Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Till Steinbach

Echtzeit-Ethernet für Anwendungen im Automobil: Metriken und deren simulationsbasierte Evaluierung am Beispiel von TTEthernet

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Till Steinbach

Echtzeit-Ethernet für Anwendungen im Automobil: Metriken und deren simulationsbasierte Evaluierung am Beispiel von TTEthernet

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

Betreuender Prüfer: Prof. Dr. Franz Korf
Zweitgutachter: Prof. Dr. Thomas Schmidt

Abgegeben am 28. Februar 2011

Till Steinbach

Thema der Masterarbeit
Echtzeit-Ethernet für Anwendungen im Automobil:
Metriken und deren simulationsbasierte Evaluierung am Beispiel von TTEthernet
Stichworte
Echtzeit-Ethernet, Fahrzeug-Netzwerke, simulationsbasierte Evaluierung,
Netzwerk-Metriken, Modellierung, Messung, time-triggered, OMNeT++, TTEthernet
Kurzzusammenfassung
Durch die Zunahme von elektronischen Systemen, insbesondere im Fahrerassistenz- und Komfortbereich, kommen die etablierten Automotive-Kommunikationstechnologien an die Grenze ihrer Leistungsfähigkeit. Ein neuer Ansatz für die Kommunikation zwischen Steuergeräten im Automobil ist der Einsatz von Ethernet. Echtzeit-Erweiterungen haben den Einsatzbereich von standard switched Ethernet auf zeitkritische Anwendungen ausgedehnt.
Diese Arbeit leistet einen Beitrag zur Bewertung dieser neuen Konzepte, indem eine simulationsbasierte Evaluierungstrategie für die Ermittlung von Kenngrößen (Metriken) Echtzeit-Ethernet-basierter Vermittlungsinfrastrukturen entwickelt wird. Eine Überprüfung anhand einer Fallstudie einer realen Applikation zeigt, dass sich der entwickelte Evaluierungsprozess für erste Untersuchungen von Ethernet-basierten Automotive-Anwendungen gut eignet. Eine gründliche Analyse des Simulationsmodells, welche die Simulationsergebnisse mit Berechnungen eines mathematischen Modells und Messungen auf echter Hardware vergleicht, belegt die Gültigkeit der Implementierung und der mit ihr ermittelten Kenngrößen.

Title of the paper
Real-time Ethernet for automotive applications:
Simulation based evaluation of network metrics using the example of TTEthernet
Keywords
Real-time Ethernet, In-vehicle network, Simulation based evaluation, Network metrics,
Modeling, Measurement, time-triggered, OMNeT++, TTEthernet
Abstract
The established automotive communication technologies reach their performance limits due to the increase of electronic systems, especially in driver assistance and comfort applications. Ethernet is a new approach for the communication between control units in cars. Real-time extensions widen the application of standard switched Ethernet into the time-critical domain. This work introduces a simulation-based evaluation strategy to determine network metrics of real-time Ethernet based communication. An evaluation on the basis of a real automotive application shows the suitability of the given approach. A careful analysis of the simulation model that compares the simulation results with calculations obtained from a mathematical model as well as with real-world measurements using hardware, assures the validity of the implementation and the determined network metrics.

Inhaltsverzeichnis

Kapitel 1

Einführung

1.1 Überblick

Ein modernes Kraftfahrzeug unterstützt den Fahrer mit einer Vielzahl an Fahrerassistenzsystemen. Beispiele sind die Antriebsschlupfregelung (ASR) oder das elektronische Stabilitätsprogramm (ESP). Auch die „X-by-Wire-Technologie[1]“, also die Übertragung von Steuer- und Regelungsbefehlen des Fahrers über Datenleitungen im Gegensatz zur mechanischen Übertragung, nimmt zu (vgl. Leen und Heffernan, 2002). Zudem verfügen Kraftfahrzeuge über immer mehr Informations- und Unterhaltungselektronik. Während in den 70er Jahren noch das Autoradio die Krönung der Fahrzeugelektronik darstellte, werden in heutigen Oberklasse-Fahrzeugen bis zu 70 Steuergeräte verbaut (vgl. Navet u. a., 2005).
Folglich sind heutige Fahrzeuge komplexe verteilte Echtzeit-Systeme, deren Kommunikationsflüsse zwischen den verteilten Modulen einen signifikanten Einfluss auf ihre Zuverlässigkeit und die Sicherheit und den Komfort der Insassen haben.
Durch die Zunahme von elektronischen Systemen im Automobil sind auch die Anforderungen an leistungsfähige Kommunikations-Verbindungen zwischen den einzelnen Steuergeräten gestiegen. Das Spitzenmodell Phaeton der Volkswagen AG (siehe Abbildung 1.1 auf der nächsten Seite) überträgt beispielsweise ca. 2500 Signale zwischen seinen über 60 Steuergeräten. Dabei wird ein Kabelbaum mit über 3800 Meter Kabel und mehr als 2100 Einzelleitungen eingesetzt (vgl. Marscholik und Subke, 2007, S. 3). Die Steigerung des Bedarfs an leistungsfähigen Kommunikations-Verbindungen wird sich auch in den kommenden Jahren, insbesondere mit kamerabasierten Assistenzsystemen, unvermindert fortsetzen (vgl. Navet u. a., 2005) (siehe Abbildung 1.2 auf Seite 3). Es sind neue Entwürfe gefragt, welche diesen zunehmenden Anforderungen gerecht werden können (vgl. Belschner u. a., 2000).

[1] X-by-Wire bezeichnet den Ersatz von mechanischen Verbindungen zur Steuerung durch die Leitung elektrischer, elektronischer, optoelektronischer oder optischer Steuersignale vom Bediener zum Aktor

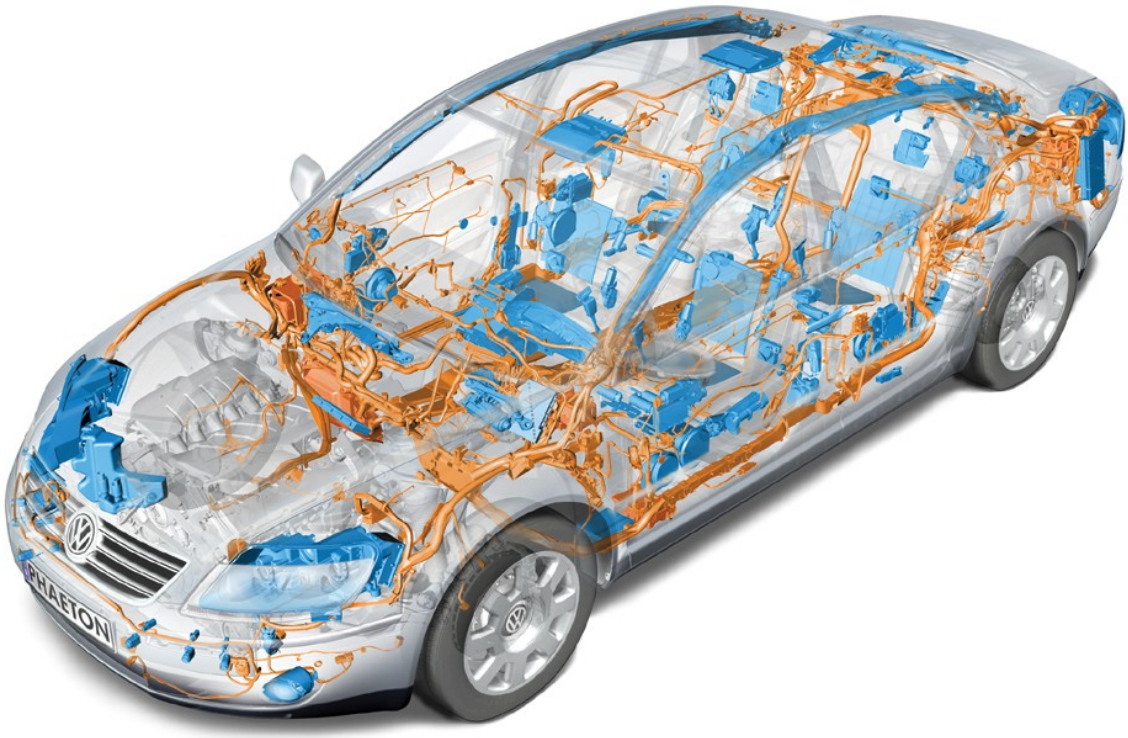

Abbildung 1.1: Bordnetz im VW Phaeton (Quelle: NYFEGA Elektro-Garage AG)

Ein neuer Architekturansatz für Kommunikation im automotive[2] Bereich basiert auf einem sogenannten *Backbone*[3]*-Netzwerk*, einer *intelligenten Vermittlungsinfrastruktur*, welche die verteilten Steuergeräte des Fahrzeugs — beispielsweise sternförmig — verbindet. Ein solcher zentraler Fahrzeug-Backbone muss starker Last gewachsen sein und dabei stets eine deterministische Nachrichtenübertragung garantieren.
Die Nutzung der Ethernet-Technologie in Fahrzeug-Netzwerken ist ein solcher Ansatz (vgl. Hammerschmidt, 2007). Ethernet hat bereits in den Computernetzen bewiesen, ein flexibles, hoch skalierbares Netzwerkprotokoll zu sein. Standardmäßig kann Ethernet jedoch nicht die erforderliche Vorhersagbarkeit und Zuverlässigkeit in der Paketweiterleitung und dem Medienzugriff garantieren, welche für eine Echtzeit-Kommunikation im Fahrzeug erforderlich ist.

