmen freizumachen (Kapitel 3.1.5).

Die zeitkritischen Ströme durchlaufen jeweils einen *Token-Bucket*-Filter, der einerseits die Einhaltung der gesetzten Kommunikationsgrenzen aus Kapitel 3.1.4 überprüft,

andererseits zur Detektion von Ausfällen im Sicherheitsmanagement (Kapitel 3.3) zuständig ist. Anschließend werden sie, nach Strömen getrennt, in eine *Round-Robin*-Warteschlange einsortiert. Nach welchem Kriterium diese Trennung am besten erfolgt, wird in Kapitel 3.1.6 erläutert. Für die analytische Betrachtung wird davon ausgegangen, dass pro Verbindung über diesen Switch eine eigene Warteschlange mit Tiefe 1 verwendet wird. Die zeitkritischen Ströme werden dann wie erwähnt in die Ausgangswarteschlange höchster Priorität einsortiert und lösen den *Dynamic-Preemption*-Mechanismus aus.

Diese Modifikationen ermöglichen eine Berechnung der WCTT, welche im nachfolgendem Kapitel erläutert wird.

3.1.4 Übertragungszeit-Analyse

Der Einsatz von „Network Calculus“, eine zum Realzeitnachweis im Softwareengeneering äquivalente Methode zum Nachweis der einwandfreien Funktion unter allen Bedingungen, wurde unter anderem in den Arbeiten von Le Boudec [LB98] sowie von Cruz [Cru91a, Cru91b] auf Kommunikationsnetze angewendet. Die Berechnung der WCTT in dem hier vorgestellten Ansatz erfolgt mit Hilfe der Warteschlangentheorie [Kle75, GH85]. Betrachtet werden die Verzögerungen durch eigene Rahmen auf den Leitungen sowie durch andere Frames, welche sich die gleiche Leitung auf Teilstücken des Übertragungsweges teilen. Da auf den Leitungen selbst durch den Medienzugriff keine Kollisionen entstehen können, sind nur die Sternknoten mit den Warteschlangen relevant für die Berechnung. Aus diesem Grund wird für jede Übertragung die Verzögerung T_T an den durchlaufenen Sternknoten berechnet.

Auch in den sendenden ECUs sind Warteschlangen vorhanden, falls Daten schneller generiert werden, als sie übertragen werden können. Die ECU muss deswegen auch in der Berechnung der WCTT betrachtet werden. In diesem Modell wird eine ECU selbst als Sternknoten mit n eingehenden Kanten mit ∞ Geschwindigkeit und einer ausgehenden Kante modelliert. Die Senke wird nicht eingerechnet, da davon ausgegangen werden kann, dass die Bandbreite des Speicherbusses zum Transferieren der Daten in den Hauptspeicher immer schneller ist als die der Kommunikationsschnittstelle.

$$T_T = T_{s0} + T_L + T_{calc} + T_{W2_{RR}} + T_{W1_{RR}} + T_{W1_{SP}} \tag{3.1}$$

Die Berechnung der Verzögerung in einem Knoten T_T ist in Gleichung (3.1) aufgestellt. Grundlage der Berechnung ist der in Abbildung 3.2 vorgestellte Aufbau der Sternknoten. Verwendet wird eine *Store-and-Forward*-Architektur innerhalb des Sternknotens, der ein vollständiges Empfangen des Rahmens vor weiterer Bearbeitung vorsieht. Dies führt in jedem Knoten damit zu einer Verzögerung $T_{s0} = \frac{\text{Message Size}}{\text{Rate}}$. Das *Stote-and-Forward*-Verfahren ist das am einfachsten realisierbare System. Als Alternative kann auch ein *Cut-Through*-Mechanismus in die Sternknoten implementiert werden, der bereits nach Empfang des Nachrichtenkopfes gleichzeitig mit dem Empfang

des Nachrichtenkörpers die Algorithmen zur Bestimmung der Ausgangswarteschlange durchführt. In diesem Fall verringert sich $T_{s0} = \frac{\text{Header Size}}{\text{Rate}} \ll \frac{\text{Message Size}}{\text{Rate}}$. Nachteil ist, dass eine Überprüfung der fehlerfreien Übermittlung mit Hilfe der Prüfsumme nicht durchgeführt werden kann, da die Daten unter Umständen bereits wieder ausgesandt wurden.

Die bei der Übertragung auftretenden Leitungsverzögerungen T_L werden nicht betrachtet. Eine bei Kupferkabeln mit einem Verkürzungsfaktor $c = 0{,}6 - 0{,}8$ und einer in Bordnetzen üblichen Länge von 20 m auftretende Verzögerung von < 1 ns ist nicht relevant für die Berechnung der Gesamtverzögerung und wird mit $T_L \approx 0$ gesetzt. Wenn längere Leitungswege erwartet werden, kann stattdessen mit einer konstanten oberen Schranke $T_{L_{max}}$ oder, mit etwas höherem Aufwand und bei bekannter Netzdimension, den konkreten Leitungslängen gerechnet werden.

Der nächste Punkt, an dem Verzögerungen entstehen können, ist das Verarbeiten der Pakete nach Eintreffen im Sternknoten. Dort müssen anhand der Media-Access-Control (MAC)-Adresse und dem IEEE 802.1q-Tag basierend auf den Einträgen in der Routingtabelle der zugehörige Ausgangsport und die entsprechende Warteschlange bestimmt werden. Problematisch wird es nun, wenn weitere Pakete während dieser Blockierung am Eingang ankommen und dort gepuffert werden müssen.

Da diese Verarbeitung der Daten, vor allem wenn Vorgänge wie Kopieren im Speicher erfolgen, von der Größe des entsprechenden Paketes abhängig sein kann, werden zwei Variablen $T_{proc\,\max} \geq T_{proc\,\min}$ eingeführt, die jeweils der benötigten Verarbeitungszeit für den kleinsten (64 Byte nach Standard) und den größten (Maximum Transmission Unit (MTU)) auftretenden Frame entsprechen. Ist die Verarbeitungszeit unabhängig von der Länge, gilt $T_{proc\,\max} = T_{proc\,\min}$.

Bei der Berechnung der WCTT ist immer der schlimmste Fall von Interesse. Dies ist im Falle der Verarbeitung ein Frame der Länge MTU gefolgt von x minimalsten Rahmen. Dafür muss gegeben sein, dass der längenabhängige Teil der Verarbeitungszeit kleiner ist als die zusätzliche Übertragungszeit $\left(T_{proc} - T_{proc\,\min}\right) \leq \left(T_A - T_{A\,\min}\right)$. Damit kann unter Verwendung von Formel (3.2) die notwendige Menge an parallel zu verarbeitenden Rahmen bestimmt werden, um eine blockierungsfreie Funktionalität zu ermöglichen. Dabei ist $T_{A\,\min}$ der minimale Abstand zwischen dem Empfang zweier Rahmen, über alle eingehenden Verbindungen gesehen. In diesem Fall ist die in Gleichung (3.1) einzusetzende Verzögerung T_{calc} nach Formel (3.3) $T_{\text{calc}} = T_{\text{proc}\,\max}$. Eine mögliche Realisierung dieses Lastenausgleichs über mehrere Prozessoren sind in den Arbeiten von Dittmann et al. zu finden [DH02].

$$N_{proc\,\max} \geq \left\lceil \frac{T_{proc\,\max}}{T_{A\,\min}} \right\rceil \tag{3.2}$$

$$T_{\text{calc}} = \begin{cases} T_{\text{proc}\,\max} + T_{\text{proc}\,\min} - \left(\left(N_{\text{proc}} - 1\right) \cdot T_{A\,\min}\right) & \text{für } N_{\text{proc}} < N_{\text{proc}\,\max},\ N_{\text{proc}} > 1 \\ T_{\text{proc}\,\max} & \text{für } N_{proc} \geq N_{\text{proc}\,\max} \\ T_{\text{proc}\,\max} + T_{\text{proc}\,\min} - T_{A\,\min} & \text{für } N_{\text{proc}} = 1 \end{cases} \tag{3.3}$$

Für den Fall, dass weniger Rahmen parallel verarbeitet werden können, tritt eine höhere Verzögerung durch Pufferung in der Eingangswarteschlange auf, die in den Formeln (3.3) bestimmt wird. Dabei wird der Prozessor durch das gerade empfangene Paket blockiert ($T_{\text{proc max}}$), während weitere Rahmen so schnell wie möglich ($T_{A\,\text{min}}$) hintereinander eintreffen. Bei mehr als zwei Prozessoren verringert sich die maximal auftretende Wartezeit um ein Vielfaches von $T_{A\,\text{min}}$, die auf anderen Prozessoren bearbeitet werden.

Für $T_{proc} > T_A$ wird ein nicht stabiler Zustand erreicht, da die Eingangswarteschlangen sich schneller auffüllen können, als sie abgearbeitet werden. Dies führt unter Umständen zu einer unendlich großen Verzögerung und damit $T_{\text{calc}} \rightarrow \infty$. Zur Vergrößerung von $T_{A\,\text{min}}$ kann die Verarbeitung der eingehenden Rahmen separiert pro Eingangsport auf dedizierten Prozessoren vorgenommen werden. Neben dieser Möglichkeiten wäre auch eine Implementierung in Hardware durchführbar, was zu einer Verringerung von T_{proc} führen kann.

Der nächste Schritt bei der Durchleitung eines Rahmens im Sternknoten ist das Ablegen in der Warteschlange und die Dauer des Aufenthaltes in derselben. Zu deren Herleitung wurde die allgemeine Warteschlangentheorie betrachtet. Der Sternknoten wird dabei als eine Ausgangswarteschlange pro Datenstrom i modelliert, welche im *Round-Robin*-Verfahren abgearbeitet wird. Es treten zwei getrennt zu beachtende Verzögerungen ein: einmal durch Rahmen in anderen Warteschlangen, also die Wartezeit T_{W1}, bis aus der eigenen Warteschlange gesendet wird. Und dann zusätzlich durch die vor einem in der Warteschlange liegenden Frames, die zu einer Verzögerung T_{W2} führen. Die entsprechenden Formeln für ein M/D/1-System sind in (3.4) dargestellt. Dabei ist T_{A_i} die minimale Ankunftszeit des Datenstroms i und T_{Si} die maximale Sendedauer eines Paketes des Datenstroms i. Zur Vereinfachung wird in diesem Fall mit konstanten maximalen Framelängen gerechnet. Die Verkehrslast ρ_i wird bei einem klassischen Ethernet Sternknoten somit aus der Sendedauer und der Ankunftsrate in Form des minimalen Abstandes berechnet. Daraus folgt, dass die Verzögerung eines Rahmens beim Durchlaufen des Sternknotens von der eigenen Verkehrslast ρ_0 und der Verkehrslast durch alle anderen diesem Ausgang zugeordneten N Datenströme $\sum_i^N \rho_i$ abhängt.

$$\begin{aligned} \rho_i &= \frac{T_{Si}}{T_{A_i}} \\ T_{W2} &= \frac{\rho_0 \cdot T_{S0}}{2 \cdot (1 - \rho_0)} \\ T_{W1} &= \frac{T_{Si} \cdot \left(N - \sum_i^N \rho_i\right)}{2 \cdot \left(1 - \sum_i^N \rho_i\right)} \end{aligned} \tag{3.4}$$

Der in dieser Arbeit vorgestellte Sternknoten wurde erweitert, um die Berechnung des WCTT zu ermöglichen. Die Details sind in Kapitel 3.1.3 vorgestellt. Durch diese

Anpassungen können die Formeln (3.4) vereinfacht werden zu den in (3.5) dargestellten Gleichungen. Betrachtet wird in der WCTT-Analyse nur der zeitkritische Verkehr. Angenommen wird, dass $T_{A_i} \geq T_{T_i}$, also der Zyklus der Nachrichten größer ist als die maximale Verzögerung und sich somit immer nur eine aktive Nachricht im Netz befindet. Durch die Trennung der einzelnen Datenströme im Sternknoten durch die *Round-Robin-(RR-)*Warteschlangen befindet sich somit nie ein weiterer Frame in der Warteschlange, und damit kann $T_{W2_{RR}} = 0$ gesetzt werden. Dies bedeutet auch, dass die maximale Warteschlangentiefe $= 1$ ist, was die Implementierung vereinfacht und den Ressourcenbedarf minimiert.

Die weiteren Verzögerungen treten also durch andere Pakete auf, welche auf den gleichen Ausgangsport ausgegeben werden. Der einfachere zu berechnende Teil ist das *Head-of-Line*-Blocking durch Rahmen niederer Priorität in der *Strict-Priority*-Queue. Angenommen, im schlimmsten Fall trifft der Sendewunsch der *Round-Robin*-Warteschlange gerade mit Beginn der Aussendung ein, ist eine Verzögerung von $T_{W1_{SP}}$ einzuberechnen. Diese setzt sich aus der Blockierung der Leitung durch einen maximal großen Frame und dem obligatorischen IFG zusammen.

Für die Berechnung der WCTT hauptsächlich ausschlaggebend ist die Verzögerung durch andere, sicherheitskritische Daten. Da es sich wie in Kapitel 3.1.3 dargestellt um eine Ausgangswarteschlange handelt, sind alle Datenströme relevant, die denselben ausgehenden Port besitzen. Berechnet wird die benötigte Zeit $T_{W1_{RR}}$ durch Aufsummieren der parallelen Datenströme und des jeweiligen IFG. Für jeden Datenstrom muss die dort auftretende MTU in die Gleichung eingesetzt werden.

Legt man nun fest, dass die Datenrate der eingehenden Kante gleich der Datenrate der ausgehenden ist, so müssen an den Sternknoten nur der Verkehr der anderen eingehenden Kanten betrachtet werden. Dies vereinfacht die Auswertung und Berechnung der WCTT. Trifft dies nicht zu, muss beim Verringern der Rate die Blockierung durch Rahmen auf der eigenen Kante in die bekannte Formel mit einberechnet werden.

$$\begin{aligned} T_{W2_{RR}} &= 0 \\ T_{W1_{RR}} &= \sum_i \left(\frac{\mathrm{MTU}_i}{\mathrm{Rate}} + T_{\mathrm{IFG}} \right) \\ T_{W1_{SP}} &= \frac{\mathrm{MTU}}{\mathrm{Rate}} + T_{\mathrm{IFG}} \end{aligned} \tag{3.5}$$

Wird die vorgestellte Formel (3.1) nun auf alle Verbindungen im Referenznetz aus Kapitel A.2 angewendet, ergibt sich die in Diagramm 3.3 dargestellte Verteilung. Dabei sind alle Verbindungen nach aufsteigender WCTT sortiert auf der y-Achse normiert auf 100% angetragen, die entsprechende WCTT auf der x-Achse. Aus dem Diagramm ist zu ersehen, dass $y\,\%$ der Verbindungen eine maximale Verzögerung $\leq x$ haben.

In dem betrachteten Referenznetz tritt laut Berechnung der WCTT keine Verzögerung oberhalb 659 µs auf. Da die Verzögerung maßgeblich von jedem durchlaufenden Sternknoten abhängt, bildet sich eine ähnliche Verteilung, wie aus Abbildung A.4 über die Statistik der Pfadlängen zu erwarten ist.

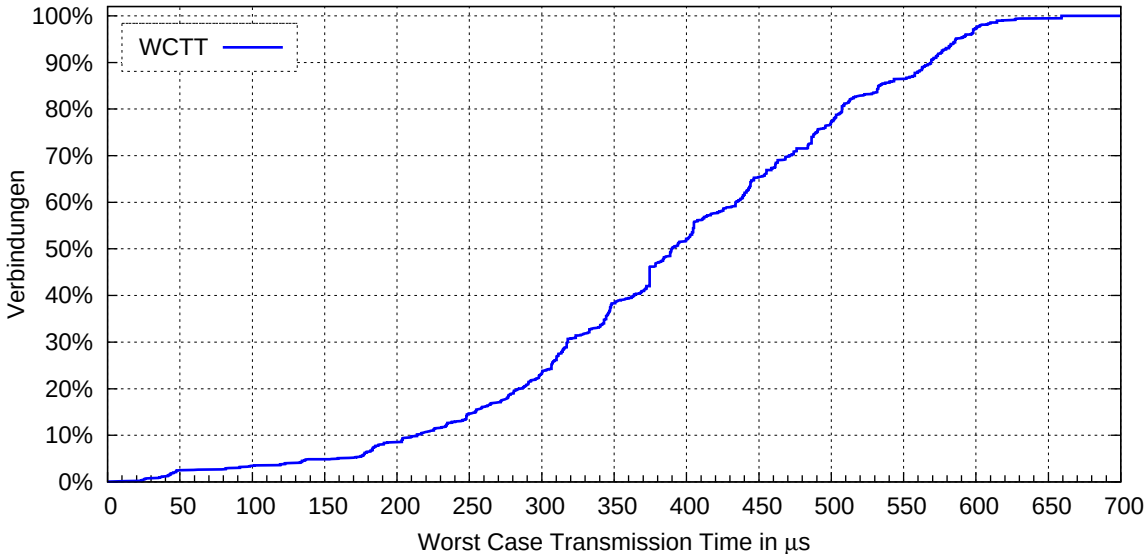


Abbildung 3.3: WCTT bei kürzestem Pfad

3.1.5 Optimierung der Übertragungszeit

Im folgenden Kapitel wird die Optimierung der WCTT behandelt. Dafür wurden zwei Ansätze betrachtet: einmal die Optimierung der verwendeten Routen, um die Last und damit die Verzögerung auf den Kanten auszugleichen. Zu diesem Zweck wurde eine Optimierung mit Hilfe eines Genetischen Algorithmus (GA) verwendet. Der zweite Ansatz betrachtet den Einfluss von dynamischer Preemption. Bei der Preemption werden Rahmen niedriger Priorität unterbrochen und die Leitung umgehend für kritische Frames freigemacht. Bei dynamischer Preemption geschieht dies nur, wenn die Leitung noch länger als die Dauer T_{offset} belegt wäre. Der Einfluss von Preemption auf Datenströme niedriger Priorität wird in Simulationen untersucht.

Lastverteilung

Bei der Berechnung der WCTT in Kapitel 3.1.4 wurden die Datenströme auf dem kürzesten Pfad durch das Netz gelegt. In diesem Kapitel wird nun die Belastung der einzelnen Kanten untersucht und mit Hilfe einer Optimierung versucht, die WCTT weiter zu verringern. Abbildung 3.4 zeigt qualitativ die Auslastung der Kanten. Dabei wurde die Datenrate der einzelnen über die Kante gerouteten Datenströme über ihre Datenmenge und die Zykluszeit aufaddiert. Grün bedeutet eine geringe Belastung, rot eine große Belastung. Die Belastung wirkt sich direkt auf die zu erwartende Verzögerung und damit die berechnete WCTT aus.

Während die Stichleitungen, also die Verbindungen der ECUs mit den Sternknoten, eine nur geringe Auslastung aufweisen, sind einzelne Kanten des ringförmigen Backbone stark ausgelastet. Es ist zu erwarten, dass die Datenströme über diese Kanten somit eine größere maximale Verzögerung aufweisen.

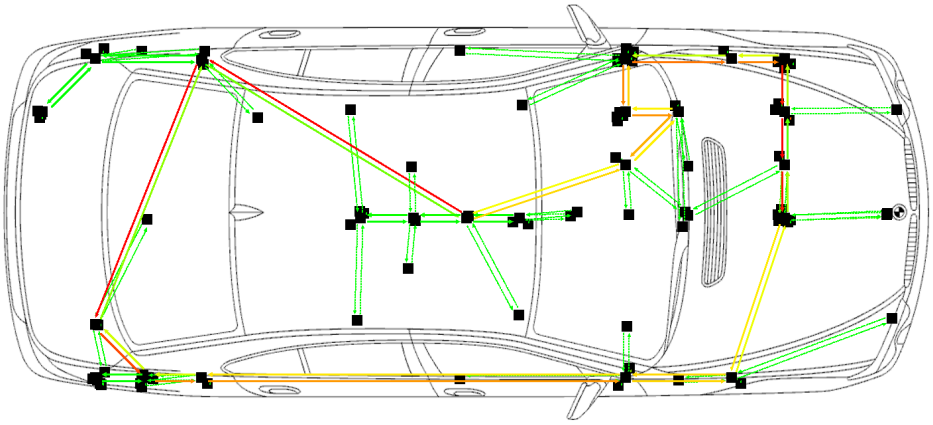


Abbildung 3.4: Auslastung Kanten Shortest Path

Mit Hilfe einer Optimierung werden nun die Routen nach festgelegter Qualitätsfunktion bewertet und verbessert. Verwendet wird ein GA, dessen Funktion in Kapitel 4.3.2 näher erläutert wird. An dieser Stelle ist nur zu erwähnen, dass eine quantitative Bewertung einer Lösung benötigt wird, um ihr Fortbestehen im evolutionären Prozess zu sichern. Diese quantitative Bewertung wird *Fitness* genannt. Um nun die WCTT zu optimieren, wurden Durchgänge mit verschiedenen Fitnessfunktionen F_d durchgeführt, die somit unterschiedliche Optimierungsziele repräsentieren. Als Vergleich ist die Verwendung von *Shortest Path*, also das Routing über die geringste Anzahl von Kanten, angegeben.

Zur Bewertung wird die Laxity verwendet, also die Differenz zwischen maximal vertretbarer Verzögerung und berechneter WCTT, $T_L = T_D - T_T$ aus Kapitel 2.4. Um eine Ressourcenbegrenzung sichtbar zu machen, wurden T_D knapp gewählt, so dass auf jeden Fall einige nicht routbare Verbindungen existieren. Vier unterschiedliche Optimierungsziele wurden gewählt: „Missed Deadline“ (*MD*), bei der die Zahl der Deadlineverletzungen, das heißt $T_L < 0$, minimiert wird; „Deadline+WCTT“ (*DL*), welche zusätzlich die WCTT verbessert; „Overall WCTT“ (*TT*), die im Gegensatz nur die berechneten T_T betrachtet, und als letzte Fitnessfunktion „80% Deadline“ (80), die T_T maximiert. Die Funktionen sind in den Gleichungen (3.6) bis (3.9) dargestellt.

$$F_{d_{MD}} = \begin{cases} \left|\frac{T_D - T_T}{T_D}\right| & \text{für } T_L < 0 \\ 0 & \text{für } T_L \geq 0 \end{cases} \tag{3.6}$$

MD in Gleichung (3.6) verwendet die prozentuale Entfernung zur gesetzten Deadline $\frac{T_D - T_T}{T_D}$. Eine prozentuale Gewichtung wird verwendet, um zeitkritische Daten mit enger gesetzter Deadline zu bevorzugen. Bei *MD* werden nur die Datenströme betrachtet, die ihre Deadline verletzen. Die Summe der Fitnessfunktion über alle Datenströme wird minimiert.

$$F_{d_{TT}} = T_T \tag{3.7}$$

TT minimiert die Summe der WCTT aller Datenströme. Ziel ist es, möglichst gleich viel Last auf alle Kanten des Netzes zu verteilen. Läge eine Gleichverteilung der Last und der Steuergeräte vor, wäre das Ergebnis analog zum kürzesten Pfad. Durch Häufung der Steuergeräte an bestimmten Knoten und bei ungleich verteilter Kommunikationslast wird eine abweichende Lösung erwartet.

$$F_{d_{DL}} = \begin{cases} 10^9 \cdot |T_D - T_T| & \text{für } T_L < 0 \\ T_T & \text{für } T_L \geq 0 \end{cases} \tag{3.8}$$

DL kombiniert die oben genannten Ziele in einer zweistufigen Optimierung. Primäres Ziel ist die Minimierung der Laxity $T_L < 0$, also die Auflösung von Deadlineverletzungen. Durch die Gewichtung mit 10^9 ist bei zeitlicher Auflösung der Laxity $min(T_L) = 1ns \cdot 10^9 = 1s$ und damit $min(T_L) > max(T_T)$. Sekundäres Ziel ist die Minimierung der WCTT der restlichen Verbindungen.

$$\begin{aligned} F_{d_{80}} &= \begin{cases} 10 \cdot \left| \frac{(0.8 \cdot T_D) - T_T}{T_D} \right| & \text{für } T_L < 0 \\ \frac{(0.8 \cdot T_D) - T_T}{T_D} & \text{für } T_L \geq 0 \end{cases} \\ &\text{mit} \quad T_L = T_D - T_T \end{aligned} \tag{3.9}$$

80 versucht den entgegengesetzten Weg. Ziel der Optimierung ist eine Maximierung der WCTT unkritischer Datenströme, dies bedeutet Datenströme mit $T_D \gg T_T$. Erhofft wird, durch die Verlegung zeitlich nicht kritischer Ströme auf längere Pfade Platz für zeitlich kritische Verbindungen auf den kurzen Pfaden zu erhalten. Um ein Oszillieren der Lösungen zu verhindern, also einen häufigen Wechsel um die Schranke $T_L = 0$, wird mit einer 80%-Grenze als Ziel gearbeitet.

Der Graph 3.5 vergleicht die Ergebnisse der Optimierung unter Verwendung der oben beschriebenen Qualitätsfunktionen. Dabei ist die Laxity T_L auf der *x*-Achse, und auf der *y*-Achse sind die Verbindungen nach aufsteigender Laxity sortiert und auf 100% normiert angetragen. Dargestellt sind die 10% zeitlich kritischen Datenströme nahe der Deadlinegrenze. Herauszuheben ist der *y*-Wert an der Stelle $T_L = 0$. Durch die Verwendung eines optimierten Routings konnten im Vergleich zum kürzesten Pfad die Verletzungen der Deadline auf fast die Hälfte verringert werden. Erreicht wurde dieser Umstand mit einer Erhöhung der WCTT der nicht zeitlich kritischen Datenströme. Die Fitnessfunktion 80 erfüllte die Erwartungen nicht und lieferte in allen Punkten schlechtere Ergebnisse. Die Verwendung der prozentualen Laxity im Gegensatz zur absoluten Laxity zeigt nur nahe der $T_L = 0$ Grenze Auswirkung, wie ein Vergleich mit der Kurve *MD* und *DL* zeigt. Ein qualitativer Vergleich der eben genannten Kurven sowie des Ergebnisses der *TT*-Optimierung im Vergleich zum kürzesten Pfad ist aus der analytischen Betrachtung heraus nicht möglich.

Um dies zu ermöglichen, wurden mit Hilfe von ITMsim aus Kapitel 6.2 Simulationen mit den unterschiedlichen Lösungen durchgeführt und dort Statistiken über auftretende Verzögerungen vorgenommen. Abbildung 3.6 zeigt die maximalen bei der

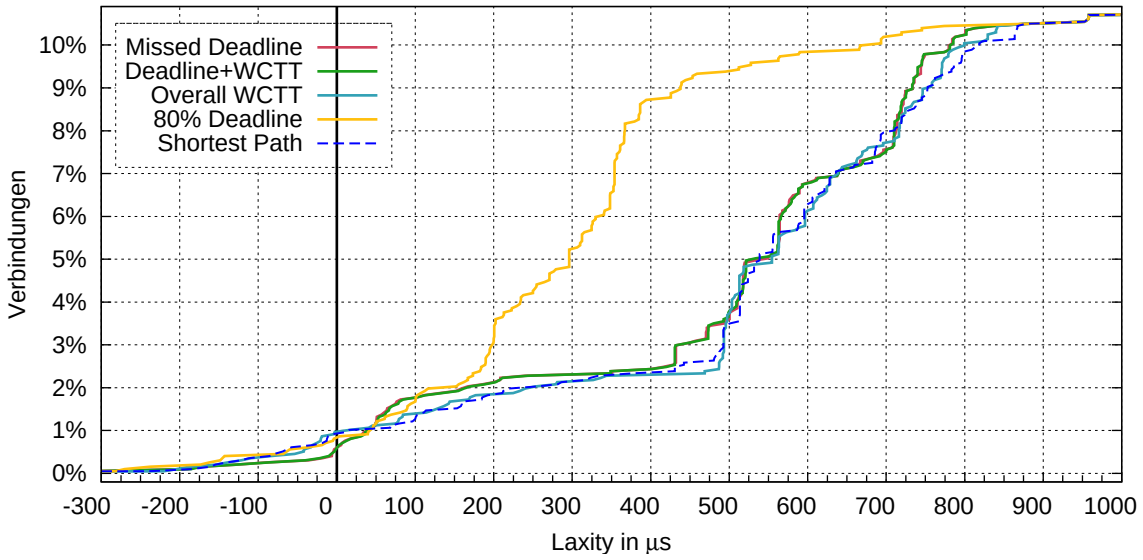


Abbildung 3.5: Laxity bei unterschiedlichen Optimierungszielen

Simulation auftretenden Verzögerungen. Das Diagramm ist wie Abbildung 3.3 zu lesen. Simuliert wurden mehrere Durchgänge mit unterschiedlicher statistischer Verteilung der Kommunikationsanforderungen, und die höchste aufgetretene Verzögerung wurde gespeichert. Nähere Angaben zu den Simulationsparametern sind in Kapitel B zusammengefasst.

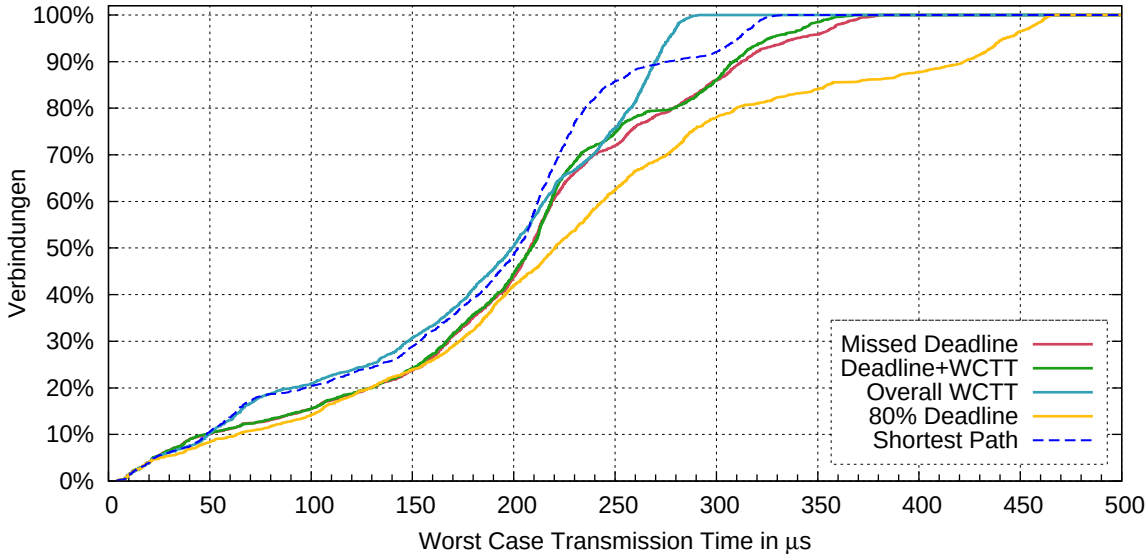


Abbildung 3.6: WCTT bei unterschiedlichen Optimierungszielen

Zu sehen ist, dass *TT* die auftretenden WCTT sichtbar reduziert. Durch diese Zielfunktion wird also wie erwartet die globale Verzögerung verringert, ohne speziell auf Datenströme mit Deadlineverletzung einzugehen. Die bessere Verteilung der Last auf

den Kanten im Gegensatz zum kürzesten Pfad wird an dem gleichmäßigeren Verlauf der Kurve sichtbar, in dem die Beule in der Kurve abgeflacht wird. Die Qualitätsfunktion eignet sich somit gut für Systeme, die für alle Datenströme i vollständig lösbar sind ($T_{L_i} > 0 \; \forall i$).

Im Vergleich dazu erhöht die Verringerung der Deadlineverletzungen die globale WCTT, wie anhand von Kurve *DL* abzulesen ist. Durch das priorisierte Routen der zeitlich kritischen Ströme erhöht sich die Verzögerung der anderen Datenströme. Die *MD*-Kurve schließt durch die fehlende Optimierung der Ströme mit $T_L > 0$ schlechter ab.

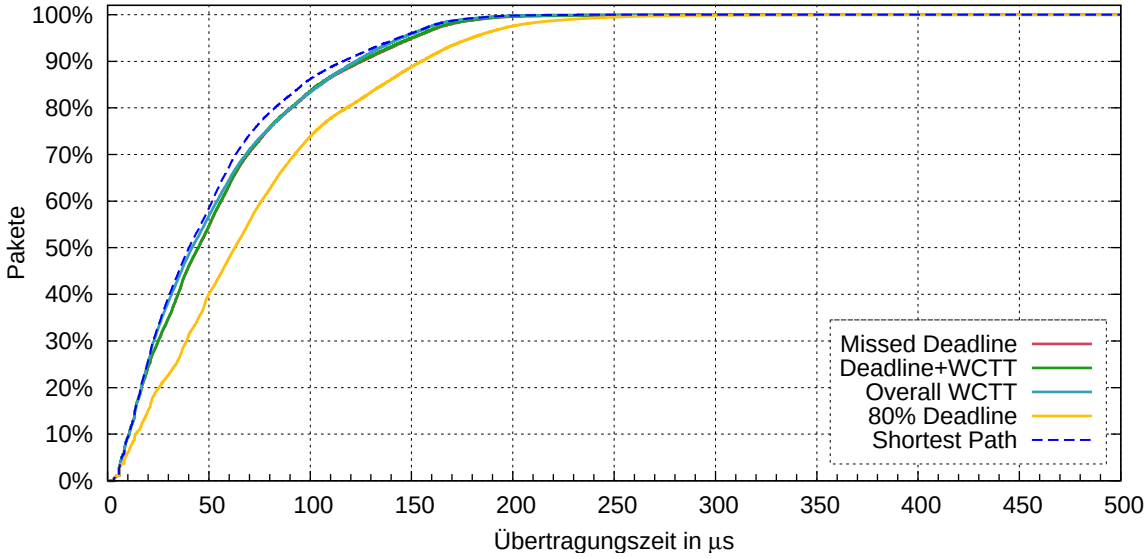


Abbildung 3.7: Übertragungszeit bei unterschiedlichen Optimierungszielen

Neben der reinen maximal auftretenden Übertragungsverzögerung der Datenströme wurde eine Statistik über die Verteilung der Ende-zu-Ende-Verzögerung erstellt, welche in Abbildung 3.7 dargestellt ist. Dies soll den Jitter, also die Varianz der Übertragungsverzögerung, sichtbar machen. Gespeichert wurde die Differenz zwischen Sendeanforderung und Empfang eines Paketes und als kumulierte Verteilungsfunktion dargestellt. Mit Ausnahme der 80-Kurve liegen die betrachteten Kurven dicht beisammen. Die Verwendung des kürzesten Pfades als Routingmethode weist einen leicht geringeren Jitter auf, *DL* und *MD* einen marginal höheren.

In Abbildung 3.8 ist die qualitative Belegung der Kanten nach Optimierung mit *TT* als Qualitätsfunktion dargestellt. Dies ist im Vergleich mit Abbildung 3.4 zu betrachten. Die globale Minimierung der WCTT führt nicht zu einer gleichmäßigeren, sondern einer stärker ungleichgewichtigen Auslastung der Kanten. Dies ist vor allem durch die nicht über die Steuergeräte gleichverteilte Kommunikationslast bedingt. Dadurch lässt sich der etwas höhere Jitter beim Vergleich von kürzestem Pfad und *TT* erklären.

Als Zusammenfassung dieses Unterkapitel ist festzuhalten, dass durch eine Optimierung der Pfade Verbesserungen im Bereich der Deadlineverletzungen erreichbar sind.

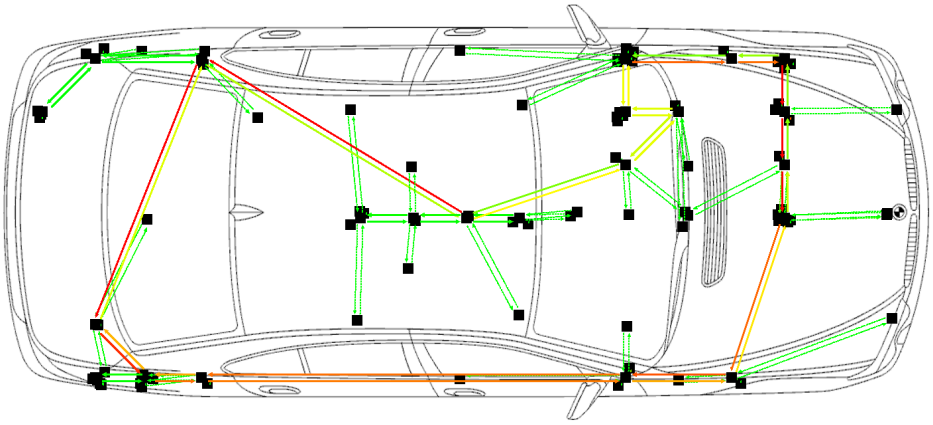


Abbildung 3.8: Auslastung der Kanten nach *TT*-Optimierung

Handelt es sich also um ein stark ausgelastetes System, können so trotzdem gültige Lösungen generiert werden. Dafür ist die „Deadline+WCTT“-(*DL*-)Qualitätsfunktion am besten geeignet. Auch lassen sich die maximal auftretenden Verzögerungen verringern. Diese Manipulierung der selten auftretenden Pakete mit hohem Delay wirkt sich negativ auf die durchschnittliche Verteilung der Verzögerung aus. Für eine möglichst geringe durchschnittliche Verzögerung ist der kürzeste Pfad durch das System mit möglichst wenig durchlaufenden Sternknoten das beste und auch am einfachsten zu realisierende Verfahren. Aus diesem Grund wird für die weiteren Betrachtungen sowie für die Netzplanung in Kapitel 4.3.5 das Routing über den kürzesten Pfad verwendet.

Dynamische Preemption

Ein weitere Möglichkeit, die WCTT zu verringern, ist die Minimierung von $T_{W1_{SP}}$ in der Formel (3.5). Durch das unter dem Begriff *Head-of-Line-Blocking* bekannte Phänomen, bei dem die Übertragung von Daten höherer Priorität durch die bereits begonnene Übertragung von Daten niedriger Priorität verzögert wird, muss pro Sternknoten eine zusätzliche Verzögerung eingeplant werden. Im Falle eines größtmöglichen Paketes mit MTU von 1500 Byte entspricht dies bei Gigabit Ethernet einer Verzögerung inklusive IFG von ≈ 12 µs ($T_{\text{IFG}} = 96$ ns).

Um dies zu umgehen, wird ein Preemption-Mechanismus in das Scheduling System des Sternknotens eingeplant (siehe Kapitel 3.1.3). Bei Eintreffen eines Rahmens höherer Priorität unterbricht der Preemption-Mechanismus umgehend die laufende Übertragung und leert den Kanal. Der unterbrochene Rahmen ist verloren und wird beim Empfänger verworfen. Nach dem obligatorischen minimalen Abstand zwischen zwei Rahmen, dem IFG, wird die Übermittlung des priorisierten Frames begonnen. So entsteht eine maximal Wartezeit von $T_{W1_{SP}} = T_{\text{IFG}}$.

Im Rahmen dieser Arbeit wurde der Mechanismus zu einer dynamischen Preemption erweitert. Bei der dynamischen Preemption wird die Übertragung nur dann gestoppt,

wenn der Kanal noch für einen Zeitraum $T_T > T_{\text{offset}}$ belegt ist. Die Belegung des Kanals kann mittels der Formel (3.10) berechnet werden. Somit wird über den Parameter T_{offset} der in die WCTT-Berechnung einfließende Faktor $T_{W1_{SP}} = T_{\text{offset}} < T_{\text{MTU}})$ festgelegt (vgl. Formel (3.5)).

$$T_{W1_{SP}} = \begin{cases} T_{\text{IFG}} & \text{für } T_{\text{offset}} \leq T_{\text{IFG}} \\ T_{\text{offset}} & \text{für } T_{\text{offset}} > T_{\text{IFG}} \end{cases} \tag{3.10}$$

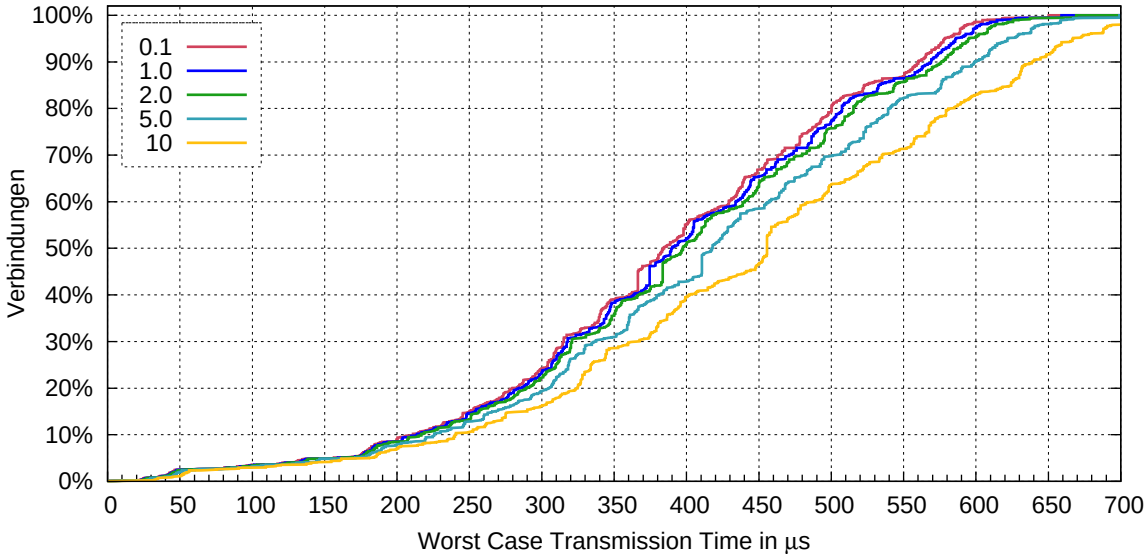


Abbildung 3.9: WCTT bei unterschiedlichem Preemption Offset

Abbildung 3.9 zeigt die Berechnung der WCTT des Referenznetzes aus Kapitel A.2 mit variiertem T_{offset} in μs. Wie erwartet, ergibt sich eine Rechtsverschiebung der Kurve mit steigendem T_{offset}. Die Kurve mit $T_{\text{offset}} = 0{,}1$ µs $\approx T_{\text{IFG}}$ repräsentiert dabei den nicht dynamischen Preemption-Mechanismus, die Kurve $T_{\text{offset}} = 10{,}0$ µs $\approx T_{\text{MTU}}$ den nahezu ungestörten Fall.

Aussagekräftiger zur Abschätzung des eingesetzten Offset ist aber das Verhalten bei normalem Netzbetrieb, das durch eine Simulation bewertet wurde. Abbildung 3.10 zeigt die maximale Ende-zu-Ende-Verzögerung. Im ablesbaren Bereich liegen alle Kurven mit Ausnahme der 10 übereinander. Daraus lässt sich schließen, dass der überwiegende Teil der Unterbrechungen bei einem $T_{\text{offset}} > 5$ µs stattfindet. Erst durch Setzen der Grenze auf 10,0 µs zeigt sich eine sichtbare Verschiebung der simulierten WCTT in Richtung einer größeren Verzögerung.

Bewertet man nun die Verteilung der Verzögerungen am Sternknoten, ergibt sich der in Diagramm 3.11 dargestellte Verlauf. Die y-Achse wurde bei 80% abgeschnitten, da unterhalb dieser Stelle eine konstante Verzögerung ausschließlich durch T_{proc} vorliegt. Das heißt, diese Rahmen treffen nicht auf andere im Sternknoten und werden ohne Verzögerung weitergeleitet. Auch hier zeigt sich wieder kein signifikanter Unterschied bei $T_{\text{offset}} \in [0{,}1; 5{,}0]$.

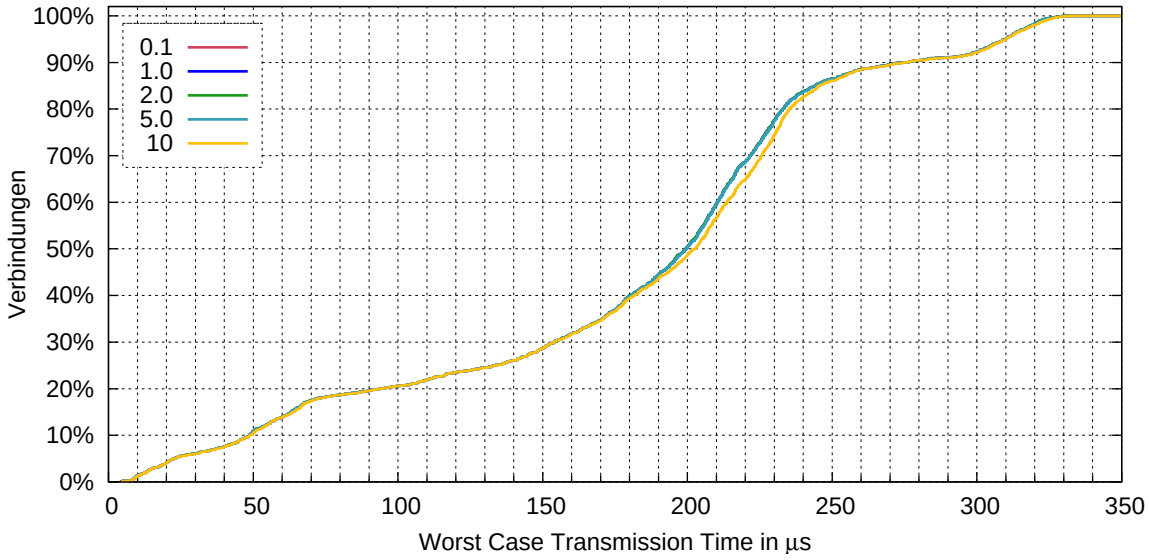


Abbildung 3.10: WCTT bei unterschiedlichem Preemption Offset (Simulation)

Abgerundet wird das Bild mit der Verzögerung der Rahmen niedriger Priorität am Sternknoten, also derjenigen, die durch die Preemption unterbrochen werden (Abbildung 3.12). Etwas über 50% der Pakete gehen ungehindert durch und werden nicht verzögert. Bei der Verwendung von $T_{\text{proc}} = 10$ µs, also dem fast ungestörten Betrieb, ist eine Verzögerung durch andere Rahmen niedriger wie hoher Priorität bis über 1 ms abzulesen. Durch den Einsatz von Preemption erhöht sich die durchschnittliche Verzögerung im Sternknoten wie zu erwarten. Ein aussagekräftiger Einfluss bei Variation des T_{offset} ist weiterhin nicht auszumachen, nur bei 5 µs zeigt sich eine leichte Verbesserung.

Zusammenfassend kann man sagen, dass der Einsatz von Preemption die rechnerische Verzögerung auf die Pakete mit hoher Priorität, wie auch die in der Simulation wirklich auftretenden Verzögerungen, verbessert. Im Gegenzug wird die Übertragung der Daten niedriger Priorität leicht verzögert. Die Auswahl des T_{offset} der dynamischen Preemption zeigte, von der analytischen Betrachtung abgesehen, im Bereich von $T_{\text{offset}} \in [0{,}1; 5{,}0]$ keine signifikanten Auswirkungen. Zur Minimierung des Jitters der zeitkritischen Datenströme wird der Einsatz von Preemption angeraten. In den weiteren Simulationen wurde $T_{\text{offset}} = 1$ µs gesetzt.

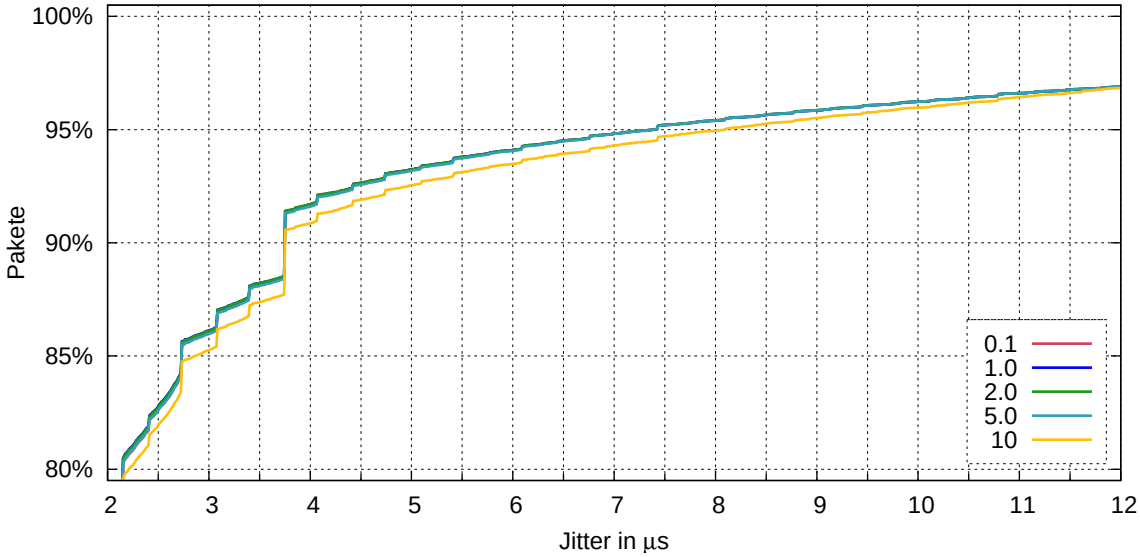


Abbildung 3.11: Jitter am Switch bei unterschiedlichem Preemption Offset

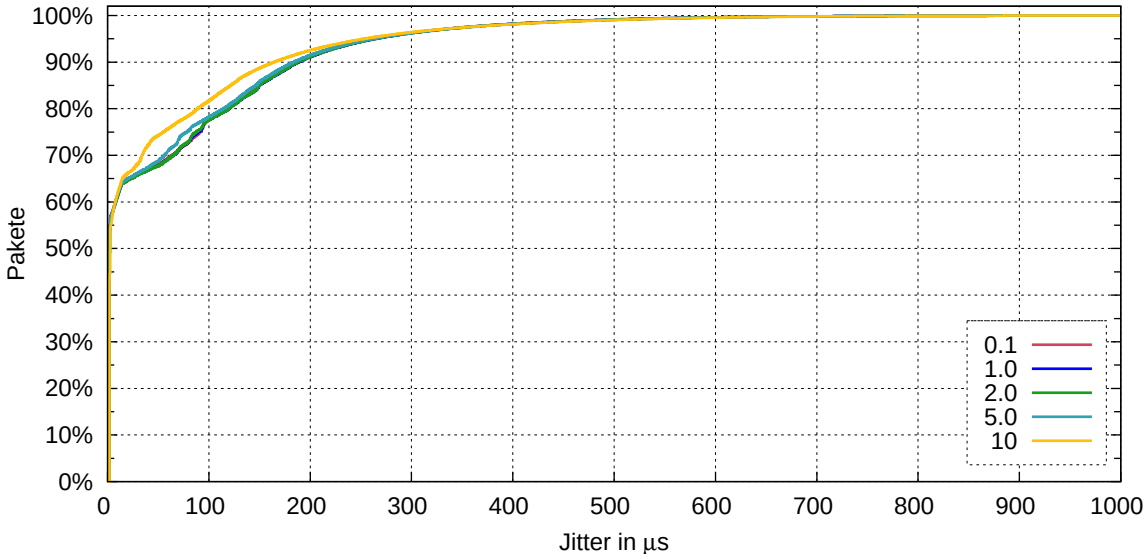


Abbildung 3.12: Jitter der Videoübertragung bei untersch. Preemption Offset

3.1.6 Bewertung der Warteschlangensysteme

Prämisse bei der Entwicklung der Sternknoten war die Möglichkeit der einfachen Berechnung einer WCTT bei unmodifiziertem Medienzugriff des eingesetzten Ethernet. Um dies zu erreichen, wurde das in Kapitel 3.1.3 eingeführte hierarchische Scheduling verwendet, um die nicht kritischen sowie die zeitkritischen Datenströme zu separieren.

Dieses Kapitel beschäftigt sich nun mit der simulativen Bewertung der Leistung der unterschiedlichen Warteschlangensysteme am Beispiel des verwendeten Fahrzeugnetzes aus Kapitel A.2. Zur Simulation wurde das eigene Framework ITMsim (siehe Kapitel 6.2) verwendet. Untersucht wurden vier unterschiedliche Warteschlangensysteme: „Flow“, „Strict Priority“, „Input Port“ und „Input Port flush“, welche unterschiedliche Methoden zur Ein- und Aussortierung der Pakete verwenden.

„Flow“ (Fl) ist der in Kapitel 3.1.3 verwendete Aufbau, bei dem alle kritischen Datenströme in einzelne, exklusiv verwendete Warteschlangen einsortiert werden. Die Warteschlangen werden im *Round-Robin*-(RR-)Verfahren bedient. Beim RR werden nacheinander die Warteschlangen abgefragt und es wird, falls belegt, ein Rahmen entnommen und weitergeleitet. Die Rahmen erreichen als nächsten Schritt eine gemeinsame Warteschlange der höchsten Priorität, die nach dem *Strict-Priority*-(SP-)Mechanismus geleert wird. Bei SP wird immer die Warteschlange mit der höchsten Priorität bedient, bis diese leer ist. Erst dann werden in absteigender Reihenfolge gemäß der Priorität andere Warteschlangen bearbeitet. Trifft ein Frame in einer Warteschlange mit höherer Priorität ein, wird diese umgehend bei freiwerdendem Kanal bedient. Durch das SP-Scheduling können die kritischen Datenströme bis auf $T_{W1_{SP}}$ (vgl. Formel (3.1)), das *Head-of-Line-Blocking*, vom restlichen Datenverkehr wie Video- und Audioverbindungen separat betrachtet und behandelt werden.

Im Gegensatz dazu verwendet „Strict Priority“ (SP) keine vorgelagerte Warteschlange nach dem RR-Prinzip für die kritischen Datenströme. Alle Rahmen dieser Kategorie werden in eine gemeinsame Warteschlange der höchsten Priorität eingeordnet. Dies bedeutet umgekehrt eine Minimierung der benötigten Hardware- und Softwarekomplexität.

Bei dem angedachten System „Input Port“ (IP) wird eine Kombination der genannten Verfahren verwendet. Es existiert neben der SP-Warteschlange eine vorgelagerte RR-Einheit. Die Einsortierung der Rahmen erfolgt aber nicht über den Datenstrom, sondern den Eingangsport. Das heißt, jeder Eingangsport des Sternknoten erhält eine exklusive RR-Warteschlange am Ausgangsport des Gerätes. Da es sich meist um eine Schnittstellenzahl von 4-8 Ports handelt, ist die benötigte Anzahl und Größe der Warteschlange geringer als im „Flow“-System.

„Input Port flush“ (IPf) ist eine Modifikation von Input Port, die bei Entnahme aus der RR-Warteschlange nicht nur einen Rahmen entfernt, sondern den Port bis zur vollständigen Entleerung bedient.

Verzögerung der Verbindung

Zuallererst wird die Verbindung und damit die Ende-zu-Ende-Verzögerung gemessen, die direkt auf die WCTT abbildbar ist. Die entsprechenden Kurven sind in Abbildung 3.13 dargestellt. „Upper Bound“ entspricht dabei der oberen, analytisch berechneten WCTT. Es bilden sich qualitativ zwei vergleichbare Kurvenverläufe aus: Fl und IPf sowie IP und SP.

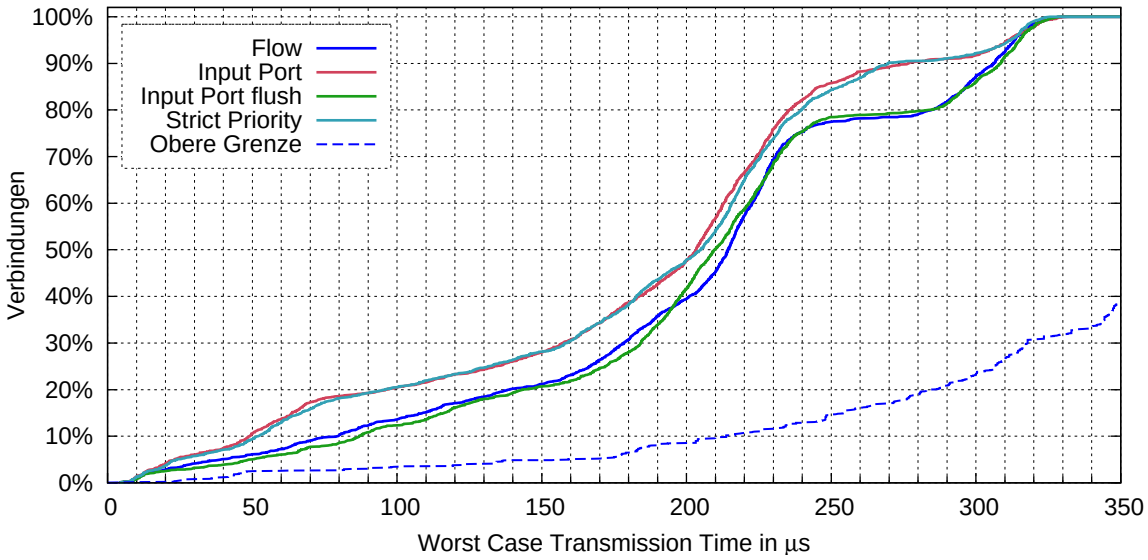


Abbildung 3.13: WCTT bei unterschiedlichen Warteschlangen

Die im Fl (beziehungsweise IPf) auftretenden höheren Verzögerungen treten durch Umsortierung der Pakete am Sternknoten auf. Bedingt durch die Einsortierung in die RR-Warteschlange ist es möglich, dass von der gleichen Kante später eintreffende Pakete früher abgesendet werden, falls der Zeiger (die gerade bediente Warteschlange) kurz hinter der eigenen Warteschlange steht. Ähnlich ist das Phänomen bei der IPf anzusehen. Bei IP und SP werden die Daten so, wie sie auf der Leitung eintreffen, einsortiert, so dass keine Umsortierung auftreten kann.

Eine etwas andere Darstellung ist in Abbildung 3.14 dargestellt, welche die Differenz zwischen berechneter und simulierter Ende-zu-Ende-Verzögerung der Datenströme aufzeigt. Die so genannte Unsicherheit zeigt die Überschätzung bei der Berechnung der WCTT. Das Diagramm bestätigt die Richtigkeit der analytischen Betrachtung: es gibt kein Ergebnis < 0, also keine Unterschätzung der Verzögerung, die zu einem nicht geplanten Verhalten führen könnte. Auf Grund der fehlenden Synchronisierung beim Medienzugriff von Ethernet kann die berechnete Verzögerung bei jeder Verbindung zutreffen, dies wird aber im realen Fall nur bei wenigen erreicht.

Der Vollständigkeit halber wurde parallel die Verzögerung der nicht kritischen Datenströme untersucht. Dort zeigte sich kein Einfluss der verwendeten Warteschlange auf das Verzögerungsverhalten der Pakete.

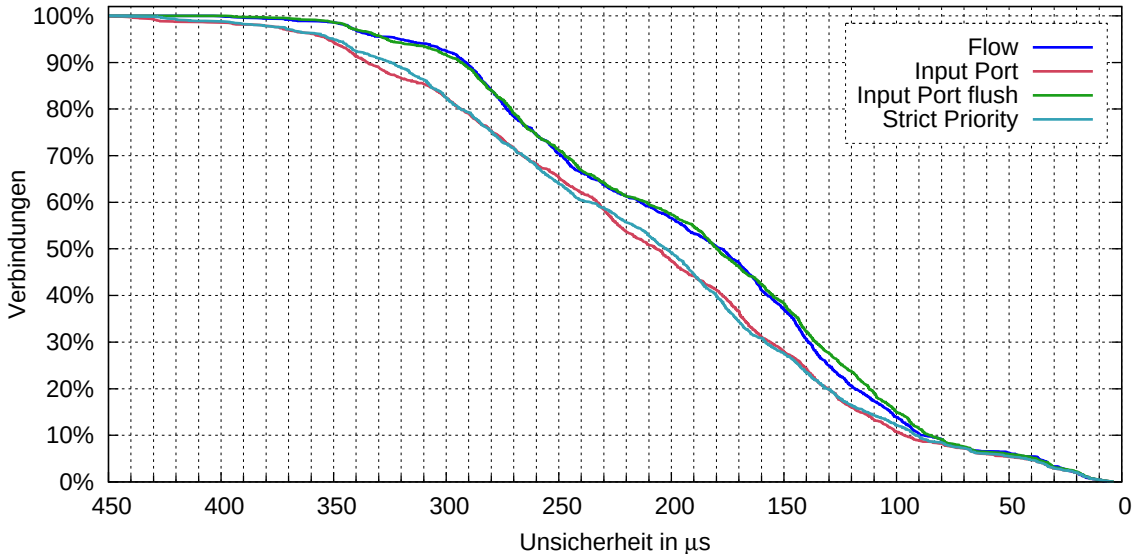


Abbildung 3.14: Unsicherheit der analytischen Betrachtung

Verzögerung am Switch

Um die Pfadlänge der Verbindungen aus der statistischen Untersuchung herauszunehmen, wurde das Verhalten am Switch untersucht. Betrachtet wird also jeder Hop auf dem Weg einzeln, dargestellt in Diagramm 3.16.

Wie im vorherigen Kapitel zeigen sich für IP und SP ein vergleichbares Verhalten. Nur den Extremwert der größten auftretenden Verzögerung betreffend zeigen auch Fl und IPf Ähnlichkeiten.

Abbildung 3.17 und 3.18 zeigen beide die Verteilung der Verzögerung einzelner Rahmen beim Durchlaufen des Sternknotens auf. Es handelt sich um zwei Ausschnitte der relevanten Stellen der Verteilung, um die Lesbarkeit zu erhöhen. Da unterhalb 80% keine Verzögerungen auftraten, dass heisst die Rahmen ungehindert durch den Sternknoten gingen, wurde dieser Bereich in den Diagrammen ausgespart. Der Verlauf der Kurven zeigt, dass Fl und IPf bei kleiner Last eine sichtbar geringere Verzögerung und damit einen geringeren Jitter aufweisen. Bei hoher Last vertauscht sich dies und die Systeme IP und SP zeigen geringere Jitter. Der Verlauf gegen 100% bestätigt auch das aus Diagramm 3.16 herausgelesene Verhalten, dass der Einsatz von Fl und IPf eine höheren maximalen Verzögerung verursacht.

Verzögerung im Fehlerfall

Neben dem Verhalten bei normalen Bedingungen ist auch der Fehlerfall zu bewerten. Zu diesem Zwecke wurde in der Simulation eine Verbindung mit größtmöglicher Datenrate geschaltet. Ein zufällig ausgewählter Knoten sendet als Quelle zu einer wiederum zufällig ausgewählten Senke so häufig Rahmen, dass die Warteschlange immer gefüllt ist. Betrachtet wird wieder die Ende-zu-Ende-Verzögerung der Datenströme

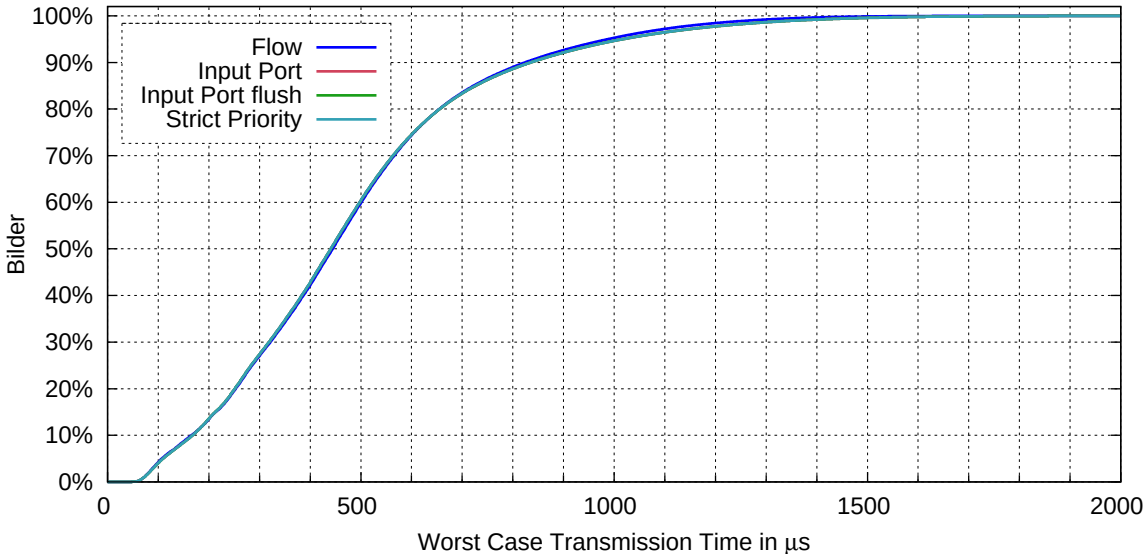


Abbildung 3.15: Verzögerung der Videoübertragung pro Frame

im Vergleich zur berechneten WCTT unter Einsatz unterschiedlicher Warteschlangensysteme (Abbildung 3.19).

Hier zeigt sich ein von den vorherigen Ergebnissen differierendes Verhalten. Die nach SP und IPf arbeitenden Warteschlangensysteme schaffen es für einen Großteil der Rahmen nicht, die berechnete WCTT einzuhalten. Tatsächlich sind alle Datenströme, die über dieselben Kanten wie der Fehlerstrom gehen, von diesem Verhalten betroffen. Ein einzelner Fehler hat also Störwirkung auf andere Funktionen. Dies führt zu dem im Kapitel 2 beschriebenen Verhalten, bei dem durch die Homogenisierung und Zentralisierung der Architektur die Verfügbarkeit sinkt.

Durch Separierung aller Datenströme in eigene Warteschlangen können die Auswirkungen des Fehlers im System unterdrückt werden. Die Kurve des Systems Fl zeigt keine Verschlechterung durch den Fehlerstrom. Die störenden Rahmen werden überwiegend in der ersten RR-Warteschlange durch Überlauf des Puffers eliminiert.

Das Verhalten des IP-Mechanismus liegt zwischen den oben genannten. Es geht ein geringer Teil der Rahmen verloren, die anderen werden nur so weit verzögert, dass sie garantiert unter der analytischen Grenze bleiben. Der Verlust findet ausschließlich an der fehlerhaften ECU statt.

Auswertung der Ergebnisse

Ein optimales Warteschlangensystem konnte nicht gefunden werden. Die beiden nach obiger Untersuchung in Frage kommenden sind „Flow“ Fl und „Input Port“ IP. Fl zeigt eine systemweite Unempfindlichkeit gegen Datenströme außerhalb der geplanten Grenzen. Im Gegensatz dazu weist es eine höhere mittlere Ende-zu-Ende-Verzögerung auf. Auch die Komplexität ist höher, da zum Herstellungszeitpunkt der

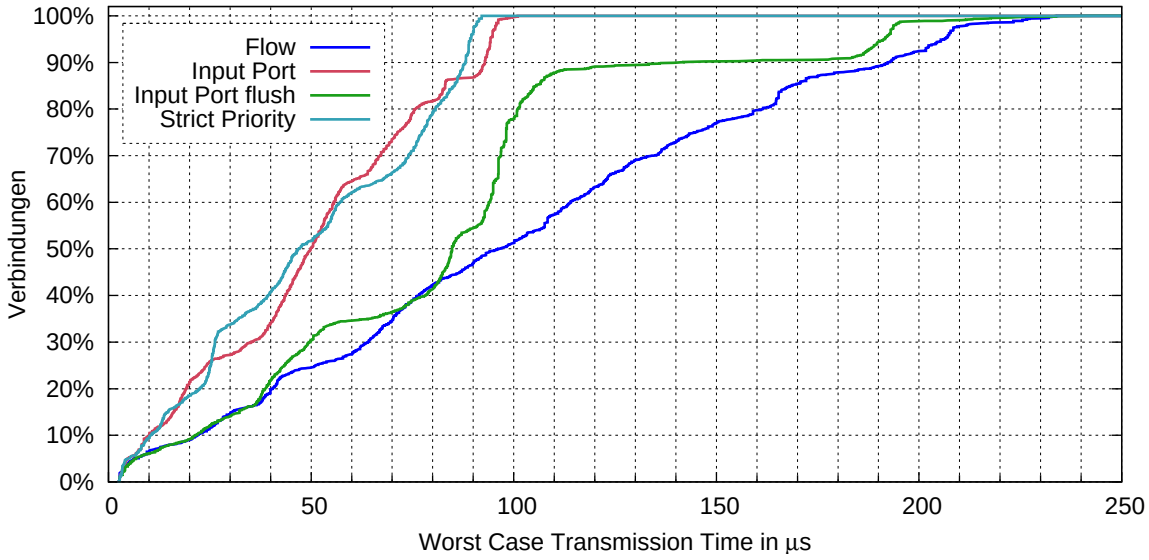


Abbildung 3.16: WCTT am Switch bei unterschiedlichen Warteschlangen

Hardware die Anzahl der Datenströme nur abgeschätzt werden kann, diese aber für die Auslegung der RR-Warteschlange bekannt sein müssen.

IP verringert die Anzahl der notwendigen RR-Queues auf eine zum Herstellungszeitpunkt bekannte Menge, die Anzahl von Eingangsschnittstellen. Dort ist aber die notwendige Tiefe der Puffer wiederum vom zu erwartenden Verkehr abhängig. Die Verwendung von IP ist sehr robust gegen Fehlerfortpflanzung, lediglich an der Störquelle kommt es zur Störung von fremden Datenströmen. Im ungestörten Betrieb zeigt dieses System einen sehr geringen Jitter und eine faire Weiterleitung der Rahmen in unveränderter Reihenfolge.

Es ist somit vom Einsatzzweck abhängig, welches System zum Einsatz kommen sollte. Verwendet man zusätzlich, wie in Kapitel 3.1.3 kurz eingeführt, einen Mechanismus der Klasse *Usage Parameter Control* (UPC), der die Datenströme kontrolliert, so ist der Einsatz von „Flow“ zum Beispiel nicht nötig.

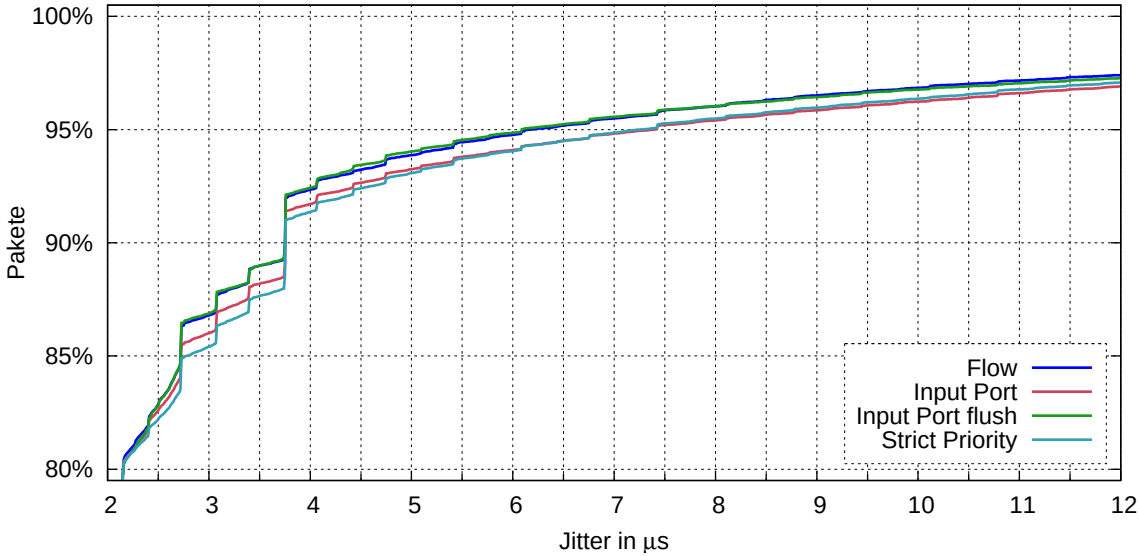


Abbildung 3.17: Jitter am Switch bei unterschiedlichen Warteschlangen

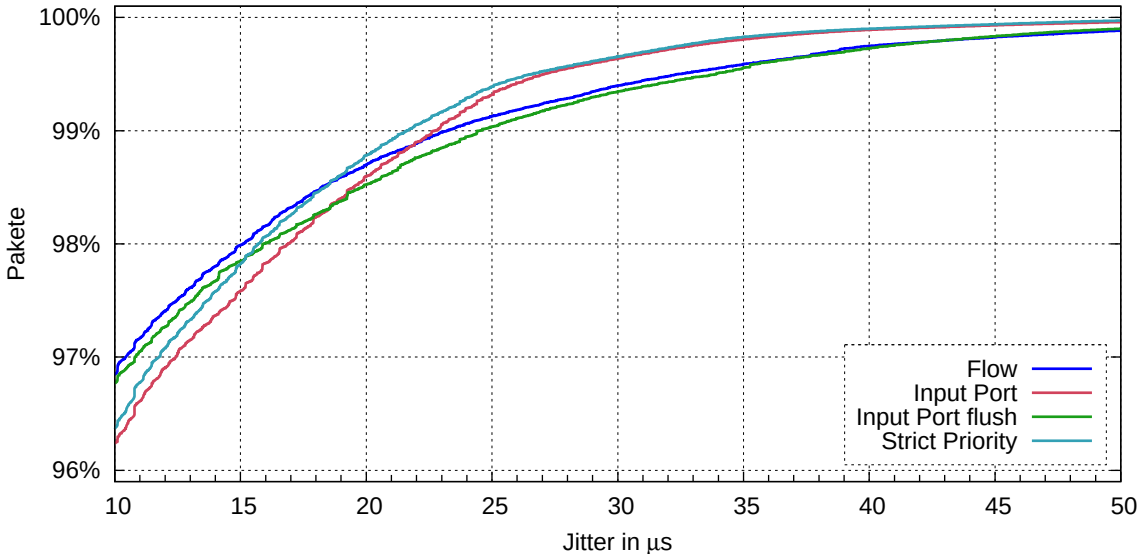


Abbildung 3.18: Jitter am Switch bei unterschiedlichen Warteschlangen

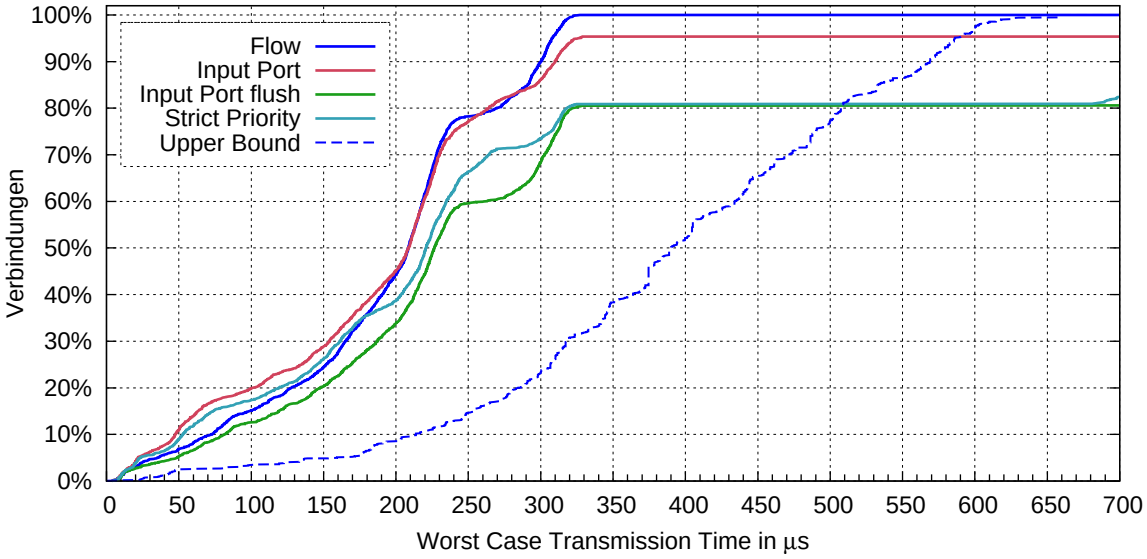


Abbildung 3.19: WCTT bei unterschiedlichen Warteschlangen im Fehlerfall

3.2 Fehlerdetektion und -behandlung

Durch die Zentralisierung der Verarbeitung und der Übertragung der Daten kommt es, wie bereits am Anfang des Kapitels erwähnt, zu einer steigenden Rückwirkung eines Ausfalls auf das Gesamtsystem. Da mehrere Datenströme auf einzelnen Leitungen gebündelt sind, führt ein Fehler dieser Leitung zum Versagen mehrerer Funktionen auf unterschiedlichen ECUs. Bei zukünftigen Bordnetzen, die auf den vorgestellten Ansätzen basieren, muss also ein erhöhter Aufwand betrieben werden, um die Zuverlässigkeit des Gesamtsystems zu erhalten.

In diesem Kapitel werden diese Anforderungen an das System definiert sowie der Datenverkehr in drei Klassen kategorisiert. Anschließend werden die im System vorhandenen Methoden zur Erhöhung der Ausfallsicherheit für die drei Klassen beschrieben. Kapitel 3.3 beschäftigt sich dann anschließend mit einer vorgeschlagenen Implementierung dieser Konzepte.

3.2.1 Anforderungen an die Robustheit

Die Anforderungen werden in quantitative sowie qualitative unterschieden. Die quantitativen Anforderungen existieren auf Grund von staatlich gesetzten Standards, die eine gewisse Verfügbarkeit dieser sicherheitskritischen Systeme festlegen. Sie sind vom jeweiligen Anwendungsfall und der jeweiligen Industrie abhängig.

Die qualitativen Anforderungen beziehen sich überwiegend auf Kundenanforderungen. Dort bestimmen die Qualitätssicherung der Übertragung und die Datensicherheit in Form von Vertraulichkeit die notwendigen Mechanismen. Auch der Angriff auf das System durch Ausspähen und Manipulieren der übertragenen Daten muss beachtet werden.

Sicherheitslevel

Eine Standardvorgabe ist das Sicherheits-Integritätslevel (SIL). Das SIL stammt aus der Norm IEC 61508 der International Electrotechnical Commission [iec99, Bro00]. In vier diskreten Stufen $1 - 4$ werden Funktionen in unterschiedliche Kategorien eingeteilt, die definierte sicherheitsgerichtete Konstruktionsprinzipien erfüllen müssen. Dies beinhaltet einerseits organisatorische Methoden, wie Versionsmanagement, Nachvollziehbarkeit von Codeänderungen, Code-Review von unabhängigen Stellen, Codecoverage-Analyse und weitere, andererseits aber auch Festlegungen von notwendiger Redundanz. Zur Einstufung der Funktion muss eine Gefährdungsbeurteilung durchgeführt werden.

Basierend auf dieser Norm wurde für Systeme in Fahrzeugen eine neue Norm entwickelt, die sich noch in der Draft-Phase der Normierung befindet. Die ISO-Norm 26262 [iso09] definiert vier Kategorien $A - D$, in die Funktionen anhand einer qualitativen Methodik durch Abschätzen des Verletzungsgrades sowie der Häufigkeit und

der Beherrschbarkeit der Situation durch den Fahrer eingeordnet werden. Basierend auf dieser Einordnung werden definierte Verfügbarkeitsgrenzen vorgegeben sowie notwendige Schritte im Entwicklungsprozess festgelegt.

Mit dem DO-178B [RTC92] existiert auch ein Pendant in der Luftfahrtindustrie. Hier werden fünf Stufen DAL $E - A$ festgelegt, die sich dabei an möglichen Schäden bei einer Fehlfunktion der Software orientieren und von „keine Auswirkungen“ bis „katastrophal“ reichen. Dies hat wiederum Rückwirkungen auf Entwicklungsmethoden, Dokumentations- und Review-Pflichten sowie die physikalische Realisierung.

In der vorliegenden Arbeit werden drei Kategorien von Übertragungen festgelegt: „Sicherheitskritische Funktionen“ Kapitel 3.2.2, „Kritische Funktionen“ Kapitel 3.2.3 sowie „Unkritische Funktionen“ Kapitel 3.2.4. Welche Kategorie bei welcher Stufe der oben genannten Standards eingesetzt werden soll, muss individuell anhand der konkreten Funktion und der physikalischen Realisierung der Komponenten festgelegt werden.

Abschottung

Eine zweite Klasse von Anforderungen betreffen die Datensicherheit, also nicht die Sicherheit der Übertragung, sondern die Sicherheit der übertragenen Informationen. Diese Anforderungen sind überwiegend qualitativer Natur. In diesem Bereich sind vor allem Stichpunkte wie Vertraulichkeit, Nachvollziehbarkeit und Unversehrtheit zu nennen. Als Anbieter eines Kommunikationsnetzes mit gemeinsam verwendetem Kommunikationsmedium muss dieses vom Betreiber, im Kontext der vorliegenden Arbeit also vom OEM, im Rahmen des Systems sichergestellt werden. Der Punkt Datensicherheit ist nicht Fokus dieser Arbeit, aus diesem Grund wird in diesem Kapitel nur eine grobe Übersicht gegeben.

Grundlage der in Kapitel 3.1.4 vorgestellten Mechanismen zur Garantie der WCTT ist der aus Kapitel 3.1.3 eingeführte Sternknoten. Wird Datenverkehr über Umwege in das Netz eingeleitet oder anderweitig verzögert, können die berechneten Grenzen nicht mehr garantiert werden. Zwingend notwendig ist damit der Aufbau eines „trusted network“, also eines Kommunikationsnetzes, das aus zertifizierten und vertrauenswürdigen Instanzen besteht. Dies kann durch Unterbinden des physikalischen Zugriffs auf die Komponenten geschehen, zum Beispiel durch einen verplombten Verbau in Industrieanlagen oder Flugzeugen. Eine weitere Möglichkeit ist ein Frage-Antwort-Protokoll auf einer höheren Ebene, das die Schnittstellen nur nach erfolgreicher gegenseitiger Authentifizierung freischaltet. Dies kann mit bekannten Methoden wie einer Public-Key-Infrastruktur (PKI) und asymmetrischen Schlüsseln realisiert werden, die vom OEM herausgegeben werden.

Da mit dieser Methode keine „Man in the Middle“-Attacke verhindert werden kann, müssen alle Datenströme an allen Eingängen aller Zwischenknoten auf Einhalten der zugelassenen Parameter überprüft werden, um die Auswirkungen von potentiellen Störungen zu minimieren. Aus Kostengründen muss ein zum Einsatzszenario passender Kompromiss zwischen Aufwand und Sicherheit gefunden werden.

3.2.2 Sicherheitskritische Funktionen

Die höchste Klasse der Übertragung besteht aus den sicherheitskritischen Funktionen (sKF). Eine sKF muss unter allen Umständen funktionieren. Dies bedeutet natürlich keine Verfügbarkeit von 1, was technisch nicht möglich ist, sondern eine sehr geringe Fehlerrate. Da die Fehlerrate hauptsächlich von der verwendeten Hardware des Sternknoten abhängt, wird hier keine quantitative Aussage darüber getroffen.

Für eine sKF wird die im Kapitel 3.1.4 berechnete WCTT garantiert. Auch bei Fehlverhalten anderer Funktionen und vorhersagbarem schadhaftem Verhalten muss diese Garantie eingehalten werden (vgl. Kapitel 3.1.6 und Kapitel 3.2.1). Der OEM als Anbieter der Dienstleistung „Kommunikation“ ist dafür verantwortlich, solange die verwendeten Ressourcen den angeforderten entsprechen.

Für eine sKF werden nun zwei generelle Annahmen festgelegt: 1) Eine sKF gilt als ausgefallen, sobald ein Datenpaket nicht innerhalb der gesetzten Deadline bei der Senke ankommt. 2) Die Deadline einer sKF ist so kritisch, dass kein erneutes Anfordern des Datenpaketes durch einen beliebigen Mechanismus möglich ist.

In einem paketorientierten Datennetz mit Zwischenpuffern, wie es für zukünftige Bordnetze vorgeschlagen wird, sind diese Vorgaben bei einem reaktiven System schwer lösbar. Um bei einem Defekt der Leitung keinen Rahmen zu verlieren, müssten einerseits alle Rahmen bis zum bestätigten Empfang des nächsten Sternknotens lokal für eine Neuaussendung vorgehalten werden, bei einem möglichen Ausfall eines Sternknoten sogar bis zum jeweils übernächsten Hop. Dies würde zu einer unverhältnismäßig hohen Last an Signalisierungsnachrichten führen. Weiter müsste für jeden Datenstrom und für jeden auftretenden Fehler persistent ein Ersatzpfad durch das Netz vorliegen, um den Rahmen noch innerhalb der vorgegebenen Deadline auszuliefern. Neben einer aufwendigen Generierung der Routinginformationen würden auch schnell die Speichergrenzen des Sternknoten gesprengt werden.

Für sKF wird deshalb eine gleichzeitige Übertragung über zwei kanten- und knotendisjunkte Wege vorgesehen. Dies wird in der Literatur auch als 1+1-Redundanz bezeichnet. Grundlage dafür ist ein Netz, welches zwischen jeder Quelle und jeder Senke einer sKF zwei mögliche Pfade besitzt. Dies ist zum Beispiel bei einer Ringtopologie vorhanden. Es ist Aufgabe der Netzplanung, wie in Kapitel 4 vorgestellt, dies sicherzustellen. Sind die physikalischen Wege vorhanden, werden entsprechende Routingtabellen berechnet und an die zuständigen Sternknoten verteilt.

Für das Routing entsprechender Rahmen können nun mehrerer Methoden verwendet werden. Einfach zu realisieren und auch im Demonstrator IT_Motive 2020 (Kapitel 6.4) eingesetzt ist das zielbasierte Routing mit Klonen der Datenrahmen beim Sender. Dabei werden vom Sender bereits zwei Frames mit gleichem Inhalt, aber unterschiedlicher Zieladresse erstellt. Beide Zieladressen weisen zwar auf den gleichen physikalischen Knoten, aber über redundante Pfade. Dieser Mechanismus benötigt nur geringe Anpassungen der Sternknoten, da das zielbasierte Routing bei Ethernet Standard ist.

Ein anderes Verfahren wäre das policybasierte Routing unter Einbeziehung des auf

der OSI-Schicht 2 vorhandenen VLAN Identifier (VID) aus dem IEEE 802.1q-Teil des Nachrichtenkopfes in Verbindung mit der Empfängeradresse. Der VID allein ist nicht ausreichend, da er auf 12 bit festgelegt ist und somit exklusiv der reservierten Werte nur 4094 Datenströme adressieren könnte. Dies könnte in Zukunft zu Limitierungen führen. Im Gegensatz zum vorherigen Ansatz kann die Duplizierung des Rahmens auch im Sternknoten an der Kante des Netzes vorgenommen werden, solange es in der Berechnung der WCTT eingeplant ist.

Durch die parallele Übertragung der sKF mit kanten- und knotendisjunkten Pfaden ist jeder Einfachfehler einer Leitung oder eines Sternknotens abgedeckt. Eine höherwertige 1+x Redundanz ist nicht vorgesehen, durch Modifikation der Netzplanung aber auch realisierbar. Eine Absicherung beliebiger Doppelfehler führt zu erhöhtem Aufwand sowie Kosten und sollte nur in begrenztem Maße eingesetzt werden, wenn die gesetzlichen Vorgaben (siehe Kapitel 3.2.1) nicht anders erreichbar sind.

3.2.3 Kritische Funktionen

Die zweite Klasse sind die kritischen Funktionen (KF). Auch hier handelt es sich um für die Funktionalität sicherheitskritischer Einrichtungen notwendige Übertragungen, welche aber nicht so strikte Bedingungen wie bei der Kategorie der sKF aufweisen.

Folgende Annahmen gelten für KF: 1) Eine KF gilt als ausgefallen, wenn über einen Zeitraum t_o keine Übertragungen durchgeführt werden können. 2) Der Verlust einzelner Nachrichten kann toleriert werden.

Gerade der Verlust von Nachrichten ist durchaus kritisch, falls Regelkreise durch springende Eingangswerte die Synchronisation verlieren. Für diese Fälle ist ein modellbasierter Fail-Safe-Mechanismus vorzuhalten, der den Regelkreis nachführt beziehungsweise über die nach dem Ausfall eintreffenden Eingangsdaten aufsynchronisiert.

Für die KF wird zur Fehlerdetektion und Fehlerbehandlung ein reaktiver Mechanismus vorgeschlagen, der aus Detektion, Verarbeitung und Reaktion besteht. In der Literatur wird auch der Begriff 1:1-Redundanz und *On-Demand*-Redundanz verwendet.

Zuerst muss der Fehler detektiert werden. Dies kann einerseits durch Betrachtung der physikalischen Verbindung auf Schicht 1 des OSI-Modells erfolgen, das die geringste Reaktionszeit von typischerweise < 1 ms aufweist. Eine weitere Methode ist die Verwendung von „Hello“-Paketen, also eine zyklische Reaktion der angeschlossenen Knoten, die mit einem Timer verknüpft die Aktivität des Knotens überwacht. Die Zykluszeit sowie der Timer müssen lang gewählt werden, um einen Verlust der Übertragungskapazität durch diese Nachrichten gering zu halten. Meist werden Werte von 100 ms $-$ 1 s verwendet. Daraus folgt auch eine sehr lange Reaktionszeit auf einen Ausfall. Die dritte Kategorie von Mechanismen überprüft das Verhalten des Datenstroms. Durch die Möglichkeit, einzelne Datenströme zu unterscheiden, und das Wissen über die Zykluszeit der Rahmen kann ein Abweichen bereits nach einer ver-

lorenen Nachricht festgestellt werden. Alle drei Mechanismen wurden prototypisch implementiert, weitere Informationen dazu sind in Kapitel 3.3.3 dargestellt.

Der nächste Schritt des reaktiven Mechanismus ist die Verarbeitung. Diese kann basierend auf zwei Eigenschaften grob eingeteilt werden: auf Bedarf oder vorgeplant, zentral oder dezentral.

Die erste Eigenschaft legt fest, ob der neue Pfad erst bei Ausfall berechnet wird oder ob er in einer Datenbank vorliegt und dort ausgelesen wird. Hier liegt ein Zielkonflikt zwischen notwendiger Rechenleistung und Speicherplatz vor. Bei n ECUs und vollvermachtem Anforderungsgraph sind anstatt $n \cdot (n-1)$ nun $2 \cdot n \cdot (n-1)$ Pfade vorzuhalten. Beim bedarfsgesteuerten Berechnen muss im Gegensatz dazu die vorhandene Topologie vollständig vorliegen, und es können nur einfache Routing-Algorithmen verwendet werden, da die Rechenleistung in eingebetteten Systemen naturgemäß beschränkt ist.

Bei der Verwendung eines zentralen Mechanismus liegt immer die Frage der Verfügbarkeit im Vordergrund. Das beinhaltet einerseits das Gerät auf dem der Mechanismus implementiert ist selbst, andererseits die nötigen Kommunikationswege dorthin. Im Gegensatz dazu hat ein zentrales System tendenziell ein vollständiges und konsistentes Abbild der Wirklichkeit vorliegen. Ein dezentrales beziehungsweise verteiltes System kann schneller Entscheidungen treffen, die im eigenen Zuständigkeitsbereich liegen. Bei Ausfall einer Kante oder eines Knotens muss die Konfigurationsänderung jedoch oft an einer anderen Stelle des Pfades weiter zur Quelle hin durchgeführt werden. Die zwischengelagerten Sternknoten können dabei einen unvorhersehbaren Einfluss auf die Reaktion des Systems haben.

Prototypisch wurde eine bedarfsgesteuerte Verarbeitung implementiert, die mit Hilfe des Dijkstra-Algorithmus [Dij59] den kürzesten möglichen Ersatzpfad berechnet. Sie ist in Software auf einem zentralen eingebetteten System realisiert. Auf den genauen Aufbau wird in Kapitel 3.3 eingegangen.