[2] Automotive ist ein Oberbegriff für maschinenangetriebene Fahrzeuge

[3] Backbone (engl. für Rückgrat, Hauptstrang, Basisnetz) bezeichnet den zentralen Kernbereich eines Telekommunikationsnetzes

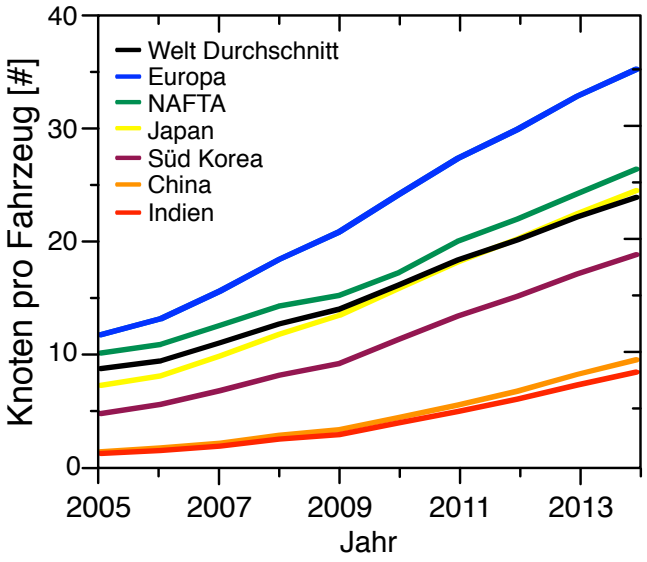

Abbildung 1.2: Durchschnittliche Anzahl Kommunikationsknoten pro Fahrzeug — Internationaler Vergleich (Bruckmeier, 2010)

Echtzeit-Erweiterungen von Standard-Ethernet versprechen zuverlässige, vorhersagbare Übertragung und haben im Bereich der Prozessautomatisierung bereits Einsatz in Produktivsystemen gefunden (vgl. Decotignie, 2005). Time-Triggered Ethernet (TTEthernet) (vgl. Steiner, 2008) ist eine Echtzeit-Ethernet-Erweiterung, welche versucht, die speziellen Bedürfnisse der Automotive-Kommunikation zu befriedigen (vgl. Elektronik, 2007). In TTEthernet werden alle sendenden Teilnehmer über ein ausfallgesichertes Protokoll synchronisiert. Anschließend operieren alle Teilnehmer nach einem vordefinierten gemeinsamen Zeitplan, der das simultane Senden zweier Echtzeit-Nachrichten über dieselbe physikalische Verbindung ausschließt. Damit erweitert TTEthernet das Standard-Ethernet-Protokoll um die Fähigkeit, zeitkritische Daten deterministisch zu transportieren.
Der Einsatz von Echtzeit-Ethernet als intelligente Vermittlungsinfrastruktur im Fahrzeug ist ein Novum (vgl. Bruckmeier, 2010). Obwohl das Interesse verschiedener Firmen und Institute in den letzten Jahren gestiegen ist und verschiedene Forschungsprojekte in diesem Bereich gestartet wurden, gibt es bisher noch wenige veröffentlichte Arbeiten, die Echtzeit-Ethernet für den Automotive-Einsatz evaluieren (Trend-Recherche: FIZ-Karlsruhe (vgl. Fachinformationszentrum Karlsruhe), ZIZ-Technik (vgl. WTI-Frankfurt eG)).
Diese Arbeit leistet einen Beitrag zur Bewertung neuer Ethernet-basierter Vermittlungsinfrastrukturen, indem eine simulationsbasierte Evaluierungstrategie für die Ermittlung von Kenngrößen (Metriken) entwickelt wird.

1.2 Motivation

Es gibt verschiedene Anreize für eine Weiterentwicklung der Datenübertragung im Automotive-Bereich. Ein Argument ist der heterogene Aufbau der Datenübertragungsnetze aktueller Serienfahrzeuge. Je nach Anwendung und zu übertragenden Daten kommen spezielle Bussysteme zum Einsatz. Jedes Bussystem ist für seine jeweilige Anwendung optimal; um jedoch eine Kommunikation zwischen den verschiedenen Technologien zu ermöglichen, müssen sogenannte Gateways eingesetzt werden, welche die Nachrichten zwischen den verschiedenen Systemen übersetzen (vgl. Dohmke, 2002). Diese heterogene Struktur macht das Fahrzeugnetzwerk kompliziert und damit in der Entwicklung teuer. Erstrebenswert ist eine Technologie, die sich weitgehend durchgängig im gesamten Fahrzeug einsetzen lässt.
Durch die heute vorherrschenden Preisunterschiede in den Komponenten ist die Vision eines homogenen, auf einer Technologie basierenden Fahrzeugnetzwerks noch in weiter Ferne. Der Einstieg in eine neue Ethernet-basierte Vermittlungsinfrastruktur ist jedoch heute bereits realistisch, erreichen doch mit dem steigenden Datenaufkommen der neuen Fahrerassistenz- und Komfortsysteme auch neue Automotive-Kommunikationstechnologien, die Grenzen ihrer Skalierbarkeit (vgl. Kämpchen u. a., 2005). „Eigentlich ist das Bordnetz im Gesamtfahrzeug bereits heute nicht mehr vernünftig zu beherrschen“ und „Die Komplexität steigt weiter“ (Badstübner, 2008, BMW Group Forschung und Technik).
TTEthernet (vgl. Steiner, 2008) ist, wie Vorarbeiten zeigen (vgl. Steinbach u. a., 2010), eine erfolgversprechende Technologie für eine zukünftige intelligente Vermittlungsinfrastruktur in Fahrzeugen. Da es Konzepte aus verschiedenen in Flug- und Fahrzeugen bewährten Protokollen wie *FlexRay* (vgl. FlexRay Consortium, 2005), *Time-triggered Protocol (TTP)* (vgl. TU Wien, 1997) oder dem *Avionics Full DupleX Switched Ethernet (AFDX)* (vgl. Aeronautical Radio Incorporated, 2002) aufgreift, ist es gut für Automotive-Anwendungen geeignet. Zudem bietet es den Transport von Nachrichten mit unterschiedlichen zeitlichen Anforderungen über eine homogene physikalische Infrastruktur. Eine Eigenschaft, die wegen der Vielschichtigkeit der Anwendungen in Fahrzeugen besonders attraktiv ist.
Um den Einsatz einer Ethernet-basierten Vermittlungsinfrastruktur im Fahrzeug zu evaluieren, ist es wichtig, ihre charakteristischen Netzwerk-Metriken zu identifizieren. Eine Abschätzung des technologischen wie wirtschaftlichen Erfolges kann auf Basis solcher aussagekräftigen Leistungsmerkmale deutlich einfacher erfolgen. Darüber hinaus unterstützt eine sorgfältige Analyse Optimierungen und Erweiterungen am Protokoll.
Während der Entwicklungsphase eines Autos, üblicherweise ca. drei Jahre (vgl. Schäuffele und Zurawka, 2010, S. 20), sind diverse Dienstleister und Zulieferer beteiligt. Verschiedene Teile und Baugruppen werden in Kooperation mit mehreren Partnern entwickelt. Der Fahrzeughersteller (Original Equipment Manufacturer (OEM)) ist in diesem Zusammenwirken für das Design und die Konfiguration der Infrastruktur zur Fahrzeugkommunikation verantwortlich. Eigenschaften des Zeitverhaltens und der Auslastung der Kommunikations-Infrastruktur sind insbesondere wegen der dezentralen Entwicklung schon zu einem sehr frühen Stadium

wichtig, noch bevor echte Komponenten zur Verfügung stehen. Spätere Erweiterungen und Build-to-Order (BTO)-Varianten[4]führen dabei zu einer großen Anzahl möglicher Konfigurationen, welche bei der Analyse berücksichtigt werden müssen.
Aus den vorgenannten Gründen eignet sich eine simulationsbasierte Evaluierungs-Strategie besonders gut für die Bewertung neuartiger Vermittlungsinfrastrukturen. Mit ihr kann der Effekt einer großen Anzahl von Konfigurationsparametern in umfassender Weise bereits vor der Realisierung überprüft werden. Aus den Simulationsergebnissen werden die Anforderungen an Dienstleister und Zulieferer definiert (vgl. Schäuffele und Zurawka, 2010, S. 31).
Darüber hinaus bietet die Simulation viele Vorteile in den verschiedenen Phasen der Entwicklung. Bei der Protokollentwicklung kann die Simulation dafür genutzt werden, frühzeitig, auch ohne Zugriff auf Hardware, Arbeiten an der Protokollschicht zu überprüfen. Testaufbauten mit großen Topologien, welche normalerweise hohe Hardwarekosten verursachen, können in einer Simulation genauso getestet werden wie der Einfluss von einzelnen Konfigurationsparametern. Beim Einsatz der Technologie kann eine Simulation verwendet werden, um frühzeitig den Einfluss verschiedener Topologien[5] zu testen und zu vergleichen.

1.3 Problemstellung und Zielsetzung

Das Hauptziel dieser Arbeit ist die Entwicklung eines simulationsbasierten Evaluierungsprozesses für die Bewertung von ausgewählten Kenngrößen in Echtzeit-Ethernet-Netzwerken. Ein dafür notwendiger Arbeitsschritt ist die Zusammenstellung, Bewertung und Kategorisierung dieser Metriken.
Der geplante Evaluierungsprozess (siehe Abbildung 1.3 auf der nächsten Seite) ist in drei Phasen unterteilt. In der Einrichtungsphase wird aus dem zu untersuchenden Netzwerkmodell ein Simulationsmodell der Vermittlungsinfrastruktur, der Datenquellen und -senken, sowie der Nachrichten erstellt. Aus den Anforderungen der zu Grunde liegenden Anwendungen können anschließend die Obergrenzen für die verschiedenen zu untersuchenden Metriken der Vermittlungsinfrastruktur gewonnen werden.
In der Simulationsphase wird das abgeleitete Modell in einer geeigneten Simulationsumgebung vermessen und die Ergebnisse werden dokumentiert. In der Auswertungsphase werden anschließend die Simulationsergebnisse mit den zu Beginn definierten Anforderungen verglichen und bewertet. Als Ergebnis dieses Prozesses erhält der Anwender einen Analysebericht über die vermessenen Metriken und ihre Konformität mit den Anforderungen. Durch eine präzise Abbildung der zeitlichen Eigenschaften im Simulationsmodell sollen die Ergeb-

[4]Build-to-Order (BTO) (deutsch: Fertigung nach Auftrag) bezeichnet eine Kombination aus Serien- und Auftragsfertigung. Details des Produktes werden dabei vom Kunden definiert, anschließend wird nach diesen Vorgaben gefertigt

[5]Mit Topologie wird in Computernetzen die Struktur der Verbindungen mehrerer Teilnehmer untereinander bezeichnet

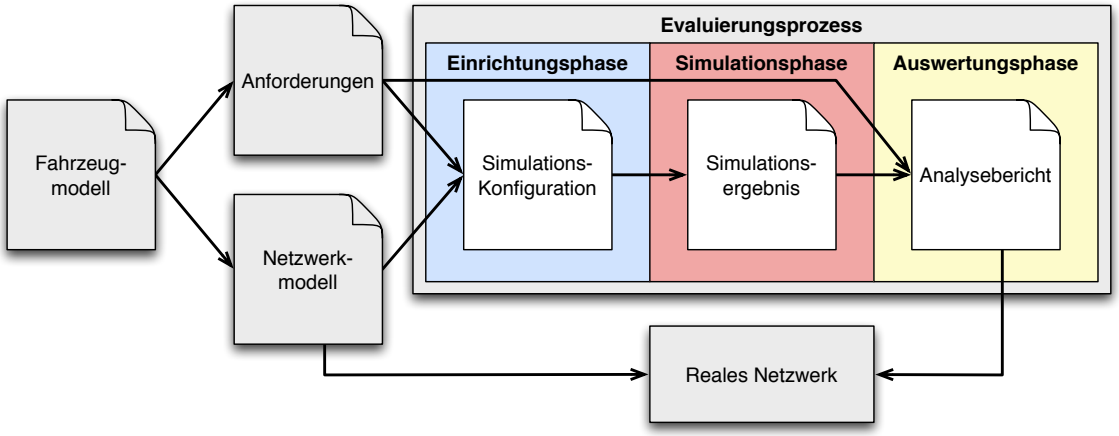

Abbildung 1.3: Überblick über die Phasen des Evaluierungsprozesses

nisse direkt auf ein mit demselben Netzwerkmodell konfiguriertes reales Netzwerk übertragen werden können.
Ein wichtiger Parameter des Netzwerkmodells ist die Topologie zwischen den Komponenten. Da sie entscheidenden Einfluss auf die Metriken der Vermittlungsinfrastruktur hat, soll es unkompliziert sein, die Simulation mit verschiedenen Topologien zu parametrisieren. Darüber hinaus ist eine geeignete Darstellung für die Anforderungen notwendig, die eine spätere automatisierte Auswertung erleichtert (vgl. Schäfer, 2004, S. 33).
Der Analysebericht, der die Simulationsergebnisse in Relation zu den Anforderungen darstellt, soll dem Nutzer in übersichtlicher Weise zeigen, an welcher Stelle das Modell nicht den Anforderungen genügt. Diese Informationen unterstützten den Nutzer bei der iterativen Optimierung der Konfiguration, bis alle Vorgaben erfüllt sind. Als Ergebnis erhält der Nutzer eine Netzwerkkonfiguration, welche direkt in vorhandene Hardware eingespielt werden kann, zusammen mit einer Übersicht über alle für ihn relevanten zu erwartenden Metriken. Dieser Prozess soll durch die enge Integration der Simulation in die Toolchain der Firma TTTech (vgl. TTTech Computertechnik AG) unterstützt werden.
Der in dieser Arbeit entwickelte Simulationsprozess soll überprüft werden, indem er sowohl auf synthetische als auch auf typische Modelle in einer Fallstudie angewandt wird.
Bei der Modellierung orientiert sich diese Arbeit am Modellbildungsprozess nach Jasperneite (2002) und Haas und Zorn (1995): Zunächst wird ein konzeptionelles Modell entwickelt, auf das sich die Implementierung stützt. Die Ausführung dieses implementierten Modells führt zu Modellkenngrößen, welche mit Messergebnissen des realen Systems (Systemkenngrößen) in einer Modellvalidierung verglichen werden können. Die Ergebnisse der Validierung fließen erneut in das Konzept ein und verbessern das Modell (siehe Abbildung 1.4 auf der nächsten Seite).

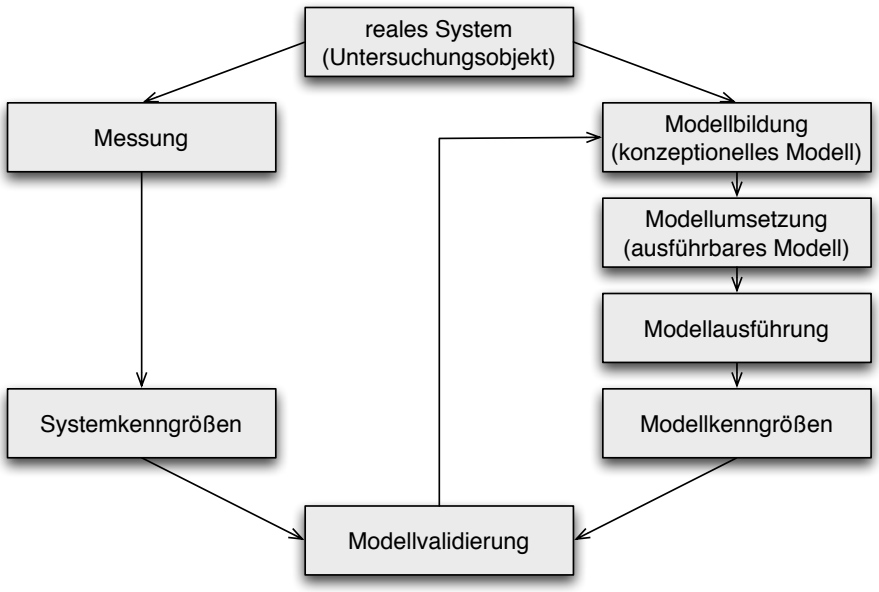

Abbildung 1.4: Arbeitsschritte bei der Modellierung nach Jasperneite (2002) und Haas und Zorn (1995)