Die Reaktion auf einen Ausfall erfolgt wiederum in den Sternknoten. Von dem Verarbeitungssystem wird die entsprechende Änderung des Routings an die betroffenen Sternknoten gesendet, dort ausgewertet und das Routing-Subsystem wird entsprechend konfiguriert. Auch in diesem Schritt können vorgeplante Routen die Reaktionszeit verbessern, wenn die Ersatzpfade bereits auf den Sternknoten vorliegen und nur der erste Sternknoten im Pfad umkonfiguriert werden muss. Auch hier müssen aber wieder die Kosten durch den erhöhten Speicherverbrauch und den Aufwand beim Warten auf die Einträge beachtet werden.

3.2.4 Unkritische Funktionen

Die letzte definierte Klasse sind die unkritische Funktionen (uF). Die uF werden nach Eintritt in den Sternknoten vom Classifier (siehe Kapitel 3.1.3) direkt aussortiert und in ein eigenes Warteschlangensystem niedriger Priorität eingeordnet. Dafür wird das im IEEE 802.1q-Header vorhandene „Prioritäten“-Feld ausgewertet. Die Menge an

uF kann wiederum in sich unterschiedliche Prioritäten aufweisen und eigene Warteschlangensysteme verwenden.

Für den Verkehr von uF wird keine WCTT berechnet und auch keine Garantie derselben gegeben. Über Simulationen des Gesamtsystems mit dem in Kapitel 6.2 vorgestellten ITMsim kann aber eine statistische Aussage über die zu erwartenden Verzögerungen und damit der Dienstqualität angegeben werden. Dies ist jedoch nicht Fokus dieser Arbeit.

3.3 Sicherheits- und Konfigurationsmanagement

In Kapitel 3.2 wurden Methoden vorgestellt, um die Robustheit gegen Fehler im Kommunikationssystem zu erhöhen. Dieses Kapitel behandelt nun die Realisierung und Implementierung dieser Methoden. Zu diesem Zweck wurde ein Sicherheitsmanagement eingeführt, das als zentrale Instanz im Bordnetz die notwendigen Funktionen übernimmt. Als zweite Funktion wird vom Sicherheitsmanagement das Konfigurationsmanagement übernommen. Das Konfigurationsmanagement beinhaltet das Konfigurieren der einzelnen Sternknoten, vor allem im Hinblick auf ein dynamisches System mit wechselnden Kommunikationsquellen und -senken. Im Projekt IT_Motive 2020 wurde dieses Sicherheits- und Konfigurationsmanagement prototypisch auf eingebetteten Systemen realisiert, um die Funktionalität und das Verhalten zu evaluieren.

Zuerst wird der funktionale Aufbau des Systems beschrieben und die Verteilung auf die verwendete Hardware dargestellt. Anschließend werden die einzelnen Module mit ihrer Funktion und den Ein- und Ausgangsschnittstellen spezifiziert. Zum Schluss wird noch auf einen speziellen Modus, den „Teilbetrieb“, eingegangen.

3.3.1 Funktionale Architektur

Das Management-System wird in drei Schichten aufgeteilt: die Management Plane (MP), welche die Entscheidungen bezüglich des Verhaltens trifft und damit die Ausgangsdaten für die Konfiguration der Sternknoten liefert; die Control Plane (CP), welche die grundlegenden Informationen für Entscheidungen der MP liefert und die Konfiguration der Sternknoten übernimmt; und die Transport Plane (TP), die für die Behandlung der in den Sternknoten vermittelten Datenrahmen zuständig ist.

Abbildung 3.20 zeigt die generelle Aufteilung. Die TP und die CP sind auf jedem der verbauten intelligenten Sternknoten vorhanden. Die verteilten Instanzen der CP kommunizieren mit einer zentralen Einheit, welche die MP beinhaltet. Zentral bedeutet nicht unbedingt eine Einheit; aus Redundanzgründen können mehrere zentrale MP existieren, wobei zwar von allen synchron Entscheidungen getroffen werden, aber immer nur eine aktiv für die Konfiguration zuständig ist. Die Verwendung eines zentralen Systems wurde in Kapitel 3.2.3 motiviert.

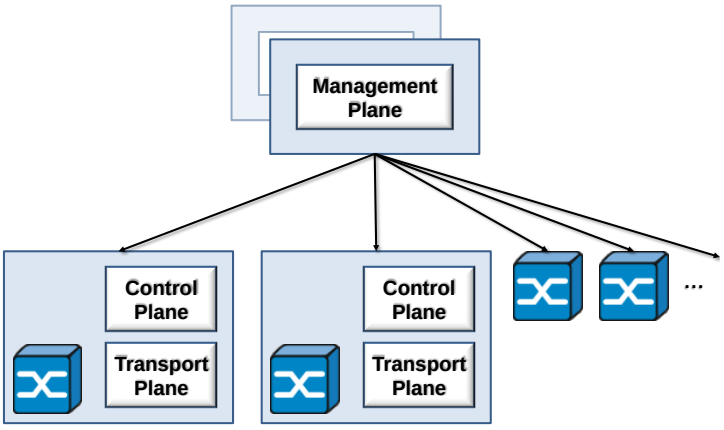


Abbildung 3.20: Übersicht Sicherheitsmanagement

In Abbildung 3.21 sind die einzelnen Blöcke innerhalb der erwähnten Schichten sowie ihr Zusammenwirken dargestellt. Die Darstellung enthält keine Information über örtliche Zusammengehörigkeit.

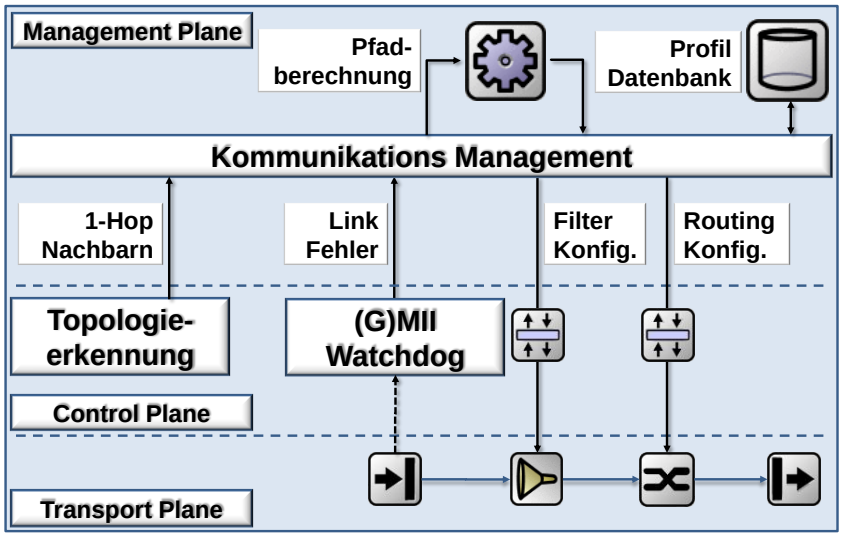


Abbildung 3.21: Blockschaltbild Sicherheitsmanagement

Die TP ist die unterste und grundlegende Schicht des Systems. Sie befindet sich auf jedem Sternknoten und ist für die Vermittlung der eingehenden Rahmen zuständig. Die TP muss also alle eingehenden Rahmen mit möglichst kurzer Verzögerung auf die entsprechende Ausgangsleitung vermitteln. Dies kann bei x Eingangsports und Gigabit Ethernet zu einer Last von $\approx x \cdot 2 \cdot 10^6 \frac{1}{s}$ Rahmen führen. Die TP muss also sehr performant sein und wird deshalb meist aus Hardwarebausteinen oder spezialisierten Prozessoren realisiert. Die Funktionen der TP sind in Abbildung 3.2 und Kapitel 3.1.3 skizziert. Sie beinhaltet neben dem Bestimmen der zutreffenden Ausgangswarteschlange durch Klassifizieren der Rahmen das Filtern der Datenströme, den Token Bucket und die Verarbeitung in den Warteschlangen selbst. Die Mechanismen in der TP werden durch die CP konfiguriert und gesteuert.

Die Hauptaufgabe der CP im beschriebenen System ist neben der Konfiguration der TP die Topologieerkennung sowie die Ausfallerkennung. Die Topologieerkennung generiert eine lokale Sicht auf die im Bordnetz vorhandene Topologie, um diese Information an die MP zu übermitteln, in der eine Gesamtsicht des Netzes aus diesen Subgraphen generiert wird. Die Ausfallerkennung arbeitet wie eine Art Serviceroutine, die bei einem durch die TP detektierten Ausfall die entsprechenden Informationen vorverarbeitet und an die MP zur Ausfallbehandlung sendet. Die notwendigen Konfigurationsinformationen für die TP werden von der MP empfangen, zu relevanten Konfigurationsänderungen verarbeitet und durchgeführt. Eine CP ist in diesem Ansatz auf jedem Sternknoten vorhanden.

Die MP existiert einmal in einer zentralen Instanz (abgesehen von Redundanzmethoden). Sie konsumiert die von der in den Sternknoten verteilten CP gesammelten Informationen und generiert daraufhin die Konfiguration der Sternknoten. Die Hauptaufgabe ist dabei, auf der Basis der lokalen Topologiesicht der einzelnen Sternknoten kombiniert mit der Verfügbarkeit der Verbindungen ein vollständiges Abbild des Bordnetzes zu generieren. Anschließend werden mit Hilfe der Routing-Algorithmen die physikalischen Pfade für die in den Ausführungsprofilen festgelegten Kommunikationsanforderungen generiert. Die Systemkonfiguration wird nun wiederum aufgeteilt in die jeweils für einen Sternknoten relevanten Einstellungen und an diesen übermittelt. Die Übermittlung der Konfiguration erfolgt zeit- und ereignisgesteuert, also umgehend nach einer Konfigurationsänderung und dann in festen zeitlichen Abständen zur Sicherstellung der Synchronität des Systems. Zur Konfiguration werden Nachrichten im XML-Format verwendet.

3.3.2 Topologieerkennung

Die in der CP implementierte Topologieerkennung dient zum Erkennen der angeschlossenen Geräte, um damit einen gültigen physikalischen Graph des Gesamtsystems zu generieren. Die Geräte werden vom Sternknoten proaktiv durch gesendete Anfragen erfasst.

In zyklischen Abständen von 1 Sekunde, um die Belastung des Netzes möglichst klein zu halten, werden auf jeden Port einzeln standardisierte Echo-Request-Nachrichten im Multicast-Verfahren an „All Systems on this Subnet“ gesendet. Alle angeschlossenen Geräte reagieren nun automatisch mit einer Echo Reply an den Absender, die somit die aktuelle Kommunikations-Adresse sowie die Hardware-Adresse des angeschlossenen Knotens enthält. Die Echo-Funktion, im Sprachgebrauch auch abgeleitet von dem verwendeten Programm „Ping“ genannt, ist als Grundfunktion in nahezu allen Kommunikationsstacks der Betriebssysteme und den *Intellectual Property Cores* zur Entwicklung integrierter Schaltkreise vorhanden. Die Nachrichten werden durch die CP im Sternknoten erstellt und dann in der Warteschlange mit niedrigster Priorität zur Aussendung abgelegt.

Die Antworten werden bis zu einem definierten Timeout gesammelt und dann durch die CP aggregiert. Die damit entstandene lokale Sicht auf die Topologie wird an die

MP zur Weiterverarbeitung übermittelt. Das entsprechende Protokoll wird in Kapitel 3.3.6 vorgestellt.

Eine weitere, jedoch hier nicht verwendete Methode wäre das Erkennen der angeschlossenen Geräte durch die Überwachung der Absendeadresse der eingehenden Pakete. Diese Funktion wird standardmäßig von Ethernet-Sternknoten im Büroumfeld verwendet. Dabei treten nun zwei Probleme auf: 1) Es können eine hohe Rechenlast und eine längere Verzögerung durch den zusätzlichen Verarbeitungsschritt bei den eintreffenden Rahmen auftreten. 2) Die Entfernung, also die Anzahl bereits durchlaufener Knoten, kann nicht festgestellt werden. Somit muss diese Information beim Verarbeiten der Daten in der MP zusätzlich extrahiert werden, um den für die Topologie relevanten ersten Sternknoten des Pfades zu erhalten.

In der MP werden zuerst anhand der empfangenen Topologieinformationen die Sternknoten ermittelt. Aus der gegenseitigen Detektierung mit dem Echo-Mechanismus wird der Aufbau des Backbones, also die Verbindung der Sternknoten untereinander, abgeleitet. Im letzten Schritt werden die Kanten zu den angeschlossenen Geräten eingefügt. Die Daten der Topologie werden in Form eines Graphen, des physikalischen Graphen, in der MP zur weiteren Verarbeitung vorgehalten.

3.3.3 Ausfallerkennung

Der zweite große Block in der CP ist die Ausfallerkennung. Sie ist einerseits Teil der Topologieerkennung, andererseits im Modul „(G)MII Watchdog“, Kurzform für Gigabit Media Independent Interface (GMII)-Überwachung, enthalten. Auch die TP ist an der Ausfallerkennung beteiligt. Aufgabe der Ausfallerkennung ist es, defekte Verbindungen zu Knoten, dies beinhaltet Softwarefehler wie auch Hardwarefehler, zu erkennen und dieses Wissen an die MP zu übermitteln.

Der einfachste Mechanismus zur Detektion von Ausfällen ist die Topologieerkennung aus Kapitel 3.3.2. Antwortet ein Knoten auf die proaktiven Anfragen nicht mehr, wird davon ausgegangen, dass er defekt ist, und es werden die notwendigen Maßnahmen eingeleitet. Dieser Mechanismus hat eine sehr lange Reaktionszeit, da erst eine Echo-Nachricht gesendet und bis zur Zeitüberschreitung der Antwort gewartet werden muss. Bei der oben erwähnten Zykluszeit von 1 Sekunde kann dies somit bis zu 2 Sekunden dauern.

Ein sehr schneller Mechanismus ist die GMII-Überwachung. Das GMII ist bei Ethernet die Schnittstelle zwischen der Medienzugriffssteuerung und der physikalischen Schicht. Über diese Schnittstelle kann der Zustand des angeschlossenen Mediums abgefragt werden. Geht die physikalische Verbindung des Mediums verloren, zum Beispiel bei Durchtrennung der Leitung oder durch Ausfall des angeschlossenen Knotens, kann dies unmittelbar aus der GMII ausgelesen werden. Dafür wird über den seriellen Management-Bus des GMII, der mit 2,5 MHz getaktet ist, das „Link Status“-Register ausgelesen. Rechnerisch dauert dies damit > 26 µs, wenn man den Management-Bus nur damit belegt. Je nach Implementierung des *Physical Coding Sublayer* (PCS) des Ethernet dauert die Erkennung des Synchronisationsverlustes

jedoch eine definierte Zeitüberschreitung von 1,6 ms − 10 ms. Die mit diesem Mechanismus erreichbare Reaktionsgeschwindigkeit liegt bei Verwendung von Standard-Komponenten bei ≈ 2 ms.

Der dritte Mechanismus verursacht den höchsten Ressourcenbedarf, kann dafür aber mehr Fehler mit einer hohen Reaktionsgeschwindigkeit detektieren. Da für die Berechnung der WCTT ein minimaler Abstand $T_{A\,\min}$ (siehe Kapitel 3.1.4) zwischen den auftretenden Rahmen eines Datenstroms bekannt ist, kann eine Übertretung der zugelassenen Rate detektiert werden. Dafür wird ein Token Bucket für jeden Datenstrom getrennt eingesetzt. Tritt ein weiterer Rahmen am Sternknoten ein, bevor der Timer einen neuen Token generiert hat, also der Token Bucket leer ist, so muss dieser Rahmen aufgehalten oder verworfen werden. Spezifiziert man nun analog dazu einen maximalen Abstand $T_{A\,\max} = x \cdot T_{A\,\min}$ als Vielfaches des minimalen Abstands, kann anhand des Füllstandes des Token Bucket die Überschreitung dieser Grenze erkannt werden. Die einzige Modifikation gegenüber dem regulären Token Bucket ist, das jeder eintreffende Rahmen den Stand des Token Bucket nicht um 1 dekrementiert, sondern diesen $= 0$ setzt. Theoretisch kann damit eine Reaktionszeit von $\approx 2 \cdot T_{A\,\min}$ erreicht werden. Um diese im praktischen Fall zu evaluieren, wurde der Mechanismus im Sternknoten prototypisch implementiert. Das Ergebnis der Auswertung ist in Abbildung 3.22 dargestellt.

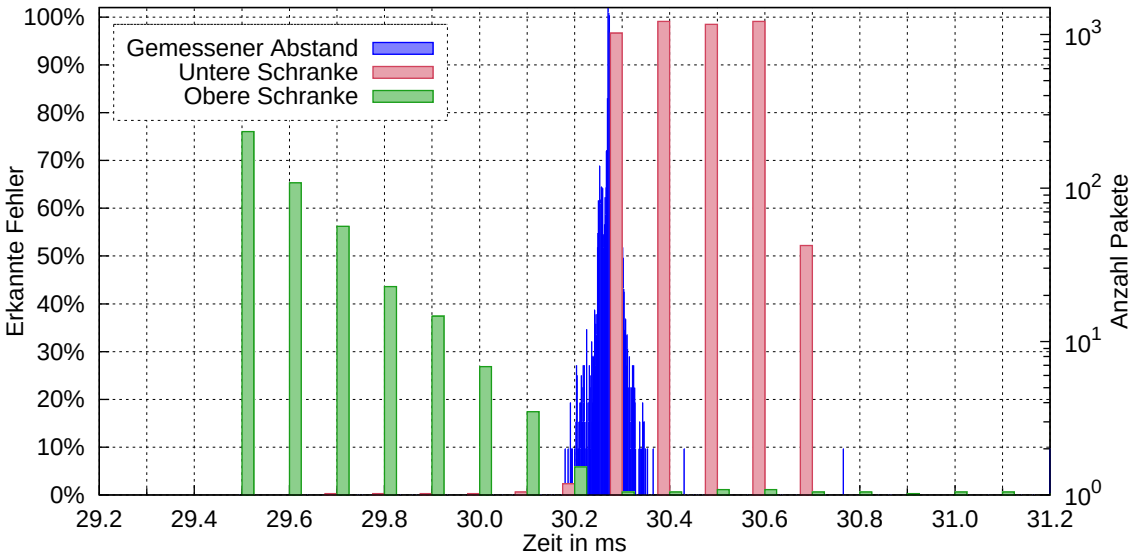


Abbildung 3.22: Fehlererkennung mit Token Bucket

Die blaue Verteilung markiert den gemessenen Abstand zwischen zwei Rahmen, die Anzahl ist an der rechten Achse angeschrieben. Durch Ungenauigkeiten beim Aussenden der Rahmen kommt es zu einer gewissen statistischen Verteilung um den gesetzten Wert. Nun wird die obere und die untere Schranke in 0,1 ms-Schritten verändert und jeweils die Anzahl der erkannten Fehler in % der aufgetretenen Unterbeziehungsweise Überschreitungen an der linken Achse aufgetragen.

Anhand der roten Balken ist das Setzen der unteren Schranke abzulesen. Diese

Schranke wird durchbrochen, wenn der minimale Abstand zwischen zwei Rahmen nicht eingehalten wird. Der Wert der x-Achse ist somit das gesetzte $T_{A\,\min}$. In dem auf Software basierenden Prototyp kann eine Unterschreitung von $T_{A\,\min}$ mit an 100% reichender Detektionswahrscheinlichkeit festgestellt und an die CP gemeldet werden.

Die grünen Balken repräsentieren ein gesetztes $T_{A\,\max}$. Aus dem Diagramm ist zu erkennen, dass eine Überschreitung der gesetzten Grenze erkannt wird, aber nur für einen Teil der betroffenen Rahmen. Im Gegensatz zum vorherigen Fall, wo beim Eintreffen des Rahmens der Stand des Token Bucket überprüft wird und damit eine Aktion ausgelöst werden kann, muss hier in regelmäßigen Abständen die Anzahl der Token aktiv überprüft werden. Je nach Einstellung des Überwachungszyklus kann ein neuer eintreffender Rahmen bereits die Token wieder entfernt haben. Aus diesem Grund ergibt sich eine gewisse Gleichverteilung der Erkennungswahrscheinlichkeit, je weiter die gesetzte Grenze $T_{A\,\max}$ von dem Abstand der eintreffenden Frames entfernt ist. Geht man von einem binären Fehlerbild aus, also entweder das System funktioniert oder es ist vollständig gestört, stellt diese Eigenschaft kein Problem dar. Müssen jedoch Fehler beim Scheduling der Daten festgestellt werden, die in einem ungenauen Timing resultieren, muss ein Abwägung zwischen dem Ressourcenbedarf durch Überwachung der Token-Bucket-Stände und der Detektionswahrscheinlichkeit durchgeführt werden.

Wird ein Fehler mit einem dieser drei Mechanismen erkannt, so wird diese Information von der CP vorverarbeitet und an die MP gesendet, um dort die notwendigen Schritte zur Umkonfiguration des Netzes vorzunehmen. Wie in Kapitel 3.2.3 diskutiert, werden nun entweder neue Pfade berechnet oder vorberechnete Ersatzpfade ausgewählt.

3.3.4 Ausführungsprofile

Die Ausführungsprofile sind in der Profildatenbank gespeichert und repräsentieren den Anforderungsgraphen im Managementsystem. Dort werden einerseits Informationen zu den angeforderten Kommunikationsressourcen, andererseits konkrete Routen der Datenströme durch das Bordnetz gespeichert.

Die Daten werden verwendet, um die in den vorherigen Kapiteln beschriebenen Mechanismen zu konfigurieren. Dies sind unter anderem die entsprechenden Warteschlangen für die (sicherheits-)kritischen Datenströme sowie der Mechanismus zur Klassifizierung der Rahmen, der Filter an den Eingangsports, um schadhaften Verkehr zu filtern, sowie die Konfiguration der mit den Token Bucket verbundenen Mechanismen.

Die Ausführungsprofile dienen auch als Grundlage zur Generierung der Routingtabellen für die einzelnen Sternknoten, da dort Informationen über die gewünschten Quellen sowie Senken hinterlegt sind. Je nach eingesetzter Methode sind dort bereits vollständige Pfadinformationen enthalten, dies gilt auch für die disjunkten Pfade bei sicherheitskritischen Funktionen (Kapitel 3.2.2) sowie die Ersatzpfade der kritischen

Funktionen (Kapitel 3.2.3), die nur noch auf Gültigkeit gegenüber der erkannten physikalischen Topologie geprüft werden müssen. Das Wissen über den Datenstrom kann auch, wie bei den unkritischen Funktionen, zur Online-Berechnung des Pfades verwendet werden (Kapitel 3.2.4).

Handelt es sich um sehr komplexe Systeme und sind nur begrenzt Rechenkapazitäten für das Managementsystem vorhanden, kann die Profildatenbank auch dafür verwendet werden, gültige Systemabbildungen zu speichern. Damit ist gemeint, dass für die Fälle von auftretenden Einzelfehlern entsprechende Konfigurationen, wie sie nach einer Ausfallbehandlung durch das Sicherheitsmanagement ausgeführt würden, bereits berechnet vorliegen. So kann bei einem klassifizierten Einzelfehler in kürzester Zeit auf eine gültige Konfiguration umgeschaltet werden. Bei Einzelfehlern ist die Menge der möglichen Kombinationen noch überschaubar und damit der benötigte Speicherbereich begrenzt.

Die Ausführungsprofile werden im Zuge der WCTT-Berechnung generiert, können jedoch über den Lebenszyklus des Bordnetzes hin angepasst werden.

3.3.5 Pfadberechnung

Die Pfadberechnung ist auch in der MP angesiedelt und hat die Aufgabe, die angeforderten Kommunikationsverbindungen auf das physikalisch vorhandene Bordnetz abzubilden. Da das Managementsystem bei einer Realisierung innerhalb des Bordnetzes auf eingebetteten Systemen in der Rechenleistung begrenzt ist, können nur relativ triviale Routing-Algorithmen zur Pfadberechnung verwendet werden. In der prototypischen Implementierung wurde ein Intel XScale-IXP425 System on a Chip (SoC) mit 133 MHz eingesetzt.

Verwendet wurde im MP der „Dijkstra“-Algorithmus [Dij59] für die Berechnung des kürzesten Pfades durch das Netz sowie der „K-disjoint shortest path“-Algorithmus [Bha99], der die zwei kürzesten kanten- und knotendisjunkten Pfade für die sicherheitskritischen Funktionen (Kapitel 3.2.2) generiert. Das Kantengewicht ist auf 1 gesetzt, es werden also die Pfade mit den wenigsten durchlaufenden Sternknoten gesucht. Diese Methode ist bei dem hier gewählten ereignisbasierten Ansatz sehr gut, wie in Kapitel 3.1.5 belegt wurde.

Berechnet wurden die Routing-Algorithmen mit Hilfe der Graphentheorie. Dafür werden das physikalische Netz, generiert durch die Topologieerkennung (Kapitel 3.3.2), und die Kommunikationsanforderungen (Kapitel 3.3.4) in Form eines physikalischen Graphen und eines Anforderungsgraphen abgebildet. Auf der Basis dieser Graphen werden die Routing-Algorithmen ausgeführt.

3.3.6 Konfigurationsprotokoll

Das Konfigurationsprotokoll dient zum Austausch der dezentral in der CP generierten Informationen mit der MP und im Gegenzug der Übertragung der Konfigura-

tionsparameter von der MP an die Sternknoten. Verwendet wird ein eigenes XML-basiertes Nachrichtenformat, das über IP/UDP übertragen wird.

XML wurde verwendet, da die übertragenen Daten aus hierarchisch strukturierten Datensätzen bestehen. Gesendet werden von der CP zur MP hauptsächlich die Information über den Sternknoten selbst, die dort existierenden Ports, die erkannten Geräte an den Ports und der Status der Verbindung zu diesen Geräten. Von der MP zur CP werden die Informationen über die Routen jedes Ports und die darüber laufenden Datenströme für die Filter übertragen. Dieser hierarchische Aufbau lässt sich gut auf das XML-Format abbilden.

Für den prototypischen Aufbau hat eine Klartextübertragung den Vorteil, dass die Nachrichten auch direkt gespeichert und schnell ausgewertet werden können, um eventuelle Fehler im Ablauf aufzuspüren. Nachteilig ist der erhöhte Ressourcenbedarf bei der Übertragung der in 8 bit kodierten Daten sowie bei der Verarbeitung der Nachrichten wegen der großen Anzahl an notwendigen Zeichenketten-Vergleichen.

3.3.7 Teilbetrieb

In diesem Kapitel soll kurz auf einen Mechanismus eingegangen werden, der für bestimmte Bordnetze wichtig ist. Der Teilbetrieb ermöglicht ein partielles Aktivieren des Bordnetzes. Ein denkbares Szenario ist hier das Bordnetz eines Fahrzeugs, das in regelmäßigen Abständen Informationen von einem zentralen Server abfragt. Gerade im Stand bei nicht laufender Stromversorgung ist ein vollständiger Betrieb des Bordnetzes aus in der Batterie gespeicherter Energie nicht möglich. Wichtig ist deshalb, dass im verwendeten Bordnetz eine Möglichkeit zur Teilinbetriebnahme vorhanden ist.

Das Konfigurationsmanagement hat die Möglichkeit, über *Wake-on-LAN* (WOL) einzelne Knoten mit Hilfe spezieller Datenpakete zu starten und über eine Management-Schnittstelle diese Knoten nach Erledigung der Aufgaben wieder zu stoppen. Die größte Herausforderung besteht jedoch daran, das Netz in der Form zu planen, dass möglichst wenige Sternknoten beteiligt und dabei vom zentralen Managementsystem für die WOL-Nachrichten erreichbar sind. Auch tritt durch das schrittweise Aktivieren der Sternknoten entlang eines Pfades eine kumulierte höhere Aufstartverzögerung auf, bis die Funktion ihre Daten übertragen kann.

In der prototypischen Implementierung wurde für die ECUs der WOL-Mechanismus implementiert und verwendet, da die Hardware der Sternknoten diese Funktionalität nicht unterstützte. Um die Energy-Star-Richtlinien bei Desktop-Rechnern zu erreichen, darf WOL maximal 0,7 W Verbrauch ausmachen, weshalb dies als typischer Wert von Standardhardware angenommen wird. Bei Bordnetzen in mobilen Szenarien muss diese Zahl jedoch deutlich unter 0,01 W gesenkt werden, um geforderte Begrenzungen des Leerlaufenergieverbrauchs einzuhalten. Alternativen stellen in diesem Bereich die Verwendung eines separaten Steuerungsbusses oder eines auf Trägerfrequenzanlagen in der Spannungsversorgung basierenden Systems dar.

3.4 Zusammenfassung

Da Ethernet als zukünftige Technologie für Kommunikation in Bordnetzen identifiziert wurde, musste untersucht werden, wie weit es für den Einsatz in diesen Systemen verwendbar ist und welche Modifikationen notwendig sind. Grundlage ist ein Ethernet nach dem Standard IEEE 802.3, um die größtmögliche Kompatibilität mit der vorhandenen Hardware und den bisherigen Entwicklungen zu erhalten.

Grundsätzlich gibt es zwei Kategorien für den Medienzugriff beim Einsatz von Ethernet für Realzeitkommunikation: ereignisbasiert und zeitschlitzbasiert. Der Standard verwendet einen ereignisbasierten Medienzugriff mit Warteschlangen, während der überwiegende Teil der kommerziellen Modifikationen von Ethernet für die Automatisierungstechnik auf festen Zeitschlitzen basiert. In dem hier vorgestellten Ansatz wird ein dem IEEE 802.3 entsprechender, ereignisbasierter Medienzugriff verwendet, um die OSI-Schicht 2, die Data-Link-Schicht, standardkonform zu belassen. Um aber die für die Realzeitkommunikation wichtigen Randbedingungen einzuhalten, vor allem die Garantie einer maximalen Übermittlungsverzögerung, mussten einige Änderungen an den verwendeten Sternknoten vorgenommen werden.

Dafür wurden die Sternknoten um ein hierarchisches Warteschlangensystem, um einen Token-Bucket-Filter sowie um eine Preemption erweitert, und die Auswirkungen auf den Datenverkehr wurde untersucht. Zum Einsatz kamen unterschiedliche Warteschlangen für die realzeitkritischen Datenströme, die quantitativ und qualitativ verglichen wurden. Mit Hilfe dieser Modifikationen kann nun eine Übertragungszeit-Analyse durchgeführt werden, um die Worst Case Transmission Time (WCTT) zu berechnen. Die Korrektheit der Berechnung wurde durch Simulationen mit verschiedenen Szenarien überprüft.

Die für die Anforderungen aus dem Referenznetz (siehe Kapitel A.2) ideale Lösung für die Sternknoten, basierend auf den gewonnenen Ergebnissen, besteht aus einem hierarchischen Warteschlangensystem, bei dem unkritische Datenströme anhand der Prioritätsinformation im Nachrichtenkopf direkt in ein System mit niedriger Priorität einsortiert werden und die zeitkritischen Rahmen in einer Reihe von vom Eingangsport abhängigen Ausgangswarteschlangen, die im Round-Robin-Verfahren bedient werden, landen. Ein vorgelagerter Token-Bucket-Filter überwacht die Einhaltung der spezifizierten Grenzen eines jeden Datenstromes. Ein dynamischer Preemptionsmechanismus verringert die in die Berechnung der WCTT eingehenden Komponenten $T_{W1_{SP}}$, ohne dabei die Weiterleitung Datenverkehrs niedriger Priorität übermäßig zu stören.

Im zweiten Teil wurde ein Sicherheits- und Konfigurationsmanagement entwickelt, das einerseits die verwendeten Sternknoten konfiguriert, andererseits bei einem Ausfall von Kommunikationsressourcen entsprechende Fehlerkorrekturmaßnahmen durchführt, um die Funktionalität des Netzes aufrechtzuerhalten. Dabei wurden verschiedene Verkehrsklassen, unterteilt nach ihre Kritikalität, mit unterschiedlichen Mechanismen zur Ausfallsicherung eingeführt. Das Managementsystem ist dabei zentralisiert aufgebaut und verwendet ein auf XML basierendes Konfigurations-

protokoll. In einer prototypischen Implementierung wurde die Funktionalität des Systems untersucht und die Reaktionszeiten auf Fehler wurden ausgewertet.

4 Planung von Bordnetzen

Neben der reinen technischen Realisierbarkeit eines Bordnetzes für verteilte heterogene Subsysteme ist vor allem der Bereich der Netzplanung und der Dimensionierung eine hohe Herausforderung. Die Electronic Control Units (ECUs) können nun nicht mehr einfach mit einem Bus hintereinander, also mit einer so genannten „Daisy-Chain“, vernetzt werden. Nach der in Kapitel 3 vorgeschlagenen Topologie müssen Sternknoten an passenden Stellen im Fahrzeug verbaut und betrieben und alle Geräte mit ihnen verbunden werden. Dieser zusätzliche Einbauplatz und Aufwand sowie die benötigten Kabel müssen minimiert werden, um eine konkurrenzfähige Lösung für zukünftige Architekturen zu erreichen.

In Kapitel 4.2 werden neue Metriken zur Quantifizierung der Aufwände einer möglichen Lösung vorgestellt und in Kapitel 4.3 ein Optimierungsalgorithmus auf der Basis des Genetischer Algorithmus (GA) für verschiedene Zielarchitekturen eingeführt.

4.1 Integration der Sternknoten

Diese grundlegende Änderung der Topologie von einer Linien- zu einer Sternstruktur bietet zwar Leistungs- und Sicherheitsvorteile, jedoch auch höhere Kosten und einen größeren Aufwand. Zusätzliche Geräte, die keine direkte Kundenfunktion leisten, sondern Infrastrukturdienste bereitstellen, müssen in der Architektur vorgesehen werden.

Da zusätzliche Geräte in eingebetteten Systemen grundsätzlich zu einem höheren Aufwand führen, müssen die Sternknoten mit den ECUs zusammen verbaut werden. Der in Abbildung 4.1 gezeigte Aufbau wurde deswegen für diese Netze entwickelt und als Realisierung vorgeschlagen.

Der Sternknoten befindet sich dabei innerhalb des Gehäuses einer ECU und verwendet eine gemeinsame Stromversorgung. Am Gehäuse befinden sich neben den zur ECU gehörenden Sensor- und Aktorleitungen Anschlüsse für weitere ECU, die selbst keinen eigenen Sternknoten benötigen. Auf der Platine des Sternknotens sind für diese Geräte so genannte *Ethernet Magnetics*, also Leitungstreiber, verbaut.

Das eigentliche Steuergerät in dem Gehäuse wird über ein Gigabit Media Independent Interface (GMII) [WWK+99] mit dem Sternknoten verbunden, was noch einmal erheblich zur Bauteilverringerung beiträgt. Die GMII-Verbindung kann als Flachbandkabel oder in Form von Leiterbahnen realisiert werden, und sie kann direkt zwei Ethernet-Media-Access-Control (MAC)-Steuerbausteine miteinander verbinden.

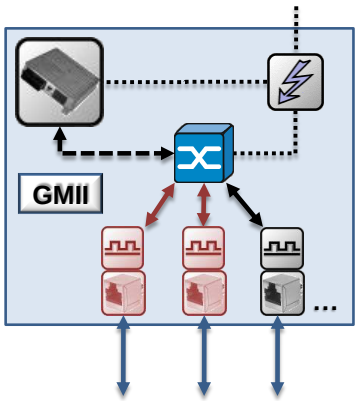
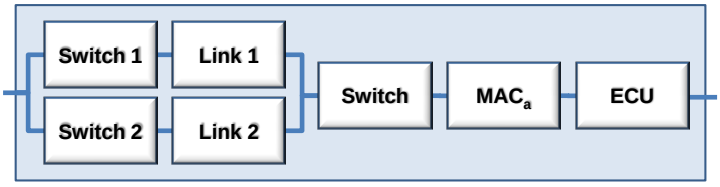


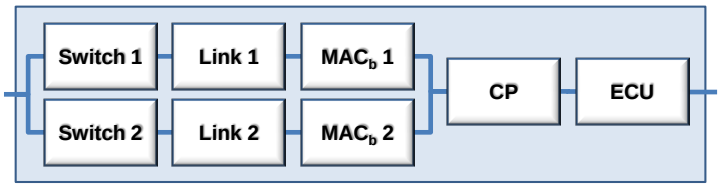
Abbildung 4.1: Blockschaltbild Steuergerät

Zur weiteren Integration wird vorgeschlagen, die Ethernet-MAC-Steuerbausteine als System on a Chip (SoC) auszuführen, das heißt in den Prozessorbaustein zu integrieren. Derartige Lösungen gibt es bereits von mehreren Herstellen, wie zum Beispiel die Freescale PowerQUICC-Serie.

Im Produktentstehungsprozess und in der Fertigung können die beiden Bauteile, Sternknoten und ECU, von getrennten Zulieferern gefertigt und auch separat in Hardware in the Loop (HIL)-Testplätzen abgesichert, jedoch im Werk als eine Einheit verbaut und verkabelt werden. Die Reduktion der Bauteile durch die direkte Anbindung des ersten Gerätes und die gemeinsame Stromversorgung tragen zur weiteren Kostenreduktion bei.



(a) Vorgeschlagene Lösung



(b) Alternative Lösung

Abbildung 4.2: Zuverlässigkeits-Ersatzschaltbild

Zur ausfallsicheren Kommunikation (vgl. Kapitel 3.2) über zwei redundante Pfade werden zwei Ports des in die ECU integrierten Sternknoten verwendet. Die alternative Lösung sind zwei getrennte Kabel zu unterschiedlichen, nicht im selben Gehäuse verbauten Sternknoten. Beide Ersatzschaltbilder sind in Abbildung 4.2 dargestellt. Der *Communication Processor* (CP) entscheidet bei der alternativen Lösung über den ver-

wendeten Pfad. Auf Grund des höheren Verkabelungsaufwands ist die erste Lösung trotz geringerer Verfügbarkeit vorzuziehen. In diesem Fall muss mehr Aufwand hinsichtlich der Robustheit der Einzelkomponenten aufgewendet werden, um eine vergleichbare Zuverlässigkeit zu erreichen. Die Arbeiten von Brand [ABBW05, BOW03] im Rahmen der Verlässlichkeitsbetrachtung des auf Ethernet aufbauenden Übertragungsprotokolls IEC 61850 sowie die Arbeiten an den Schweizer Engineering Laboratories [SD07, CTS08] bieten einen guten Ausgangspunkt zur Abschätzung der Verfügbarkeit.

4.2 Kostenabschätzung

Das Kapitel Kostenabschätzung beschäftigt sich damit, den Aufwand greifbar zu machen, der durch das Bereitstellen der Kommunikationsinfrastruktur notwendig geworden ist. Es werden Metriken eingeführt und es wird die Anpassung an reale Verhältnisse über die Parametrisierung gezeigt.

4.2.1 Metriken

Zur Abschätzung der Qualität einer möglichen Lösung und deren Optimierung wird eine zu den erwarteten Kosten repräsentative Metrik benötigt, die innerhalb eines Algorithmus berechenbar ist. Wie in den vorhergehenden Kapiteln beschrieben, sind Stückkosten ausschlaggebend für Designentscheidungen im heutigen Produktentstehungsprozess. Diese Metrik ist aber nicht weitreichend genug, da sie unter anderem den Aufwand im Werk und die Besonderheiten des Fahrzeuges nicht widerspiegelt. Aus diesem Grund wurde eine Formel für die Virtuelle Kostenfunktion (VK) eingeführt, die aus dem Verkabelungsaufwand, dem Bauraum und dem Energieverbrauch zusammengesetzt wird. „Virtuell“ ist sie, weil nicht direkt monetäre Kosten im Sinne von Geld abdeckt werden, sondern sich vielmehr ein auf diese Zahl abbildbarer Wert ergibt.

Parameter	Wert	Parameter	Wert
l	10.0	b	8.00
c	2.00	e	7.00

Tabelle 4.1: Standardwerte der Kostenfunktion

Die Komponenten der VK werden mit den Parametern l, c, b und e gewichtet. Falls nicht angegeben, gelten in den nächsten Kapiteln die in Tabelle 4.1 festgesetzten Standardwerte.

Verkabelung

Zuerst muss der höhere Aufwand durch die Verkabelung der einzelnen ECUs untereinander betrachtet werden. Da es um die Kosten zur Bereitstellung der Kommunikationsinfrastruktur geht, werden auch nur diese Aktorleitungen und keine Stromversorgungs- oder Sensorleitungen betrachtet.

Die Positionen der ECU selbst werden dabei als gegeben angenommen. Dies ist ein praxisnaher Ansatz, weil die ECU möglichst an Orten verbaut werden, wo die notwendigen Aus- und Eingabedaten generiert beziehungsweise verarbeitet werden können. Ein Repositionieren derselben allein zur Optimierung der Kommunikationsinfrastruktur ist nur bei einzelnen Geräten ohne direkte Anbindung sinnvoll und wird deshalb in diesem Prozess nicht weiter betrachtet. Die in Abbildung 4.3 gezeigte Architektur wurde aus dem Referenznetz in Kapitel A.2 extrahiert. Die Abbildung zeigt die ECUs als schwarze Vierecke an dem physikalischen Verbauort im Fahrzeug. Auf Grund der teilweisen Zentralisierung der ECUs auf einige Geräteträger bildet sich an einigen Stellen im Fahrzeug eine hohe Dichte an Geräten aus, wie zum Beispiel unterhalb des Lenkrads, in den Kofferraumseiten und im Motorraum. Trotzdem ist eine große Anzahl weiterhin über das gesamte System verteilt.

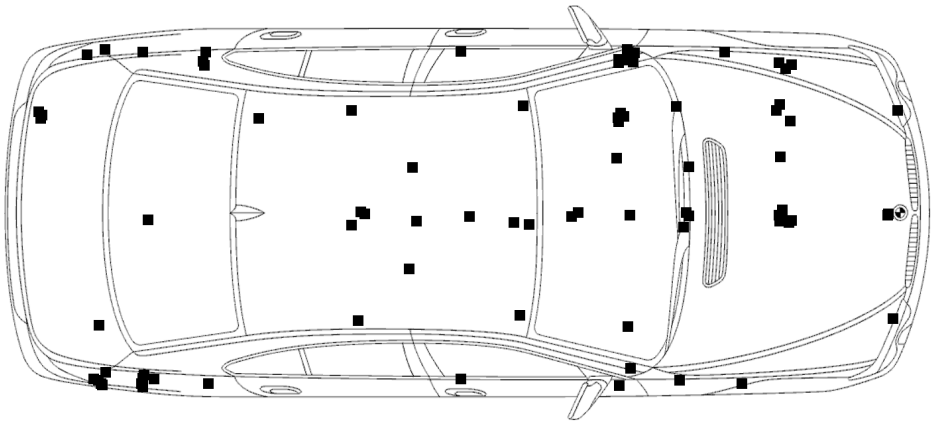


Abbildung 4.3: Positionen der Steuergeräte

Zur Vereinfachung der Positionsdatengenerierung gerade in einer frühen Phase des Produktentstehungsprozess (PEP) werden nicht exakte Ortsinformationen verwendet, sondern die Position der ECU anhand eine Koordinate in einem dreidimensionalen Gitter beschrieben. Dafür wird das Fahrzeug in einzelne Kuben zerschnitten, mit Nullpunkt am Bodenblech im Fahrzeugschwerpunkt. Für einen Personenkraftwagen hat sich auf Grund der Abmessungen eine Kantenlänge von 3,0 dm als gut geeignet erwiesen, da somit die Position einzelner Geräteträger ausreichend aufgelöst werden kann. Jeder Kubus hat somit einen Inhalt von 27 dm^3. Befindet sich ein Gerät an irgendeinem Punkt innerhalb des Kubus, wird ihm diese Position in $x,y,z \in \mathbb{Z}$ zugewiesen. Das *ICM* in Abbildung 4.4 hat zum Beispiel die Positionsinformation $(4,2,1)$.

Für Geräte, die sich im gleichen Kubus befinden, wird ein kugelförmig approximierter mittlerer Abstand von 1,5 dm gesetzt. Für jeden Schritt von einem zum anderen

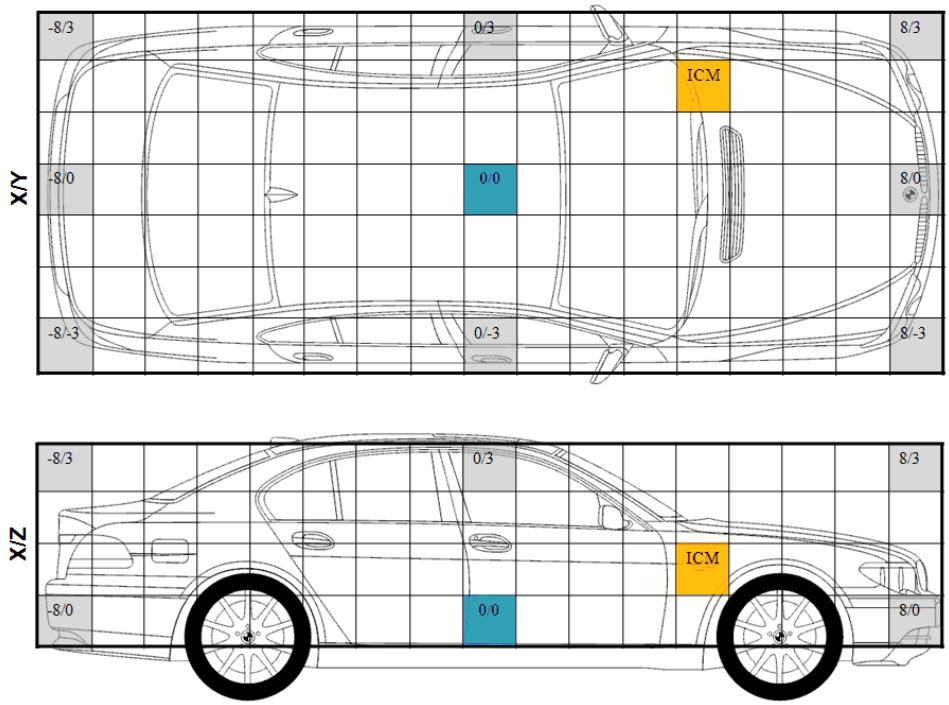


Abbildung 4.4: Cube Grid

Objekt in x,y,z-Richtung werden 3,0 dm hinzugefügt. Es wird sozusagen die kürzeste Strecke über die Kanten der Kuben zwischen zwei Geräten gewählt und die Anzahl der durchlaufenen Kanten N_C aufsummiert. Dies ergibt bildlich eine Treppenkurve und keine direkte Verbindung, was nahe genug an den physikalisch möglichen Kabelwegen in Fahrzeugen liegt. Somit wird die genährte Kabellänge zwischen zwei Objekten mit der Gleichung (4.1) berechnet.