1.4 Struktur der Arbeit

Die vorliegende Arbeit gliedert sich wie folgt:
In **Kapitel 2** wird der *Hintergrund* zu dieser Arbeit und die *vergleichbaren und verwandten Arbeiten* vorgestellt. Neben den Merkmalen der Kommunikation in Fahrzeugen werden Echtzeit-Ethernet und insbesondere die time-triggered Ethernet-Technologie erklärt. Darüber hinaus enthält das Kapitel einen Überblick über Simulationstechniken. **Kapitel 3** gibt eine Einführung in die echtzeitrelevanten *Metriken* und ihre Bedeutung. Weiterhin wird die analytische und simulationsbasierte Ermittlung von Metriken erklärt. In **Kapitel 4** wird das *Konzept* der simulationsbasierten Evaluierung und die *Architektur* der Plattform vorgestellt. Neben den Einsatzszenarien und Arbeitsabläufen werden die verwendeten Frameworks und Komponenten gezeigt. **Kapitel 5** zeigt die *Umsetzung* des Simulationsmodells, der Konfiguration, der NED-Generierung und der weiteren Werkzeuge. **Kapitel 6** stellt die *Ergebnisse*, die aus der Anwendung des Evaluierungsprozesses auf synthetische Modelle gewonnen werden, sowie das Ergebnis der Validierung des Simulationsmodells und einer *Fallstudie* vor. Abschließend enthält **Kapitel 7** eine *Zusammenfassung* der Ergebnisse dieser Arbeit und das *Fazit*.

Kapitel 2

Hintergrund und verwandte Arbeiten

Dieses Kapitel gibt einen Überblick über die verwendeten Techniken und Technologien. Nachdem die Merkmale von verteilten Anwendungen im Automobil und ihre Anforderungen sowie heutige Bussysteme vorgestellt werden (Abschnitt 2.1), werden Ethernet und die verschiedenen Technologien zur Erweiterung von Ethernet um Echtzeitfähigkeit erklärt (Abschnitt 2.2). Weiterhin werden TTEthernet (2.3) und die Simulationsverfahren und Werkzeuge (2.4) als Basistechnologien dieser Arbeit vorgestellt. Das Kapitel endet mit einer Übersicht über die Vorarbeiten (2.5) und die mit dieser Arbeit verwandten und vergleichbaren Veröffentlichungen und Projekte (2.6).

2.1 Merkmale eines verteilten Echtzeit-Systems im Automobil

Der folgende Abschnitt stellt zunächst die Eigenschaften eines typischen verteilten Echtzeit-Systems im Automobil vor, um dann auf die Anforderungen, die sich aus diesem Kontext ergeben, einzugehen.
Ein Fahrzeugnetzwerk ist ein wohldefiniertes, geschlossenes System. In der Regel werden Komponenten nicht spontan in das System eingefügt oder entfernt. Damit kann der Determinismus und das Zeitverhalten des Kommunikationssystems gut im voraus berechnet werden. Das System kann statisch konfiguriert werden und muss sich nicht auf sich ändernde Topologien oder variierende Teilnehmerzahlen anpassen, wie es etwa in lokalen (Local Area Network (LAN)) oder ausgebreiteten Computer-Netzwerken (Wide Area Network (WAN)) erforderlich ist.
Echtzeitanwendungen im Automobil haben hohe Anforderungen an das Zeitverhalten der Datenübertragung zwischen den Steuergeräten. So muss beispielsweise bei einem Aufprall die Aufnahme der Sensorwerte, die Datenübertragung und die Aktivierung des Airbags abgeschlossen sein, bevor die Relativbewegung der Insassen beginnt (vgl. Rokosch, 2002).

Die Güte von Regelungssystemen wie beispielsweise der Antriebsschlupfregelung (ASR) hängt ebenfalls vom Zeitverhalten der Datenübertragung, insbesondere der Nachrichtenlaufzeit (Latenz) und der Varianz der Nachrichtenlaufzeit (Jitter) ab, denn sie hat wesentlichen Einfluss auf die Totzeit der verteilten Regelung. Die Totzeit ist ein Begriff aus der Regelungstechnik, der die Zeitspanne zwischen einer Änderung am System-Eingang und der entsprechenden Antwort am System-Ausgang bezeichnet. Je länger die Totzeit ist, desto schlechter wird das Regelverhalten. Dies kann soweit führen, dass der Regler unbrauchbar wird (vgl. Koller u. a., 2001).

2.1.1 Anforderungen an Echtzeit-Kommunikation im Automobil

Die Automobilindustrie definiert einen umfangreichen Anforderungskatalog an Kommunikationssysteme für Echtzeitanwendungen im Fahrzeug (vgl. Belschner u. a., 2000). Die für diese Arbeit wichtigsten Anforderungen sind im folgenden Abschnitt erläutert.

Übertragung von Daten mit unterschiedlichen Zeitanforderungen

Anwendungen im Fahrzeug haben unterschiedliche Anforderungen an Determinismus und Zeitverhalten des Kommunikationssystems. Daten für Fahrerassistenzsysteme und Sicherheitseinrichtungen des Fahrzeugs müssen mit absolut deterministischem Zeitverhalten, also innerhalb definierter Grenzen für Nachrichten-Latenz und Jitter[6], übertragen werden. Weiterhin gibt es Systeme mit „weichen“ Zeitanforderungen. Zu diesen gehören beispielsweise die Telekommunikations- und Unterhaltungselektronik im Fahrzeug. Eine dritte Klasse von Daten hat keine feste Zeitanforderung an die Datenübertragung.

Synchrone und asynchrone Übertragung

Datenübertragung kann synchron, das heißt zyklisch zeitgesteuert, oder asynchron, ereignisgesteuert, organisiert sein. Verteilte Regelungssysteme im Fahrzeug basieren meist auf synchroner Kommunikation. Telekommunikations- und Unterhaltunganwendungen, Konfigurations- und Diagnosedaten werden dagegen in der Regel asynchron und nur bei Bedarf übertragen. Ein Kommunikationssystem muss daher beide Übertragungsklassen unterstützen (vgl. Hillebrand u. a., 2007).

[6] Mit Jitter (deutsch: Fluktuation/Schwankung) wird in der Nachrichtentechnik die Varianz der Laufzeit von Datenpaketen bezeichnet

Geringe Latenz und geringer Nachrichtenjitter

Die Regelgüte einer verteilten Regelung hängt maßgeblich von der Latenz und der Höhe des Nachrichtenjitters ab (vgl. Koller u. a., 2001). Die Latenz ist dabei die Dauer der Nachrichtenübertragung. Mit Jitter wird die Varianz der Nachrichtenlaufzeit bezeichnet.

Hohe Übertragungsbandbreite

Steuerelektronik im Antriebs- und Fahrwerksbereich von Automobilen hat für heutige Kommunikationssysteme bereits eine sehr hohe Bandbreitenanforderung. Mit dem Einsatz von kamerabasierten Fahrerassistenzsystemen nimmt der Bandbreitenbedarf weiter zu. Diese Daten müssen stets mit deterministischem Zeitverhalten ausgeliefert werden.
Multimediaanwendungen aus dem Entertainmentbereich wie beispielsweise hochauflösende Videodaten haben einen ebenfalls hohen Bandbreitenbedarf. Hier sind jedoch die Anforderungen an das Zeitverhalten geringer.

Redundanzkonzept

Im Gegesatz zur Flugzeugindustrie, in der die Fehlertoleranz durch massiven Einsatz von Redundanz[7] erreicht wird, spielen redundante Kommunikationswege im Fahrzeug derzeit noch keine große Rolle (vgl. Benz, 2004). Dies liegt insbesondere daran, dass im Fahrzeug derzeit die kritischen Anwendungen über mechanische Rückfallkomponenten verfügen.
Um fehlertolerant sicherheitsrelevante Systeme — wie beispielsweise X-by-Wire-Anwendungen — zu vernetzen, muss die Kommunikations-Technologe ein einfaches Redundanzkonzept anbieten. Dabei dürfen aus Kosten- und Gewichtsgründen nicht grundsätzlich alle Verbindungen redundant ausgelegt sein, sondern nur solche, über die fehlertolerant kommuniziert werden muss (Paret, 2007).

Einfacher Austausch des Physical-Layers

Wegen ihrer leichten Konfektionierung und der unproblematischen Verlegung im Fahrzeug sind derzeit Kabel mit metallischen Leitern im Fahrzeug dominant. Trotz ihrer anspruchsvollen Verlegung haben Lichtwellenleiter (LWL) Vorteile wie geringes Gewicht, beständige Kontakte an Steckern, keine elektromagnetische Ausstrahlung und keine Anfälligkeit für Störungen (vgl. Malek, 2001).
Je nach Anwendung und Einsatzgebiet sind unterschiedliche physikalische Medien optimal für die Datenübertragung. Ein fortschrittliches Kommunikationssystem abstrahiert daher vom

[7] Redundanz (lat. redundare – im Überfluss vorhanden sein) bezeichnet in der Technik das Vorhandensein zusätzlicher, funktional gleicher oder vergleichbarer Ressourcen, wenn diese bei einem störungsfreien Betrieb nicht benötigt werden

physikalischen Medium und lässt damit den Austausch und die Kombination verschiedener Technologien zu.

Einfache Anwendung

Kürzere Produkteinführungszeiten fordern eine schnelle Entwicklung. Daraus ergibt sich die Anforderung nach einfachen und flexiblen Schnittstellen. Insbesondere die Nutzung von Techniken, welche sich auch in anderen Bereichen wie beispielsweise der IT oder der Prozessautomatisierung durchgesetzt haben, ermöglichen die Nutzung von Standardwerkzeugen und den Einsatz von branchenfremden Fachkräften (vgl. Bruckmeier, 2010).

Nutzung von Standardkomponenten

Standardkomponenten, sogenannte Components-of-the-shelf (COTS), helfen Kosten in Entwicklung und Produktion zu sparen. Trotz der enormen Stückzahlen für Steuergeräte im Automotive-Bereich (vgl. Bruckmeier, 2010, S. 6-7) ist hier ein beträchtliches Einsparungspotential gegeben.

Darüber hinaus gibt es eine ganze Reihe von allgemeinen Anforderungen an Elektronik im Fahrzeug wie: Kosteneffizienz, schnelles Start-Up, lange Wartbarkeit, gute Skalierbarkeit für weitere Extras, Eignung für kritische Umweltbedingungen (Temperatur, Vibrationen, Feuchtigkeit, elektromagnetische Strahlung), geringes Gewicht, geringer Platzbedarf und niedriger Energieverbrauch.
In dieser Kombination sind die Anforderungen einzigartig in der Industrie. So hat die Flugzeugindustrie beispielweise ähnliche Sicherheits-Anforderungen, die Kosten von Komponenten sind hier jedoch weniger bedeutend als im Automobil-Bereich. Im Consumerelektronik-Bereich ist der Kostendruck ähnlich, aber die Sicherheitsanforderungen und Umweltbedingungen sind deutlich geringer. Zudem ist beispielsweise bei Computern die geforderte Dauer der Wartbarkeit weit kürzer als bei einem Automobil (vgl. Bruckmeier, 2010, S. 3).

2.1.2 Heutige Automotive-Bussysteme

Zur Zeit ist das Datenübertragungsnetz in Serienfahrzeugen heterogen aufgebaut. Je nach Anwendung und zu übertragenden Daten kommen spezielle Bussysteme zum Einsatz (vgl. Dohmke, 2002). Einige wichtige Kommunikations-Technologien des Automotive-Bereichs werden im folgenden Abschnitt vorgestellt.

CAN-Bus

Der *Controller Area Network-Bus (CAN)* (vgl. International Organization for Standardization, 2003) ist das heute am meisten verbreitete Kommunikationssystem im Automobil. Er wurde bereits 1983 von Bosch entwickelt und ist nach ISO 11898 standardisiert. Der CAN-Bus ist ein asynchroner Bus zwischen mehreren verteilten Knoten. Seine time-triggered (zeitgesteuerte) Erweiterung *Time-triggered Controller Area Network (TTCAN)* (vgl. International Organization for Standardization, 2004) konnte sich bisher kommerziell nicht nennenswert durchsetzen. Der Medienzugriff des CAN-Bus ist nicht-präemptiv prioritätengesteuert nach dem Collision-Detection-Multiple-Access-Verfahren (CDMA). Die Priorisierung ist dabei über eine Arbitrierungsphase[8] realisiert. Über die Konfiguration kann echtzeitfähige Kommunikation über den CAN-Bus erreicht und bewiesen werden; aufgrund des unsynchronisierten medienzugriffs, variiert die Laufzeit einer Botschaft jedoch stets.

LIN-Bus

Der *Local Interconnect Network-Bus (LIN)* (vgl. LIN-Administration) ist das „low-cost“ Übertragungsmedium im Automobil. Er ist ein für die kostengünstige Verbindung von Steuergeräten und abgesetzten Sensoren und Aktoren entwickelter Feldbus. Der Medienzugriff erfolgt beim LIN-Bus nach dem Master-Slave-Verfahren. Der Master fordert einen Slave durch Anlegen der Slave-Adresse explizit zum Senden auf.

MOST-Bus

Mit der steigenden Verbreitung von Multimedia-Anwendungen und dem damit verbundenen hohen Bandbreitenbedarf wurde der *Media Oriented System Transport-Bus (MOST)* (vgl. MOST Cooperation) eingeführt. Er arbeitet seriell über elektrische- oder Lichtwellenleiter. Die Teilnehmer des MOST-Bus sind in einem Ring verbunden und kommunizieren zyklisch. Die Synchronisierung wird von einem sogenannten Timing-Master übernommen, welcher kontinuierlich Frames in den Ring sendet. An diesen Botschaften können sich die Teilnehmer synchronisieren. Seit MOST150, welches im Oktober 2007 eingeführt wurde, bietet MOST auch einen gesonderten Kanal, über den Ethernet-Frames getunnelt werden können.

FlexRay

FlexRay (vgl. FlexRay Consortium, 2005) ist ein neues zeitgesteuertes Bussystem für Steuergeräte im Automobil. FlexRay zielt insbesondere auf die Verwendung in komplexen verteilten Regelungen wie Fahrerassistenzsystemen ab. Insbesondere Anwendungen wie die

[8] Die Arbitrierung oder Arbitration ist ein Zugangsverfahren, welches gleichberechtigten Teilnehmern durch gegenseitige Vereinbarung das Zugangsrecht zu einer geteilten Ressource erteilt

System	Typisches Anwendungsfeld
Controller Area Network (CAN)	Vernetzung von Steuergeräten, event-triggered
Local Interconnect Network (LIN)	Vernetzung von Aktoren und Sensoren
Media Oriented System Transport (MOST)	Vernetzung von Audio- und Video-, Sprach- und Datenanwendungen
FlexRay	Vernetzung von Steuergeräten (höhere Bandbreite), event- und time-triggered

Tabelle 2.1: Überblick — Bussysteme im Fahrzeug und ihre typische Verwendung

Ansteuerung von adaptiven Fahrwerken haben hohen Bedarf an synchroner zeitgesteuerter und zyklischer Kommunikation. Durch die geringe Verbreitung sind FlexRay-Systeme momentan teuer und werden nur in Oberklasse-Fahrzeugen wie dem BMW X5 (vgl. Schedl, 2007) oder dem Audi A8 (vgl. AUDI AG, TTTech Automotive GmbH, 2009) eingesetzt.

Tabelle 2.1 zeigt wichtige, heute eingesetzte Bussysteme und ihre Verwendung.

2.2 Die Echtzeit-Ethernet-Technologie

Derzeit ist nicht bekannt, welche Hersteller und Zulieferer an Lösungen für Ethernet im Automobil arbeiten. Nur von BMW existieren umfangreichere Veröffentlichungen (siehe auch Abschnitt 2.6). Dort wurde ein Versuchsfahrzeug mit Ethernet ausgerüstet. Die verwendete Technologie wurde nicht veröffentlicht. Zu den Ergebnissen der Studie sagt der Projektmanager Richard Bogenberger: „Our experiments with prototypes demonstrated, that the real-time behavior far exceeded the requirements — even when we ran multimedia applications across the same network“ (vgl. Hammerschmidt, 2007). Dabei wird deutlich, dass die klassische Trennung von Übertragungsleitungen sicherheitskritischer und unkritischer Anwendungen reduziert werden kann und soll.
Der folgende Abschnitt stellt die Echtzeit-Ethernet-Technologie vor. Nach einem Überblick über Standard-Ethernet werden die verschiedenen Konzepte für Echtzeitübertragung über Ethernet erklärt.