Die Komponente „Verkabelungsaufwand“ L der VK ist die gewichtete Summe über alle Kabelverbindungen im System. Sie besteht aus der mit l proportional gewichteten Länge in m und einem konstanten Faktor c für die Steckverbindungen, vgl. Gleichung (4.2).

$$\text{length}_{ij} = 150\ \text{mm} + 300\ \text{mm} * N_C \tag{4.1}$$

$$L = \sum_{\text{Switch}_i} \sum_{\text{Entity}_j} \left(l \cdot \frac{\text{length}_{ij}}{\text{m}} \right) + (2 \cdot c) \tag{4.2}$$

$$\text{für } ij = \text{linked}$$

Eine proportionale Gewichtung wurde gewählt, da Gewicht und Materialmenge linear mit der Länge steigen. Auch ist beim Verbau im Fahrzeug die Dicke, also die Kreisfläche des Kabelbündels, ausschlaggebend, wenn man Biegeradieneinschränkungen und Durchführungen im Blech betrachtet. Geht man von einer konstanten Gesamtlänge des Kabelbaums aus, steigt die mittlere Anzahl parallel liegender Kabel linear

zur akkumulierten Länge der Einzelverbindungen. Die in einem runden Bündel liegenden Einzelstrippen erhöhen die Kreisfläche wiederum höchstens linear. Somit ist dies eine gute, jedoch pessimistische Schätzung für die erwarteten Kosten der Verkabelung.

Bauraum

Der Bauraum ist eine weitere Komponente der Kostenfunktion. Er spiegelt den zusätzlichen Platzbedarf durch die Sternknoten, siehe Kapitel 4.1, wider und wird grundsätzlich über die benötigte Platinenfläche quantifiziert. Zu diesem Zweck wurden die Datenblätter größerer Hersteller von Switching-Bausteinen wie Micrel, Marvell und Broadcom ausgewertet sowie Geräte aus dem Büro- und Industriebereich vermessen.

Die Auswertungen gaben einen linearen Verlauf im Bereich von 4-8 Ports, der für den Einsatz in verteilten Systemen ausreichend ist. Die Fläche ist abhängig von einer konstanten und einer portabhängigen Komponente. Die Konstante besteht hauptsächlich aus der zentralen *Switching Fabric*, also dem Hauptprozessor und dessen Stromversorgung, die portabhängige aus den bereits erwähnten Leitungstreibern mit Versorgung. Das Mittel der ausgewerteten Daten gibt einen konstanten Anteil von 10 cm^2 und einen zur Anzahl der Ports N_P proportionalen von 5 cm^2 pro Port, was in der Gleichung (4.3) zusammengefasst ist. Die hier beschriebenen Zahlen sind exemplarisch, die Mechanismen funktionieren auch mit anderen Werten in der Gleichung.

Die berechnete Platinenfläche wird in der Gleichung (4.4) auf dm^2 normiert weiterverwendet. Mit dem Faktor b gewichtet dient sie als Argument einer Exponentialfunktion auf der Basis e.

$$\text{size}_i = 1.000\ \text{mm}^2 + 500\ \text{mm}^2 * N_P \tag{4.3}$$

$$B = \sum_{\text{Switch}_i} \exp\left(b \cdot \frac{\text{size}_i}{\text{dm}^2}\right) \tag{4.4}$$

Die Exponentialfunktion wurde gewählt, weil es viel schwieriger ist, große Objekte in einem festgelegten Raum unterzubringen und es in einer größeren Verschwendung von Raum resultiert, als wenn dies in vielen kleineren geschieht. Zum Beispiel kann man auf einer $10x10$ großen Fläche 100 $1x1$ große Quadrate unterbringen, vier $4x4$ große Quadrate, aber nur ein $6x6$ großes Quadrat. Auf ein Fahrzeug übertragen bedeutet dies, dass es unmöglich ist, alle ECUs an einem Platz im Fahrzeug unterzubringen, da die Kosten unendlich wären. Geräte der Größe einer Streichholzschachtel können im Gegensatz dazu fast überall mit sehr geringem Platzaufwand verbaut werden. Die Komponente „Bauraum“ der VK muss also $\lim_{\text{size}_i \to 0} \approx 0$ und $\lim_{\text{size}_i \to 1\ \text{m}^2} = \infty$ abbilden. Die Wahl fiel deshalb auf die Exponentialfunktion mit linear gewichtetem Argument.

Energieverbrauch

Die letzte Komponente, die weder mit der Produktion noch mit dem Betrieb des Fahrzeugs direkt zu tun hat, ist die notwendige Energie zur Unterhaltung der Kommunikationsinfrastruktur. Diese einzeln und nicht mit anderen Kosten zusammen auszuweisen, ist vor allem dann sinnvoll, wenn wie bei Autos bei stehendem Fahrzeug Teilfunktionaliäten aus Batteriesystemen bereitgestellt werden müssen. Hier dient es weiter zur Abschätzung, ob die vorgeschlagene Kommunikationsarchitektur überhaupt realisierbar ist.

Wie in Kapitel 4.2.1 wurde eine Auswertung von aktuell auf dem Markt befindlichen Bausteinen durchgeführt, die zu der in der Gleichung (4.5) gezeigten Abhängigkeit von der Anzahl der Schnittstellen $N_{P_{\text{active}}}$ führte. In diesem Fall werden nur die aktiven Schnittstellen beachtet, da diese bei einem Teilbetrieb des Kommunikationsnetzes ausschlaggebend sind. Für den zentralen Prozessor werden nun ungefähr 420 mW veranschlagt und weitere 101 mW für jeden aktiven Port. Die hier beschriebenen Zahlen sind exemplarisch , die Mechanismen funktionieren auch mit anderen Werten in der Gleichung.

Der auf W normierte, mit e gewichtete Energieverbrauch aller aktiven Sternknoten liefert das Argument für die quadratische Kostenfunktion E. Gleichung (4.6) spiegelt damit die VK für den Stromverbrauch der Kommunikationsinfrastruktur wider.

$$W_i = 420\ \text{mW} + 101\ \text{mW} * N_{P_{\text{active}}} \tag{4.5}$$

$$E = \sum_{\text{Active Switch}_i} \left(e \cdot \frac{W_i}{\text{W}} \right)^2 \tag{4.6}$$

Eine Verdoppelung der benötigten Stromstärke führt zu einer Verdoppelung des Kabeldurchmessers der Stromversorgung und damit zu einer Erhöhung der Kabelquerschnittsfläche im Quadrat. Gewicht und Materialmenge, zwei Hauptfaktoren der Kosten, sind linear zum Durchmesser. In dem Bereich der Mikroprozessoren zeigen Abhandlungen wie [GBCH01], dass auch die Kosten der Wärmeabfuhr quadratisch zur Leistungsaufnahme steigen. Dies begründet die Verwendung einer quadratischen Gewichtung des Energieverbrauchs in der Gesamtkostenfunktion.

Kostenfunktion

Die Virtuelle Kostenfunktion F_p besteht aus den aufsummierten Komponenten der vorangegangenen Kapitel nach Gleichung (4.7).

$$F_p = L + B + E \tag{4.7}$$

Sie kann innerhalb eines Algorithmus mit Hilfe der Position der Electronic Control Units sowie dem Wissen der physikalischen Verbindungen untereinander berechnet

werden. Damit dient sie später als so genannte *Fitness*-Funktion zur Optimierung der Infrastruktur in Kapitel 4.3. Die aufsummierten Teilkomponenten length_{ij}, size_i und W_i geben einen Anhaltspunkt, ob die vorgeschlagene Lösung in automotiven oder anderen speziellen Einsatzzwecken realisierbar ist.

4.2.2 Parametrisierung

Diagramm 4.5 zeigt die VK bei unterschiedlichen Werten von l sowie die berechnete Kabellänge in Abhängigkeit zu der Anzahl von Kubenkanten.

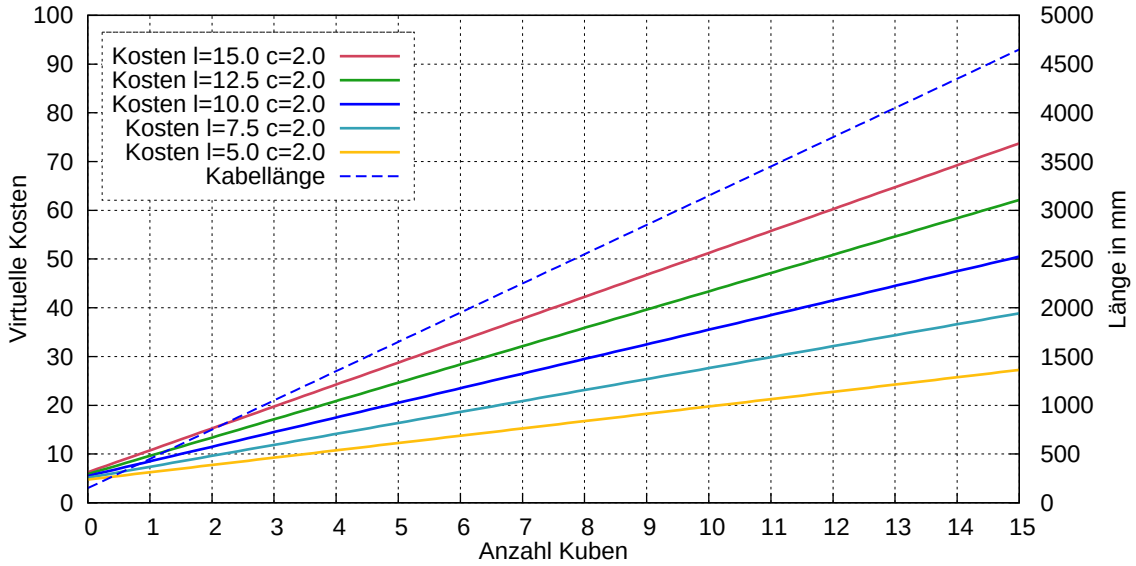


Abbildung 4.5: Metrik Kabellänge

Diagramm 4.6 zeigt die VK bei unterschiedlichen Werten von c sowie die berechnete Kabellänge in Abhängigkeit von der Anzahl von Kubenkanten.

Diagramm 4.7 zeigt die VK eines Sternknotens bei unterschiedlichen Werten von b sowie die berechnete Platinenfläche in mm^2 in Abhängigkeit von der Anzahl verbauter Ports.

In Abbildung 4.8 ist die resultierende Portdichte der Sternknoten bei Variation des Parameters b innerhalb der Optimierung des Referenznetzes (siehe Kapitel A.2) aufgezeigt.

Diagramm 4.9 zeigt die VK eines Sternknotens bei unterschiedlichen Werten von e sowie der berechnete Energieverbrauch in mW in Abhängigkeit von der Anzahl verbauter Ports.

In Abbildung 4.10 ist die resultierende Portdichte der Sternknoten bei einer Variation des Parameters e innerhalb der Optimierung des Referenznetzes (siehe Kapitel A.2) aufgezeigt.

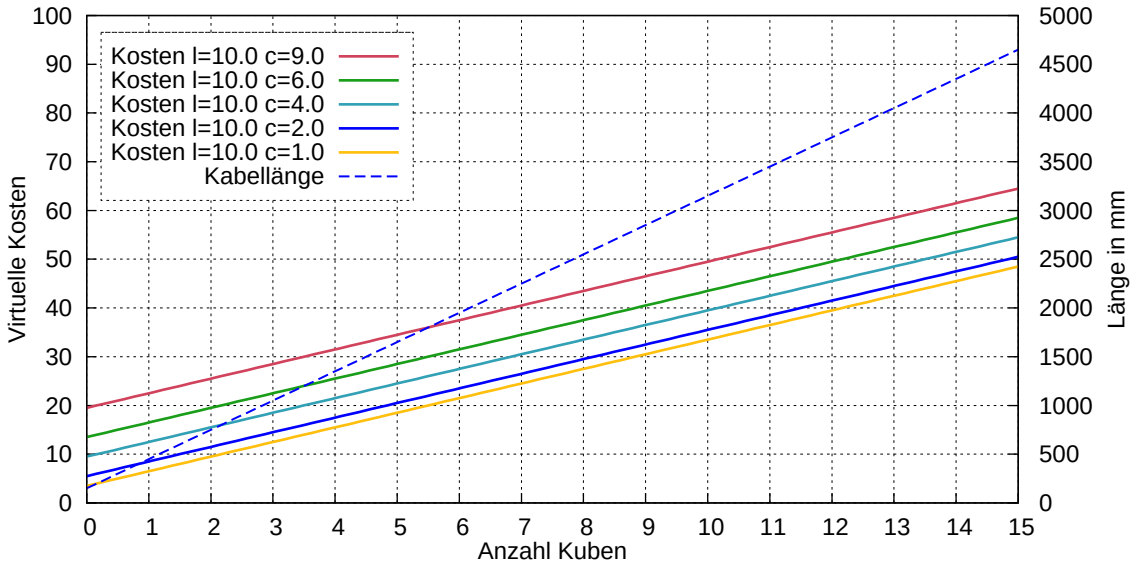


Abbildung 4.6: Metrik Kabellänge

Es wurde eine große Anzahl von Optimierungen unter Varianz der Parameter l, c, b und e durchgeführt und die jeweils resultierenden Lösungen wurden im Hinblick auf die Anzahl der Sternknoten sowie die Anzahl der Ports jedes Sternknotens ausgewertet. Hauptgegenspieler bei der Generierung der Lösungen sind der Bauraum und der Energieverbrauch. Je höher der Parameter b, desto mehr Sternknoten mit jeweils nur drei Ports (Eingang, Ausgang und ECU) werden in der optimalen Lösung verbaut. Aus Sicht des Bauraums sind möglichst viele, aber kleine Sternknoten vorteilhaft, da so die im Fahrzeug vorhandenen Hohlräume besser ausgenutzt werden können. Betrachtet man diese Lösung aber unter wirtschaftlichen Gesichtspunkten, ist zu erkennen, dass dies voraussichtlich nicht die günstigste Lösung sein kann.

Der Gegenspieler bei der Optimierung ist hauptsächlich der Parameter e, also der Energieverbrauch. Eine Erhöhung von e führt zu weniger eingesetzten Sternknoten mit jeweils im Durchschnitt mehr angeschlossenen Kanten. Ausschlaggebend ist dafür der Energiegrundverbrauch durch die Switching Fabric. Die Portdichte erhöht sich nur in Grenzen, da sie durch die quadratisch steigenden Kosten gebremst wird.

Die VK der Kabellängen haben bei einer Variation der Parameter l und c kaum Einfluss auf die resultierende Portdichte der optimalen Lösungen. Sie haben aber grundlegende Bedeutung für die Positionierung der Sternknoten und für die Geschwindigkeit des GA.

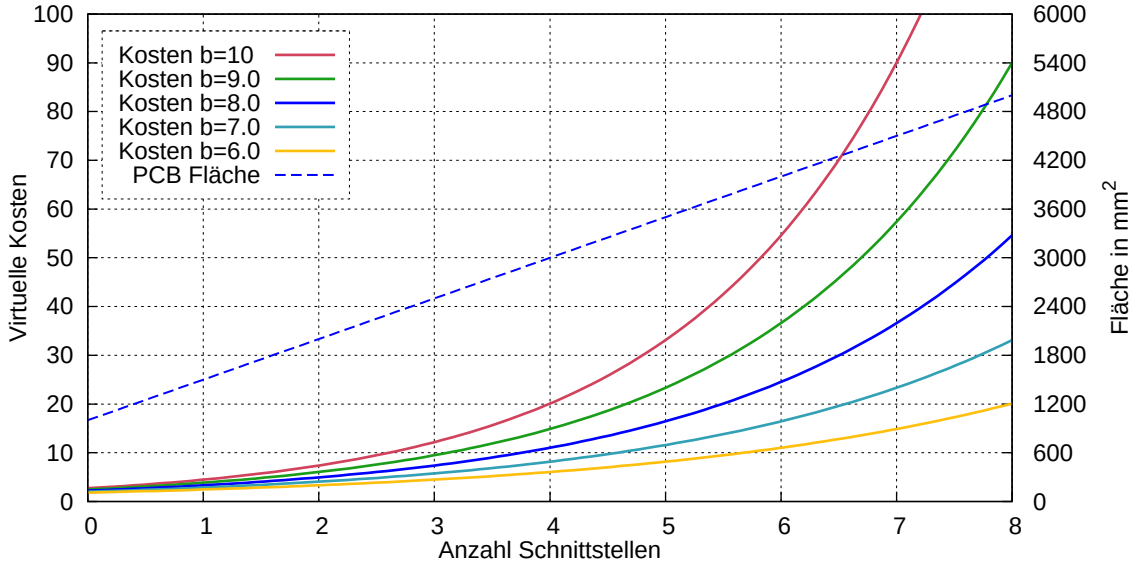


Abbildung 4.7: Metrik Bauraum

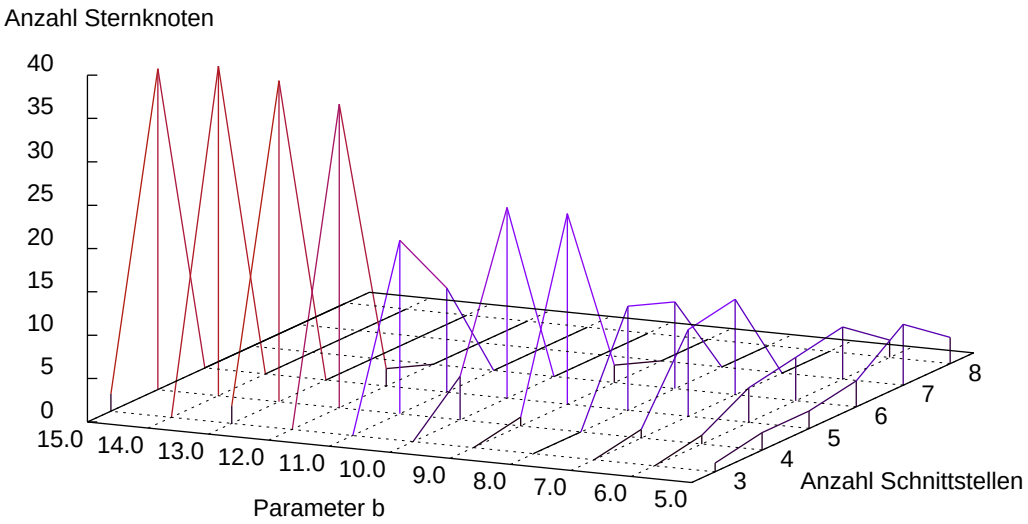


Abbildung 4.8: Portdichte gegen Parameter b

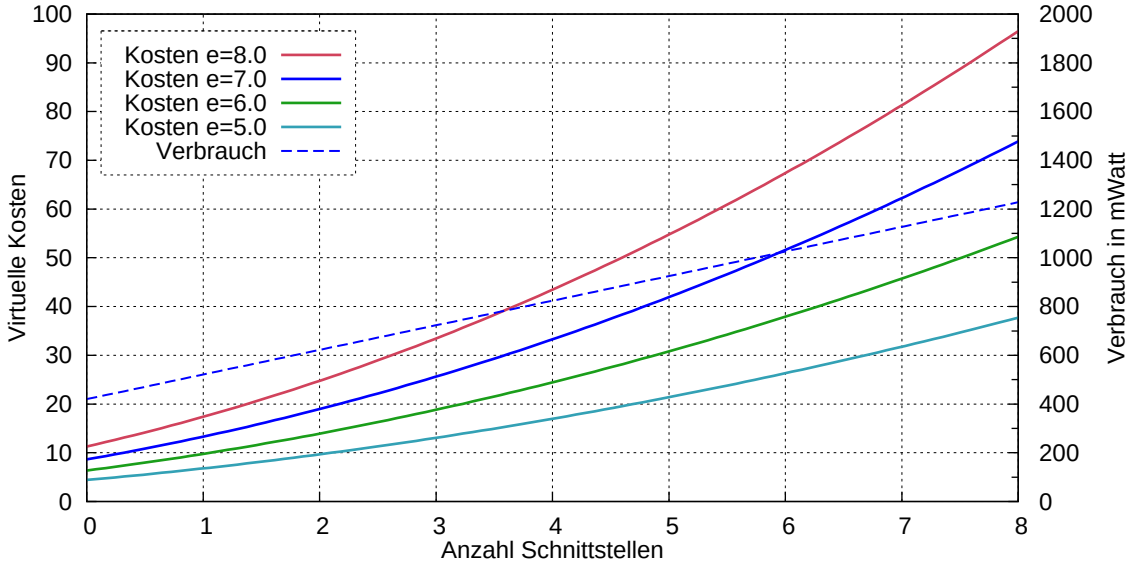


Abbildung 4.9: Metrik Energieverbrauch

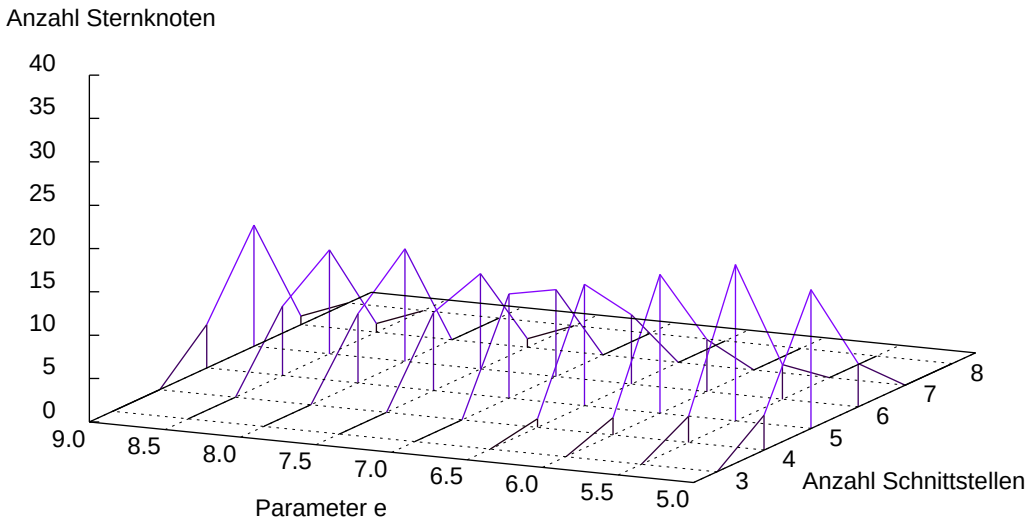


Abbildung 4.10: Portdichte gegen Parameter e

4.3 Optimierung mit Hilfe eines Genetischen Algorithmus

Nachdem mit Kapitel 4.2 nun eine Möglichkeit geschaffen wurde, die Qualität einer möglichen Realisierung der Kommunikationsarchitektur zu quantifizieren, beschäftigt sich dieses Kapitel mit der computergestützten Optimierung derselben.

Um mögliche Algorithmen zur Lösung des Problems zu finden und zu evaluieren, muss zunächst die Komplexität der Aufgabe bestimmt werden. Das Buch von Evans und Minieka [EM92] geben einen guten Überblick über die Graphentheorie und mögliche Optimierungen. Das hier vorliegende Optimierungsproblem lässt sich grundsätzlich auf zwei bekannte Probleme zurückführen. Die Anordnung der sicherheitskritischen Knoten mit jeweils zwei knoten- und kantendisjunkten Pfaden, wie sie in Kapitel 4.3.4 vorgestellt wird, entspricht dem „Problem des Handlungsreisenden" [Men32]. Da die Verbindungen bei Ethernet bidirektional verwendet werden und die gleichen Kosten in beide Richtungen ausweisen, handelt es sich um den symmetrischen Fall. Aus den in Tabelle A.1 gezeigten Attributen des Referenznetzes ergeben sich ungefähr $\frac{(n-1)!}{2} = \frac{(18-1)!}{2} \approx 2 \cdot 10^{14}$ Möglichkeiten. Dieses Problem ist NP-äquivalent, es kann jedoch mit Hilfe von ganzzahlig linearer Optimierung gelöst oder mit Heuristiken eine optimale Lösung angenähert werden.

Um die restlichen ECUs zu einem die Kosten optimierenden Netz zu verbinden, muss ein minimaler Spannbaum, auch *Minimum Spanning Tree (MST)* genannt, im vollvermaschten, ungerichteten Graphen gefunden werden. Nach dem Satz von Cayley [Cay89] gibt es $n^{n-2} = 65^{63} \approx 2 \cdot 10^{114}$ Möglichkeiten, die Knoten zu verbinden. Für dieses Problem gibt es mehrere Lösungsalgorithmen, die den kostengünstigsten Teilgraphen finden.

Alle Algorithmen für die oben erwähnten Probleme haben jedoch gemeinsam, dass die Kosten allein den Kanten des Graphen zugeordnet werden und diese Kosten, egal für welche Lösung, konstant bleiben müssen. Bei dem hier vorliegenden Optimierungsproblem entstehen durch die Sternknoten Kosten an den Knoten, die von der Anzahl der wirklich dort zusammengeführten Leitungen abhängen. Verallgemeinert heißt das, dass die Kosten der Kanten abhängig von den anderen Kanten sind. Damit funktionieren die klassischen Algorithmen für diese Probleme nicht. Die hohe Menge an möglichen Kombinationen machen es noch dazu unmöglich, alle in Polynomialzeit auszuprobieren.

Die Kosten an den Knoten sind, nach den in 4.2.1 aufgestellten Metriken, zwar ganzzahlig linear abhängig von der Anzahl der verbundenen Kanten, werden jedoch mit nichtlinearen Kostenfunktionen gewichtet. So ist eine Beschreibung als lineares Programm (LP), genauer als ein gemischt ganzzahlig lineares Optimierungsproblem, *mixed integer programming (MIP)*, grundsätzlich möglich, die Komplexität dieses Problems würde aber zu extrem langen Rechenzeiten führen.

Zur Lösung wird deswegen der Genetischer Algorithmus (GA) [Bar57, Fra57, Bre62, Gol89] eingesetzt. Der GA gehört zur Klasse der heuristischen Optimierungsverfah-

ren und ist ein evolutionärer Algorithmus. Das heißt, nach dem Beispiel der biologischen Evolution wird aus einer Menge an initialen Zufallslösungen über Vererbung und Selektion der erfolgversprechenden Lösungen auf ein Optimum hingearbeitet. Als heuristisches Optimierungsverfahren kann keine analytische Schranke angegeben und damit auch nicht garantiert werden, dass die zu einem bestimmten Zeitpunkt gefundene Lösung dem Optimum entspricht. Wissenschaftliche Untersuchungen bestätigen aber die Konvergenz der evolutionären Algorithmen zu einem globalen Optimum [EAVH91].

Großer Vorteil des GA in diesem Einsatzzweck ist, dass man am Ende des Prozesses eine ganze Generation an Lösungen hat und damit aus einer Vielzahl von Alternativen auswählen kann. So kann der Netzplaner ungeeignete Lösungen, zum Beispiel bei falschen Kabeldurchführungen, aussondern, ohne dass der Prozess von neuem durchgeführt werden muss.

Die weiteren Unterkapitel beschreiben den verwendeten Algorithmus und die zu Beginn getroffenen Einschränkungen genauer. Details über die Implementierung des Algorithmus und die Einbindung dieses Computerprogramms im Entwurfs- und Planungssystem sind in Kapitel 6.3 erläutert.

4.3.1 Voraussetzung und Annahmen

Betrachtet wird erst einmal ein Netz ohne sicherheitskritische Knoten. Dies bedeutet, eine gültige Lösung besteht aus der kostengünstigsten Verbindung aller Knoten, so dass jeder Knoten mit jedem anderen kommunizieren kann.

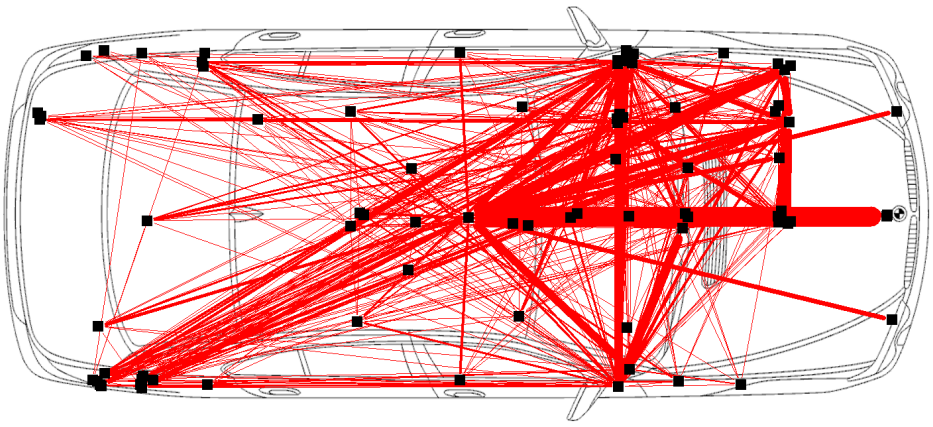


Abbildung 4.11: Typischer Anforderungsgraph des Referenznetzes

Eingabedaten sind die dreidimensional angegebene Position der ECUs im Fahrzeug und die angeforderten Verbindungen. Die angeforderten Verbindungen sind, im Gegensatz zu den physikalischen Verbindungen, nur als logische anzusehen, was bedeutet, dass die Quelle der Verbindung zur Senke Daten überträgt. Dies lässt sich in dem Anforderungsgraphen in Abbildung 4.11 visualisieren. Für den Planungsprozess werden keine weiteren physikalischen Informationen des Fahrzeuges betrachtet. Speziell

mögliche Kabeldurchführungen durch Trennbleche oder Kabelkanäle werden nicht als Eingangsdaten verwendet, sie sind eher als Ergebnis der Optimierung zu sehen. Diese Einschränkung der Realität durch die freie Wegewahl muss eingeführt werden, da erstens in frühen Phasen des PEP diese Informationen nur rudimentär vorliegen und es zweitens die Komplexität des Algorithmus vervielfachen würde, wenn im gleichen Prozess zusätzlich die Laufwege optimiert werden müssten. Durch die in Kapitel 4.2.1 eingeführte Approximation mit der Treppenkurve sind die Einschränkungen der möglichen Pfade in den Kabellängen bereits realistisch genug eingerechnet.

Die Informationen werden aus der Datenbank des Fahrzeugherstellers gewonnen und im XML-basierten Standard Field Bus Exchange Format (FIBEX) exportiert. Der Export dient als Eingabemedium für den Optimierungsalgorithmus.

4.3.2 Algorithmus

Der GA startet mit einer initialen Population und durchläuft bis zur Abbruchbedingung iterativ die Funktionen Immigration, Rekombination, Mutation und Selektion. Eine Population besteht aus einer definierten Anzahl von möglichen Lösungen. Jeweils ein vollständiger Durchlauf der Schleife bildet eine neue Generation der Population. Sobald eine stabile Lösung erreicht wird, sich also durch weitere Durchgänge keine Verbesserung einstellt, wird die Schleife abgebrochen und die aktuelle Generation gesichert. Der Ablauf ist noch einmal in Abbildung 4.12 visualisiert.

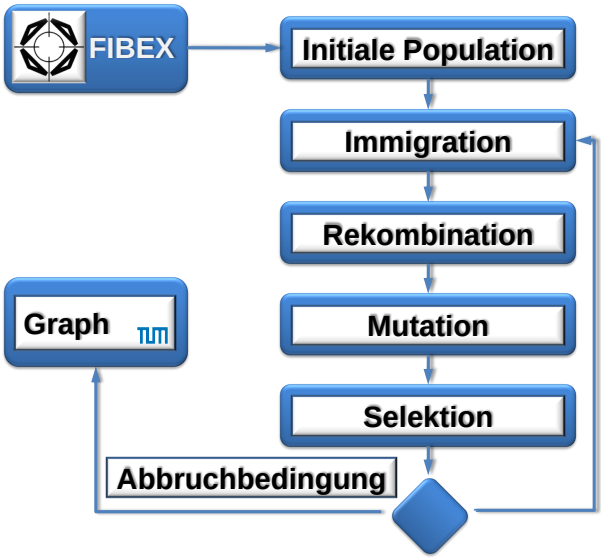


Abbildung 4.12: Ablaufdiagramm des Genetischen Algorithmus

Nach der Eingabe der Daten im FIBEX wird eine initiale Population generiert. Diese besteht aus einer festgelegten Anzahl absolut zufälliger, aber gültiger Lösungen. Sie werden mit Hilfe eines Zufallsgenerators erstellt und dienen als Startpunkt der Evolution. Es ist beabsichtigt, dass in dieser Phase nicht augenscheinlich gute Lösungen als

Startgeneration verwendet werden, um einen möglichst vollständigen Ergebnisraum aufzuspannen.

Die weiteren Unterkapitel beschreiben die einzelnen Stufen des Evolutionsprozesses, wie er im GA abgebildet wird, genauer.

Genom

Algorithmisch kommt der Repräsentation der Individuen große Bedeutung zu. Sie wird in Anlehnung an die Biologie als Genom bezeichnet. Das Genom muss einem gewissen Format entsprechen, so dass lokale Mutationen und Rekombinationen überhaupt möglich sind. Zum Beispiel müssen zwei Genome die gleiche Größe haben und eine Veränderung an einem Eintrag darf das Genom nicht ungültig werden lassen.

In dem hier vorgestellten Algorithmus wird eine Liste von Kanten verwendet. Dafür werden alle verbauten ECUs eindeutig von 1 steigend durchnummeriert. Im Genom gespeichert wird nun nicht die Kante, sondern die Senke derselben. Das heißt, eine 11 an Position 2 in der Liste repräsentiert eine Kante von Knoten 2 zu Knoten 11.

Dies bedeutet, dass es genauso viele Kanten im Genom gibt, wie es Knoten gibt, $|E| = |V|$. Für ein einfach verbundenes Netz, bei dem nur ein möglicher Pfad zwischen einem Paar beliebiger Knoten vorhanden sein muss, müssen mindestens $|E| = |V| - 1$ Kanten existieren [Cay89]. Um jedoch eine Ausfallsicherheit über alternative Wege bereitstellen zu können, wird gegenüber dem minimalen Spannbaum mindestens eine Kante mehr benötigt, was wiederum zu der Menge $|E| = |V|$ führt. Damit zeigt sich, dass der Ansatz, eine feste Anzahl von Kanten zu verwenden, die der Anzahl der Knoten entspricht, tragfähig ist. Zwei zufällige Genome haben damit automatisch eine konstante Größe $|V|$ für den Rekombinationsschritt. Auch die Modifikation eines isolierten Eintrages, sprich das Austauschen der Senke einer Kante, löst nicht die algorithmische Gültigkeit des Genoms auf.

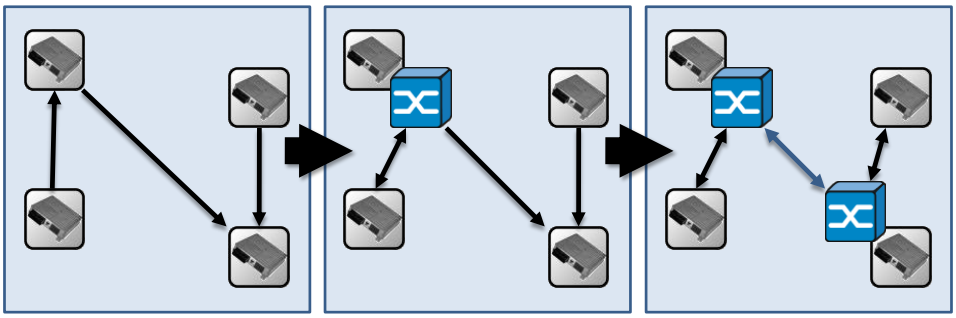


Abbildung 4.13: Aufbau des Netzes aus dem Genom

Voraussetzung ist jedoch, dass sich das Genom eindeutig auf eine physikalische Realisierung abbilden lässt. Für diese ist die Platzierung der Sternknoten genauso relevant wie die Verbindungen zwischen den ECUs. Dafür wurde ein Ablauf basierend auf dem Genom entwickelt, der in Abbildung 4.13 dargestellt ist. Dabei wird jeder Knoten gesucht, der mehr als eine Kante besitzt. Der Knoten, der einer ECU entspricht,

wird nun gegen einen Sternknoten ausgetauscht und die ECU als neu erstellter Knoten mit diesem verbunden. Damit werden die bestehenden Verbindungen zu diesem Knoten automatisch auf den Sternknoten umgezogen. Führt man diesen Prozess für alle Knoten mit einem Kantengrad > 1 aus, bildet sich eine Art Backbone von Verbindungen zwischen den Sternknoten, im Bild blau dargestellt, und den einfachen Kanten zu den ECUs, hier schwarz dargestellt, aus. Wichtig ist, dass jede ECU genau eine Verbindung zu einem Sternknoten und sonst keine weiteren Kanten hat. Mit dem beschriebenen Vorgehen wird dies sichergestellt und resultiert bei gleichem Genom immer in dem gleich aufgebauten physikalischen Kommunikationsnetz.

Eine zufällige, aber gültige Lösung wird generiert, indem jeder Position i der Liste $[1; |V|]$ im Genom ein Wert j aus dem Intervall $[1; |V|]$ zugewiesen wird, so dass $i \neq j$ ist.

Immigration

Der erste Schritt der Iteration ist die Immigration. Sie dient dazu, regelmäßig frisches Erbgut in den Prozess zu bringen. Dieser Schritt stellt eine Erweiterung zum klassischen GA dar und wurde von Moed et al. eingeführt [MDS+91].

Eine Herausforderung, der sich der hier dargestellte Algorithmus stellen muss, ist, dass es viele gleichwertige Lösungen gibt. Da die Position der Steuergeräte, wie in Kapitel 4.2.1 beschrieben, nur näherungsweise in Kuben festgelegt ist, gibt es daraus folgend mehrere ECUs mit derselben Position. Bei welchem nun der Sternknoten sitzt, wirkt sich, bei Vernachlässigung der Übertragungsverzögerungen, auf die Kosten weiter nicht aus. Diese Verarmung des Ergebnisraumes führt dazu, dass der Algorithmus mit höherer Wahrscheinlichkeit in einem Nebenminimum des Lösungsraumes verweilt und dort sogar stecken bleiben kann. Um dies zu verhindern, wurden zwei Modifikationen eingeführt.

Erstens wurden gleiche Lösungen aussortiert. Genome, die exakt die gleichen Kosten wie bereits in der Population vorhandene Lösungen verursachen, werden bei der Selektion aus der Population entfernt. Dafür werden für jedes Genom in der Generation die berechneten Kosten als *Double*-Variable, also mit einer Genauigkeit von ungefähr 16 Dezimalstellen, vorgehalten. Ergeben sich bei der Berechnung exakt dieselben Kosten, wird davon ausgegangen, dass es sich um eine redundante Lösung handelt, und dieses Genom wird mit Kosten von ∞ versehen.

Zweitens erfolgt eine vorgelagerte Immigrationsphase. Bei der Immigration wird eine festgelegte Anzahl an zufälligen neuen Lösungen in die aktuelle Generation aufgenommen. Dabei erhöht sich bis zur Selektion die Größe der Population um die Anzahl der Immigranten. Diese Genome sind, wie schon bei der initialen Population, mit Hilfe eines Zufallsgenerators erstellt. Sie dienen vor allem in der Rekombinationsphase der Generierung frischer Lösungen. Die Anzahl der immigrierten Genome darf nicht zu hoch angesetzt werden, um den normalen Evolutionsprozess nicht zu verhindern.

Rekombination

Zentraler Vorgang im evolutionären Vorgehen ist die Rekombination. Mit Hilfe der Rekombination werden potentiell gute Lösungen miteinander kombiniert, um bessere Lösungen entstehen zu lassen. Es ist vergleichbar mit der Kreuzung des Erbguts beim Menschen. Verwendet wird das Verfahren *uniform crossover* [Syw89].

Für die Rekombinationsphase ist es wichtig, dass zwei zufällig ausgewählte Genome die gleiche Länge haben. Dies wurde mit der Festlegung in Kapitel 4.3.2 bereits festgesetzt. Es werden nun zwei Genome der aktuellen Generation mit Hilfe eines Zufallsgenerators ausgewählt. Es ist wichtig, das dabei Genome über ihre Güte gleichverteilt ausgewählt werden, also nicht zum Beispiel nur die Kreuzung der besten Lösungen durchgeführt wird. Die beiden Genome dienen nun als Mutter und Vater für den anschließenden Prozess.

Als Nächstes wird eine Kreuzungsliste erstellt. Über einen binären Zufallsgenerator wird mit der definierten Wahrscheinlichkeit P_{comb} für die Anzahl der Knoten $|V|$ mal eine 0 oder eine 1 gewählt und in einer Liste hinterlegt. Für das Kind-Genom wird für jede Stelle, bei der die Kreuzungsliste ein 0 enthält, der Eintrag der Mutter übernommen und bei einer 1 die des Vaters. Mit der Inversen der Kreuzungsliste wird ein zweites Kind erzeugt.

Eine klassische Rekombination wird mit der Wahrscheinlichkeit $P_{comb} = 0{,}5$ erreicht, wenn also gleich viele Einträge des Mutter- wie des Vater-Genoms verwendet werden. In dem hier verwendeten Optimierungsproblem zeigte diese Einstellung gute Ergebnisse und wurde nicht weiter variiert. Dabei ist sicherzustellen, dass weder Mutter noch Vater, noch die beiden Kinder im weiteren Vorgehen wieder als Eltern gewählt werden.

Mutation

Ein weiteres Mittel, um neue Kombinationen in die Evolution einzubringen, ist die Mutation. Hierbei werden einzelne Einträge des Genoms zufällig verändert.

Mit dieser Implementierung des GA werden, in Anlehnung an die menschliche Evolution, nur die Kinder des Rekombinationsprozesses in den Mutationsvorgang einbezogen. Einerseits würde das Neuberechnen der Kosten aller Lösungen die Laufzeit des Algorithmus deutlich verlängern, andererseits sollen gute Lösungen durch die Mutation nicht zerstört werden.

Bei der Mutation werden alle Einträge aller Genome durchlaufen und mit der Wahrscheinlichkeit P_{mut} ausgewählt. Gewählte Einträge der Liste, also Senken der Kanten, werden mit einem zufällig ausgesuchten Knoten ersetzt, der ungleich der Quelle ist. Auch dabei hilft der Aufbau des Genoms, da isolierte Veränderungen innerhalb der Liste die Gültigkeit des Genoms nicht antasten.

Selektion

Der wichtigste Schritt im evolutionären Algorithmus ist die Selektion als Ende der Iteration. Beim Selektionsprozess wird die Populationsgröße auf einen definierten Wert verringert und dabei werden jene Lösungen ausgewählt, die am besten dafür geeignet sind, im nächsten Durchlauf noch bessere Ergebnisse zu erzeugen. Diese Lösungen bilden die nächste Generation im Ablauf des Algorithmus.

Zuerst muss über die Gültigkeit nachgedacht werden. Diese wird unterschieden in ein gültiges Genom und eine gültige Lösung. Ein Genom ist gültig, wenn die Konstruktionsregeln eingehalten wurden. Dies ist dann der Fall, wenn das Genom eine Größe von $|V|$ hat und Quelle und Senke, das heißt die Position in der Liste und der Wert, nicht identisch sind. Dies wird bereits im Ablauf des GA in jedem Schritt durch die Implementierung sichergestellt.

Eine Lösung wiederum ist gültig, wenn sie die Verbindungsregeln einhält. In einem normalen, nicht ausfallsicheren Netz ist dies erfüllt, wenn jeder Knoten mit jedem anderen kommunizieren kann, also eine gültiger Pfad existiert. Dafür muss zuerst aus dem Genom ein physikalischer Graph erstellt werden. Mit dem Genom werden die Verbindungen zwischen den Knoten als Kanten eingetragen und dann mit dem Ablauf aus Abbildung 4.13 die Sternknoten platziert. Danach wird ausgehend von einem zufällig ausgewählten Knoten ein Spannbaum aufgestellt und überprüft, ob er alle vorhandenen ECUs einschließt. Ist dies nicht der Fall, ist die Lösung ungültig und wird mit Kosten von ∞ belegt, so dass sie bei der nachfolgenden Selektion aussortiert wird. Zu beachten ist, dass in diesem Vorgang nicht zu viele Lösungen verworfen werden, weil sie nicht den Regeln entsprechen. Beim Durchlauf des Prozesses waren jedoch über $2/3$ der neuen Lösungen korrekt, also lebensfähig, und damit bestand keine Notwendigkeit, diesen Schritt zu modifizieren. Die Anzahl der neuen Lösungen einer Generation vor der Selektion, also der noch nicht vorhandenen und gültigen Lösungen, ist in Abbildung 4.15 dargestellt.

Die Verbindungsregeln für ein ausfallsicheres Netz werden in Kapitel 4.3.4 eingeführt, die bei zusätzlicher Einbeziehung des Datenverkehrs in Kapitel 4.3.5.

Im nächsten Schritt muss die Fitness des Genoms, also die Güte einer Lösung bestimmt werden. In diesem Fall sind dies die Kosten, die über die VK aus Kapitel 4.2 ermittelt werden müssen. Dieser Schritt wird für jede gültige Lösung, die durch Immigration oder Rekombination hinzugekommen ist, sowie für die mutierten Genome durchgeführt. Ergebnis ist eine positive, reelle Zahl $\mathbb{R} > 0$. Die Berechnung der Kostenfunktion wird ebenso nicht direkt am Genom, sondern an dem daraus resultierenden, eindeutigen Graphen durchgeführt.

Als Auswahlkriterium [GD91] für die nächste Generation wurde die Roulette-Selektion [Bak87] evaluiert. Bei dem auch unter der Bezeichnung Fitness-proportionale Selektion bekannten Verfahren wird jedes Genom mit einer zu den Kosten proportionalen Wahrscheinlichkeit für die nächste Generation ausgewählt. Bildlich werden jeder Lösung eine Anzahl Felder auf einem Roulette-Rad zugewiesen, die von ihrer Fitness abhängt. Bei einem auf die Populationsgröße normierten Verfahren bekommt

die schlechteste Lösung ein Feld und die beste Lösung a Felder, wobei a gleich der aktuellen Populationsgröße ist. Aus dem gesamten Pool der Felder wird nun sooft zufällig gezogen, bis die nächste Generation vollständig gebildet ist. Wurde ein Genom selektiert, wird es aus dem nächsten Ziehvorgang ausgeschlossen und seine Felder werden entfernt. Nachteil dieses Verfahren ist es, dass die besten Lösungen zwar mit einer hohen Wahrscheinlichkeit gewählt werden, aber auch mit einer gewissen Wahrscheinlichkeit verworfen werden. Für das hier beschriebene Optimierungsproblem führte dies zu überwiegend nicht konvergierenden Abläufen. Auch die Modifikation zu einem Rangstellen-basierten Auswahlverfahren [Whi89] brachte keine Besserung. Deswegen wurde ein künstliches Verfahren verwendet, das garantiert die besten Lösungen selektiert.

Die Verkürzungsmethode (*Truncation Selection*) [MSV93] verwendet immer die besten Lösungen für die nächste Iteration. Dafür werden die Genome nach ihren Kosten sortiert und die besten n ausgewählt, wobei n der Startgröße der nächsten Generation entspricht. Die Auswahl ist damit unabhängig von den konkreten Kosten, sondern nur vom Rang der Lösung in der Population abhängig. So wird sichergestellt, dass die besten Genome auch weiter fortbestehen und unrealistische Lösungen sofort entfernt werden. Es zeigte sich, dass sich der hohe Verlust an Diversität bei jeder Selektion durch die Verkürzungsmethode [BT96] bei diesem Problem positiv auf die Konvergenz des Algorithmus auswirkt.

Abbruchbedingung

Am Ende des GA steht die Abbruchbedingung, die das weitere Durchlaufen der Schleife beendet. Da, wie oben bereits erwähnt, bei heuristischen Optimierungsverfahren keine analytische obere Schranke angegeben und damit auch nicht garantiert werden kann, dass die zu einem bestimmten Zeitpunkt gefundene Lösung dem Optimum entspricht, muss die Abbruchbedingung die Qualität einer Generation bewerten.

Naheliegend ist, das beste Genom zu betrachten. Im Diagramm 4.14 sind die obere und die untere Schranke der besten Lösung bei einer bestimmten Generation über 20 Durchläufe des Algorithmus als blaue Fläche dargestellt. Am Anfang des Optimierungsprozesses sind die Verbesserungen pro Generation sehr stark, was sich an der Steigung der Kurve erkennen lässt. Jedoch zeigt der Prozess die Tendenz, für stärker wie linear wachsende Zeiträume bei der gleichen Lösung zu verharren, was an der zunehmenden Stufenlänge der blauen Kurve zu sehen ist. Dies führt gerade gegen Ende der Iterationen zu einer hohen Anzahl an Generationen mit konstanter Qualität. Somit ist es schwer, eine Grenze zu setzen, bei der mit hoher Wahrscheinlichkeit das Optimum erreicht wurde.

Aus diesem Grund wurde die durchschnittliche Fitness der gesamten Generation als Qualitätsmerkmal derselben verwendet. Der Verlauf ist zum Vergleich in Abbildung 4.14 als rote Fläche eingezeichnet. Im Gegensatz zur besten Lösung nähert sich der Wert monoton (aber nicht streng monoton) fallend einem Minimum an. Wenn die

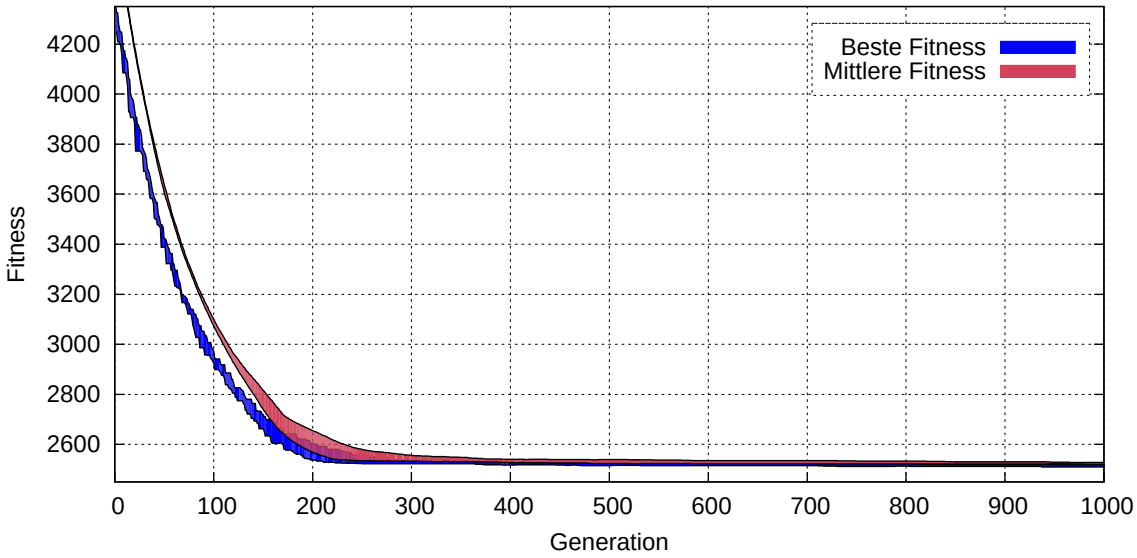


Abbildung 4.14: Abbruchbedingung des Genetischen Algorithmus

durchschnittliche Fitness über eine Anzahl von Generationen stabil bleibt, ist der Algorithmus in einem Sättigungszustand, und das Minimum ist erreicht.

Ob dies dann ein lokales oder ein globales Minimum und damit die Lösung ein lokales oder globales Optimum ist, kann von der berechneten Qualität der Generation nicht abgeleitet werden. Dies lässt sich am besten durch mehrere Durchgänge und manuelle Überprüfung der Ergebnisse bewerten. In den hier durchgeführten Untersuchungen mit mehreren Durchläufen ergab sich für das Ergebnis im lokalen Seitenoptimum eine Abweichung von nur $0{,}33\%$.

Parameter

Zur Festlegung der Parameter wurden mehrere Testläufe gemacht und die Konvergenzgeschwindigkeit wurde ausgewertet. Für den in der Arbeit verwendeten Netzaufbau und die verwendeten Metriken haben sich die in Tabelle 4.2 angeführten Werte als optimal erwiesen, um eine schnelle und sichere Konvergenz zu einem Endresultat zu erreichen. Die Wahrscheinlichkeit des Steckenbleibens in Nebenminima zeigte sich damit als sehr gering.

Parameter	Wert	Parameter	Wert
Population	1000	Mutation	1400
Immigration	200	Mut. Stärke P_{mut}	0,001
Abbruch $\lvert Gen. \rvert$	300	Rekomb. P_{comb}	0,5

Tabelle 4.2: Standardwerte des Genetischen Algorithmus

Es wird eine initiale Population von 1.000 Individuen erzeugt und auch während der Selektionsphase kontinuierlich auf diese Größe verringert. Die 1.400 Genome in der Mutation sind alle immigrierten sowie alle durch Rekombination gewonnenen Kinder. Der Abbruch der Iteration erfolgt, sobald sich die mittlere Fitness aller Genome über mehr als 300 Generationen nicht verbessert hat.

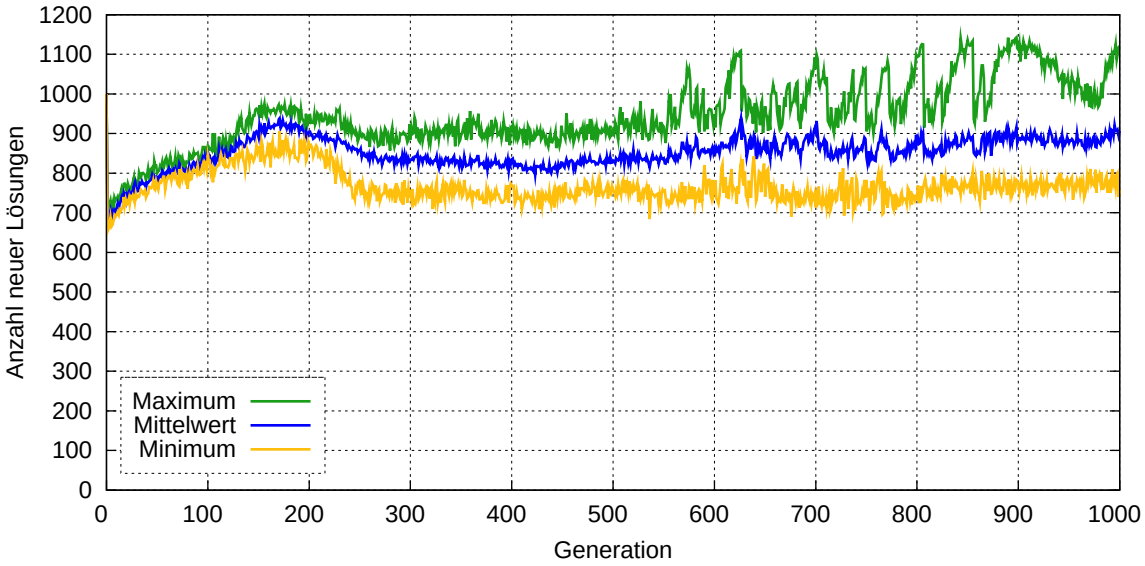


Abbildung 4.15: Neue Lösungen pro Generation

Abbildung 4.15 visualisiert die pro Generation neu gefundenen Lösungen. Bearbeitet werden mussten 1.400 Lösungen pro Generation, also $1.400.000$ in jedem dieser Durchläufe. Durchschnittlich wurden, aufsummiert, 852.578 neue Lösungen gefunden, was einem Verhältnis von $\approx 60\%$ entspricht. Der nicht in die Selektion aufgenommene Rest sind ungültige und bereits in der Population vorhandene Lösungen. Das Diagramm zeigt die neuen Lösungen pro Generation als arithmetischen Mittelwert über die Durchläufe sowie die maximal und minimal aufgetretenen Werte aller Durchläufe.

4.3.3 Kostenoptimierte Vernetzung

Der erste Schritt zum Testen und zur realen Anwendung des Algorithmus war die einfache Vernetzung aller ECUs, so dass, wie bereits in Kapitel 4.3.1 erwähnt, zwischen jedem Endgerät genau ein Pfad existiert. Dafür wurde das in Kapitel A.2 beschriebene Fahrzeugnetz verwendet. Ziel war es, die Kosten für dieses Netz mit Hilfe des GA zu minimieren. Aus diesen Durchgängen wurden auch diejenigen in Kapitel 4.3.2 gewonnen und zur Lösung der Probleme der weiteren Kapitel verwendet.

Auf den ersten Blick entspricht dieses Problem zwar grundsätzlich dem lösbaren Problem des minimalen Spannbaums, wie in Kapitel 4.3 bereits erwähnt. Jedoch entstehen je nach Anzahl der zusammengefassten Leitungen beziehungsweise Kanten an

den Knoten Kosten durch die Sternknoten. Diese dynamisch sich verändernden Kosten pro Kante sind in den verfügbaren Algorithmen nicht abgedeckt und damit nicht direkt lösbar.

Der hier vorgestellte GA löst dieses Optimierungsproblem mit der in Diagramm 4.14 gezeigten Konvergenz. Die in Kapitel 4.3.2 beschriebene Eigenschaft des gewählten Genoms, dass die Anzahl der Kanten gleich der Anzahl der Knoten ($|E| = |V|$) ist, stellt kein Fehler bei dem als Ergebnis generierten minimalem Spannbaum dar. Beim Aufbau des Ergebnisgraphen aus dem Genom, der auch Grundlage der Kostenberechnung ist, wird die zusätzliche Kante, sobald sie mit einer bereits vorhandenen parallel liegt, automatisch entfernt. Für eine Lösung mit minimalsten Kosten ist dies automatisch der Fall, da sie als redundante Kante nur Kosten verursacht, aber keine Erhöhung der Konnektivität bringt.

Die beste Lösung aus 1.000 per Zufallsgenerator erzeugten möglichen Kombinationen der Generation 0 liegt bei 4.301. Der Algorithmus optimiert diese Kosten in unter 1.000 Generationen auf 2.513, was einer Einsparung von $\approx 42\%$ entspricht.

Da aber die Grundvoraussetzung des Ansatzes aus Kapitel 3.2 ist, dass festgelegte sicherheitskritische Knoten über kanten- und knotendisjunkte Pfade verbunden werden müssen, dient dieses Setup nur zur Evaluierung des Algorithmus und zur Optimierung der Parameter.

4.3.4 Ausfallsichere Netze

Der in Abbildung 4.12 gezeigte Ablauf des klassischen GA funktioniert nachgewiesenermaßen für einfach verbundene Netze wie die in Kapitel 4.3.3 verwendeten. In diesem Kapitel werden die notwendigen Erweiterungen aufgezeigt, um auch Netze mit einem Clusterkoeffizienten > 0 zu realisieren.

Als Ergänzung zu Kapitel 4.3.2 ist Voraussetzung für ein gültiges Netz, dass zwischen allen sicherheitskritischen ECUs mindestens zwei kanten- und knotendisjunkte Pfade existieren und zwischen allen nicht sicherheitskritischen Knoten mindestens einer. Die Einstufung der ECU erfolgt manuell und wird im FIBEX zur Eingabe in den Algorithmus vorgehalten.

Es wurden drei Ansätze verfolgt, um die zufälligen Lösungen für die initiale Population zu erzeugen. Der erste arbeitet mit dem vorhandenen Genom der festen Größe $|E| = |V|$. Die Anzahl der Kanten reicht theoretisch aus, um eine gültige Lösung zu generieren (Kapitel 4.3.2). Wird nun über den Zufallsgenerator das Genom befüllt, bildet sich mit einer gewissen Wahrscheinlichkeit ein gültiges Netz aus. Bereits bei vier sicherheitskritischen Knoten zeigt sich, dass nur sehr wenig gültige Lösung gefunden wurden. Bei dem bekannten Testnetz aus Kapitel A.2 mit 18 Knoten konnte auch nach 24 Stunden Laufzeit keine initiale Population mit 1.000 Genomen generiert werden, tatsächlich lag die Anzahl der gefundenen Lösungen unter 10. Angesichts des schlechten Verhaltens des Ansatzes im Testaufbau unterhalb jedes Maßstabes wurde er nicht weiter verfolgt, und es wurde auch keine analytische Auswertung des Vorgangs durchgeführt.

Eine weitere Methode ist das Entfernen oder Hinzufügen von Kanten bis zur gültigen Lösung. Dies wird in den Arbeiten von Monma et al. [MS89, MMP90] näher betrachtet. Nachteil dieser Methode ist, dass unterschiedliche Lösungen eine differierende Anzahl von Kanten haben und damit die Genome unterschiedliche Größe. Dies führt im Rekombinationsprozess zu erhöhtem Aufwand. Des Weiteren hängen die VK der Lösung Kapitel 4.2 zufolge ausschließlich von den Kanten ab, so dass ein direkter Zusammenhang zwischen der Anzahl der Kanten und den Kosten und damit der Güte der Lösung vorhanden ist. Methoden, die mehr Kanten als notwendig erzeugen, stehen damit dem gesetzten Optimierungsziel entgegen.

Inspiriert von den Arbeiten von Konak et al. [KS05] wurde ein Ansatz mit einer Reparaturphase entwickelt. Dafür wurde der Ablauf des klassischen GA um diesen Schritt erweitert, visualisiert in Grafik 4.16, sowie die Generation der initialen Population modifiziert.

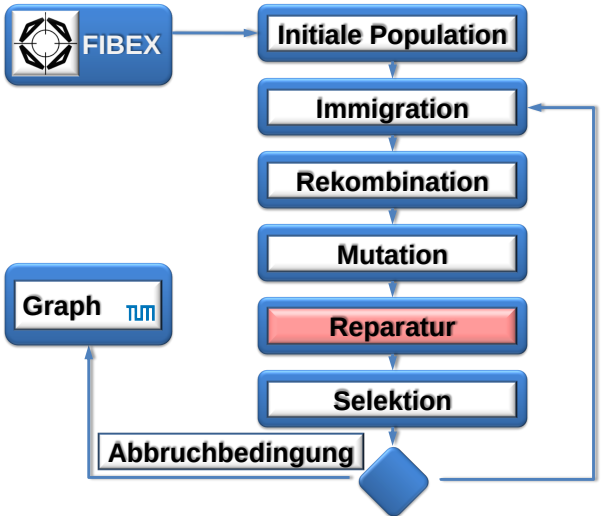


Abbildung 4.16: Erweiterung für ausfallsichere Netze

Bei generierten Lösungen, wie in der Generation 0 und während des Immigrationsvorgangs, wird das Vorgehen in zwei Phasen getrennt. Erste Phase behandelt nur die sicherheitskritischen Knoten. Mit einem zufällig gewählten Knoten beginnend, wird eine Kante zu einem beliebigen anderen Knoten gelegt, der noch keine Verbindung besitzt. Der letzte Knoten muss sich mit dem Startknoten verbinden. Mit dem nun vorhandenen Ring wird in der zweiten Phase, wie in Kapitel 4.3.2 gezeigt, das vollständige Genom gebildet, indem die verbleibenden Positionen der Liste mit Knoten aus $[1; |V|]$ (aber nicht sich selbst) belegt werden. Diese Lösung wird nach den oben genannten Gültigkeitsrichtlinien untersucht.

Dies bedeutet, jede von diesem Ablauf generierte gültige Lösung besteht aus einem Netz mit $|E| = |V|$ Kanten, die einen Spannbaum mit $|E| = |V| - 1$ Kanten bilden, und einem durch die letzte Kante geschlossenen Hamilton-Zyklus über die sicherheitskritischen Knoten. Diese Konstellation erfüllt alle gesetzten Anforderungen, vor allem die redundanten Kommunikationspfade zwischen den sicherheitskritischen Knoten.

Diese spezielle Konstellation im Graphen wird jedoch durch die Mutation und vor allem durch den Rekombinationsprozess mit hoher Wahrscheinlichkeit zerstört. Für evolutionäre Algorithmen untypisch und auch nur bedingt einsetzbar, wurde ein zwischengelagerter Reparaturprozess eingefügt, der in einem Struktogramm (Abbildung 4.17) visualisiert ist. Zu beachten ist bei externen Eingriffen in den Ablauf, dass die ursprüngliche Information des Genoms weitestgehend erhalten bleibt.

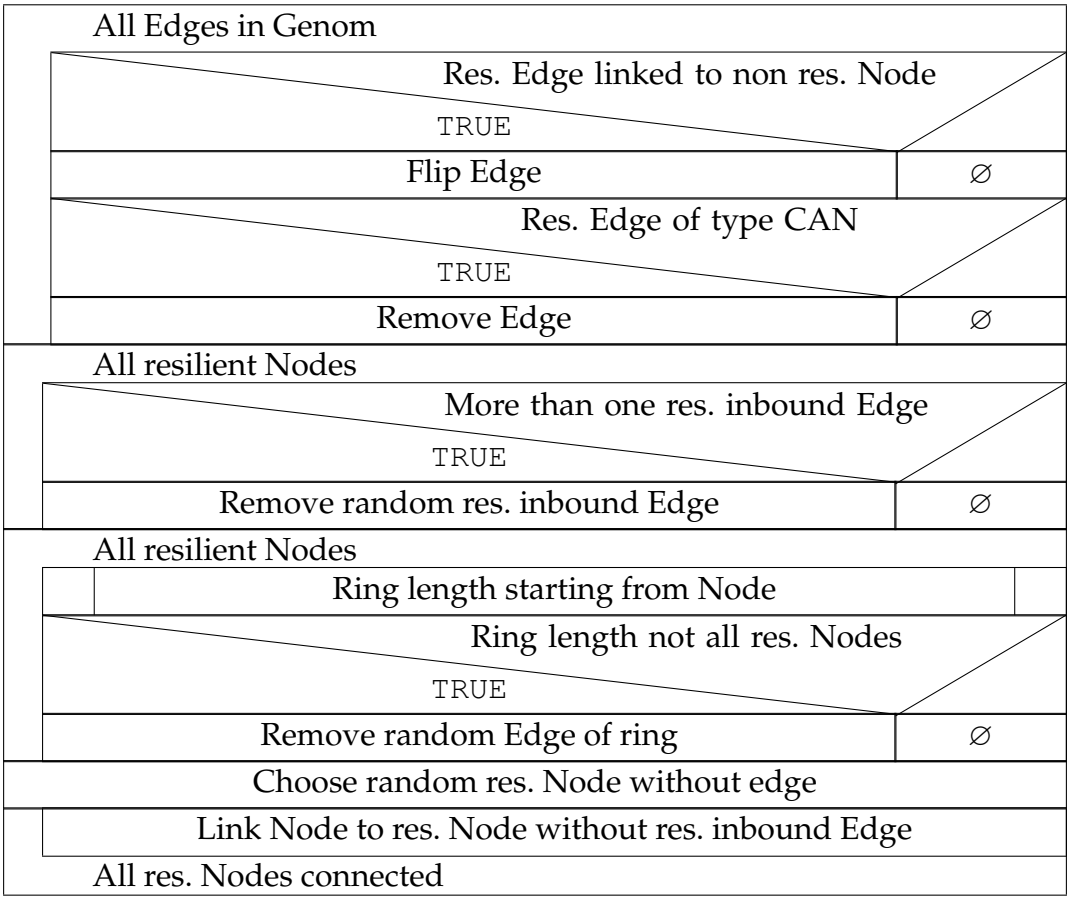


Abbildung 4.17: Struktogramm „Repair“

Im Genom wird noch mit gerichteten Kanten gearbeitet. Zuerst müssen alle der Lösung zuwiderhandelnden Kanten entfernt werden. Dies sind erstens Kanten von einem sicherheitskritischen Knoten (SK), die an einem nicht sicherheitskritischen Knoten (nSK) enden, da jeder Knoten nur eine gerichtete Kante hat und laut Hamilton-Zyklus diese an einem SK enden muss. Diese Kanten werden nicht entfernt, sondern umgedreht, also Quelle und Senke werden vertauscht, um nicht zu viel Information zu verlieren. Des Weiteren wird sichergestellt, dass die Kante vom Typ Ethernet ist (siehe Kapitel 4.3.5).

Im nächsten Schritt werden Subringe und Kreuzungen bei den SK entfernt. Dabei wird primär bei jedem SK, der mehr wie eine eingehende Kante mit einem anderen SK als Quelle besitzt, eine zufällig ausgewählt und entfernt. Danach wird von jedem

SK aus untersucht, ob ein geschlossener eigenständiger Subring existiert. Trifft dies zu, wird eine beliebige Kante dieses unvollständigen Subrings entfernt.

Zur Reparatur des nun unvollständigen Netzes wird das oben genannte Vorgehen zur Bildung des Ringes vorgenommen. Keiner der nSK ist in diesen Vorgang involviert.

Ergebnis

In Abbildung 4.18 ist ein resultierendes Netz mit optimalen Kosten beispielhaft dargestellt. Grundlage war das Testnetz aus Kapitel A.2. Die blauen Striche entsprechen dem Backbone, also den Verbindungen zwischen zwei Sternknoten. Analog dazu sind die schwarzen Striche die Anschlussleitungen der ECUs. Die Quadrate kennzeichnen die Position der ECU und die rot gefärbten sind dabei die als sicherheitskritisch eingestuften.

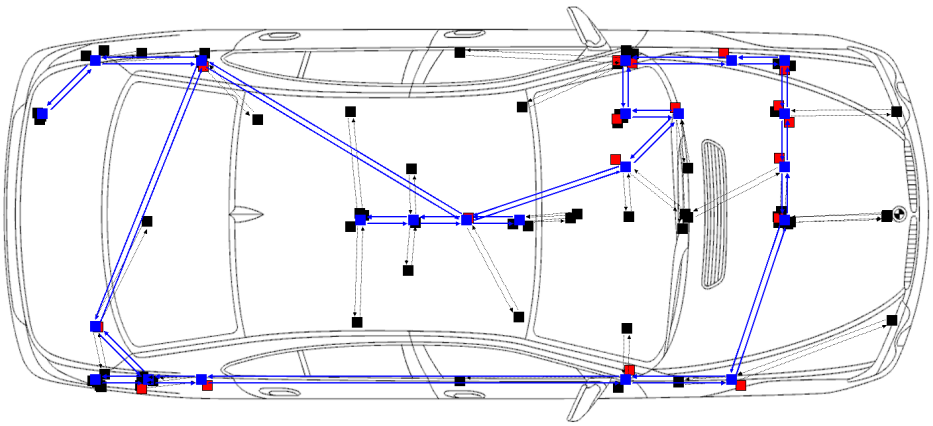


Abbildung 4.18: Resultierendes ausfallsicheres Netz

Das gezeigte Netz entspricht auch visuell einer optimalen Lösung. Der Ring zwischen den sicherheitskritischen Knoten ist klar ausgeprägt, die restlichen Knoten sind an räumlich nahe Sternknoten angeschlossen. Bei Häufung von Steuergeräten hat sich der Backbone erweitert, um die Kabelwege zu verringern, zum Beispiel im Kofferraum fahrerseitig.

4.3.5 Heterogene Netze

Die in den letzten Kapiteln erarbeiteten Lösungen sind unter der Annahme der Verwendung einer homogenen Kommunikationsinfrastruktur, basierend auf Ethernet, optimal. Nun soll weiteres Optimierungspotential untersucht werden. Betrachtet man die Realisierungen in Personenkraftfahrzeugen und untersucht die einzelnen ECUs hinsichtlich ihrer Komplexität, ergibt sich ein sehr divergierendes Bild. Einige Knoten bestehen nur aus wenigen Schaltern mit Kosten im unteren Euro-Bereich, für die es daher nicht indiziert ist, eine aufwendige Ethernet-Verkabelung vorzunehmen.

Um diese Datenquellen abzudecken, werden Subnetze eingeführt. Unter Verwendung von Bussystemen mit niedriger Datenrate und Komplexität werden ECUs auf Grund der Position im Fahrzeug und nicht klassisch durch die Funktionalität aggregiert und über Gateways an den zentralen Backbone angeschlossen. Am besten eignet sich dafür Controller Area Network (CAN) infolge der weiten Verbreitung in allen Industriebereichen und der geringen Komplexität bei dennoch zukunftstauglichen Datenraten von bis zu 1 Mbit/s. Die vorgeschlagene Integration ist in Blockschaltbild 4.19 dargestellt.

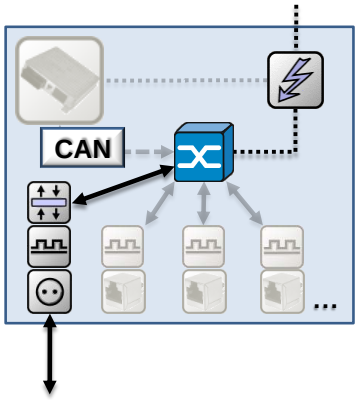


Abbildung 4.19: Blockschaltbild CAN-Integration

Der in Kapitel 4.1 gezeigt Sternknoten wird um eine Schnittstelle für CAN erweitert. Über einen Kommunikationsprozessor oder eine Hardwarerealisierung mit einem ASIC beziehungsweise FPGA wird die Gatewayfunktionalität realisiert. Das Protokoll uMAP, zu dem sich Details in Kapitel 5.3 finden, wird als Transportprotokoll vorgeschlagen.

Die Integration in die Sternknoten wird bevorzugt, weil dort bereits eine passende Stromversorgung vorhanden ist. Auch kann der Gatewaybaustein direkt als Ethernet-Endgerät über GMII mit dem Koppelelement verbunden werden, was weitere Hardwarebausteine einspart.

Erweiterung des Algorithmus

Um eine Optimierung unter diesen veränderten Voraussetzungen zu ermöglichen, muss der Algorithmus angepasst werden. Zuerst wird das Genom um einen weiteren Eintrag erweitert, den Typ. Eine im Genom festgelegte Kante hat nun neben der Senke noch die Eigenschaft CAN oder Ethernet gespeichert. Dies wird, wie die anderen Einträge, per Zufallsgenerator erstellt und bei der Rekombination zusammen mit der Senkeninformation kopiert. Die weiteren Schritte im Algorithmus sind analog zum unmodifizierten Vorgehen.

Wichtig ist, dass aus dem Genom eine eindeutige, gültige Topologie im Ergebnisgraphen generiert werden kann. CAN ist ein klassisches Bussystem mit „Daisy-Chain“-Topologie, das heißt mit Serienschaltung der Knoten. Dafür werden an einer durch-

gehenden Kommunikationsleitung über einen Abzweig, den so genannten „Stub“, die einzelnen ECUs angeschlossen. Dieser „Stub“ darf auf Grund der Interferenz durch Reflexionen nur eine begrenzte Länge aufweisen, nach den für Fahrzeuge gültigen Normen zwischen 30 cm [iso03] und 100 cm [SAE02]. Der Bus kann damit als Punkt-zu-Punkt-Verbindung zwischen den Knoten approximiert werden. Eine CAN-Topologie ist somit gültig, wenn nicht mehr als zwei Kanten vom Typ CAN mit einem Knoten verbunden sind. Ungültige Lösungen werden, wie bereits erwähnt, mit Kosten von ∞ belegt und beim Selektionsschritt aussortiert.

Als Erweiterung zu dem Netzaufbau aus Kapitel 4.3.2 wird auch dann ein Sternknoten gesetzt, wenn nur eine Kante vom Typ Ethernet, dafür aber zusätzlich eine oder zwei Kanten vom Typ CAN am Knoten enden.

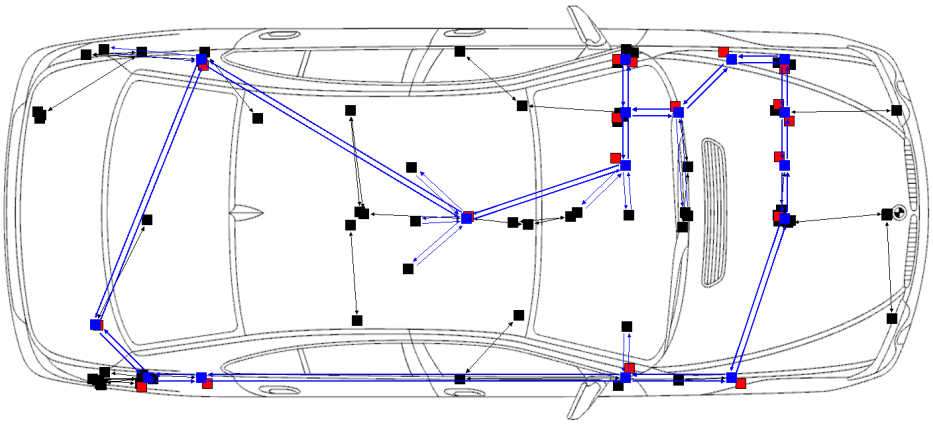


Abbildung 4.20: Heterogenes Referenznetz ohne Demand

Zur Kalibrierung und Evaluierung der Konvergenz des modifizierten Algorithmus wurden mehrere Testläufe durchgeführt. Ein ausgewähltes Ergebnis ist im Graphen von Abbildung 4.20 visualisiert. Dabei handelt es sich bei den blauen Linien um Kanten vom Typ Ethernet und bei den schwarzen um jene vom Typ CAN. Es bildet sich, wie zu erwarten, ein Ring zur Verbindung der sicherheitskritischen Knoten aus. Knoten, die als Quelle oder Senke für Video- und Audiocontent dienen, sind auch über Ethernet mit dem Backbone verbunden. Alle weiteren Knoten bilden lokale Subnetze und sind über nahe liegende Gateways mit dem Kernnetz verbunden.

Dieses weitestgehend vorhersagbare Ergebnis wird nun unter Berücksichtigung der Datenübertragung betrachtet. Dafür wird nun die Worst Case Transmission Time (WCTT) mit in den Prozess aufgenommen und die Berechnung der Fitness einer Lösung entsprechend erweitert, dargestellt in Abbildung 4.21.

Nachdem über die Abbildungsregeln aus dem Genom ein physikalischer Graph generiert wurde, werden die VK für die Verkabelung und die Bauteile F_p wie gewohnt berechnet. Anschließend erfolgt ein Routing der Kommunikationsverbindungen, basierend auf dem kürzesten Pfad (siehe Ergebnisse aus Kapitel 3.1.5). Für diese Verbindungen wird nun analog zu dem in Kapitel 3.1.4 beschriebenen Verfahren die WCTT berechnet. Der CAN ist mit 1 Mbit/s und dem Base-Frame-Format festgelegt. Der

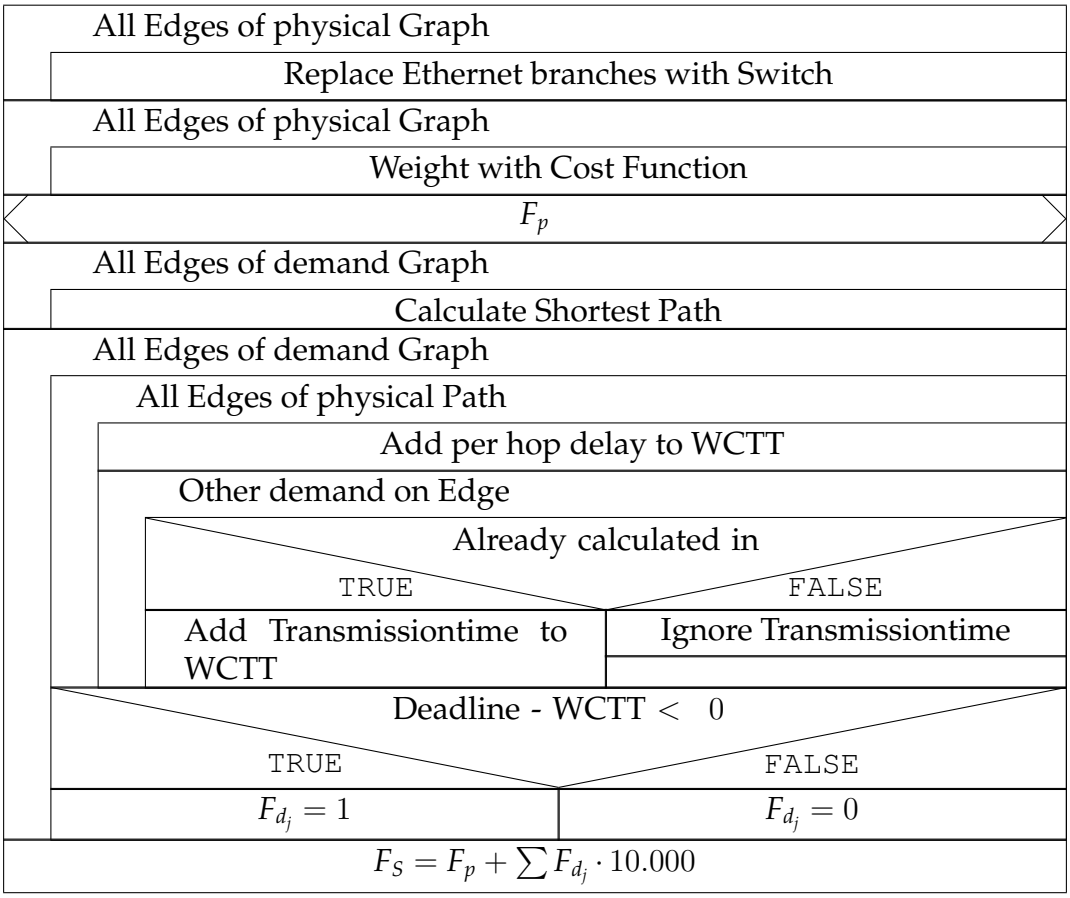


Abbildung 4.21: Struktogramm „Calculate Fitness“

gesamte Bus wird als eine Kante modelliert, das heißt, Daten auf einem beliebigen Teilstück des CAN-Netzes belegen zu diesem Zeitpunkt das gesamte Subnetz. Implementierungsdetails sind in 6.3 näher erläutert. Diese berechnete WCTT wird mit der im FIBEX gespeicherten Deadline verglichen und bei Verletzung der Deadline $F_{d_j} = 1$ gesetzt.

$$F_p \in \mathbb{R}\,[0; 9.999] \tag{4.8}$$

$$F_{d_j} \in \mathbb{N}\,[0; 1] \tag{4.9}$$

$$F_s = F_p + \sum_j F_{d_j} \cdot 10.000 \tag{4.10}$$

In Formel (4.10) ist die modifizierte Berechnung der Fitness F_s einer Lösung zusammengefasst. In der verwendeten Methode wird also nicht direkt die WCTT, sondern die Anzahl der überschrittenen Deadlines eingerechnet. Dies hat sich bei der Evaluierung verschiedener Methoden in Kapitel 3.1.5 als besonders zielführend dargestellt. Da das primäre Ziel sein muss, ein möglichst kostenoptimiertes, aber gültiges Netz zu erhalten, wird die Deadlineverletzung in der Gesamtsumme stark gewichtet. Durch

die Verwendung von 10.000 als Gewichtungsfaktor wird sichergestellt, das bereits eine Deadlineverletzung nicht durch Optimieren der Topologie ausgeblendet werden kann.

Parameter	Wert	Parameter	Wert
l_{Eth}	10,0	l_{CAN}	10,0
c_{Eth}	2,00	c_{CAN}	2,00
b_{Eth}	8,00	b_{CAN}	50%
e_{Eth}	7,00	e_{CAN}	50%

Tabelle 4.3: Standardwerte bei heterogenem Netz

$$F_p = L_{Eth} + B_{Eth} + E_{Eth} + L_{CAN} + b_{CAN} \cdot B_{CAN} + e_{CAN} \cdot E_{CAN} \tag{4.11}$$

Als Gewichtungsfaktoren der physikalischen Kosten F_p werden die Parameter aus Tabelle 4.1 um die Kosten der CAN-Integration (Tabelle 4.3) erweitert. Dabei wurde als möglichst realitätsnahe Annahme der Faktor 50% angesetzt. Die Gewichtungen für Steckverbindungen und Kabel sind damit zwar gleich hoch, CAN wird jedoch mit ungerichteten Kanten, also mit nur einer Leitung pro Verbindung und nicht wie bei Ethernet mit zwei Leitungen für den bidirektionalen Betrieb modelliert. Das CAN-Gateway in den Sternknoten wird als ein mit den Korrekturfaktoren b_{CAN} und e_{CAN} gewichteter zusätzlicher Ethernet-Port dargestellt.

Ergebnis

Durch die stärkeren Einschränkungen bei der Aufgabe, eine gültige Topologie zu generieren, wird ein Einfluss auf die Konvergenz des Algorithmus erwartet. Dieser ist, im Vergleich zum unmodifizierten Algorithmus, in Diagramm 4.22 dargestellt. Zu erkennen ist ein deutlich schlechterer, jedoch stetiger Verlauf des evolutionären Algorithmus. Das optimale Ergebnis wird erst später, im Bereich der 2.000 Generation, erreicht. Auch ist zu sehen, dass gegenüber der homogenen Lösung eine Einsparung der VK erreicht wurde. Die Betrachtung des Verkehrsaufkommens und der damit verbundenen Verzögerungen ist in diesem Aufbau notwendig, da durch die geringere Datenrate auf dem CAN sowie durch das gemeinsame Medium in der Busarchitektur hohe Verzögerungen auftreten und somit Knoten teilweise nicht am CAN angeschlossen werden können.

Interessant ist das sich ergebende Netz, das als physikalischer Graph in Abbildung 4.23 visualisiert ist. Die Knoten mit den höchsten Anforderungen sind über minimale Zwischenstationen miteinander verbunden. Dies sind in unserem Fall die ECUs, welche für die Höhenstandsregelung (vgl. Kapitel 2.4.2) verantwortlich sind, da dort Regelkreise mit hoher Frequenz bearbeitet werden müssen. Dadurch bildet sich eine Kreuz-Struktur aus, da diese ECUs hauptsächlich an den vier Rädern sitzen. Damit

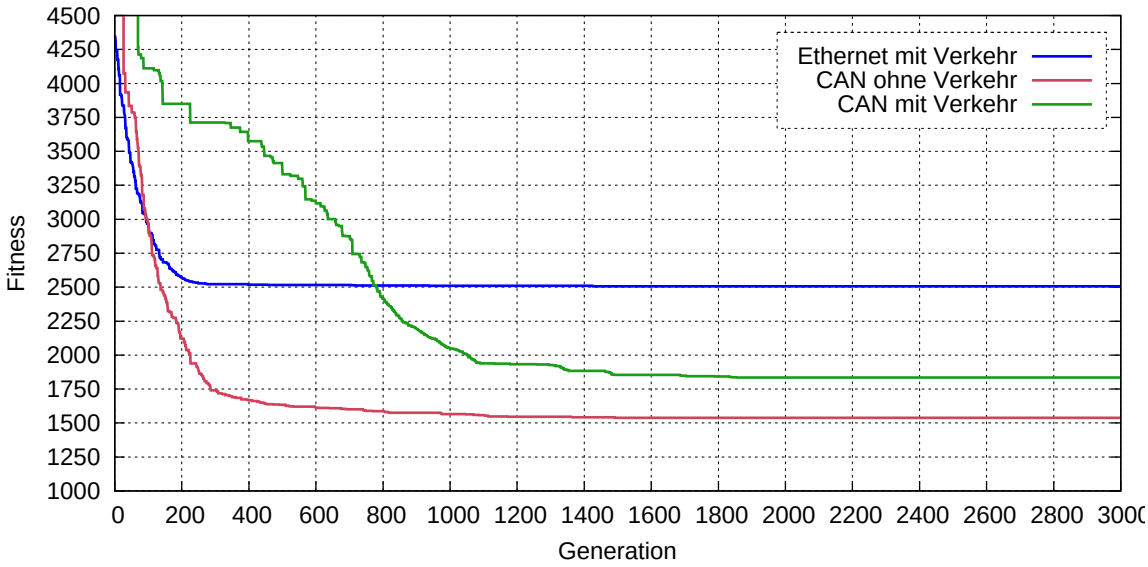


Abbildung 4.22: Konvergenz des Genetischen Algorithmus

verringert sich die Verzögerung auf den anderen Kanten und mehr Knoten können in Subnetze verschoben werden. An 15 der 20 verbauten Sternknoten ist ein CAN-Bus angeschlossen, dort werden entsprechende Gateway-Bausteine benötigt.

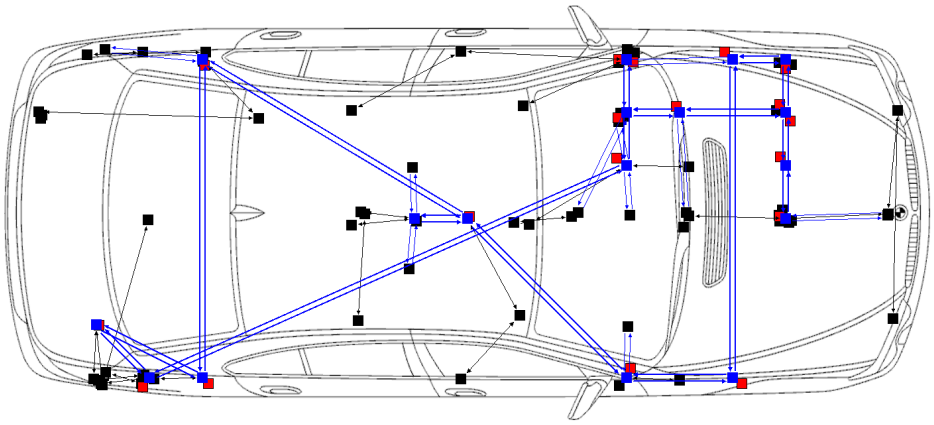


Abbildung 4.23: Resultierendes heterogenes Netz

Zur Überprüfung des Algorithmus wurden Verbindungen mit nicht erreichbaren Zielen hinzugefügt. Die Verbindungen, welche ungefähr 1% ausmachen, haben beim Routing über Ethernet nach dem kürzesten Pfad eine Laxity < 0 (Definition Kapitel 2.4.1), verletzen also die Deadlinebedingungen. Durch den Einsatz von Subnetzen auf CAN verringert sich global betrachtet die Laxity, weil durchschnittlich eine höhere WCTT auftritt. Dies ist anhand der kritischen Verbindungen aus Diagramm 4.24 auch herauszulesen. Die Anzahl der Verbindungen mit Laxity < 0, im Diagramm an der vertikalen Linie bei 0 abzulesen, bleibt erwartungsgemäß konstant. Damit verschlechtert der Einsatz von heterogenen Netzen das Ergebnis durch die hohe Gewichtung der

Deadlineverletzung nicht weiter.

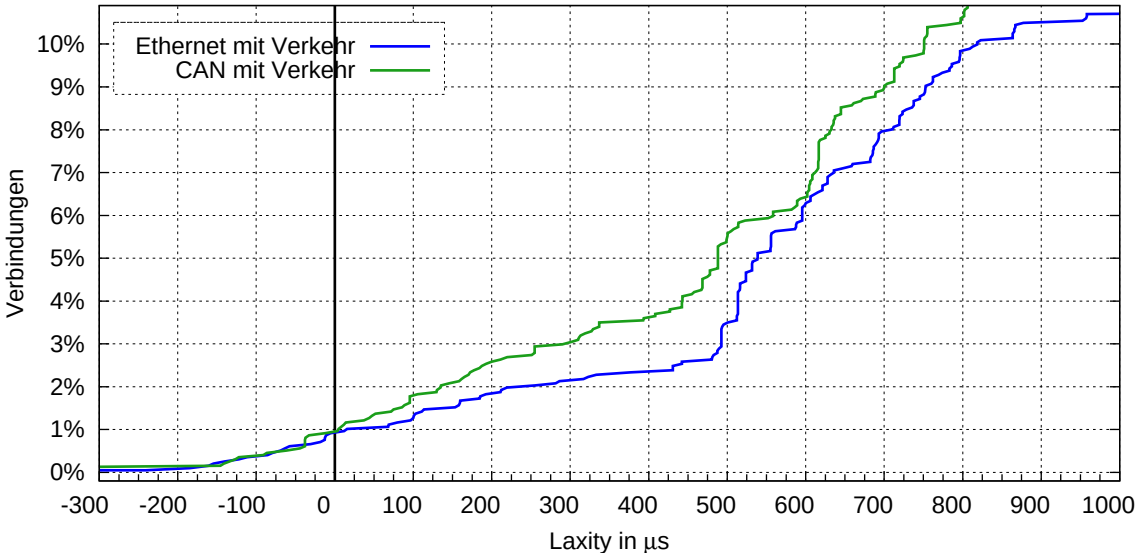


Abbildung 4.24: Laxity im heterogenem Netz

4.4 Komplexität

Ein großer Vorteil des hier vorgestellten Algorithmus ist die geringe Komplexität. Dies bezieht sich einerseits auf die Implementierung, die für den reinen GA nur etwas über 1.000 Zeilen Programmcode in C++ benötigt hat, andererseits auf den Speicherverbrauch, da nur die begrenzte Anzahl der Einträge der aktuellen Population gespeichert wird und keine Informationen aus vergangenen Generationen vorgehalten werden müssen.