2.2.1 Standard-Ethernet als Basis

Ethernet ist eine bewährte Technologie, die mit dem Internet und der damit verbundenen Vernetzung von Rechnern ihren Durchbruch erlebte. Ethernet bietet eine hohe Übertragungs-Bandbreite. Während sich Systeme mit bis zu 10 Gbit/s bereits am Markt etabliert haben

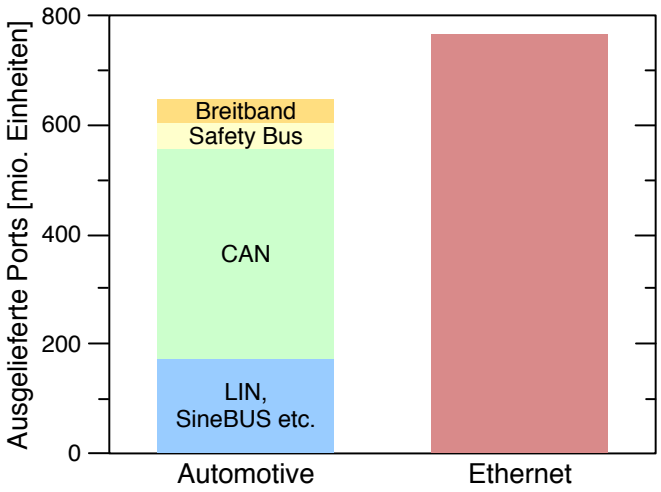

Abbildung 2.1: Ausgelieferte Einheiten in 2010 — Automotive-Systeme und Ethernet (Bruckmeier, 2010)

(vgl. Tanenbaum und Wetherall, 2011), werden in Laborbedingungen bereits 100 Gbit/s, auch über weite Strecken, erreicht (vgl. D'Ambrosia, 2010). Durch die gestiegene Verbreitung sind die Preise für Ethernet-Komponenten stark gesunken (vgl. Greifeneder und Frey, 2007). Heute haben sehr viele Microcontroller bereits Hardware für die Anbindung an Ethernet integriert und es werden heute mehr Ethernet-Einheiten ausgeliefert als Einheiten der klassischen Automotive-Technologien (siehe Abbildung 2.1).
Ein großer Vorteil bei der Verwendung von Ethernet ist die Entkopplung der Kommunikation vom Übertragungsmedium. Der Ethernet-Standard definiert neben der Nutzung von Kupferkabel in verschiedenen Konfigurationen beispielsweise auch die Übertragung über Lichtwellenleiter (LWL) (vgl. Institute of Electrical and Electronics Engineers, 2005). Die Wahl des Übertragungsmediums kann somit gut an die Anforderungen der Übertragungsstrecke — wie beispielsweise besonders hohe Bandbreite oder besonders hohe Sicherheit gegenüber elektromagnetischen Einstrahlungen — angepasst werden. Im nicht zeitkritischen Bereich stehen bereits eine große Anzahl anwendungsspezifischer Ethernet-basierter Protokolle zur Verfügung. Der Einsatz solcher Standardprotokolle, beispielsweise im Diagnose- oder Unterhaltungsbereich eines Fahrzeugs, kann dabei helfen, Kosten in der Entwicklung einzusparen und vereinfacht die Integration von herstellerfremden Standardkomponenten, sogenannten Components-of-the-shelf (COTS).
Durch die weite Verbreitung von Ethernet gibt es bereits eine große Palette an Entwicklungs- und Diagnosewerkzeugen. Die Zahl an fachkundigen Entwicklern ist beim Ethernet-Protokoll weit größer als bei speziellen Automotive-Lösungen wie LIN oder MOST.

Ursprünglich ist Ethernet nicht echtzeitfähig, da weder Teilnehmer noch Vermittlungsknoten einer deterministischen Nachrichtenvermittlung unterliegen. In Switched-Ethernet kann es dabei zu Datenstaus an den Switchen (Knotenpunkten) kommen, welche die Übertragung der Nachrichten verzögern.

2.2.2 Echtzeit-Ethernet-Übersicht

Im folgenden Abschnitt werden Technologien und Produkte vorgestellt, die es ermöglichen, Echtzeitfähigkeit auf Ethernet-Netzwerken herzustellen.
Grundsätzlich lassen sich die am Markt befindlichen Technologien und Systeme nach verschiedenen Kriterien wie beispielsweise Nachrichtenlatenz, Jitter, Nutzung von Standardhardware, Offenheit der Technologie oder Kosten, klassifizieren. Im folgenden Abschnitt sind die Technologien in die drei Hauptfelder *time-triggered* Systeme, *token-basierte* Systeme und *bandbreitenbasierte* Systeme eingeordnet, da sie jeweils die wichtigsten Eigenschaften vereinen. Die Tabelle A.1 (Anhang auf Seite 100) bietet eine Aufschlüsselung weiterer Kriterien.

Time-triggered Systeme

Time-triggered (zeitgesteuerte) Protokolle sind ein verbreiteter Ansatz, um Echtzeitfähigkeit auf Kommunikationssystemen herzustellen. Dabei synchronisieren sie sich über eine globale Zeit. Jeder Teilnehmer hat eine lokale Uhr, welche möglichst genau mit den Uhren der anderen Teilnehmer synchronisiert werden muss. Zugewiesene Zeitschlitze definieren für jeden Teilnehmer, wann er das Medium zum Senden nutzen darf. Damit ist sichergestellt, dass immer nur ein Teilnehmer zur Zeit auf das Medium zugreift. Diese Technik wird auch als *koordiniertes Time Division Multiple Access (TDMA)* bezeichnet.
Es gibt mehrere Verfahren, um eine globale Zeit herzustellen. Bei den Zeit-Master-Slave-Verfahren gibt es einen sogenannten *Timemaster*, der den Kommunikationsablauf steuert, indem er regelmäßig Synchronisierungsnachrichten verschickt. Bei der Konfiguration von time-triggered Systemen ist zu beachten, dass die Teilnehmer nie über ihren Zeitschlitz hinaus senden. Je genauer die Teilnehmer synchronisiert werden können, desto enger können die Zeitschlitze aneinander gehängt werden.
Diverse Echtzeit-Ethernet-Produkte basieren auf der time-triggered Technologie:

Profinet (vgl. PROFIBUS & PROFINET International) ist ein Echtzeit-Ethernet-Standard, der von der PROFIBUS Nutzerorganisation e.V. als industrieller Fachverband organisiert wird. Es definiert drei Klassen von Daten: *Non Real Time (NRT)*, *Real Time (RT)* und *Isochronous Real Time (IRT)*. Im Bereich NRT kommen Standard IP-Protokolle zum Einsatz. Im Bereich RT werden die Daten im Ethernet-Protokoll transportiert. Die

Zykluszeiten liegen hier größer 10 ms. Für harte Echtzeitanforderungen definiert Profinet mit Isochronous Real Time ein Verfahren, das Zykluszeiten kleiner 1ms und Jitter kleiner 1μs erreichen kann und auf spezielle Ethernet-Controller angewiesen ist (vgl. Felser, 2005).

SynqNet ist ein von der Danaher Motion Corporation entwickeltes und 2001 vorgestelltes Kommunikationssystem für Antriebssteuerungen im Automatisierungsbereich. SynqNet bietet keine Sterntopologie an; damit sind keine Switche erforderlich. Jeder Teilnehmer hat zwei Interfaces, wodurch sowohl die Ringtopologie als auch die Linientopotogie ermöglicht wird. Leitungslängen und Anzahl an Knoten auf dem Leitungsweg werden während der Initialisierung ermittelt, was zu einer genauen Vorhersage der Laufzeiten führt (vgl. SynqNet Interest Group).

RTnet entstand an der Universität Hannover in der Real-Time Systems Group (vgl. Real-Time Systems Group). Die Referenzimplementierung basiert auf Linux und der Echtzeiterweiterung *Real Time Application Interface (RTAI)* (vgl. RTAI Team, 2010). Da RTnet ohne spezielle Switche arbeitet, ist es erforderlich, dass das Netz abgeschlossen ist und alle Teilnehmer ausschließlich das RTnet Protokoll sprechen. RTnet arbeitet nach dem koordinierten TDMA-Verfahren, wobei nicht-zeitkritische Daten ungenutzte Zeitslots verwenden (vgl. Kiszka und Wagner, 2005). Die Synchronisation der Teilnehmer erfolgt durch einen Master, der periodisch den Beginn eines neuen Zyklusses durch eine Startnachricht (Start of Frame) signalisiert. RTnet ist als freies Softwareprojekt entstanden und kann ohne Lizenzkosten weiterentwickelt und genutzt werden (vgl. Kiszka und Schwebel, 2004).

POWERLINK wurde ursprünglich von der Firma B&R als proprietäre Technologie entwickelt und 2001 vorgestellt. 2002 entschied sich B&R, die Technik offenzulegen, und gründete die Ethernet POWERLINK Standardization Group (EPSG), die seitdem die Entwicklung vorantreibt. POWERLINK ist ein Timemaster-basiertes Verfahren. Ein Zyklus wird durch ein Start of Cycle-Paket (SoC) gestartet. An dieser Nachricht synchronisieren sich die Teilnehmer und senden in dem ihnen zugewiesenen Zeitschlitz ihre Nachrichten. Das Ende dieser synchronen Phase wird durch ein Start of Asynchron-Paket (SoA) markiert. Bis zum nächsten SoC-Paket können nun zeitunkritische und asynchron auftretende Daten versendet werden (vgl. Ethernet POWERLINK Standardization Group).