Zur Laufzeit lassen sich nur schwer vergleichende Angaben machen, da sie fundamental von der verwendeten Hardware abhängt. Der zur Erstellung der Ergebnisse der vorliegenden Arbeit verwendete Rechner ist ein 8 Kern Intel-Xeon E5520 mit 2,26 GHz und einer CUDA-kompatiblen Graphikkarte nVidia Corporation G80 GeForce GTX 260 mit 192 Kernen und 576 MHz. Pro Generation benötigte diese Hardware $\approx$ 5 Sekunden für das Problem aus Kapitel 4.3.3 und $\approx$ 9 Sekunden für das komplexere Problem aus Kapitel 4.3.5.

Zusammengefasst ergibt dies unter $1{,}5$ Stunden (kostenoptimierte Vernetzung) beziehungsweise $7{,}5$ Stunden (heterogene Netze) für eine sichere Konvergenz zur optimalen Lösung. Dies und der Speicherverbrauch sind angesichts des Optimierungsproblems eines aus 83 Knoten bestehenden Netzes als sehr gering einzuordnen.

4.5 Zusammenfassung

Der Einsatz von Ethernet, und damit die Veränderung der vorherrschenden Topologie von einer Daisy-Chain zu einer Sternform mit zentralen Sternknoten zur Verbindung der einzelnen Electronic Control Units (ECUs), ist für Bordnetze im automotiven Bereich ein neues Themenfeld. Ziel der Arbeiten, die in diesem Kapitel dargestellt wurden, ist es, die notwendigen Änderungen in der verwendeten Hardware greifbar zu machen.

Zu diesem Zweck wurden Metriken entwickelt, um mit Hilfe einer Virtuelle Kostenfunktion (VK) den notwendigen zusätzlichen Aufwand beim Verbau der Sternknoten im Bordnetz quantifizierbar zu machen. Dabei wurden Verkabelungsaufwand, Energieverbrauch und Bauraum als die treibenden Elemente der entstehenden Kosten identifiziert. Es wurde ein Modell entwickelt, um aus einem physikalischen Graphen mit ECUs als Knoten und Leitungen als Kanten einen eindeutigen Netzaufbau mit Sternknoten zu generieren. Es wurde eine Methode zur Berechnung der auftretenden Kosten auf der Basis dieses Netzes entwickelt und die Auswirkung von unterschiedlichen Gewichtungen der Teilkomponenten dargestellt. Annahme ist dabei der Verbau der notwendigen Sternknoten im Gehäuse einer ECU. Diese Maßnahme wurde unter Berücksichtigung der notwendigen Bauteile und der üblichen Prozesse beim Fahrzeughersteller als ideale Lösung identifiziert.

Die Quantifizierung der zu erwartenden VK ermöglicht es nun, Methoden zu entwickeln, um genau diese notwendigen Aufwände zu minimieren. Da die betrachteten Netze eine hohe Komplexität aufweisen, wie an dem Referenznetz (siehe Kapitel A.2) mit über 80 Knoten und fast 2.000 Kommunikationsanforderungen dargelegt wurde, mussten aufwendige Methoden zur Optimierung angewendet werden. Angepasst wurde ein Genetischer Algorithmus (GA), um auf das vorliegende Optimierungsproblem angewendet werden zu können. Neben dem Aufbau des Genoms wurde ein zusätzlicher Immigrationsschritt sowie ein Reparaturschritt entwickelt, der wegen der Besonderheiten des Optimierungsproblems notwendig ist, aber keine negativen Auswirkungen auf den evolutionären Ablauf haben darf. Weiter wurden unterschiedliche Rekombinations- und Selektionsmechanismen untersucht und eine jeweils modifizierte Version davon wurde eingesetzt.

Als weitere Möglichkeit der Kosteneinsparung wurden heterogene Ansätze in der Kommunikationsarchitektur untersucht. Die Idee ist, ein physikalisches Kommunikationsnetz mit geringerer Komplexität zur örtlichen Aggregation der Daten zu verwenden, die dann über ein Gateway in das zentrale System eingespeist werden. Gerade ECUs der Komfort-Domäne mit geringen Kommunikationsanforderungen und späten Deadline-Grenzen können so mit niedrigeren Kosten an das Netz angekoppelt werden. Die Gatewayfunktionalität übernimmt dabei ein erweiterter Sternknoten, was nur geringen zusätzlichen Hardwareaufwand erfordert. Untersucht wurde der Einsatz von Controller Area Network (CAN) als Aggregationsnetz. Da CAN eine geringere Datenrate und eine höhere Verzögerung aufweist, muss im GA die Einhaltung der Kommunikationsanforderung bei jedem Schritt und für jeden Demand

durch Routing über den kürzesten Pfad (vgl. Kapitel 3.1.6) und Berechnung der Worst Case Transmission Time (WCTT) (vgl. Kapitel 3.1.4) überprüft werden. Die Ergebnisse zeigen, das der entwickelte GA diese Komplexität handhaben kann und dass durch den heterogenen Ansatz beim Referenznetz (siehe Kapitel A.2) die VK deutlich gesenkt werden konnte.

5 Ein effizientes Kommunikationsprotokoll

Neben dem verwendeten Medienzugriffsverfahren wirken sich auch die Protokolle der höheren Schichten auf die Performance der Datenübertragung aus. In diesem Kapitel werden die auf Ethernet aufbauenden Protokolle untersucht und ein neues, schichtübergreifendes Verfahren wird vorgestellt, das die Kommunikation innerhalb von Bordnetzen effizienter gestaltet.

Besonders wurde darauf geachtet, dass die für den größten Teil der Funktionen benötigten Mechanismen, die sonst in höheren Schichten implementiert wurden, bereits in diesem Protokoll verankert sind. Dies beinhaltet Zeitstempel und Zeitsynchronisierung sowie dynamische Datensicherung. Auch wurde speziell auf Multicast und Subnetting, also das Aufteilen des Gesamtnetzes in einzelne Unternetze, auch mit unterschiedlichen physikalischen Übertragungsprotokollen, Wert gelegt.

5.1 Protokolle der physikalischen Schicht

Das bekannteste und überwiegend zum Transport von Daten verwendete Netzwerkprotokoll der Schicht 3 ist das Internet-Protokoll (IP). Es ist die Basis des Internets und auch im Büroumfeld Standard in der digitalen Kommunikation. Der IPv4-Header ist dabei mindestens 20 Byte lang, der IPv6-Header sogar 40 Byte.

IP ermöglicht eine Kommunikation zwischen Teilnehmern über mehrere Netzsegmente, so genannte Transportnetze, hinweg. Die IP-Adresse ist hierarchisch aufgebaut, womit im lokalen Netz liegende Knoten von entfernten unterschieden werden können. Über Router werden für andere Segmente bestimmte Pakete mit Hilfe der Routingtabelle weitergeleitet.

Bordnetze in verteilten Systemen sind abgeschlossene Einheiten. Der Einsatz von Routing innerhalb eines Bordnetzes ist nicht notwendig. Damit ist der Hardwareadresse immer die gleiche Internetadresse zugewiesen, und die Funktionalität von IP als Abstraktionsschicht wird überwiegend nicht benötigt. Nur zur Kommunikation mit Geräten außerhalb des Bordnetzes muss IP verfügbar sein.

Eine weitere wichtige Funktionalität des Internet-Protokolls ist die Fragmentierung. Bei der Übertragung mit Ethernet gibt es eine maximal mögliche Framelänge, die Maximum Transmission Unit (MTU). Diese Grenze wurde zur Optimierung der Speicherzugriffszeiten und der Auslegung der Hardware eingeführt und beträgt bei Standard-Ethernet 1.500 Byte. Um größere Datenpakete zu übertragen, wie dies bei Multimedia-Strömen häufig der Fall ist (siehe Kapitel B.2), müssen diese von

Protokollen der höheren Schichten in kleinere Portionen aufgeteilt und nach der Übertragung wieder assembliert werden.

Auch diese Funktion übernimmt IP mit Hilfe der Felder *Identification* und *Fragment Offset* im Nachrichtenkopf. Dies ermöglicht, Datenpakete mit bis zu 64 kByte zu übertragen. In den untersuchten Systemen wurde diese Funktionalität fast ausschließlich von Videoübertragungen verwendet, die von Regelkreisen gesendeten Informationselemente sind weit kleiner als die MTU, wie die Auswertung in Kapitel A.2 und die Abbildung A.3 zeigen. Für die Prozesskommunikation wird das Fragmentieren deswegen nicht benötigt.

Andere Schicht-3-Protokolle wie Apples Datagram Delivery Protocol (DDP) und Novells Internetwork Packet Exchange (IPX) sind vollständig vom Markt verschwunden und spielen keine Rolle mehr.

5.2 Ineffizienz der Informationsverteilung

Es gibt drei Hauptverursacher für die Ineffizienz bei der Übertragung der Daten: der Header, die Prüfsummen und die Adressierung. Betrachtet wird die Übertragung über *UDP over IP over Ethernet*. Diese Kombination wird von Konsortien und Fachzeitschriften als zukünftiges Kommunikationsprotokoll vorgeschlagen und dient hier als Grundlage zum Effizienzvergleich.

Die Verschachtelung der Header, also der Nachrichtenköpfe, über die Schichten hinweg hat den Vorteil, dass unterhalb liegende Protokolle transparent gegen andere ausgetauscht werden können, sie hat jedoch den Nachteil, dass häufig dieselbe Information in mehreren Schichten vorhanden ist. Zum Beispiel befindet sich die Längeninformation in Schicht 3 und 4, die Art der Übertragung und damit die notwendigen Mechanismen zur Sicherstellung der Übertragungsqualität in Schicht 2 und 3. Bei der Festlegung auf ein gemeinsames Kommunikationsprotokoll sind auch die Angaben zu der verwendeten Version und dem Protokollidentifier in jedem Paket überflüssig.

Prüfsummen, auch *Checksum* genannt, dienen zur Überprüfung der fehlerfreien Übermittlung des Nachrichtenframes. Unbeabsichtigte Veränderungen der Nachricht auf dem Übertragungsweg, zum Beispiel durch Interferenz oder Speicherfehler, können mit der Prüfsumme in Maßen detektiert und korrigiert werden. Da Fehler bereits in den unteren Schichten erkannt werden und dort fehlerhafte Frames aussortiert werden können, kommen die Prüfsummen der oberen Schichten selten zum Einsatz, müssen aber für jedes Paket individuell berechnet und im Nachrichtenkopf übertragen werden.

In Ethernet/IP-Netzen besitzen alle Teilnehmer eine weltweit eindeutige, fest eingestellte Adresse, die so genannte Hardware- oder MAC-Adresse. Von ihrem Internetprovider werden dem Gerät eine oder mehrere Internetadressen zugewiesen, umgangssprachlich IP genannt. Diese kann sich je nach Ort und Zugangsmethode ändern. Die Verknüpfung zwischen MAC und IP muss von jedem Gerät für jeden Verbindungswunsch mit Hilfe des Address Resolution Protocol (ARP) durchgeführt wer-

den. Solange Hardware nicht ausgetauscht wird und sich im selben Netz befindet, bleibt die Internetadresse und damit die Verknüpfung zur Hardwareadresse gleich und ist als redundante Information nicht notwendig.

Für den Einsatz über offene Systeme wie das Internet sind die verwendeten Protokolle ideal, was eine Durchdringungsrate von nahezu 100% bestätigt. In abgeschlosseneren Systemen, wie sie Bordnetze darstellen, führt diese Universalität jedoch zu erhöhtem Aufwand bei der Verarbeitung der Kommunikationsdaten, was sich wiederum auf die Softwarekomplexität, den benötigten Speicherplatz und die Rechenressourcen und damit auch auf den Stromverbrauch auswirkt.

5.3 uMAP - universal Multilayer Automotive Protocol

Auf den in Kapitel 5.2 festgestellten negativen Eigenschaften des Internet-Protokoll basiert die Entwicklung des universal Multilayer Automotive Protocol (uMAP). Kernpunkte sind neben einer effizienten Übertragung von Regelungsdaten adaptive Adressierung und Kompatibilität zu aktuell im Fahrzeug verwendeten Kommunikationsprotokollen.

5.3.1 Grundlegende Protokolleigenschaften

uMAP vereint, technisch gesehen, mehrere Schichten des OSI-Modells und trägt deswegen die Bezeichnung *Multilayer* im Namen. In Abbildung 5.1 ist das OSI-Modell und die Einordnung von uMAP in den Schichten 3-5, das heißt in Vermittlungs-, Transport- und Sitzungsschicht, dargestellt.

- Als Aufgaben der **Vermittlungsschicht** übernimmt uMAP vor allem die Fragmentierung sowie die Umsetzung und Weiterleitung zwischen den Teilnetzen.
- Für die **Transportschicht** werden die Fehlersicherungs- und Fehlerbehebungsverfahren sowie die Adressierung der Dienste auf einem Gerät ausgeführt.
- Die **Sitzungs-** beziehungsweise **Kommunikationssteuerungsschicht** beinhaltet die Synchronisation der Kommunikation durch Zeitstempel und Quittierungsmechanismen.

uMAP setzt auf unmodifiziertes IEEE 802.3 Ethernet auf. Damit können vorhandene Komponenten wie Netzwerkkarten oder Hardwareblöcke für Ethernet weiter verwendet werden. Grundlegend ist ein gemeinsames Zeitverständnis. Die bei der Kommunikation involvierten Geräte müssen eine synchronisierte, absolut Zeit besitzen. Die Präzision ist dabei einzig von der implementierten Funktionalität abhängig und wird aus diesem Grund nicht im Protokoll spezifiziert.

Das Feld *Timestamp*, der Zeitstempel, umfasst 4 Byte. Es wird dabei eine komplette Stunde in absoluter Zeit adressiert, er repräsentiert also die Minutenangabe auf eine μ-Sekunde genau. Dabei legt das erste Bit die zuständige halbe Stunde als *even* =

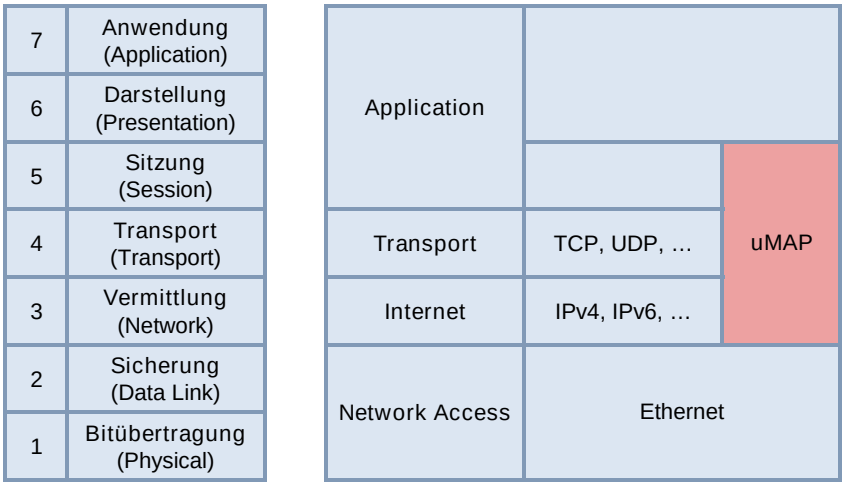


Abbildung 5.1: uMAP im OSI-Modell

$00-29$ und *odd* $= 30-59$ Minuten fest, die weiteren Bits die μ-Sekunde innerhalb dieses 30-minütigen Blocks. Damit können Zeitpunkte, die 30 Minuten in der Zukunft oder 30 Minuten in der Vergangenheit liegen, auf eine μ-Sekunde genau angegeben werden. Bei einer Genauigkeit von einer μ-Sekunde können damit Zeitpunkte mit einer Auflösung von 1 MHz festgelegt werden, was die in dieser Klasse von Bordnetzen vorkommenden Regelalgorithmen weit übersteigt, wie in Kapitel 2.4 dargelegt.

Diese zeitliche Auflösung muss aber indirekt auf bis zu 100 kHz eingeschränkt werden, weil das Nachrichtenkopffeld *Timestamp* auch die Aufgabe der Serialisierung übernimmt. Die Zuordnung von Quittierungen, Löschaufträgen und Fragmenten wird über den Zeitstempel vorgenommen. Deswegen müssen sich Nachrichten mit den gleichen Adressdaten um mindestens 1 µs unterscheiden, denn Nachrichten mit dem gleichen Zeitstempel und der gleichen Adresse überschreiben die vorhergehenden Nachrichten. 1 µs entspricht bei Gigabit-Ethernet der Übertragung von nur 125 Byte. Die Verwendung der Zeitstempel wird in Kapitel 5.3.2 aufgezeigt.

Zur Synchronisation der Uhren wird das Precision Time Protocol (PTP) auf der Basis des IEEE 1588-Standards vorgeschlagen. Dies ermöglicht eine Synchronisation der internen Uhren auf mindestens eine μ-Sekunde Genauigkeit, mit dem Einsatz von Hardwareunterstützung sogar auf wenige Nanosekunden genau. Alternativ können sich die Steuergeräte auch über DCF77-Uhren oder das GPS-Signal mit hinreichend genauen Zeitinformationen versorgen.

5.3.2 Versand und Verarbeitung von Information

Die Verarbeitung der übertragenen Informationen in diesem Protokoll basiert auf den Prinzipien des *memory mapping*, also dem Zugriff auf Daten in Speicherblöcken. Dafür wurden vier Arbeitsmodi des Protokolls festgelegt, die im Nachrichtenkopffeld *Mode of Operation (MD)* gesetzt werden: *Write*, *Modify*, *Read* und *Poll*.

00 **Write:** Schreibt Daten in einen Speicherblock. Die im Datenpaket empfangenen Daten werden zu dem angegebenen Zeitpunkt in den adressierten Speicherbereich geschrieben. Bis zu diesem Zeitpunkt werden die Daten in einem Puffer im Kommunikationsmanagement vorgehalten. Dies entspricht weitestgehend dem normalen Vorgehen beim Empfang von Datenpaketen in anderen Protokollen.

01 **Modify:** Entspricht dem *Write*-Modus, nur werden die Daten im adressierten Speicherbereich mit den empfangenen Daten über eine XOR-Logic verknüpft. Das bedeutet, jedes empfangene 1-Bit alterniert das entsprechende Bit im Speicher.

10 **Read:** Kennzeichnet einen gelesenen Wert aus dem Speicher. Die Aussendung eines Paketes im Modus *Read* kann von höheren Schichten beziehungsweise der Funktion selbst initiiert werden, zum Beispiel zyklisch oder bei definierten Ereignissen. Der Zeitstempel definiert den Lesezeitpunkt der Daten. So kann trotz Jitter bei der Übertragung das exakte Alter der Daten festgestellt werden. Ein *Read*-Rahmen ohne Datenteil löst einen Lesezugriff auf den adressierten Speicherbereich und eine Antwortnachricht zu dem gesetzten Zeitpunkt aus. Ein Zeitstempel in der Vergangenheit führt zum sofortigen Lesezugriff.

11 **Poll:** Ähnelt dem leeren *Read*-Rahmen, nur muss die Antwort in höheren Schichten, zum Beispiel der Funktion, generiert werden. Der Modus wird eingesetzt, wenn die gewünschte Information exakt zu dem angegebenen Zeitpunkt generiert werden soll.

Alle über die Modi gesetzten Befehle, die eine Ausführung in der Zukunft mit dem angegebenen Zeitstempel auslösen, können natürlich bis zu diesem Zeitpunkt wieder deaktiviert werden, indem ein Löschbefehl gesendet wird.

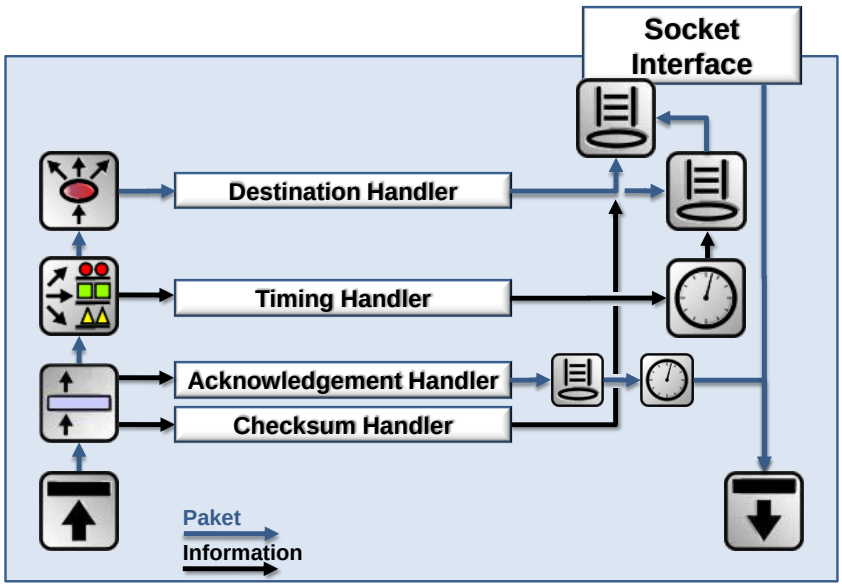


Abbildung 5.2: Blockschaltbild uMAP-Socket

Das Blockschaltbild der vorgeschlagenen Implementierung ist in Abbildung 5.2 dargestellt. Das Paket landet nach dem Durchlaufen der physikalischen Übertragungsschicht mit deren Mechanismen am Eingang des uMAP-Blocks. Dort wird im *Checksum Handler* die Prüfsumme kontrolliert und optionale Fehlerbehandlungsmechanismen werden durchgeführt. Das Ergebnis hinsichtlich der Frage, ob ein Übertragungsfehler im Paket vorliegt, wird als Metainformation für die höheren Kommunikationsschichten beigefügt und das Paket trotzdem nicht vernichtet. Diese Entscheidung wird nicht in der uMAP-Schicht getroffen. Der *Acknowledgement Handler* sendet, falls angefordert, die Quittierung bei korrektem Empfang des Paketes.

Für die Auswertung der Zeitstempel und das Planen der Übermittlung ist der *Timing Handler* zuständig. Aus Implementierungssicht wird ein Ringpuffer vorgeschlagen, in dem die Pakete inklusive Metadaten einsortiert werden. Der *Timing Handler* steuert dann über Sprungbefehle den Zeiger im Ringpuffer, um die Aussortierung nach Zeitplan zu gewährleisten.

Der *Destination Handler* sortiert das Paket in den zur Adresse passenden Ausgangspuffer. Dies kann ein globaler Puffer sein, in dem alle Schreibzugriffe auf den Speicher zwischengepuffert werden, bis diese ausgeführt werden können. Dies entspricht mehr der *Memory-mapped*-Architektur. Alternativ kann ein lokaler Puffer verwendet werden, bei dem die Funktion aus der höheren Schicht die Daten abholen kann, was dem klassischen Berkeley Socket entspricht. Über eine spezielle Art des Socket ist auch eine systemtransparente Tunnelung von IP-, TCP- und UDP-Daten in uMAP-Paketen möglich. Diese können dann direkt den dem Betriebssystem zugehörigen Schnittstellen übergeben werden, was die Kommunikation mit nicht für den Einsatz in uMAP modifizierten Anwendungen ermöglicht.

5.3.3 Header-Format und Frameaufbau

Eine Besonderheit von uMAP ist, dass mehrere Datenabschnitte in einem Frame übertragen werden können. Einige Schicht-2-Protokolle haben minimale Paketlängen, wie zum Beispiel Ethernet mit 64 Byte. Werden nur wenig Nutzdaten übertragen, müssen diese mit unnützen Daten bis zur minimalen Größe aufgefüllt werden, das so genannte *Padding*. Die Aggregation mehrerer Nutzdatenblöcke in einen Übertragungsframe, um den Verlust zu verringern, muss nicht durch ein höher liegendes Protokoll organisiert werden, sondern ist bereits in uMAP integriert. Abbildung 5.3 zeigt den Rahmen von uMAP im Vergleich zu einem konventionellen Ethernet/IP-Rahmen.

Dabei kennzeichnet das Kopffeld *Length*, Länge, die Menge in Bytes bis zum nächsten Datenblock. Maximal möglich sind 1.021 (0x3FD) Bytes pro Block oder nur ein Block, der den gesamten Rahmen füllt, definiert über die Längeninformation 0x3FE. Ein uMAP-Kopf mit der Länge 0 kennzeichnet das Ende des Rahmens. Die Längeninformation bezieht sich dabei auf den Nachrichtenkopf mitsamt den Nutzdaten.

Der Aufbau des Nachrichtenkopfes selbst ist in Abbildung 5.4 zu sehen. Die einzelnen Felder werden in den nächsten Unterkapiteln beschrieben oder wurden bereits erläutert. Genaue Inhalte der Felder sind im Anhang C nachzulesen.

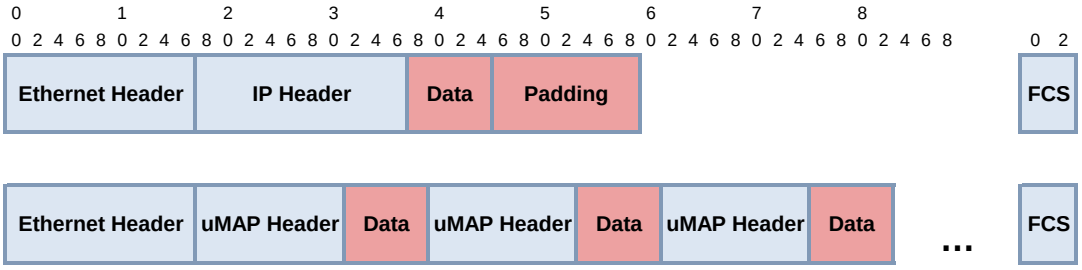


Abbildung 5.3: uMAP-Frame Aufbau

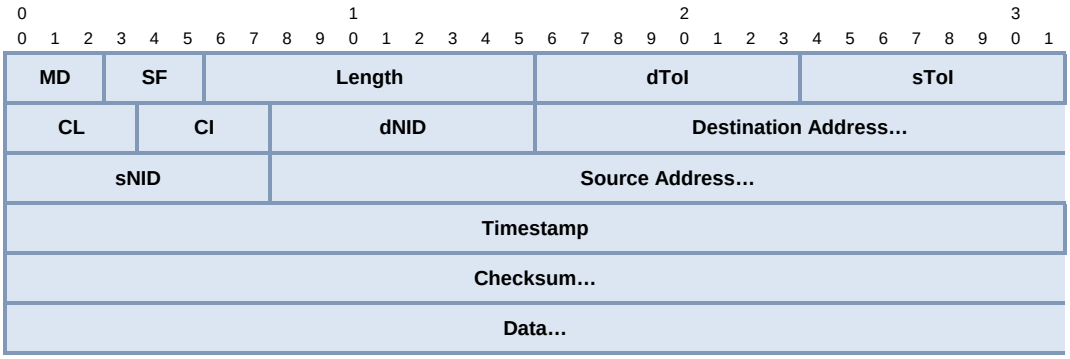


Abbildung 5.4: uMAP-Header Aufbau

Adressierung

Für die Adressierung sind im Nachrichtenkopf jeweils für Sender und Empfänger drei Felder vorgesehen. Dabei handelt es sich um den *Type of Identifier (ToI)*, welcher das verwendete Adressformat und die Adresslänge angibt, den optionalen *Network Identifier (NID)*, der das adressierte Netzwerk festlegt, sowie die Adresse selbst.

Eine Besonderheit von uMAP ist, dass das Format und die Länge des Adressfeldes flexibel definiert werden können. Somit wird eine direkte und transparente Adressierung von Geräten über Netzgrenzen und sogar über verschiedene physikalische Schichten hinweg ermöglicht. Dies ist sinnvoll, da es in Bordnetzen öfters mehrere verschiedene Standards gibt, was einerseits durch Spezialisierung für eine bestimmte Anwendung, anderseits, wie in Personenkraftwagen, durch den hohen Kostendruck notwendig ist.

Der *ToI* definiert, ob ein *NID* im Adressfeld angegeben ist und welche Länge die Adresse hat. Dies kann zwischen 16 und 144 bit in diskreten Stufen von $2^{n+1} + \{0; 2\}$ Byte, also $2{,}4{,}6{,}8{,}10{,}16{,}18$ Byte, gesetzt werden. Eine Adressfeldlänge von 1 Byte wäre für Protokolle wie Local Interconnect Network (LIN) zwar vorteilhaft, eine ungerade Länge des Nachrichtenkopfes auf Grund der Speicheraufteilung und des Speicherzugriffs moderner Rechensysteme jedoch nicht sinnvoll. Die weiteren 4 bit des Feldes spezifizieren das verwendete Protokoll.

Der *NID* adressiert das gewünschte Subnetz. Dabei kann das lokale Netz, also das Netz direkt an der Netzwerkkarte, ausgewählt werden, was auch dem Verhalten oh-

ne *NID*-Feld entspricht. Einen besonderen Stand hat das Standard-Gateway, Netz 01, das zum Beispiel der Knoten für den Internetzugang sein kann. Dorthin werden alle Nachrichten für Geräte, die sich nicht im Bordnetz befinden, weitergeleitet. Im *NID* können dann 126 Subnetze explizit adressiert werden.

Um eine Trennung von Electronic Control Units (ECUs) im Bordnetz zu ermöglichen, die zwar physikalisch im gleichen Netz, aber logisch durch ihre Funktion oder ihren Einsatzzweck von anderen getrennt operieren, unterstützt uMAP Multicast. Damit können Gruppen von Steuergeräten zusammengefasst und direkt als Gruppe angesprochen werden. Dies bildet sehr gut den aktuellen Aufbau eines Fahrzeugbordnetzes (Kapitel 2.2.1) ab, wo durch separate Netze eine Trennung der Domänen durchgeführt ist. uMAP unterstützt dies auf der logischen Schicht durch die Multicast-Gruppen, von denen 126 Stück adressiert werden können. *NID FF*, als höchste Multicast-Gruppe, sendet in alle erreichbaren Netze durch Fluten.

Mit dem Einsatz von Multicast wird auch die Bustopologie, wie sie in Controller Area Network (CAN), Flexray und anderen Netzen überwiegend verwendet wird, ressourcenschonend abgebildet. In diesen Netzen werden nicht direkt Zielgeräte, sondern eher Informationen adressiert. Jede ECU empfängt alle Daten und filtert nach den für sie relevanten. Eine weitere Senke bedarf keiner Umkonfiguration der vorhandenen Quellen und verursacht keine höhere Datenlast. Dies ist mit dem normalerweise verwendeten Unicast nicht möglich, eine weitere Senke bedeutet dort ein mehrfaches Versenden der Daten. Durch Multicast liegt die Entscheidung, welche Informationen empfangen werden, nicht mehr beim Sender, sondern beim Empfänger.

Die mit dem *NID* und damit in uMAP adressierbaren 253 Netze werden in der physikalischen Schicht auf spezielle Multicast-Ethernetadressen abgebildet, um neben einer direkten Adressierung eine Gruppenadressierung schichtübergreifend zu ermöglichen.

Übertragungssicherung

Zur Steuerung der Übertragungssicherung sind die Felder *Safety Flag (SF)* sowie die optionalen Felder *Checksum Length (CL)*, *Checksum Identifier (CI)* und die Prüfsumme selbst im Nachrichtenkopf vorgesehen. Eine Besonderheit von uMAP bei der Übertragungssicherung ist, dass für Übertragungen individuell verschiedene Methoden mit unterschiedlichem Mehraufwand, aber auch mit angepasster Sicherheit ausgewählt werden können.

Mit dem *SF* wird einerseits festgelegt, ob eine Empfangsbestätigung des Paketes beim Empfänger angefordert wird, wie es zum Beispiel bei verbindungsorientierten Protokollen wie TCP üblich ist. Diese Bestätigung wird bei erfolgreichem Empfang automatisch versendet. Ein Bestätigungspaket wird über den Wert 100 im Kopffeld *MD* gekennzeichnet, die Zuordnung zum versendeten Paket erfolgt über den Zeitstempel.

Des Weiteren wird im *SF* gekennzeichnet, ob eine Prüfsumme mit Paket mitgesendet wird und welches Format beziehungsweise welchen Algorithmus sie hat. Es ist durchaus nicht in jedem Fall sinnvoll, den Mehraufwand durch Berechnung und Versand

der Prüfsumme zu betreiben. Vor allem wenn im physikalischen Übertragungsprotokoll, wie zum Beispiel bei Ethernet, bereits eine Übertragungssicherung durchgeführt wird, können dort Fehler erkannt werden, und die Prüfsummen weiterer Schichten bringen keine weitere Erkenntnis. In uMAP gibt es nun zwei Erweiterungen dafür: Erstens kann die Prüfsumme komplett deaktiviert werden, wenn diejenige der physikalischen Schicht ausreicht, zweitens werden von der physikalischen Schicht als fehlerhaft detektierte Pakete nicht sofort verworfen, sondern zur weiteren Behandlung an den uMAP-Socket weitergegeben, um dort mit optional vorhandenen Prüfsummen eine genauere Fehlererkennung oder -beseitigung durchzuführen.

In uMAP können nun neben den Standardverfahren zur Berechnung der Prüfsumme auch auf die Verbindung zugeschnittene Verfahren eingesetzt werden. Dafür kann im Feld *CL* eine gewünschte Länge des Prüfsummenfeldes im Nachrichtenkopf reserviert und mit dem *CI* das verwendete Protokoll spezifiziert werden. Die Länge wird dabei in 2^n-Byte-Schritten festgelegt. Natürlich müssen beide Kommunikationspartner über eine entsprechende Implementierung des verwendeten Algorithmus verfügen, trotzdem dürfen in abgeschlossenen Systemen auch proprietäre Methoden verwendet werden. Neben den klassischen Verfahren können deshalb im *CI* auch benutzerdefinierte gesetzt werden.

Fragmentierung

Fragmentierung ist, wie bereits erwähnt, das Aufteilen von größeren, zusammenhängenden Datenmengen über Paketgrenzen hinweg. Vor allem bei einer Begrenzung der maximalen Framegröße in unterliegenden Schichten, wie die MTU von 1.500 Byte bei Ethernet, ist dies notwendig. Wie aus der Testsequenz in Kapitel B.2 zu ersehen, sind sogar einzelne Bilder der Videoströme größer als diese Grenze und überschreiten sogar teilweise die für das Internet-Protokoll festgelegte obere Schranke von 64 kByte.

Auch uMAP unterstützt die Fragmentierung. Auf Grund der oben erwähnten Probleme wird keine höchste erlaubte Paketgröße angegeben. Die Steuerung der Fragmentierung erfolgt über die bereits vorhandenen Kopffelder *Length* und *Timestamp*. Ein Frame mit dem Eintrag 0x3FF kennzeichnet ein fragmentiertes Datenpaket. Das Paket verwendet den kompletten Frame, es darf keinen weiteren Block mehr im Rahmen geben. Um eine *out-of-order transmission* zu verhindern, also das falsche Zusammensetzen der Fragmente durch zeitlich vertauschten Empfang der Pakete, wird das *Timestamp*-Feld für jedes Fragment um $+1$ inkrementiert. Der Zeitstempel im ersten Paket entspricht dabei dem für die Nachricht gesetzten Zeitpunkt. Eine Längenangabe $\neq$ 0x3FF kennzeichnet das Ende der fragmentierten Übertragung. Eine Einschränkung besteht insofern, als kein erneutes Fragmentieren ohne vorheriges Reassemblieren möglich ist, wie es im Internet-Protokoll vorgesehen ist.

Eine Fragmentierung wird in uMAP zwar unterstützt, auf Grund von möglichen Übertragungsfehlern und dem potentiellen Verlust der vollständigen Nachricht wird dies jedoch nur bedingt empfohlen. Falls es von der zeitlichen Abfolge möglich ist,

sollte ein Protokoll oberhalb des uMAP verwendet werden, das ein separates Anfordern der bei der Übertragung verlorenen Segmente ermöglicht.

5.3.4 Subnetze und Router

Wie bereits in der Einleitung zu uMAP erwähnt, ist die transparente Kommunikation mit anderen Teilnetzen grundlegende Funktionalität. Wie bei der Adressierung beschrieben, ist dafür das Feld *NID* zuständig.

Neben einer reinen Übermittlung der Pakete in andere Subnetze ist aber auch die direkte Adressierung von Geräten möglich, die über andere physikalische Protokolle als Ethernet angebunden sind. Als stellvertretendes Beispiel wird in diesem Kapitel ein an das Netz angeschlossener CAN-Bus betrachtet, wie er in Kapitel 4.3.5 als kostenoptimierte Vernetzung vorgeschlagen wird. Die Grundlagen sind aber auch auf andere Netze anwendbar. Der Aufbau eines CAN-Frame und die entsprechenden Header bei der Übertragung von Ethernet nach CAN und umgekehrt sind in Abbildung 5.5 dargestellt.

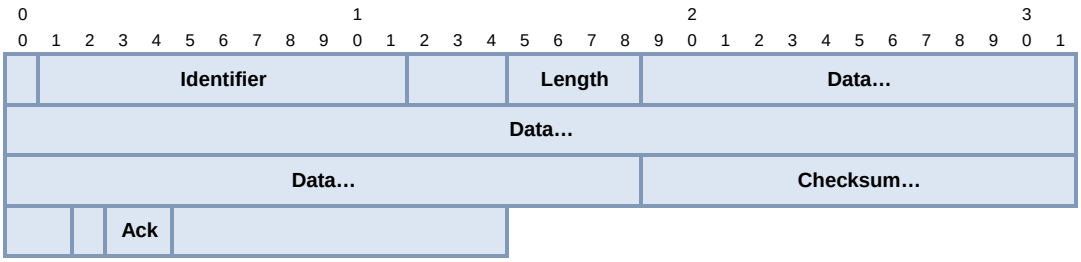
Durch die flexible Adressierungsmöglichkeit mit dem *dToI* kann direkt ein CAN-Empfänger angegeben werden. Der Inhalt des Feldes ist in diesem Beispiel 0x81, die Adresslänge damit 2 Byte zuzüglich des *dNID*-Feldes. Von diesen 16 bit werden die ersten 5 mit Nullen aufgefüllt und die hinteren 11 für den CAN-Identifier verwendet.

Dem Router, also dem vermittelnden Knoten zwischen den Netzen, wird ein entsprechender Identifier für das Subnetz zugewiesen. Dieser muss dann in der Nachricht im Feld *dNID* angegeben werden. Die eigentliche Adressierung des Routers kann nun direkt vorgenommen werden, indem dessen Hardwareadresse direkt im darunterliegenden Ethernet-Frame verwendet wird, oder indirekt mit Hilfe der dem *NID* zugeordneten Multicast-Adresse.

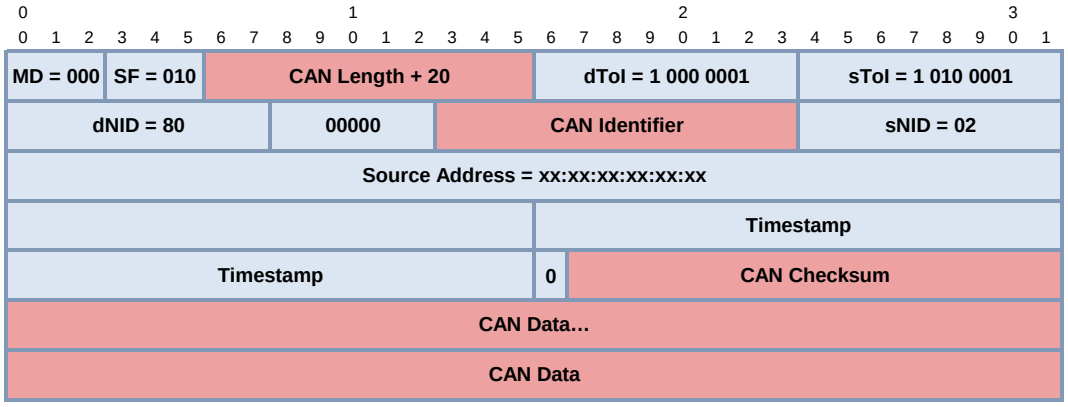
Die Felder *CL* und *CI* sind nicht notwendig, da bereits über das *Safety Flag* und den *dToI* indirekt die Länge des Feldes und die Verwendung des für CAN spezifizierten Algorithmus festgelegt wurde. Würde hier ein benutzerdefinierter Algorithmus gesetzt werden, müsste dieser im Gateway überprüft und dann durch einen CAN-spezifischen ersetzt werden, bevor der Nachrichtenframe weitergeleitet wird. Mit Blick auf den Ressourcenverbrauch wird dies nicht empfohlen.

5.4 Implementierungsdetails

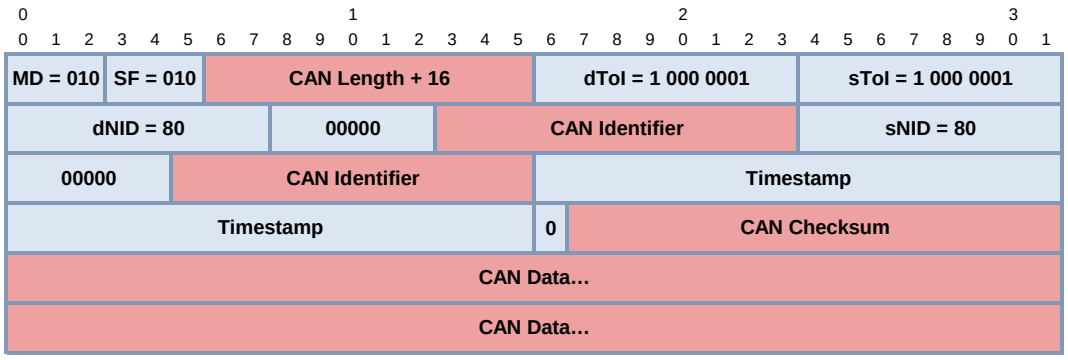
In diesem Kapitel wird kurz auf die Details des Referenz-Implementierungsvorschlages und die notwendigen Einschränkungen für eingebettete Systeme eingegangen. Für eine Implementierung in einer hardwarebasierten Verarbeitung ist gerade das dynamische Verändern der Nachrichtenkopflänge durch unterschiedliche Adresstypen eine hohe Herausforderung.



(a) CAN Frame



(b) Ethernet nach CAN



(c) CAN nach Ethernet

Abbildung 5.5: uMAP-zu-CAN Übersetzung

Eine rein hardwarebasierte Verarbeitung wird vor allem bei ECUs mit wenig Komplexität verwendet, zum Beispiel bei der Sensordatenvorverarbeitung, der Aktoransteuerung und bei Schaltpanels, wo keine Kundenfunktionalität direkt läuft, sondern nur Informationen zwischen Kommunikationsnetz und Fahrzeugkomponenten ausgetauscht werden. Gerade dort ist die Flexibilität der Adressierung in diesem Maße nicht zwingend notwendig.

Das Paket durchläuft erst die physikalische Schicht in Form des Ethernet-Layers, wo bereits eine Aussortierung nicht für das Gerät bestimmter Nachrichten vorgenommen wird. Dies ist vor allem wichtig, damit die ECU baldmöglichst wieder in einen Stromsparzustand wechseln kann, um nicht potentiell knappe Energieressourcen zu verschwenden.

Die Längeninformation des Nachrichtenkopfes wird aus den Feldern *dToI*, *sToI* und dem *SF* in Verbindung mit dem *CL* berechnet. Die entsprechenden Berechnungsgrundlagen sind in den vorherigen Kapiteln erläutert. Die Felder mit flexibler Länge werden dabei immer auf ein volles Datenwort, also ein Vielfaches von 2 Byte aufgefüllt, um ein problemloses Ablegen im Speicher oder serielles Verarbeiten über einen 8 bit-Bus zu ermöglichen. Dieses Padding mit 0-bits erfolgt dabei am Anfang des Feldes, um die Vergleichsoperationen zu vereinfachen. Trotzdem muss natürlich die höchstmögliche Länge im Speicher bzw. im Register reserviert werden, was aber durch die Begrenzung der Felder in ihrer Größe möglich ist.

Notwendig ist es gerade bei rein hardwarebasierter Verarbeitung, aber auch bei leistungsschwachen Mikrocontrollern, die Vielfalt der verwendeten Adressen und Prüfsummenalgorithmen einzuschränken. Diese Einschränkungen können in Lastenheften der Kommunikation während des Produktentstehungsprozess (PEP) festgelegt werden, ohne die Gültigkeit des uMAP für die anderen Steuergeräte einzuschränken.

5.5 Zusammenfassung

Das universal Multilayer Automotive Protocol (uMAP) als ein schichtübergreifendes Protokoll zur Übertragung von Daten in Bordnetzen hat viele qualitative Vorteile. Diese, aber auch der bei dem hier untersuchten Referenznetz sichtbare quantitative Vorteil wurden in diesem Kapitel diskutiert. uMAP beinhaltet eine Funktionalität, die direkt auf den Einsatz in Bordnetzen abgestimmt ist und normalerweise von auf TCP/IP oder UDP/IP bauenden Protokollen noch höherer Schichten im OSI-Modell zur Verfügung gestellt wird. Gerade die Zeitsynchronisierung ist neben anderen Funktionen hier besonders zu erwähnen. Die Verarbeitung vieler Schichten in der Kommunikation mit dem Ein- und Auspacken der Nutzdaten aus den verschachtelten Rahmen führt zu einem quantifizierbar höheren Speicher- und Rechenleistungsverbrauch bei der Verarbeitung. Mit uMAP können somit potentiell Kosten bei den Bauteilen eingespart werden. Des Weiteren ermöglichen die an die *Memory-mapped*-Architektur angelehnten Betriebsmodi eine Verarbeitung in auf leistungsschwachen Mikrocontrollern oder reinen Logikelementen basierenden Electronic Control Units.

Die Möglichkeit, größere zusammenhängende Datenmengen verteilt über mehrere Pakete mit Hilfe der Fragmentierungsfunktion zu übertragen, unterstützt das direkte Senden von Bildern eines Videodatenstroms. Die Planung der Verarbeitung und Aussendung der Daten auf wenige μ-Sekunden genau vereinfacht die synchrone Darstellung verteilter Audio- und Videoströme trotz des durch die ereignisbasierte Übertragung unweigerlich auftretenden Jitters der Übertragungsverzögerung.

Bei zukünftigen Bordnetzen mit Ethernet als Backbone und diversen, für den Einsatz und die Kosten optimierten Subnetzen mit differierenden physikalischen Übertragungsprotokollen unterstützt uMAP die transparente Adressierung und die transparente Kommunikation mit Geräten über diese Netze. Dies erleichtert Planung und Design während des Produktentstehungsprozess (PEP), und vor allem vereinfacht es die nachträgliche Erweiterung des Bordnetzes während des Lebenszyklus.

Zum Schluss sei noch die relativ einfache Implementierung erwähnt. Da im Gegensatz zu UDP/IP und Vergleichbarem nur eine Schicht realisiert werden muss und durch die *Memory-mapped*-Arbeitsweise eine klare und im Fehlerfall einfach zu untersuchende Schnittstelle besteht, wird der Aufwand bei der Implementierung weiter gedrückt.

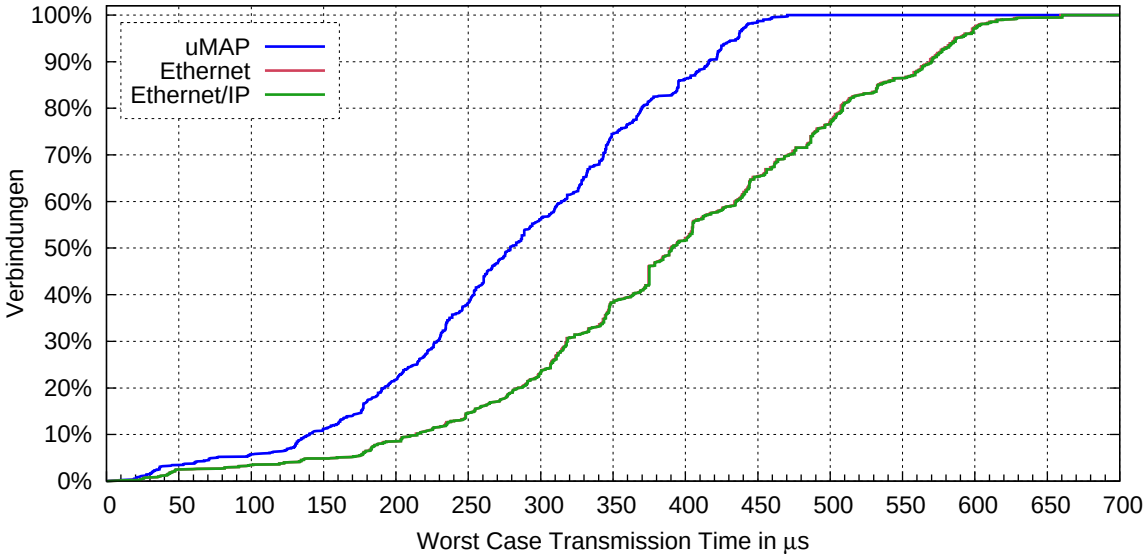


Abbildung 5.6: WCTT bei uMAP

Neben den beschriebenen qualitativen Vorteilen beim Einsatz von uMAP wurde auch der Einsatz im verwendeten Testnetz aus Kapitel A.1 untersucht. Der Ergebnisgraph ist in Abbildung 5.6 dargestellt. Aufgezeigt ist die analytisch berechnete Worst Case Transmission Time (WCTT) über alle gewünschten Kommunikationsbeziehungen hinweg. Zwischen dem reinen Ethernet und der Verwendung von IP als zusätzlicher Schicht bestehen keine signifikanten Unterschiede, da nahezu alle der bei einer Verbindung übertragenen Daten weniger als die Maximum Transmission Unit (MTU) des verwendeten Ethernet sind (vgl. Diagramm A.3). Beim Einsatz von uMAP können nun alle Nachrichten mit gleicher Zykluszeit, gleicher Quelle und gleichem Ziel

in einen zu sendenden Rahmen aggregiert werden. Diese optimierte Übertragung verringert die pro Verbindung zu übertragende Menge an Daten und damit signifikant die maximal auftretende Verzögerung. Dieser Effekt tritt stärker auf, je belegter die Netzressourcen sind. Multicast wird in diesem Fall nicht unter der Annahme einer „Worst Case“-Situation betrachtet, würde aber die auftretenden Verzögerungen weiter senken.

6 Realisiertes Entwurfs- und Planungssystem

Zur Anwendung und Überprüfung der in den vorangegangenen Kapiteln dargestellten Ansätze wurde ein Entwurfs- und Planungssystem realisiert. Dieses Softwareframework erlaubt es, die Netzplanungsalgorithmen und den Einsatz von Ethernet als Kommunikationsprotokoll für verschiedene Netze zu evaluieren. Dabei wird auf eine möglichst realitätsnahe Abbildung der Wirklichkeit Wert gelegt.

Um die verwendeten Modelle zu parametrisieren, wurden Messungen an einer Referenzhardware vorgenommen sowie ein Ausschnitt der vorgeschlagenen Architektur in einem Prototypenfahrzeug realisiert. Gerade reaktiv arbeitende Systeme, wie das in Kapitel 3.3 vorgestellte Sicherheitsmanagement, können so am besten auf ihre korrekte Funktionalität und ihr zeitliches Verhalten überprüft werden.

Dieses Kapitel beschreibt die einzelnen Komponenten des Systems und deren Zusammenspiel. Weiter wird ein Überblick über die Implementierung und die daraus gewonnenen Ergebnisse gegeben.

6.1 Komponenten des Systems

Das vollständige Framework steht auf drei Säulen: Berechnung, Simulation und Messung. Entsprechend sind auch die einzelnen Komponenten in diese drei Kategorien unterteilt. In Abbildung 6.1 sind alle Blöcke und ihr Zusammenspiel dargestellt.

Die Säule „Berechnung“, also die analytische Betrachtung der Architektur, besteht zum einen aus dem Block „FIBEX“. Diese Komponente liest Daten im Field Bus Exchange Format (FIBEX) ein und stellt sie dann den anderen Blöcken zur Verfügung. FIBEX ist eine auf XML basierende Beschreibungssprache, um die Kommunikation in Feldbussen zu beschreiben [KBD+04]. Ursprünglich aus der Automatisierungstechnik stammend, wurde das Format erweitert und findet vor allem in der Automobilindustrie verbreiteten Einsatz. Es wurde eine in C++ geschriebene, auf TinyXML basierende Bibliothek entwickelt und um Original Equipment Manufacturer (OEM)-proprietäre Kategorien erweitert. Die eingelesenen und ausgewerteten Daten werden intern als Graph vorgehalten, also als Kanten mit definierter Quelle und Senke. Die Art der zu übermittelnden Daten sowie Angaben zum Zeitverhalten sind Eigenschaften dieser Kanten. Der Block „FIBEX“ stellt diese Daten dem Szenarien-Generator für den Teil Simulation sowie der Komponente VEHInet zur Verfügung.

Der Block „VEHInet“ vereint die analytische Auswertung der Daten, hauptsächlich durch die in Kapitel 3.1.4 aufgezeigte Worst Case Transmission Time (WCTT)-

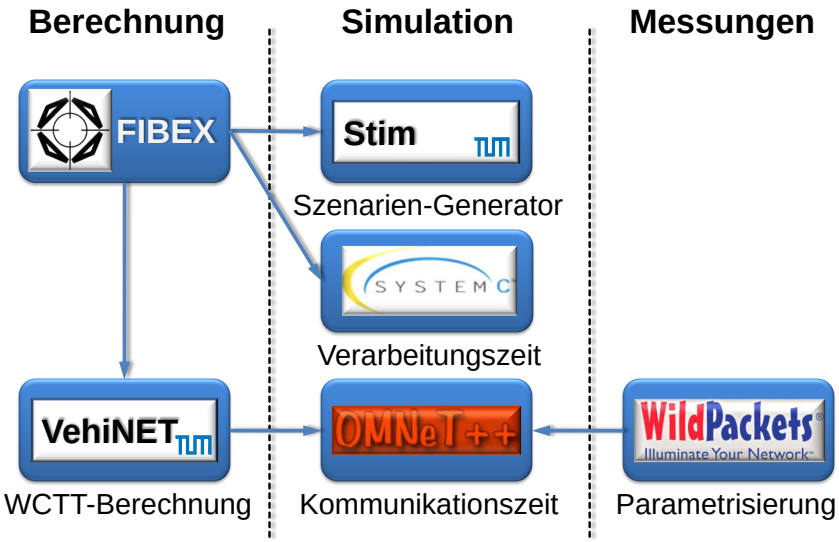


Abbildung 6.1: Komponenten des Frameworks

Berechnung sowie den in Kapitel 4.3 vorgestellten Genetischen Algorithmus (GA). Ergebnis ist ein physikalischer Graph, der dann als Grundlage der Simulation des Kommunikationsverhaltens dient.

Die Säule „Simulation“ stellt ein kooperierendes System von Blöcken dar, die es ermöglichen, komplette Wirkketten zu simulieren. Dies bedeutet nicht nur die Verzögerung während der Kommunikation, sondern aller Verarbeitungsschritte zwischen der Generierung der Daten in einer Quelle und dem Empfang der verarbeiteten Informationen an der Senke. Dies kann eine mehrfache Verarbeitung in Recheneinheiten und Kommunikationsschritte beinhalten. Der Block „Stim“ als Szenarien-Generator (Kapitel 6.2.1) erstellt die Wirkketten und startet die Verarbeitung auf der Basis der Daten aus dem FIBEX. Die Komponente „SystemC“ simuliert die in den Electronic Control Units (ECUs) auftretenden Verzögerungen und der Block „OMNeT++“ die durch Kommunikation entstehenden Verzögerungen (Kapitel 6.2.3).

Der Block „Verarbeitungszeit - SystemC“ ist nicht im Rahmen der vorliegenden Arbeit entstanden; ein kurzer Überblick über ihn und darüber, wie er in das Gesamtsystem eingebunden ist, wird in Kapitel 6.2.2 gegeben.

Die letzte Säule, „Messungen“, beinhaltet die Komponente „WildPackets“. Mit der Hardware von WildPackets zur Messung und Analyse von Datenpaketen bei der Übertragung mit Ethernet konnten Statistiken mit einer zeitlichen Auflösung von $\approx 10ns$ generiert werden [AF07]. Als Grundlage dienen einerseits Aufbauten im Laborumfeld, wie das in Kapitel A.1 beschriebene Testnetz, andererseits Messungen an einer prototypischen Implementierung, die in Kapitel 6.4 näher beleuchtet wird. Die Statistiken dienen zum Parametrisieren der in der Kommunikationszeitsimulation verwendeten Modelle.

6.2 ITMsim Simulationsframework

Die Entwicklung des Simulationsframeworks entstand unter der Prämisse, Wirkketten abzubilden und simulieren zu können. Bei Wirkketten wird, im Gegensatz zu klassischen Simulatoren, eine vollständige Kette von Einzelaktionen betrachtet und es werden Statistiken über das Verhalten dieser kompletten Kette erstellt. Als Einzelaktionen sind dabei natürlich die Kommunikation über ein Bordnetz sowie die Verarbeitung in einem Rechenknoten implementiert, aber auch das Aufsplitten der Daten und die Synchronisierung einzelner Datenströme vorgesehen.

Ein Faktor bei der Planung des Frameworks war, möglichst vorhandene und ausgereifte Komponenten zu verwenden, bei der eine große Community vorhanden ist und somit auf viel Wissen und bereits vorhandene Implementierungen zurückgegriffen werden kann. Die Entscheidung fiel daher auf OMNeT++ [V+01] für den Bereich der Kommunikation und SystemC bei der Modellierung der ECUs.

Beim Einsatz von zwei getrennten Simulatoren ist deren Synchronisierung eine große Herausforderung. Es entstand eine universelle Schnittstelle basierend auf Remote Procedure Call (RPC) zur Verknüpfung der Systeme. In Kapitel 6.2.4 wird diese Schnittstelle näher erläutert.

6.2.1 Szenarien-Generator

Der Szenarien-Generator erstellt die Wirkkette aus den im FIBEX gespeicherten Informationen und hält diese in einer nach Startzeitpunkt sortierten Liste vor. Er besitzt zwei unterschiedliche Modi. Der erste, der die Eigenschaften von Regelkreisen abbildet, startet zyklisch mit überlagerten zufälligen Jitter-Wirkketten mit fester Datengröße. Details zu Parametrierungsoptionen sind in Kapitel B.1 dargestellt. Der zweite Modi erstellt eine für die Übertragung von komprimierten Videodaten charakteristische Last. Eine exemplarische Auswertung ist in Kapitel B.2 zu finden. Basierend auf der statistischen Analyse [MRSZ92] werden mit einer definierten GOP-Sequenz Einzelbilder im Abstand von 40 ms mit variabler Größe erstellt. Dies deckt die besonderen Herausforderungen bei der Übertragung und Verarbeitung von Daten mit variabler Rate ab.

Die Informationen der Übertragungen, wie Datengröße, Zyklus sowie Quelle und Senke, werden aus dem von FIBEX generierten Anforderungsgraphen extrahiert. Die statistische Überlagerung dieser Werte werden für verschiedene Simulationsläufe variiert, um möglichst das gesamte Ereignisfeld abzudecken.

6.2.2 Verarbeitungszeit

Die Modellierung der ECUs erfolgt in SystemC. Dort wird auch das Handling der ein- und ausgehenden Ereignisse durchgeführt. Die Modelle sind im Rahmen des Pro-

jektes IT_Motive 2020 von Partnern entwickelt worden und nicht Bestandteil dieser Arbeit.

Mit den SystemC-Modellen wird die Belegung realer Hardwarekomponenten in den ECU nachgebildet. Dies umfasst mehrere Prozessoren und unterschiedliche Scheduling-Varianten, also die Zuteilung der Kapazitäten. Modelliert wird auch die Belegung eines Prozessorbusses sowie Speicherzugriffe.

Bei der Verarbeitung in SystemC wird auch die Synchronisation von Wirkketten durchgeführt. Mit den Funktionen der Synchronisation, die auch Aufsplittung und Vereinigen von Wirkketten beinhaltet, können komplexe Funktionen im Fahrzeug realitätsnah nachgebildet werden. Dies trifft zum Beispiel auf komplexe Fahrerassistenzfunktionen zu, bei denen die Daten mehrerer Sensoren aggregiert werden. Mit nur einem durchlaufenden Datenelement wie bei klassischen Simulationen können diese Funktionen nur unzureichend dargestellt werden.

Die Wirkketten beginnen dabei am Szenarien-Generator und werden von dem Quellen-SystemC-Modul nach Bearbeitung dem Kommunikationsnetz übergeben. Die Modellierung der Verzögerung durch die Kommunikation zwischen den ECUs wird im Block „OMNeT++“ durchgeführt.

6.2.3 Kommunikationszeit

Im Block „OMNeT++“ erfolgt die Berechnung der durch Kommunikation auftretenden Verzögerungen bei der Verarbeitung der Wirkketten. OMNeT++ ist ein ereignisbasierter Simulator für Kommunikationsnetze, der eine hohe Verbreitung in der Community hat. Dadurch sind viele Modelle für bestehende Netze bereits vorhanden. Das Simulationsframework ist in C++ geschrieben und damit sehr performant und leicht erweiterbar.

OMNeT++ wird überwiegend zur Simulation von Netzen im Office- und im Weitverkehrsnetzbereich eingesetzt. Deswegen mussten die in dieser Arbeit verwendeten Modelle neu erstellt werden. So entstand, auf Standard-Ethernet basierend, das hier verwendete Realzeit-Ethernet sowie das zeitschlitzgesteuerte Ethernet. Zum Vergleich mit herkömmlichen Systemen wurde ein Modell von Flexray sowie von Controller Area Network (CAN) implementiert.

Da OMNeT++ vollständig separat zu SystemC arbeitet, muss eine abbildbare Topologie in beiden Systemen existierten. Dafür wurde zu OMNeT++ ein Topologiegenerator entwickelt, der direkt den physikalischen Graphen einliest und daraus ein für OMNeT++ verständliches Netz erstellt. Dieses Netz ist beispielhaft in Abbildung 6.2 visualisiert. Der Graph wird automatisch eingelesen und verarbeitet. So können in mehreren Durchläufen automatisch verschiedene Topologien simuliert und die Ergebnisse verglichen werden. Der physikalische Graph stammt aus dem Ergebnis der Netzplanung aus Modul „VEHInet“ (siehe Kapitel 6.3).

Bei Ethernet wurde, wie bereits in Kapitel 3 dargelegt, hauptsächlich der Switch, also der Ethernet-Sternknoten, modifiziert. Neben dem Queuing der Pakete und den

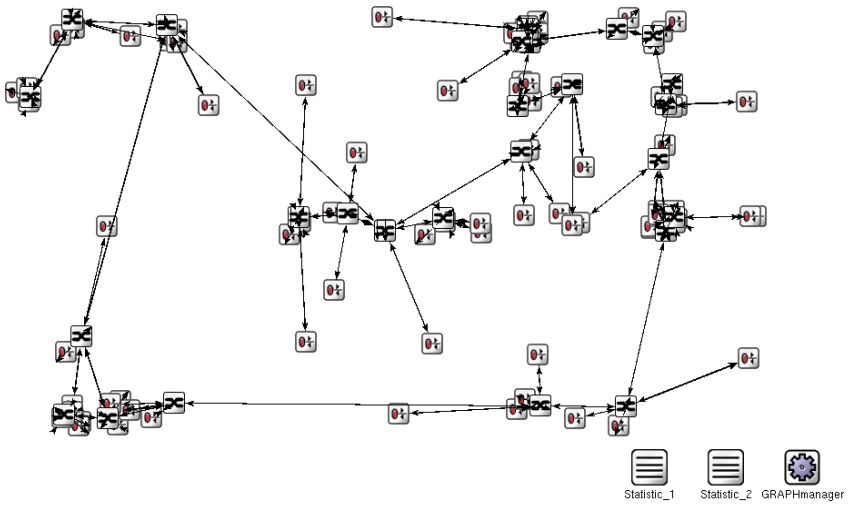


Abbildung 6.2: Importiertes Simulationsszenario

Warteschlangen ist ein großer Punkt die Rechenleistung und damit die Verarbeitung der Daten. Um die Verzögerung durch die Verarbeitung der Pakete zu bestimmen, wurden Hardwaremessungen an einem Gigabit-Ethernet-Switch aus dem Büroumfeld vorgenommen. Da die zu erwartenden Verzögerungen sehr klein sind, wurde mit einer Messung der Ein- und Austrittszeiten der Rahmen in der Hardware mit dem Equipment von WildPackets vorgenommen. Durch eine hardwareunterstützte Messung sind eine Auflösung und eine Genauigkeit von 10 ns möglich. Der Vergleich zwischen der Messung am Serienprodukt, der Messung in der Simulation und der modellierten Verzögerung sind in Diagramm 6.3 dargestellt.

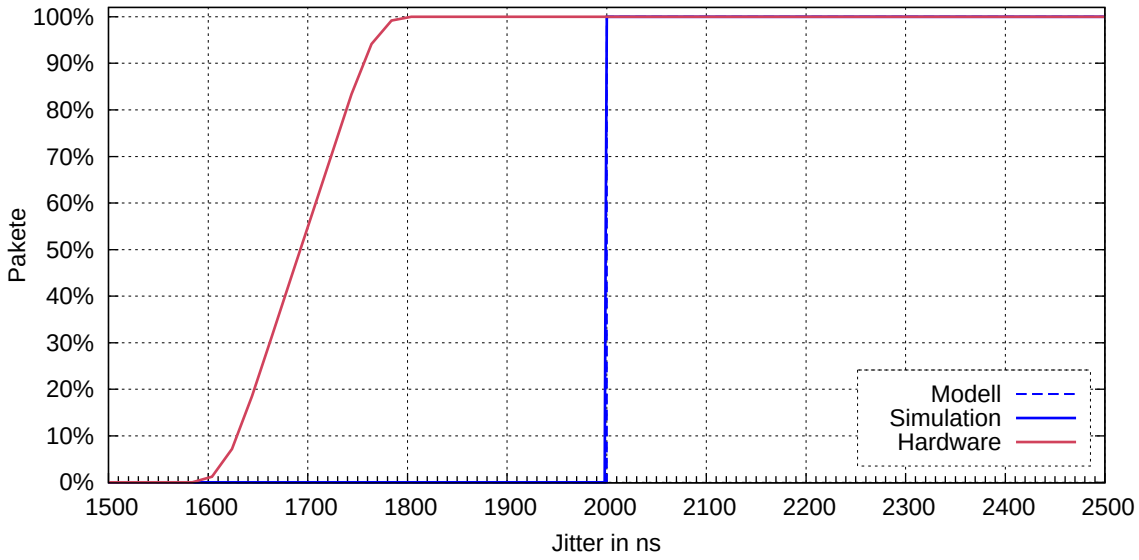


Abbildung 6.3: Verarbeitungszeit-Modell Switch

Die Messungen zeigen eine Verteilung der Verzögerungen im Bereich von $\approx$ 1.600 ns - 1.800 ns, wobei 100% der mehreren Millionen ausgewerteten Pakete unter 1.850 ns

verzögert wurden. Zur Sicherheit wurde deswegen im Modell 2 µs als feste Verzögerung gesetzt. So sind die real zu erwartenden Ergebnisse immer besser als die modellierte Verzögerung und die obere Grenze der berechneten WCTT kann nicht durchbrochen werden.

Die Simulation betrachtet den Data Link Layer (Layer 2) nach dem OSI-Modell und die darauf aufbauenden Schichten. Die physikalische Impulsformung wird nicht betrachtet, jedoch die Laufzeitverzögerung auf den Leitungen. Die Sternknoten sind vom Medienzugriff und vom Queuing realitätsnah abgebildet, zu Evaluierungszwecken wurden beliebig tiefe Warteschlangen verwendet, um eine Aussage über die Belegung zu machen. Die in Hard- und Software ablaufenden Managementfunktionen und die daraus resultierenden Verzögerungen auf die Rahmen sind in einem zentralen, an der Eingangswarteschlange auftretenden Delay von 2 µs zusammengefasst. Ein weiterer interner Signalisierungsoverhead ist damit abgedeckt.

Innerhalb der Simulation des Kommunikationsnetzes wurden Statistiken über die Zeitpunkte des Eintritts der Nachrichten in das Netz am Quellknoten, des jeweiligen Empfangs und der Aussendung an den zwischengelagerten Sternknoten und des Eintreffens der Nachricht am Zielknoten erstellt. Somit konnte die Auswirkung von verschiedenen Warteschlangendesigns und unterschiedlicher Wegewahl im Netz untersucht werden.

6.2.4 Synchronisation

Um zwei Simulationsframeworks für unterschiedliche Zwecke miteinander zu verknüpfen, gibt es mehrere mögliche Methoden. Die meisten dieser Lösungen kommen aus dem Gebiet der Hardware/Software-Cosimulation. Die Arbeiten von Fummi et al. [DFP02, FPG+03, FLM+05] sind in diesem Bereich besonders zu erwähnen. Eine dabei häufig eingesetzte Methode ist das *rollback*. Dabei laufen beide Simulatoren unabhängig weiter, bis ein Ereignis in einem Simulator ein Ereignis im anderen auslöst. Ist dieser in der simulierten Zeit zurückgeblieben, wird der Event in die Warteschlange eingereiht. Ist der Simulator aber bereits in der simulierten Zeit weiter fortgeschritten, wird er auf den Zeitpunkt des Ereignisses zurückgesetzt, die Simulation also zurückgerollt (*rollback*). Da es sich bei beiden eingesetzten Simulatoren, SystemC und OMNeT++, um kausale diskrete ereignisorientierte Systeme handelt [Bal96], ist ein Zurückspringen in der Zeit direkt nicht möglich und könnte nur über das regelmäßige Zwischenspeichern des Systemzustandes realisiert werden. Dies führt zu hohem Speicherverbrauch und einer deutlichen Geschwindigkeitseinbuße.

Eine weitere Möglichkeit ist das Verknüpfen der beiden getrennten Eventkernels, also des Teils der Software, der den Ereigniskalender verwaltet. Dabei werden über einen Adapter die Ereignisse von System b in den Ereigniskalender von System a als Übersetzung $a = f(b)$ geschrieben. Diese Lösung ist sehr schnell, erfordert aber starke Anpassungen an der Software, da es nur rudimentär vorhandene Schnittstellen gibt.

In ITMsim wurde eine asymmetrische, nachrichtenbasierte Synchronisation entwickelt. Da der Beginn der Wirkkette immer an ein Ereignis eines Sensors gekoppelt

ist, beginnt die Verarbeitung immer in einer Hardwareeinheit, welche in SystemC modelliert ist. Somit übernimmt SystemC die Aufgabe des Synchronisationsmasters. OMNeT++ folgt im Sinne der Simulationszeit als Slave. Dies wird über zwei einfache Befehle gesteuert: Mit *runUntil(t_x)* wird OMNeT++ aufgefordert, den Ereigniskalender bis zu den Ereignissen $t \leq t_x$ abzuarbeiten. SystemC ist zu diesem globalen Zeitpunkt bereits bei der Simulationszeit t_x. Anschließend wird an den Master über das Kommando *nextEvent(t_y)* der Zeitpunkt übermittelt, wann ein für SystemC relevantes Ereignis vorliegt. OMNeT++ wird dann zu einem Zeitpunkt $t \leq t_y$ wieder aufgerufen.

Der Austausch dieser Nachrichten erfolgt über eine simple, auf dem Prinzip der Remote Procedure Calls (RPC) basierende Schnittstelle. Die Informationen werden dabei über eine TCP/IP-Verbindung übertragen. Die Filterung der Ereignisse im Ereigniskalender von OMNeT++ ist ein fundamentaler Teil der Schnittstelle. Jeder unnötige Sprung zwischen den Frameworks kostet Prozessorzeit und bringt keinen Fortschritt. So ist es zum Beispiel nicht sinnvoll, bei jedem der bis zu 20 Verarbeitungsschritte eines Rahmens im Sternknoten eine Synchronisation durchzuführen. Eine Möglichkeit wäre die Verringerung der zeitlichen Auflösung auf $> 1ns$, was jedoch zu größeren Simulationsfehlern führen könnte. Eine andere Möglichkeit ist das Verwenden von prädizierten Dummy-Ereignissen. Dabei wird bei Eintreffen des Paketes in einem Kommunikationsgerät der Zeitpunkt ermittelt, wann das Paket frühestens beim nächsten Gerät ankommt, und dieser vorhergesagte Wert t_d übermittelt. Es muss sichergestellt sein, das der effektive Zeitpunkt t_e immer hinter dem prädizierten liegt, $t_e \geq t_d$. Aber auch wenn keine Filterung des Ereigniskalenders in OMNeT++ durchgeführt wird, zeigt die Auswertung der Synchronisation ein akzeptables Verhalten: Bei einer zeitlichen Auflösung von $1ns$ der beiden Simulationsframeworks wurde im Durchschnitt nur alle 1.391 ns zwischen den Simulatoren gewechselt.

6.3 VEHInet-Planung und -Analyse

Im Block „VEHInet“ wird aus dem Anforderungsgraphen und der Liste und Position der ECUs der physikalische Graph gebildet. Dies geschieht mit dem in Kapitel 4 vorgestellten GA. Der physikalische Graph enthält also neben den ECUs die Anzahl und Position der Sternknoten sowie die Kanten zwischen den einzelnen Geräten mit Typ, Rate und Länge. Weiter wird die Information zur Wegewahl, also dem Routing durch das Netz, im Graph hinterlegt.

VEHInet ist eine in C++ programmierte Eigenentwicklung. Es verwendet die Layonics GRAPH-Library zum Speichern der Graphen und Durchführen einzelner Algorithmen wie *Dijkstra* und Pfadanalysen.

Der in VEHInet verwendete GA aus Kapitel 4.3.2 verwendet in vielen Schritten einen Zufallsgenerator. Die Arbeiten von Cantú-Paz [CP02] und Meysenburg [MF97] zeigen, das die Auswahl des Zufallsgenerators Einfluss auf die Effizienz des GA hat. Vor allem bei der Generierung der initialen Population sind zufällig verteilte Lösungen wichtig, um den ganzen potentiellen Ergebnisraum aufzuspannen. Deshalb wurde in

VEHInet der Mersenne-Twister als Pseudozufallszahlengenerator implementiert, der nicht nur schnell ist, sondern auch eine sehr lange Periode besitzt.

Bei der Optimierung von Netzen und Demand in dieser Größenordnung gibt es zwei Herausforderungen: Speicherverbrauch und Rechenleistung. Durch den Einsatz eines GA, der jeweils nur die aktuelle Population gespeichert hat und keine Ergebnisbäume mit früheren Lösungen, ist der Speicherverbrauch des Optimierungsalgorithmus stark verringert und mit einigen Gigabyte problemlos auf aktuellen Rechensystemen durchführbar. Deswegen wurde in VEHInet speziell die Rechenleistung betrachtet und es wurden Methoden eingeführt, die Ausführungszeiten zu verringern. Neben Verbesserungen im Code geschah dies hauptsächlich durch den Einsatz von paralleler Programmierung und Ausführung auf symmetrischen Multiprozessoren. Dafür wurde ein Dual-Prozessor mit Quad-Core verwendet, also acht verfügbaren Rechenkernen. Zum Einsatz kam die OpenMP-Library.

Gerade durch den GA ist es möglich, einzelne Schritte parallel abzuarbeiten. Zum Beispiel erfolgt die Generierung zufälliger Lösungen bei der initialen Population und der Immigration unabhängig voneinander (siehe Kapitel 4.3.2). Auch der Rekombinationsschritt erfolgt weitestgehend unabhängig. Und der komplexeste Schritt des GA, die Bewertung der Qualität einer gefundenen Lösung durch die Fitnessfunktion, kann für jedes der 1.400 pro Generation gefundenen Genome unabhängig und damit parallel erfolgen. Sobald entsprechende Abschnitte im Ablauf des Programms identifiziert und entsprechend umgearbeitet wurden, damit keine Überschneidung der Variablen und Abhängigkeiten zwischen den Blöcken auftreten, können sie durch den Einsatz von OpenMP automatisch parallelisiert werden. Da eine Synchronisation zwischen den Schritten notwendig ist und auch seriell zu bearbeitende Blöcke wie die Selektion existieren, ist der Geschwindigkeitszuwachs bei acht Prozessorkernen $\approx 7{,}5 < 8$.

Bei der Berechnung der WCTT aus Kapitel 3.1.4 wurde eine andere Methode verwendet. Die Berechnung der WCTT stellt sich hauptsächlich als *Search-and-Compare*-Algorithmus dar. Untersucht wird ein Datenstrom, wobei alle Datenströme gesucht werden, die er auf dem Weg durch das Netz trifft und durch die er so potentiell verzögert wird. Es wird die Verzögerung durch diese Datenströme berechnet und aufsummiert. Derselbe Vorgang mit derselben Datenbasis im Speicher wird auf alle der ≈ 2.000 Demand-Verbindungen des Referenznetzes aus Kapitel A.2 angewendet. Es handelt sich dabei somit um ein klassisches SIMD-Problem (*Single Instruction, Multiple Data*) nach der Flynnschen Taxonomie. Zum Einsatz kommt aus diesem Grund ein Streamprozessor.

Verwendet wird eine NVIDIA GeForce GTX260 mit 192 Prozessoren zu je 576 MHz und einer Speicherbandbreite von 112 GByte/s. Zur Ansteuerung wurde die CUDA-Library von NVIDIA verwendet. Dafür werden in einem Vorverarbeitungsschritt die Graphen in zweidimensionale Arrays mit entsprechendem Alignment konvertiert und in den Speicher kopiert. Dann wird der Prozesscode geladen und auf allen Prozessoren parallel auf die Daten im Speicher angewendet. Nach Abschluss der Berechnungen wird das Ergebnis zur weiteren Verarbeitung auf dem Hauptsystem aus dem Speicher kopiert.

Ein Quantifizieren des Geschwindigkeitsvorteils durch den Einsatz von CUDA war nicht möglich, denn dafür hätte ein vergleichbarer Code für beide Methoden separat erstellt werden müssen, da das Vorgehen in CUDA nicht auf symmetrische Multiprozessoren angewendet werden kann. Da die Berechnung der WCTT aus mehreren hundert parallelen Aufgaben besteht, die jeweils von geringer Komplexität sind und messbar hohen Speicherzugriff benötigen, der über der Cache-Größe liegt, ist qualitativ betrachtet eine hoher Gewinn bei der Geschwindigkeit von > 10 zu erwarten.

6.4 Anwendung in IT_Motive 2020

Das Projekt IT_Motive 2020 war ein interdisziplinäres Projekt mit mehreren beteiligten Lehrstühlen der Technischen Universität München, der BMW Forschung und Technik GmbH und der BMW Group AG im Rahmen der CAR@TUM Initiative. In IT_Motive 2020 wurden neue Konzepte für IT-Architekturen im Fahrzeug entwickelt und untersucht, um den steigenden Kosten und der steigenden Komplexität aktueller Ansätze entgegenzuwirken. Durch die Mitwirkung der Lehrstühle für Integrierte Systeme, Realzeit-Computersysteme, Datenverarbeitung, Medientechnik und Kommunikationsnetze sowie des Fachgebiets Höchstfrequenztechnik konnten die Herausforderungen und Lösungen von mehreren unterschiedlichen Seiten betrachtet werden, um so ein Gesamtkonzept zu generieren. Ergebnis war ein Vorschlag für eine zukünftige IT-Architektur, die neben analytischer und simulativer Validierung auch in Form einer prototypischen Implementierung in ein Serienfahrzeug realisiert wurde.

Im Rahmen dieser Arbeit wurde in den Prototypen das in Kapitel A.1 beschriebene Testnetz implementiert. Schwerpunkt war die Realisierung der Funktionen in den Sternknoten, insbesondere das Scheduling aus Kapitel 3.1.3 und das Sicherheitsmanagement aus Kapitel 3.3. Zentrales Element ist das ausfallsichere Netz mit den drei Sternknoten, basierend auf Gigabit-Ethernet. Bei der Auswahl der zu verwendenden Prototypenhardware für die Sternknoten waren hauptsächlich folgende Punkte maßgebend: Zugriff und Manipulierbarkeit der Routingtabellen, schnelle Anpassung an Änderungen in Protokollen, Prozessor zur Vorverarbeitung der Daten in der Control Plane (CP), sechs Ports pro Sternknoten. Da der Zweck des Demonstrators ein *Proof-of-Concept* war und keine Grundlage zur Zertifizierung, konnte das Timing der Hardware freizügiger ausgelegt werden.

Nach Untersuchung der auf dem Markt befindlichen Produkte fiel die Entscheidung auf eine reine Software-Lösung, die auf einem Freescale-PowerQUICCIII-MPC8555-Kommunikationsprozessor basiert. Dadurch war die Anforderung des Rapid Prototyping und der vollständigen Manipulierbarkeit ideal abgedeckt. Nachteil einer Transport Plane (TP) in der Software ist die schlechtere Vorhersagbarkeit der Paketverzögerung. Der Backbone des Bordnetzes wird über die zwei als System on a Chip (SoC) integrierten Ethernet-Schnittstellen aufgebaut. Die peripheren ECUs werden über eine Quad-Ethernet-Karte an den PCI-Bus angebunden. Als Software kam ein embedded Linux von DENX zum Einsatz.

Für die TP wurde der Linux-Kernel Ethernet Bridge verwendet, der die Funktionalität eines Ethernet Switch in der Software abbildet. Erweitert wurde das System um eine Schnittstelle zum Auslesen und zum Manipulieren der Einträge in der Routingtabelle. Verwendet wurde *Destination-Based Routing* wie bei Ethernet. Die hierarchischen Warteschlangen wurden mit dem Linux-Kernel Traffic Control realisiert. Die TP läuft somit vollständig im Kernelspace ab und hat damit bestmögliche Bedingungen für Timing und Datendurchsatz. Beim Kompilieren des Kernels wurde speziell auf die Optimierung des Timings parametrisiert, da dies ausschlaggebend für die geplanten Demonstrationsszenarien ist.

Die CP wurde in C++ entwickelt und läuft auf dem Prozessor im Userspace. Die teilweise komplexen Aufgaben der CP stören den Transport der Pakete somit weitestgehend nicht. Neben der Ansteuerung der Schnittstelle zur TP ist die Topologie-Erkennung Hauptaufgabe der CP. Die prototypische Implementierung der Topologie-Erkennung verwendet ICMP Echo-Request-Nachrichten an die Multicast-Adresse 224.0.0.1 („All Systems on this Subnet“) gesendet. Es wird ein TTL („Time To Live“) von 0 im Internet-Protokoll (IP)-Kopffeld gesetzt, sodass die Nachrichten im Fehlerfall nicht weitergeleitet werden. Alle angeschlossenen Geräte reagieren nun automatisch mit einer ICMP Echo-Reply an den Absender, die somit die aktuelle IP-Adresse sowie die Hardware-Adresse des angeschlossenen Knotens enthält. Dies wird in regelmäßigen Abständen wiederholt. Im Gegensatz zur proaktiven Erkennung über „Hello“-Nachrichten kann dies je nach Systemereignis umgehend ausgelöst werden, zum Beispiel wenn eine Statusänderung der Netzwerkschnittstelle festgestellt wird. Die Information, welche Adressen an welchen Ports zu finden sind, wird an die zentrale Management Plane (MP) übermittelt.

Für die Kommunikation zwischen CP und MP, die sich im zentralen Managementsystem befindet (siehe Kapitel 3.3), werden XML-Nachrichten als IP/UDP-Pakete verwendet. Die Codierung mit XML benötigt zwar eine höhere Rechenleistung und eine höhere Datenrate, die Übertragung der Daten als hierarchisch organisierter Text ermöglicht aber eine schnellere Fehlersuche.

Als Schnittstelle zum Fahrzeug wurde ein Local Interconnect Network (LIN) zu Ethernet Gateway sowie ein CAN zu Ethernet Gateway implementiert. Damit wurde einerseits der Zugriff von Applikationen auf das Serienbordnetz über Ethernet ermöglicht, andererseits der Transport von realistischen Fahrzeugdaten über das neue Bordnetz getestet. Die bei der Implementierung gewonnenen Erkenntnisse und die quantitativen sowie qualitativen Ergebnisse sind iterativ in die Entwicklung der vorgestellten Lösungen eingeflossen.

7 Zusammenfassung und Ausblick

In der Arbeit wurden neue Konzepte zur Realisierung von Bordnetzen für verteilte heterogene Subsysteme entwickelt. Dazu wurden vorhandene Lösungen untersucht, um die Anforderungen an diese Art von Netzen zu bestimmen. Zusätzlich wurden die Forschungslandschaft und die in der Entwicklung befindlichen Produkte näher betrachtet. Der Einsatz von IEEE 802.3 Ethernet wurde als zukunftssichere Lösung identifiziert, da seine Verbreitung am größten ist und somit die Verwendung vorhandener Hardware und Intellectual Property potentiell zu größeren Kosteneinsparungen führen als proprietäre Lösungen. In diesem Zusammenhang wird oft der Begriff „components-off-the-shelf“ (COTS) verwendet. Es hat sich jedoch gezeigt, dass der Ethernet-Standard nicht direkt übernommen werden kann.

Es wurde ein hierarchisches Schedulingsystem entwickelt, um trotz des ereignisbasierten Verhaltens von Ethernet eine deterministische Übertragungsverzögerung zu garantieren. Als Eingangsparameter wird der minimale Frameabstand $T_{A\,\min}$ und die maximale Framegröße der Datenströme benötigt. Ein entwickelter Algorithmus berechnet dann die Worst Case Transmission Time (WCTT), also die maximal auftretende Verzögerung jedes Datenstroms. Diese Verzögerung ist garantiert und wird in jedem Fall eingehalten. Das Schedulingsystem befindet sich in den Sternknoten, somit ist keine Modifikation des IEEE 802.3-Standards und der Endgeräte notwendig. Es hat sich gezeigt, dass das ereignisbasierte Ethernet grundsätzlich für zeitkritische Kommunikation eingesetzt werden kann, dabei aber eine höhere Überreservierung vorgenommen werden muss, da immer vom selten auftretenden schlimmsten Fall ausgegangen werden muss, was man aus den Vergleichen der simulierten Szenarien herauslesen kann. Diese reservierte Datenrate muss von niederpriorem Datenverkehr verwendet werden, um die Auslastung des Netzes zu erhöhen. In einem wie dem hier vorliegenden Szenario mit vielen Video- und Audiodatenströmen stellt dies kein Hindernis dar.

Es wurde ein Sicherheits- und Konfigurationsmanagement entwickelt, welches als verteiltes System auf den Sternknoten in Verbindung mit einer zentralen Instanz definiert ist. Durch die Topologieerkennung und den zentralen Routingalgorithmus können neue Geräte dynamisch in das vorhandene Bordnetz eingefügt werden. Verwendet man Netztopologien mit zwei und mehr möglichen Pfaden zwischen Quelle und Senke, kann durch intelligente Wegesuche eine Minimierung der auftretenden Verzögerungen erreicht werden, andererseits können bei Ausfall von Ressourcen die Datenverbindungen durch neue Routen aufrechterhalten werden. Dies ermöglicht eine systemweite Ausfallsicherung und damit eine Erhöhung der Gesamtverfügbarkeit.

Der Paradigmenwandel von klassischen Busnetzen zu einem Ethernet in Sterntopologie bedeutet Veränderungen in der Planung der Netze. In der Arbeit wurde eine

Virtuelle Kostenfunktion (VK) entwickelt, um mit Hilfe von auf Bauraum, Energieverbrauch und Verkabelungsaufwand basierenden Metriken die Aufwände beim Einsatz von Ethernet in Bordnetzen zu quantifizieren. Darauf aufbauend wurde ein Optimierungsverfahren basierend auf einem evolutionären Algorithmus entwickelt. Der Algorithmus arbeitete trotz der hohen Komplexität des Optimierungsproblems mit über 80 Knoten und fast 2000 Verbindungen sehr schnell und generierte in Zeiträumen von < 8 Stunden optimale Lösungen. Es hat sich gezeigt, dass man durch den Einsatz von Controller Area Network (CAN) zur örtlichen Aggregation der Daten weitere Aufwände und damit Kosten sparen kann. Auf der Basis dieser Ergebnisse hat sich ein heterogenes Netz aus CAN und Ethernet für den Backbone als eine ideale Lösung für Bordnetze der Zukunft herausgestellt.

Basierend auf den gewonnenen Kenntnissen über die Abläufe bei der Verarbeitung von Daten in verteilten heterogenen Subsystemen wurden die auf dem Transportprotokoll Ethernet aufbauenden Schichten 3 und höher des OSI-Modells näher betrachtet. Das in dieser Arbeit entwickelte universal Multilayer Automotive Protocol (uMAP) vereint Funktionen der Schichten 3 bis 5. Viele der sonst erst mit zusätzlichen Protokollen möglichen Mechanismen sind direkt verwendbar, zum Beispiel eine zeitliche Zuordnung der Daten sowie eine zeitsynchrone Verarbeitung im μ-Sekunden Bereich. Auch ist uMAP darauf ausgelegt, über einfache Gateways Daten in Netze mit anderen Transportprotokollen zu übertragen. So ist eine direkte Adressierung von Electronic Control Units (ECUs) in angeschlossenen CAN-, Local Interconnect Network (LIN)- oder Flexray-Netzen möglich. Auch können Video- und Audiodatenströme mit Hilfe von Fragmentierung und Zeitstempeln ohne zusätzliche Protokollschichten an verteilte Systeme übertragen werden. Dies minimiert den Effizienzverlust auf Grund der zu übertragenden Nachrichtenköpfe und vereinfacht die Verarbeitung in den Endgeräten.

Im Projekt IT_Motive 2020 entstand in Zusammenarbeit mit Partnern an der Technischen Universität München ein Entwurfs- und Planungssystem für Bordnetze. Es beinhaltet eine Cosimulation für die Verarbeitung der Daten in der ECUs und die Kommunikation der Daten über das Bordnetz. Im Rahmen dieser Arbeit wurden eine Schnittstelle zum Einlesen von Szenarien, ein realistisches Modell des Kommunikationsnetzes sowie Modelle für den angeforderten Datenverkehr entwickelt. Des Weiteren wurde die Schnittstelle zur Synchronisation zwischen der Simulation der Verarbeitung in SystemC und der Simulation der Kommunikation in OMNeT++ definiert und implementiert. Bei der im Laufe des Projektes entstandenen prototypischen Implementierung in einem Serienfahrzeug wurden Teile der in dieser Arbeit entwickelten Konzepte auf Rapid-Prototyping-Hardware realisiert. Vor allem das Kommunikationsnetz sowie das Sicherheits- und Konfigurationsmanagement mit dem automatischen Umleiten bei Ausfällen wurden als *Proof-of-concept* implementiert.