TTEthernet entstand aus einem Projekt der Real-Time Systems Group (vgl. Real Time Systems Group (RTS)) der TU Wien aus dem Jahre 2004 und wird heute von der österreichischen Firma TTTech Computertechnik AG (vgl. TTTech Computertechnik AG) in modifizierter Form weiterentwickelt und kommerziell angeboten. TTEthernet steht

für time-triggered Ethernet. Wie Profinet definiert auch TTEthernet drei Nachrichten-Klassen : *time-triggered* Messages werden in vordefinierten Zyklen gesendet und haben Vorrang vor allen anderen Nachrichten. *Rate-constrained* Messages sind Nachrichten mit weniger strengen Echtzeitanforderungen. Für sie ist nur sichergestellt, dass ausreichend Bandbreite für die Übertragung zur Verfügung steht. *Best-effort* Messages sind Standard-Ethernet-Nachrichten. Für best-effort[9] Messages werden keine Annahmen zur Laufzeit gemacht. Sie nutzen die übrige Bandbreite, nachdem time-triggered und rate-constrained Messages übertragen wurden. TTEthernet ist ein offenes Protokoll. Zum Einsatz kommen spezielle Field Programmable Gate Array-basierte (FPGA) Switche. Die Controller der Teilnehmer können in Software auf Standardhardware implementiert werden (vgl. Grillinger u. a., 2006).

Token-basierte Systeme

Die Grundlage von Token-basierten Systemen ist der wechselseitige Ausschluss mit Hilfe einer zwischen allen Teilnehmern geteilten Ressource, dem *Token*. Das Token wandert von Teilnehmer zu Teilnehmer, wobei nur derjenige senden darf, der das Token hält. Damit ist sichergestellt, dass das Medium stets frei ist, wenn Daten gesendet werden. Das System bleibt über das Token synchronisiert. Ein Beispiel für eine token-basierte Technologie ist das Token Ring Protokoll (IEEE 802.5) (vgl. Institute of Electrical and Electronics Engineers, 1997) aus dem Bereich der Computernetze. Hier ist für die Topologie ein geschlossener Ring erforderlich. Der IEEE-Standard 802.4 definiert solche Token-basierten Systeme auch für die Sterntopologie. Eine Ringstruktur wird dabei nicht über das physikalische Medium hergestellt, sondern auf logischer Ebene. Das Token wird jeweils zum Teilnehmer mit der nächst niedrigeren Adresse weitergegeben.
Token-basierte Systeme können mit Standard-Ethernet-Switchen implementiert werden, da sich die gesamte Protokoll-Logik auf die Teilnehmer verschieben lässt. Dabei ist stets sicherzustellen, dass nur ein Token im Umlauf ist. Das Komplexe an token-basierten Systemen ist die Erkennung und Neuerzeugung eines verlorenen Tokens, was bei Paketverlust oder Ausfall eines Teilnehmers eintreten kann. Die Dauer dieses Vorgangs muss dabei bei der Berechnung des Jitters in jedem Fall berücksichtigt werden und hat direkten Einfluss auf die worst-case Übertragungszeit.
Bei Token-basierten Systemen kann zwar die Dauer der eigentlichen Datenübertragung sehr gut berechnet werden, jedoch hängt der Zeitpunkt der Übertragung wesentlich von der Last ab. Mit variabler Teilnehmeranzahl verändert sich auch die Zykluslänge. Somit kann zwar eine definierte Aussage über Laufzeit und Jitter getroffen werden, dieser hängt jedoch von der Anzahl der aktiven (sendenden) Teilnehmer ab. Neben dem klassischen Tokensystem mit

[9] Best-effort (deutsch: größte Bemühung) bezeichnet die geringste Dienstgütezusicherung in Netzwerken. Übermittlungsanfragen werden im Rahmen der zur Verfügung stehenden Ressourcen bedient

einem Token für das gesamte Netz können bei umfangreicheren Topologien auch mehrere Token eingesetzt werden. Dabei dürfen Nachrichten von Sendern mit verschiedenen Token jedoch nicht über dieselbe Leitung gesendet werden.
In der Prozessautomatisierung wird das Token-basierte *EtherCAT-Protokoll* eingesetzt:

EtherCAT wird von der Ethercat Technology Group (vgl. EtherCAT Technology Group), einer Interessengemeinschaft aus Anwendern und Herstellern, welche von der Firma Beckhoff gegründet wurde, entwickelt. EtherCAT wurde 2003 veröffentlicht und ist ein klassisches Token-basiertes System. Ein Master sendet einen Standard-Ethernet-Frame, der in einem logischen Ring von Teilnehmer zu Teilnehmer weitergeleitet wird. Die Slaves empfangen dieses Telegramm und entnehmen für sie bestimmte Daten, bzw. fügen Daten ein, während sie die Nachricht weiterleiten. EtherCAT unterstützt verschiedene physikalische Medien und Topologien. Die Wahl der Topologie hat dabei Auswirkung auf Nachrichtenlatenz, Jitter und Redundanz, sowie auf die Möglichkeit, im laufenden Betrieb Teilnehmer hinzuzufügen oder zu entfernen. Der EtherCAT-Master kann komplett in Software und auf Standardhardware implementiert werden. Da die Ethercat-Slaves die Nachrichten während des Durchlaufs bearbeiten, können hier keine standard Ethernetcontroller eingesetzt werden. Hier kommen spezielle Application Specific Integrated Circuit- (ASIC) oder FPGA-Lösungen zum Einsatz.

Bandbreitenbasierte Systeme

Bandbreitenbasierte Systeme arbeiten ohne eine Form der Synchronisierung. Es werden virtuelle Bandbreitenkontingente konfiguriert. Jeder Teilnehmer sendet mit einem solchen Kontingent. In diesen Systemen überwachen die Switche, dass die Teilnehmer ihre Kontingente nicht überschreiten.
Da keine Synchronisierung stattfindet, ist in Systemen mit Bandbreitenkontingenten die berechnete worst-case Verzögerungszeit und damit auch der Jitter weit höher als in der Realität zu beobachten. Dies hängt damit zusammen, dass im worst-case alle Teilnehmer zur gleichen Zeit ihre Nachrichten versenden.
Um die zur Verfügung stehende Bandbreite zu verteilen, gibt es verschiedene Verfahren. Beim sogenannten Token-Bucket-Verfahren werden beispielsweise Tickets an die Busteilnehmer verteilt. Jeder Teilnehmer verbraucht Tickets, wenn er sendet. Die Tickets können auch bis zu einer definierten Menge angespart werden. Dadurch, dass stets nur eine definierte Anzahl an Tickets im Umlauf ist, ist sichergestellt, dass in endlicher Zeit alle Nachrichten zugestellt werden.

Avionics Full DupleX Switched Ethernet (AFDX) (vgl. Aeronautical Radio Incorporated, 2009) ist ein Protokoll, das für den Einsatz im Flugzeug entwickelt wurde. AFDX definiert maximale Transferraten für jeden Busteilnehmer und ist damit ein klassisches

bandbreitenbasiertes Verfahren. Jeder Teilnehmer sendet in einem vorkonfigurierten Intervall. Dabei überwachen spezielle Switche, dass die Teilnehmer ihr Sendeintervall nicht unterschreiten.

2.3 Time-triggered Ethernet

In diesem Abschnitt wird TTEthernet, sowie seine Nachrichtenarten, Komponenten und Werkzeuge vorgestellt.