Mit dieser Arbeit wurde somit ein weiterer Grundstein zum Einsatz von Ethernet in Bordnetzen für verteilte heterogene Subsysteme gelegt. Als mögliche weiterführende Arbeiten in diesem Umfeld sei die Untersuchung zur Verringerung der Überreservierung bei der analytischen Berechnung der Übertragungsverzögerung erwähnt. Durch laxe zeitliche Synchronisation der Übertragungen beziehungsweise einer de-

finierten Asynchronität der ECUs könnte die berechnete WCTT deutlich verringert werden und läge damit näher an den simulierten Ergebnissen. Eine vollständige Aufnahme der Wirkketten in die Berechnung könnte dies bereits zum großen Teil erreichen. Zum Zeitpunkt der Arbeit lag leider keine ausreichende Datenbasis für diese Untersuchungen vor. Auch der Punkt Security, also die Sicherung der Daten bei der Übertragung gegen Manipulation und Ausspähen, konnten im Rahmen dieser Arbeit nicht näher betrachtet werden. Das Verbundprojekt SEIS, Sicherheit in eingebetteten IP-basierten Systemen, behandelt dieses Thema im Rahmen der Innovationsallianz Automobilelektronik (E | ENOVA).

A Referenznetze

In der Arbeit wurden zwei unterschiedliche Referenznetze verwendet. Einerseits das Testnetz, welches in Kapitel A.1 beschrieben wird. Es ist ein kleiner repräsentativer Ausschnitt aus einem Fahrzeug, welches stellvertretend die in Zukunft vorherrschenden Funktionen enthält. Dieses Bordnetz wurde in einem Laboraufbau sowie in einer prototypischen Implementierung innerhalb eines Serienfahrzeuges realisiert (siehe Kapitel 6.4).

Das zweite verwendete Referenznetz ist der Export eines aktuellen, in Serie befindlichen Fahrzeuges der Luxusklasse. Die Daten wurden anonymisiert vom Hersteller zur Verfügung gestellt. Grundlage ist eine so genannte „120%“-Konfiguration, also alle für den Verbau vorgesehenen Electronic Control Units (ECUs). Die unterschiedlichen Sonderausstattungsvarianten schließen aber teilweise den gleichzeitigen Einbau aus. Der größte Teil dieser doppelten Belegung von Funktionen mit ECUs wurde manuell ausgefiltert, so dass die entsprechende Datenbasis möglichst der Vollausstattung entspricht. Die Eigenschaften dieses Bordnetzes sind in Kapitel A.2 dargestellt.

A.1 Testnetz

Das Testnetz besteht nur aus wenigen Komponenten, enthält aber repräsentative Vertreter für die verschiedenen Funktionsklassen. Das verwendete Kommunikationsnetz ist gerade groß genug, um die in dieser Arbeit vorgeschlagenen Mechanismen zu untersuchen. Dafür wurden drei Sternknoten in einer ausfallsicheren Ringtopologie miteinander verbunden, so dass zwischen den Sternknoten jeweils zwei knoten- und kantendisjunkte Pfade existieren. Der Aufbau des Testnetzes ist in Abbildung A.1 dargestellt.

Als Abnehmer der Kommunikationsressourcen wurden in dem Testnetz von Partnern im Rahmen des Projektes IT_Motive 2020 mehrere Funktionen realisiert: Videostreaming von einer Live-Quelle, einer Kamera sowie von einem Streamingserver. Stellvertretend für komplexe Sensorik wurden zwei Lidar-Scanner in das Bordnetz integriert. Als Beispiel für eine realzeitkritische Anwendung mit sicherheitskritischem Datenverkehr wurde eine „schwebende Kugel“ verwendet. Die durch ein Magnetfeld in der Schwebe gehaltene Metallkugel stellt ein instabiles Gleichgewicht dar. Die Stärke des Magnetfeldes wird durch einen Regelkreis mit einem Regelzyklus von 1 kHz gesteuert. Die Kundenfunktionen in Form der Regelung der schwebenden Kugel, einer Lidar-Datenaggregation und Objekterkennung sowie eines Transcoders für Videodaten werden auf zwei ECUs ausgeführt. Mehrere Gateways verbinden das Testnetz mit dem Audiosystem des Fahrzeuges über den Media Oriented

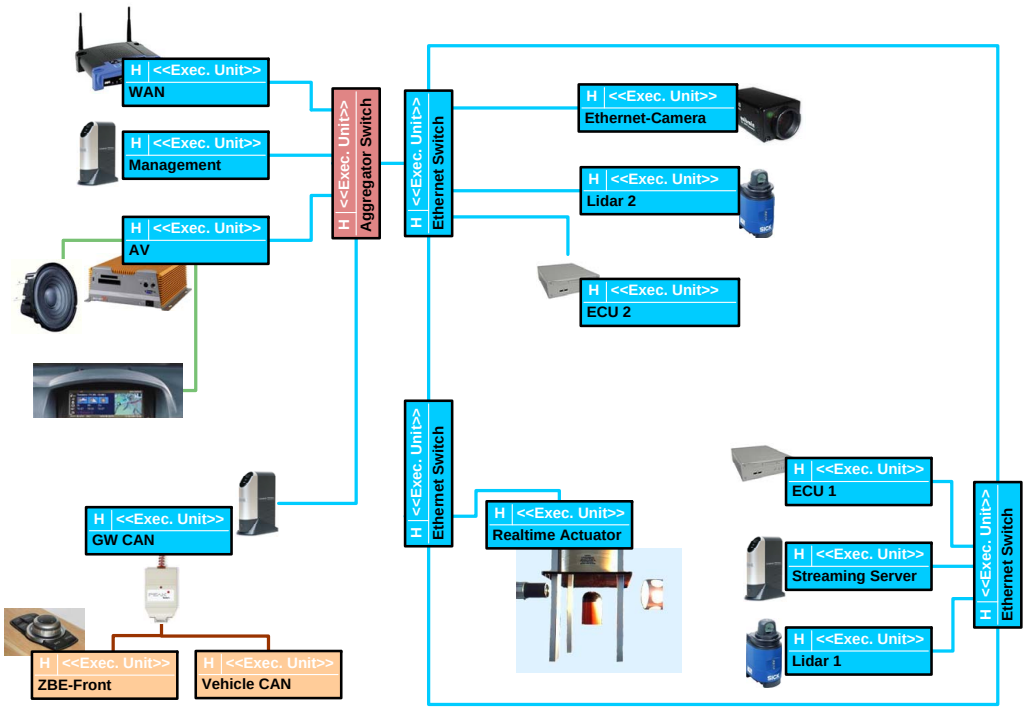


Abbildung A.1: Testnetz

Systems Transport (MOST)-Bus sowie mit Fahrzeugzustandsinformationen und Eingabegeräten über Controller Area Network (CAN)-Busse. Zur Verknüpfung mit dem Wide Area Network dient ein Wireless-LAN-Router.

In dem Kommunikationsnetz werden also drei Videoübertragungen, drei Lidarübertragungen sowie zwei Datenströme des Regelkreises parallel über gemeinsame physikalische Kanten übertragen. Daneben müssen noch Pakete der Topologieerkennung sowie des Sicherheits- und Konfigurationsmanagements transportiert werden.

A.2 Fahrzeugnetz

Die Daten des Fahrzeugnetzes basieren auf anonymisierten Informationen des Herstellers, die intern zur Verifikation der Kommunikationsarchitektur verwendet werden. Sie wurden im Field Bus Exchange Format (FIBEX) gespeichert und eingelesen. Die Quelldaten beinhalten die ungefähre Position aller ECUs im Fahrzeug sowie Sende- und Empfangsschnittstellen. Aus der Beschreibung der Sendeschnittstellen mit der Datengröße der übertragenen Informationen und der entsprechenden Empfangsschnittstellen der Informationen an den unterschiedlichen Bussystemen wurden die Kommunikationsbeziehungen erstellt. Die im FIBEX eingelesenen Daten sind in dem Anforderungsgraphen in Abbildung A.2 visualisiert. Eine rote Kante entspricht dabei einem übertragenen Informationsblock (unabhängig von der übertragenen Datenmenge), dicke rote Kanten entsprechen somit einer Vielzahl dieser Übertragungen.

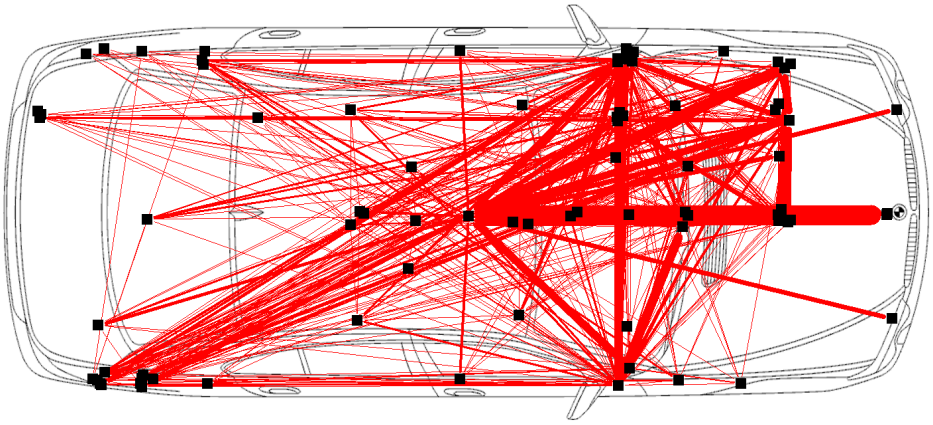


Abbildung A.2: Referenznetz des Fahrzeuges

Die Daten wurden manuell erweitert, um ein möglichst realistisches Abbild eines vom Band gelaufenen durchschnittlichen Fahrzeuges der Luxusklasse zu haben. Dazu wurden im ersten Schritt sich auf Grund der Konfiguration gegenseitig ausschließende ECUs entfernt. Dies sind überwiegend ECUs, welche in verschiedenen Sonderausstattungsvarianten die gleiche Aufgabe übernehmen (im Sprachgebrauch als „120%“-Konfiguration dargestellt). Im nächsten Schritt wurden die Videoverbindungen in den Anforderungsgraphen eingepflegt. Videoverbindungen werden in aktuellen Bordnetzen nicht über Busnetze übertragen, sondern über klassische direkte Videoleitungen. Ein Videoumschalter an einer zentralen Stelle verknüpft Quellen und Senken. In den in dieser Arbeit vorgeschlagenen Lösungen für ein Bordnetz sollen aber auch diese konventionellen Leitungen ersetzt werden. Anhand der Funktionsbeschreibung der Ausstattungen wurden aus diesem Grund die gewünschten Videoverbindungen manuell dem Anforderungsgraphen hinzugefügt. Als Letztes wurde eine einfache Gefährdungsanalyse durchgeführt. In diesem Schritt wurden einem Teil der ECUs eine Markierung hinzugefügt, die sie als sicherheitskritisch klassifizieren. Für die sicherheitskritischen ECUs gelten im Netzplanungsprozess besondere Nebenbedingungen, wie zum Beispiel zwei kanten- und knotendisjunkte Pfade zur ausfallsicheren Kommunikation. Die Einstufung erfolgte anhand der Funktionsbeschreibung der ECUs.

Parameter	Wert
Anzahl der ECU	83
davon sicherheitskritische	18
Angeforderte Verbindungen	1984

Tabelle A.1: Daten des Fahrzeugnetzes

Das Fahrzeugnetz besteht nach Modifikation der eingelesenen Daten aus 83 ECUs, wobei 18 als sicherheitskritisch eingestuft sind. Es werden insgesamt 1984 Kommu-

nikationsverbindungen zwischen Quellen und Senken angefordert. Die Werte sind in Tabelle A.1 zusammengefasst.

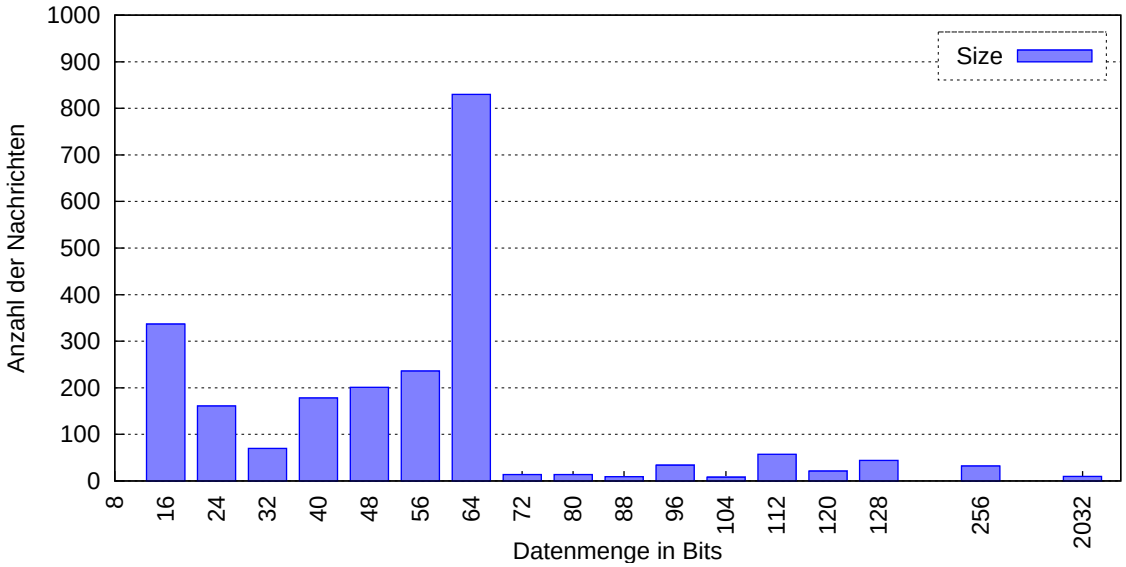


Abbildung A.3: Verteilung der Nachrichtengröße

Eine Auswertung der übertragenen Informationsblöcke ergibt die in Abbildung A.3 dargestellte Verteilung. Es handelt sich hierbei um die reine Größe der Daten ohne Nachrichtenkopf. Daten bis 64 bit beziehungsweise 8 Byte können über den CAN-Bus transportiert werden, aus diesem Grund entspricht dies der Mehrheit in der Verteilung. Größere Pakete können in dem diesem Referenznetz zugrunde liegenden physikalischen Bordnetz ausschließlich über Flexray-Technologie übertragen werden.

Anhand des optimierten Netzes aus Kapitel 4.3.3 wurden, ohne die Kommunikationsanforderungen mit einzubeziehen, die beim kürzesten Pfad entstehenden Pfadlängen aller Anforderungen ausgewertet, um eine Verteilung der Senken und Quellen im Fahrzeug zu untersuchen. Das Ergebnis ist in Diagramm A.4 visualisiert. Die mittlere Pfadlänge liegt bei ≈ 7 durchlaufenen Sternknoten. Es zeigen sich zwei Maxima im Bereich um 4 Hops sowie im Bereich um 10 Hops, vergleichbar mit der Überlagerung zweier Normverteilungen. Verglichen mit der physikalischen Verteilung des Einbaus der ECUs bilden sich zwei Schwerpunkte im Fahrzeug ab: der Kofferraum sowie der Geräteträger hinter der vorderen Spritzwand. Innerhalb dieser beiden Schwerpunkte und zwischen den beiden Schwerpunkten ist ein hohes Verkehrsaufkommen. In grober Näherung kann also von einer Gleichverteilung der Quellen und Senken im physikalischen Bordnetz ausgegangen werden.

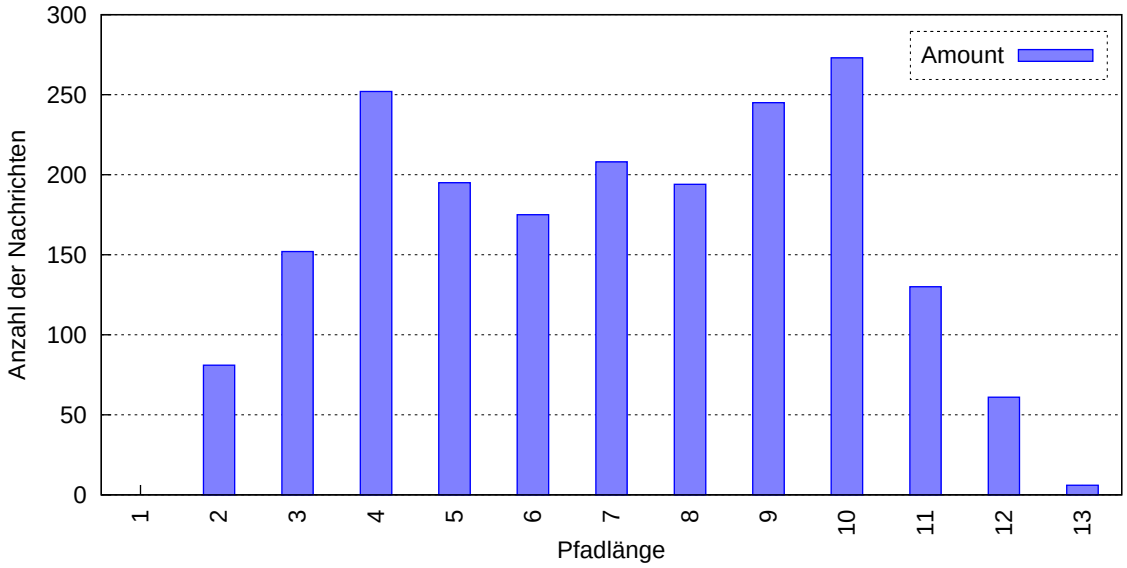


Abbildung A.4: Pfadlänge bei kürzestem Pfad

B Simulationsparameter

Im Gegensatz zum prototypischen Aufbau oder der analytischen Betrachtung der vorgeschlagenen Lösungen muss bei der Simulation eine definierte statistische Verteilung in den Eingangsparametern erfolgen, um den Ergebnisraum möglichst vollständig abzudecken. Dieses Kapitel stellt die im Simulationsframework ITMsim verwendeten Quellenmodelle für Steuernachrichten von Regelkreisen sowie für die Übertragung von Videodaten dar.

B.1 Verkehrsgenerator für Steuernachrichten

Die in Bordnetzen vorherrschenden Nachrichten sind zyklischer Natur. Dies ist vor allem aus historischen Gründen durch die klassischen Regelkreise bedingt. Aus diesem Grund wurden auch die Verkehrsgeneratoren für Steuernachrichten in ITMsim zyklisch ausgelegt. Alle in der vorliegenden Datenbasis als ereignisgesteuert markierten Datenübertragungen wurden auf zyklische übersetzt, indem der angegebenen Mindestabstand zwischen zwei Übertragungen als Zykluszeit gesetzt wurde.

Das verwendete Modell hat somit vier Eingangsparameter: „Cylce“, „Size“, „Offset“ und „Jitter“. Die Informationen für Cycle t_c und Size liegen im Anforderungsgraphen vor und werden für alle Durchläufe der Simulation gleichbleibend von dort eingelesen. Der Parameter Offset t_o legt die Verschiebung des Sendezeitpunktes t_a des Paketes x im Verhältnis zum Zeitpunkt 0 fest. Damit wird die Synchronisation aller Regelkreise untereinander festgelegt. Der Offset wird nur beim Start der Simulation benötigt. Der Jitter t_j ist dann nach Berechnung des zyklischen Sendezeitpunktes die Varianz der tatsächlichen Aussendung des Paketes um diesen Zeitpunkt herum.

Daraus folgt: $t_{ax} = 0 + t_o + x \cdot t_c + t_{jx} \cdot t_c$, wobei $t_j \in]-1{,}1[$ und $t_o \in \mathbb{Q} \geq 0$.

Durch Variation des Parameters Offset wird dem Umstand, dass nicht alle Regelkreise gleichzeitig auf den entsprechenden Verarbeitungseinheiten laufen können, Rechnung getragen. Der Jitter spiegelt die Ungenauigkeit des Schedulers in den Electronic Control Units (ECUs) wieder. Der ungünstigste generierbare Fall sind Durchläufe mit $t_o = t_j = 0$, also eine unrealistische Situation, bei der alle Steuerungsdaten jeweils gleichzeitig gesendet werden. Weitere Durchläufe der Simulationen wurden mit gleichverteiltem $t_j \in [-0.01{,}0.01]$ und gleichverteiltem $t_o \in [0; 10\,\text{ms}]$ durchgeführt. Dabei wurde auf entsprechende Zufälligkeit der Werte geachtet.

B.2 Verkehrsgenerator für Videodaten

Das zweite implementierte Quellenmodell bildet den Datenstrom eines komprimierten Videos ab. Die Implementierung basiert auf den Ansätzen und Auswertungen von Melamed et al. [MRSZ92]. Modelliert wird die GOP-Sequenz „IBPBPBPBPBPBIBPBPBPBPBPBI...“, eine klassische Folge für H.263-kodierte Video-Inhalte, mit einer PAL-Bildrate von 25 Hz beziehungsweise je einem Bild im Zyklus von 40 ms. Die mit diesem Modell berechnete Abfolge von Frames ist für einen repräsentativen Zeitabschnitt in Abbildung B.1 dargestellt.

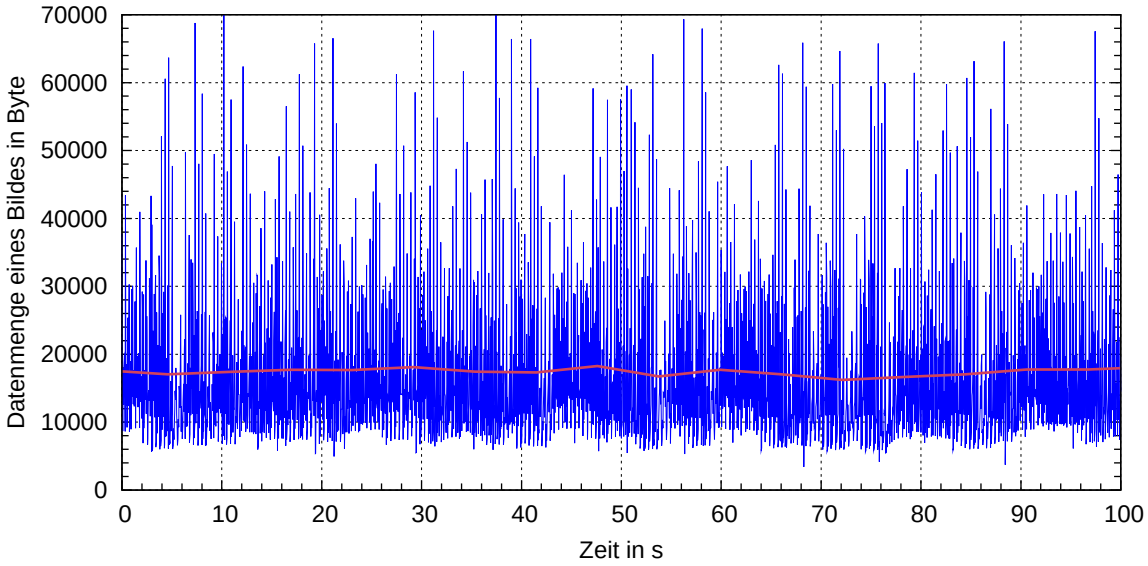


Abbildung B.1: Datenmenge eines Bildes und mittlere Datenrate

Aufgetragen ist die Datenmenge eines jeden mit dem Modell berechneten Bildes über die Zeit. Die Spitzen repräsentieren die I-Frames, welche weit größer wie die im Ethernet festgelegte Maximum Transmission Unit (MTU) von 1500 Byte ist. Somit benötigen diese Datenbursts einen Mechanismus zur Fragmentierung der Bilddaten und blockieren den Kanal über eine längere Zeit. Auch eine statistisch auftretende Variabilität der Komplexität des zu komprimierenden Inhaltes und damit eine schwankende mittlere Datenrate wird in dem verwendeten Modell abgebildet. Die durchschnittliche Datenrate, die rote Linie in Abbildung B.1, beträgt $\approx 3{,}6$ Mbit/s und ist damit mit den SD-Inhalten einer handelsüblichen DVD vergleichbar.

C uMAP Nachrichtenkopf-Felder

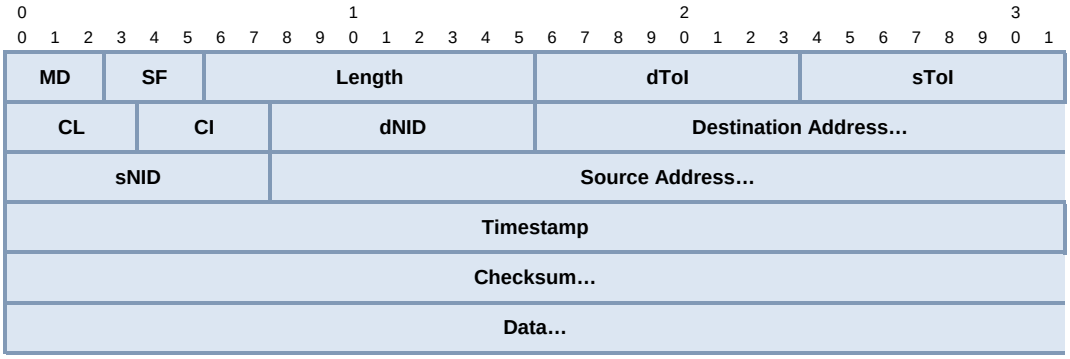


Abbildung C.1: uMAP Header Aufbau

MD

Mode of operation.
Predefined values:

- 000: Write data into controller
- 001: Modify data in controller
- 010: Read data from controller
- 011: Poll data from controller
- 100: Acknowledgement
- 101: Cancel Write/Modify request
- 110: Withdraw Read data
- 111: Cancel Poll request

SF

Safety Flag.
Predefined values:

- 000: no error checking
- 100: send acknowledge on successful reception
- 010: uses checksum in this section with generic algorithm and length derived from destination Type of ID (dToI)
- 011: uses checksum in this section with user defined algorithm
- 111: or combinations from above

Length

Number of Bytes to next section, so including all headers and data.
Predefined values in Bit:

- 0 00: no further section in this packet
- 0 01 - 3 FD: size in Byte (max. 1021 Byte)
- 3 FE: section utilize whole packet, size of an Ethernet MTU (typically 1500 Byte)
- 3 FF: Data is fragmented, section utilize whole packet

dToI / sToI / Address

ToI stands for Type of ID and specifies the format of the destination identifier.
The first bit specifies if the identifier uses extended format and so includes a Network Identifier (NID) for routing through different gateways:

- 0: without NID
- 1: with NID

The next 3 Bit preset the length of the identifier field. Definded values:

- 000: 2 Byte
- 001: 4 Byte
- 010: 6 Byte
- 011: 6 Byte
- 100: 8 Byte
- 101: 10 Byte
- 110: 16 Byte
- 111: 18 Byte

The subsequent 4 bits specifies the type of the identifier. Predefined values vor dToI in HEX:

- 01: CAN Base frame format
- 02: CANopen
- 07: LIN 1.3
- 08: LIN 2.0
- 09: Flexray
- 11: CAN Extended frame format
- 12: CAN Gateway frame format
- 19: IPv4
- 21: Ethernet MAC-Address
- 69: IPv6
- 31: IPv4/TCP
- 71: IPv6/TCP
- 39: IPv4/UDP
- 79: IPv6/UDP

dNID / sNID

Optional network identifier, specifies the subnet within the network. With the help of the NID, the needed gateway to access the subnet is addressed.
Predefined values in HEX:

- 00: this (local) network
- 01: default Internet access point
- 02 - 7F: unicast networks
- 80 - FE: multicast groups
- FF: all networks

Timestamp

The Timestamp specifies the wall clock time.
The resolution are μseconds. The timestamp for the same identifier must differ in time for minimum of $1\mu s$. Messages with same timestamp and identifier will overwrite the old message.

CL / CI / Checksum

Optional field. It starts with an optional 4 Bit Checksum Length (CL) and a 4 Bit Checksum Identifier (CI), specify the length and the used algorithm. Subsequent a variable sized checksum field depending on the chosen length.

Calculated values CL:

- 0000: 1 Byte
- 0001: 2 Byte
- 0010: 4 Byte
- 0011: 8 Byte
- ...
- 1010: 1024 Byte

Stuffing

0 Bit to pad header till next full word corresponding to the length field minus data field length.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

Veröffentlichungen des Autors

[EMR05] EICHLER, Stephan ; MÜLLER-RATHGEBER, Bernd: Performance Analysis of Scalable Certificate Revocation Schemes for Ad Hoc Networks. In: *LCN '05: Proceedings of the The IEEE Conference on Local Computer Networks 30th Anniversary*. Washington, DC, USA : IEEE Computer Society, 2005. – ISBN 0–7695–2421–4, S. 382–391

[LBBMR08] LEIPOLD, F. ; BOVELLI, S. ; BRANDNER, S. ; MÜLLER-RATHGEBER, B.: Performance evaluation of the WiMedia MAC protocol in high density wireless environments. In: *2nd International Symposium on Advanced Networks and Telecommunication Systems, 2008 (ANTS '08)*, 2008, S. 1–3

[MREM08a] MÜLLER-RATHGEBER, B. ; EICHHORN, M. ; MICHEL, H.U.: A unified Car-IT Communication-Architecture: Design guidelines and prototypical implementation. In: *2008 IEEE Intelligent Vehicles Symposium (IV08)*, 2008, S. 709–714

[MREM08b] MÜLLER-RATHGEBER, B. ; EICHHORN, M. ; MICHEL, H.U.: A unified Car-IT Communication-Architecture: Network switch design guidelines. In: *IEEE International Conference on Vehicular Electronics and Safety (ICVES), 2008*, 2008, S. 16–21

[MRM09] MÜLLER-RATHGEBER, B. ; MICHEL, H.U.: Automotive Network Planning - a genetic approach. In: *2009 IEEE Intelligent Vehicles Symposium (IV09)*, 2009

[MRR08] MÜLLER-RATHGEBER, B. ; RAUCHFUSS, H.: A Cosimulation Framework for a Distributed System of Systems. In: *IEEE 68th Vehicular Technology Conference, 2008. VTC 2008-Fall*, 2008, S. 1–5

[RMRS07] RAHMANI, M. ; MÜLLER-RATHGEBER, B. ; STEINBACH, E.: Error Detection Capabilities of Automotive Network Technologies and Ethernet-A Comparative Study. In: *2007 IEEE Intelligent Vehicles Symposium (IV07)*, 2007, S. 674–679

[SEMR06] SCHWINGENSCHLÖGL, Christian ; EICHLER, Stephan ; MÜLLER-RATHGEBER, Bernd: Performance of PKI-based security mechanisms in mobile ad hoc networks. In: *AEU - International Journal of Electronics and Communications* 60 (2006), Nr. 1, S. 20 – 24. – ISSN 1434–8411

Allgemeine Veröffenlichungen

[ABBW05] ANDERSSON, L. ; BRAND, K. P. ; BRUNNER, C. ; WIMMER, W.: Reliability investigations for SA communication architectures based on IEC 61850. In: *Proc. IEEE Russia Power Tech*, 2005, S. 1–7

[AF07] ARLOS, P. ; FIEDLER, M.: A Method to Estimate the Timestamp Accuracy of Measurement Hardware and Software Tools. In: *Lecture Notes in Computer Science* 4427 (2007), S. 197

[AKG+06] ADEMAJ, A. ; KOPETZ, H. ; GRILLINGER, P. ; STEINHAMMER, K. ; HANZLIK, A.: Fault-tolerant time-triggered ethernet configuration with star topology. In: *Dependability and Fault Tolerance Workshop*, 2006

[AKGS06] ADEMAJ, A. ; KOPETZ, H. ; GRILLINGER, P. ; STEINHAMMER, K.: Integration of Predictable and Flexible In-Vehicle Communication Using Time-Triggered Ethernet / Society of Automotive Engineers, 400 Commonwealth Dr, Warrendale, PA, 15096, USA,. 2006. – Forschungsbericht

[Alb04] ALBERT, A.: Comparison of event-triggered and time-triggered concepts with regard to distributed control systems. In: *Embedded World* (2004), S. 235–252

[ari05] Airlines Electronic Engeneering Committee, AERONATICAL RADIO: *Aircraft Data Network Part 7, Avionics full duplex switched Ethernet (AFDX) network, 664P7*. 2005

[Bak87] BAKER, J.E.: Reducing bias and inefficiency in the selection algorithm. In: *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application table of contents* L. Erlbaum Associates Inc. Hillsdale, NJ, USA, 1987, S. 14–21

[Bal96] BALL, P.: Introduction to discrete event simulation. In: *2nd DYCOMANS workshop on Management and Control: Tools in Action* (1996), S. 367–376

[Bar57] BARRICELLI, N.A.: Symbiogenetic evolution processes realized by artificial methods. In: *Methodos* 9 (1957), Nr. 35-36, S. 143–182

[bel02] FlexRay Consortium: *FlexRay Requirements Specification*. `http://www.flexray.com`. Version: 2002

[Bha99] BHANDARI, R.: *Survivable networks: algorithms for diverse routing*. Kluwer Academic Pub, 1999

[Bos91] Robert Bosch GmbH: *CAN Specification Version 2.0 Part A*. 1991

[BOW03] BRAND, K.P. ; OSTERTAG, M. ; WIMMER, W.: Safety related, distributed functions in substations and the standard IEC 61850. In: *Power Tech Conference Proceedings, 2003 IEEE Bologna* 2 (2003)

[Bre62] BREMERMANN, H.J.: Optimization through evolution and recombination. In: *Self-Organizing Systems* (1962), S. 93–106

[Bro00] BROWN, S.: Overview of IEC 61508. Design of electrical/electronic/programmableelectronic safety-related systems. In: *Computing & Control Engineering Journal* 11 (2000), Nr. 1, S. 6–12

[BT96] BLICKLE, T. ; THIELE, L.: A comparison of selection schemes used in evolutionary algorithms. In: *Evolutionary Computation* 4 (1996), Nr. 4, S. 361–394

[Cay89] CAYLEY, A.: A theorem on trees. In: *The Quarterly Journal of Mathematics* 23 (1889), S. 376–378

[Cha10] CHAKRABORTY, Samarjit ; EBERSPÄCHER, Jörg (Hrsg.): *Advances in Real-Time Systems*. Springer Verlag, erscheint 2010

[CP02] CANTÚ-PAZ, E.: On random numbers and the performance of genetic algorithms. In: *Proceedings of the Genetic and Evolutionary Computation Conference* Citeseer, 2002, S. 311–318

[Cru91a] CRUZ, RL: A calculus for network delay, part I: Network elements in isolation. In: *IEEE Transactions on information Theory* 37 (1991), Nr. 1, S. 114–131

[Cru91b] CRUZ, R.L.: A calculus for network delay, part II: Network analysis. In: *IEEE Transactions on Information Theory* 37 (1991), Nr. 1, S. 132–141

[CTS08] COPPEL, S. ; TIBBALS, T. ; SILGARDO, A.: Practical Considerations for Ethernet Networking Within Substations / American Electric Power and Schweitzer Engeneering Laboratories, Inc. 2008 (TP6317-01). – Forschungsbericht

[DFP02] DRAGO, N. ; FUMMI, F. ; PONCINO, M.: Modeling network embedded systems with NS-2 and SystemC. In: *1st IEEE International Conference on Circuits and Systems for Communications, ICCSC'02.*, 2002, S. 240–245

[DH02] DITTMANN, G. ; HERKERSDORF, A.: Network processor load balancing for high-speed links. In: *Proceedings of the 2002 International Symposium on Performance Evaluation of Computer and Telecommunication Systems* Bd. 735 Citeseer, 2002

[Dij59] DIJKSTRA, E.W.: A note on two problems in connexion with graphs. In: *Numerische Mathematik* 1 (1959), Nr. 1, S. 269–271

[EAVH91] EIBEN, A. ; AARTS, E. ; VAN HEE, K.: Global convergence of genetic algorithms: A Markov chain analysis. In: *Parallel Problem Solving from Nature* (1991), S. 3–12

[EM92] EVANS, J.R. ; MINIEKA, E.: *Optimization algorithms for networks and graphs*. M. Dekker New York, 1992

[FFMT04] FERRARI, P. ; FLAMMINI, A. ; MARIOLI, D. ; TARONI, A.: Experimental evaluation of PROFINET performance. In: *2004 IEEE International Workshop on Factory Communication Systems, 2004. Proceedings*, 2004, S. 331–334

[FFV05] FERRARI, P. ; FLAMMINI, A. ; VITTURI, S.: Response Times Evaluation of PROFINET Networks. In: *Industrial Electronics, 2005. ISIE 2005. Proceedings of the IEEE International Symposium on* Bd. 4, 2005

[FLM+05] FUMMI, F. ; LOGHI, M. ; MARTINI, S. ; MONGUZZI, M. ; PERBELLINI, G. ; PONCINO, M.: Virtual Hardware Prototyping through Timed Hardware-Software Co-Simulation. In: *Proceedings of the Design, Automation and Test in Europe (DATE'05) Volume 2-Volume 02* (2005), S. 798–803

[FPG+03] FUMMI, F. ; PERBELLINI, G. ; GALLO, P. ; PONCINO, M. ; MARTINI, S. ; RICCIATO, F.: A Timing-Accurate Modeling and Simulation Environment for Networked Embedded Systems. In: *Proceedings of the 40th Design Automation Conference (DAC03)* 1 (2003), S. 58113–688

[Fra57] FRASER, A.S.: Simulation of Genetic Systems by Automatic Digital Computers. In: *Australian Journal of Biological Sciences* 10 (1957), S. 484–491

[FSN04] FELD, J. ; SIEMENS, AG ; NURNBERG, G.: PROFINET-Scalable Factory Communication for all applications. In: *2004 IEEE International Workshop on Factory Communication Systems, 2004. Proceedings*, 2004, S. 33–38

[GBCH01] GUNTHER, S. ; BINNS, F. ; CARMEAN, D.M. ; HALL, J.C.: Managing the impact of increasing microprocessor power consumption. In: *Intel Technology Journal* 5 (2001), Nr. 1, S. 9

[GD91] GOLDBERG, D.E. ; DEB, K.: A comparative analysis of selection schemes used in genetic algorithms. In: *Foundations of Genetic Algorithms* 1 (1991), S. 69–93

[GH85] GROSS, D. ; HARRIS, C.M.: *Fundamentals of queueing theory* . John Wiley & Sons, Inc. New York, NY, USA, 1985

[GJ+79] GAREY, M.R. ; JOHNSON, D.S. u. a.: *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman, San Francisco, 1979

[Gol89] GOLDBERG, David E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*. Massachusetts : Addison-Wesley Publishing Company, Inc., 1989. – ISBN 0–201–15767–5

[Hed98] HEDENETZ, B.: A development framework for ultra-dependable automotive systemsbased on a time-triggered architecture. In: *Real-Time Systems Symposium, 1998. Proceedings., The 19th IEEE*, 1998, S. 358–367

[iec99] International Electrotechnical Commission: *61508: Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems*. 1999

[iso03] International Organization for Standardization, ISO: *Road vehicles – Controller area network (CAN) – Part 2: High-speed medium access unit ISO 11898-2:2003*. 2003

[iso09] International Organization for Standardization, ISO: *Road vehicles – Functional safety ISO 26262*. 2009

[KAGS05] KOPETZ, H. ; ADEMAJ, A. ; GRILLINGER, P. ; STEINHAMMER, K.: The time-triggered Ethernet (TTE) design. In: *Object-Oriented Real-Time Distributed Computing, 2005. ISORC 2005. Eighth IEEE International Symposium on* (2005), S. 22–33

[KB03] KOPETZ, H. ; BAUER, G.: The time-triggered architecture. In: *Proceedings of the IEEE* 91 (2003), Nr. 1, S. 112–126

[KBD+04] KRAMMEN, J. ; BORNAT, P. ; DENGEL, G. ; KUTTENKEULER, S. ; LUKAS, R. ; OBERHOFER, K. ; RUH, J. ; QUECKE, H.: FIBEX: An exchange format for networks based on field busses. In: *Ingénieurs de l'automobile(Paris)* (2004), S. 78–83

[Kle75] KLEINROCK, L.: *Queueing systems*. Wiley Interscience, New York, 1975

[Kop91] KOPETZ, H.: Event-triggered versus time-triggered real-time systems. In: *Proceedings of the International Workshop on Operating Systems of the 90s and Beyond* Springer-Verlag London, UK, 1991, S. 87–101

[Kop93] KOPETZ, H.: Should responsive systems be event-triggered or time-triggered? In: *IEICE TRANSACTIONS on Information and Systems* 76 (1993), Nr. 11, S. 1325–1332

[KS05] KONAK, A. ; SMITH, A.E.: Designing resilient networks using a hybrid genetic algorithm approach. In: *Proceedings of the 2005 conference on Genetic and evolutionary computation* ACM New York, NY, USA, 2005, S. 1279–1285

[LB98] LE BOUDEC, J.Y.: Application of network calculus to guaranteed service networks. In: *IEEE Transactions on Information Theory* 44 (1998), Nr. 3, S. 1087–1096

[LH04] LOESER, J. ; HAERTIG, H.: Low-latency hard real-time communication over switched Ethernet. In: *Real-Time Systems, 2004. ECRTS 2004. Proceedings. 16th Euromicro Conference on* (2004), S. 13–22

[LHD04] LOESER, J. ; HAERTIG, H. ; DRESDEN, TU: Using switched ethernet for hard real-time communication. In: *Parallel Computing in Electrical Engineering, 2004. International Conference on* (2004), S. 349–353

[lin99] Audi AG, BMW AG, DaimlerChrysler AG, Motorola Inc. Volcano Communication Technologies AB, Volkswagen AG, and Volvo Car Corporation: *LIN specification and LIN press announcement*. 1999

[MB76] METCALFE, R.M. ; BOGGS, D.R.: Ethernet: distributed packet switching for local computer networks. In: *Communications of the ACM* 19 (1976), Nr. 7, S. 395–404

[MDS+91] MOED, M.C. ; DANBURY, CT ; STEWART, C.V. ; TROY, NY ; KELLY, R.B.: Reducing The Search Time Of A Steady State Genetic Algorithm Using the Immigration Operator. In: *Third International Conference on Tools for Artificial Intelligence, TAI'91: November 5-8, 1991, San Jose, California* IEEE Computer Society Press, 1991

[Men32] MENGER, K.: Das Botenproblem. In: *Ergebnisse eines mathematischen Kolloquiums* 2 (1932), S. 11–12

[MF97] MEYSENBURG, M.M. ; FOSTER, J.A.: The quality of pseudo-random number generators and simple genetic algorithm performance. In: *Proceedings of the Seventh International Conference on Genetic Algorithms*, 1997, S. 276–282

[MMP90] MONMA, C.L. ; MUNSON, B.S. ; PULLEYBLANK, W.R.: Minimum-weight two-connected spanning networks. In: *Mathematical Programming* 46 (1990), Nr. 1, S. 153–171

[Moo65] MOORE, G.E.: Cramming more components onto integrated circuits. In: *Electronics* 38 (1965), Nr. 8

[mos04] Most Cooperation: *MOST Media Oriented System Transport-Multimedia and Control Networking Technology Specification Rev.* 2004

[MRSZ92] MELAMED, B. ; RAYCHAUDHURI, D. ; SENGUPTA, B. ; ZDEPSKI, J.: TES-based traffic modeling for performance evaluation of integrated networks. In: *INFOCOM '92. Eleventh Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE*, 1992, S. 75–84 vol.1

[MS89] MONMA, C.L. ; SHALLCROSS, D.F.: Methods for designing communications networks with certain two-connected survivability constraints. In: *Operations Research* 37 (1989), Nr. 4, S. 531–541

[MSV93] MUEHLENBEIN, H. ; SCHLIERKAMP-VOOSEN, D.: Predictive models for the breeder genetic algorithm I. Continuous parameter optimization. In: *Evolutionary Computation* 1 (1993), Nr. 1, S. 25–49

[MW04] MITSCHKE, M. ; WALLENTOWITZ, H.: *Dynamik der Kraftfahrzeuge*. Springer, 2004

[Rah09] RAHMANI, Mehrnoush: *A Resource-Efficient IP-based Network Architecture for In-Vehicle Communication*. Dr. Hut Verlag, 2009

[RSBH98] RINGLER, T. ; STEINER, J. ; BELSCHNER, R. ; HEDENETZ, B.: Increasing System Safety for by-wire Applications in Vehicles by Using a Time-Triggered Architecture. In: *Lecture Notes in Computer Science* (1998), S. 243–253

[RTC92] RTCA, R.D.O.: *DO-178B Software Considerations in Airborne Systems and Equipment Certification*. 1992

[SAE02] Society of Automotive Engineers, SAE: *High-Speed CAN (HSC) for Vehicle Applications at 500 KBPS, J2284*. March 2002

[SD07] SCHEER, G.W. ; DOLEZILEK, D.J.: Selecting, Designing, and Installing Modern Data Networks in Electrical Substations. In: *proceedings of the 9th Annual Western Power Delivery and Automation Conference, Spokane, WA, April*, 2007

[SGAK06] STEINHAMMER, K. ; GRILLINGER, P. ; ADEMAJ, A. ; KOPETZ, H.: A time-triggered ethernet (TTE) switch. In: *Proceedings of the conference on Design, automation and test in Europe: Proceedings* European Design and Automation Association 3001 Leuven, Belgium, Belgium, 2006, S. 794–799

[Syw89] SYWERDA, G.: Uniform crossover in genetic algorithms. In: *Proceedings of the third international conference on Genetic algorithms table of contents* Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1989, S. 2–9

[V+01] VARGA, A. u. a.: The OMNeT++ discrete event simulation system. In: *Proceedings of the European Simulation Multiconference (ESM2001)* (2001). `http://www.omnetpp.org/`

[Vol00] Robert Bosch GmbH: *Supplement to IDB-1394: System and Interoperability Layer*. 2000

[Wei] WEINMANN, U.: *Software im Automobil - Technische und Wirtschaftliche Perspektiven*. BMW Car IT GmbH

[Whi89] WHITLEY, D.: The GENITOR algorithm and selection pressure: why rank-based allocation of reproductive trials is best. In: *Proceedings of the third international conference on Genetic algorithms table of contents* Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 1989, S. 116–121

[WWK+99] WITKOWSKI, M.L. ; WALKER, W.J. ; KHAN, M.A. ; KOTZUR, G.B. ; MAYER, D.J.: Direct media independent interface connection system for network devices. In: *United States Patent 5,892,926* (1999)

[ZS06] ZIMMERMANN, W. ; SCHMIDGALL, R.: *Bussysteme in der Fahrzeugtechnik*. Vieweg, 2006