2.3.1 Das TTEthernet-Protokoll

Ursprünglich entstand TTEthernet in der Real-Time Systems Group (vgl. Real Time Systems Group (RTS)) der TU Wien in 2004. Die universitätsnahe österreichische Firma TTTech Computertechnik AG (vgl. TTTech Computertechnik AG) übernahm die Konzepte, entwickelt TTEthernet weiter und bietet es kommerziell an. Die TTEthernet-Spezifikation (vgl. Steiner, 2008) wird momentan von der Society of Automotive Engineers (SAE) in der Arbeitsgruppe AS-2D standardisiert (vgl. SAE - AS-2D Time Triggered Systems and Architecture Committee, 2009).
TTEthernet ist eine time-triggered Echtzeit-Ethernet-Erweiterung, welche auf die speziellen Anforderungen der Automotive-Kommunikation zugeschnitten ist. Es erlaubt best-effort Ethernet-Datenverkehr zusammen mit harter Echtzeit-Kommunikation auf derselben Layer 2 Infrastruktur.
TTEthernet basiert auf Switched-Ethernet. Die Topologie setzt sich aus Switchen zusammen, welche die Nachrichten weiterleiten. Redundanz kann durch redundante Verbindungen zwischen den Komponenten erreicht werden.
Die Datenübertragung in TTEthernet ist synchron und zyklisch aufgebaut. Für die Echtzeit-Kommunikation werden jedem Teilnehmer offline konfigurierte Zeitslots zugewiesen. Dieses Netzwerk-Zugriffsverfahren wird als koordiniertes TDMA bezeichnet und erlaubt vorhersagbare Übertragungszeiten in definierten Grenzen ohne Warteschlangenbildung in den Switchen und damit Kommunikation mit niedriger Latenz und niedrigem Jitter. Damit jeder Teilnehmer zur richtigen Zeit seinen Slot in Anspruch nehmen kann, haben alle Teilnehmer eine lokale Uhr und einen Zeitplan (Schedule) für die Übertragung. Da eine globale Zeit über alle Teilnehmer erforderlich ist, definiert die TTEthernet-Spezifikation ein fehlergesichertes Synchronisierungsprotokoll.
Neben den *time-triggered* Nachrichten definiert TTEthernet noch zwei weitere Nachrichtenarten. *Rate-constrained* Nachrichten sind für die Übertragung von Daten mit weniger harten Echtzeitanforderungen gedacht und basieren auf dem AFDX-Protokoll (vgl. Aeronautical Radio Incorporated, 2009). *Best-effort* Nachrichten basieren auf Standard-Ethernet und haben

in der Übertragung die niedrigste Priorität. Sie werden verwendet, um Daten im Hintergrund zu übertragen. Eine Besonderheit von TTEthernet-Netzwerken ist die Fähigkeit, Teilnehmer zu integrieren, die das time-triggered Protokoll nicht unterstützen. Diese Teilnehmer bleiben unsynchronisiert und kommunizieren ausschließlich über best-effort Nachrichten.

Synchronisierung

Das TTEthernet-Synchronisierungsprotokoll definiert drei Rollen:

- *Synchronisation Master* starten die Synchronisierung
- *Compression Master* berechnen eine neue globale Zeit für alle Teilnehmer
- *Synchronisation Clients* empfangen die globale Zeit

Das Synchronisierungsprotokoll in TTEthernet basiert auf sogenannten *Protocol Control Frames (PCFs)*. Jeder Synchronisation Master sendet einen PCF an einem bestimmten offline festgelegten Zeitpunkt im Schedule. Die Compression Master sammeln diese PCF und berechnen eine globale Zeit aus allen Nachrichten. Der so berechnete Mittelwert wird vom Compression Master in einem neuen PCF an alle Teilnehmer versandt, welche ihre lokale Zeit an die neu berechnete globale Zeit angleichen.
Jeder Teilnehmer im TTEthernet-Netzwerk kann mehrere Rollen aus dem Synchronisierungsprotokoll übernehmen. So müssen beispielsweise alle Synchronisation Master gleichzeitig auch Synchronisation Clients sein, um die neue globale Zeit zu empfangen. Grundsätzlich kann jede Funktion auf jedem Gerät im Netzwerk implementiert werden. Abbildung 2.2 auf der nächsten Seite zeigt ein Beispiel mit zwei Synchronisation Mastern und zwei Synchronisation Clients auf Endsystemen und einem Switch als Compression Master.

Nachrichtenarten in TTEthernet

TTEthernet definiert drei Nachrichtenarten: Time-triggered, rate-constrained und best-effort, die im folgenden beschrieben werden.

Time-triggered Nachrichten: Ein globaler offline konfigurierter Zeitplan (Schedule) definiert die Übertragung und Weiterleitung von Time-triggered (TT) Nachrichten. Dieser Zeitplan ist für alle TTEthernet-Geräte, also Hosts und Switche, gültig. Der Zeitplan enthält die Zeiten der Ereignisse, die das Senden und Empfangen von time-triggered Nachrichten auslösen. Dabei ist das Routing der Nachrichten statisch, was eine vollständig deterministische — offline geplante — Nachrichtenübertragung erlaubt.

Time-triggered Nachrichten benutzen, ähnlich wie Ethernet-Multicast, ein inhaltsorientiertes Adressformat. Während sonst der Empfänger einer Nachricht aus der Zieladresse ermittelt werden kann, beschreibt bei time-triggered Nachrichten die Adresse

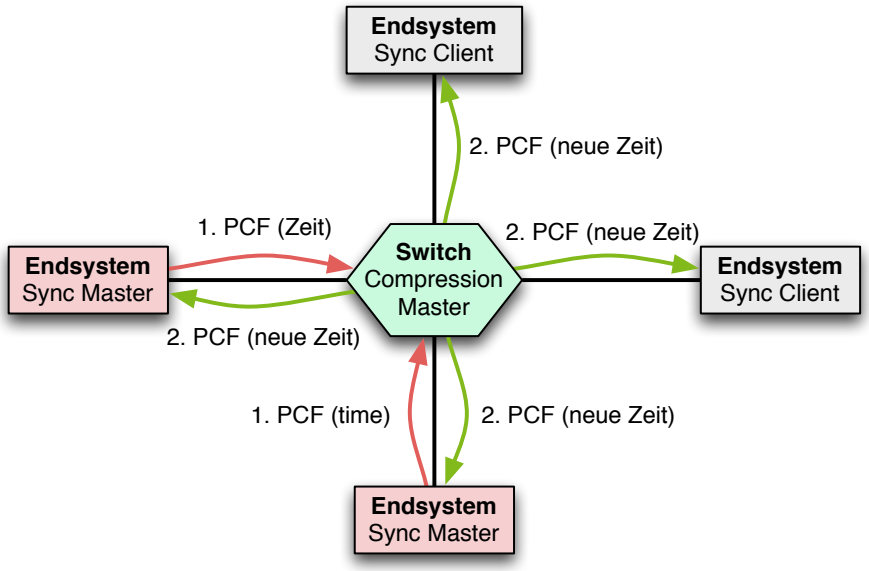

Abbildung 2.2: Zweistufiges Synchronisationsprotokoll in TTEthernet — Beispielkonfiguration mit zwei Sync Mastern, zwei Sync Clients und einem Compression Master im Switch

den Inhalt der Nachricht. Um zuverlässig und performant time-triggered Nachrichten zu erkennen, wird die 48 Bit Zieladresse aus dem Ethernet-Header in zwei Teile geteilt (siehe Abbildung 2.3 auf der nächsten Seite). Der erste Teil enthält den sogenannten *Critical Traffic Marker (CT-Marker)*. Durch Matching kann anhand des CT-Markers eine zeitkritische Nachricht erkannt werden. Der zweite Teil ist die *Critical Traffic ID (CT-ID)*. Die CT-ID ist eindeutig für jede Nachricht im Zyklus und wird verwendet, um Nachrichten in den TTEthernet-Schedules zu adressieren. Die Weiterleitungs-Entscheidungen werden auf Basis der CT-ID getroffen.

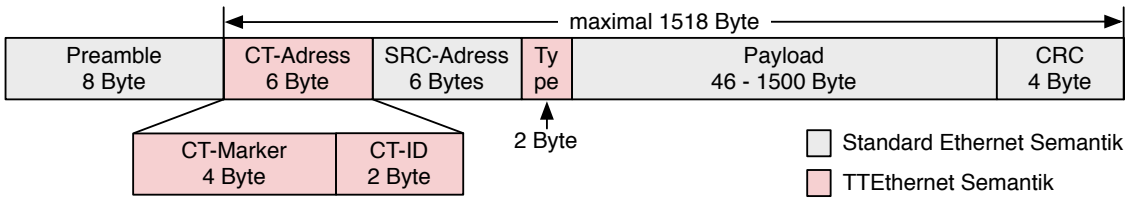

Abbildung 2.3: Aufbau des TT-Ethernet-Frames (vgl. Bartols, 2010)

Rate-constrained Nachrichten: Rate-constrained (RC) Datenverkehr basiert auf dem *ARINC 664 Standard* (vgl. Aeronautical Radio Incorporated, 2009), der auch unter der Bezeichnung *Avionics Full DupleX Switched Ethernet (AFDX)* bekannt ist. Für rate-constrained Nachrichten wird sichergestellt, dass ausreichend Bandbreite zur Übertragung verfügbar ist. Um dies zu erreichen, werden sogenannte *Bandwidth Allocation Gap-Accounts (BAG)* eingerichtet. BAG-Accounts legen das minimale Intervall von Nachrichten mit derselben CT-ID und ihre maximale Länge fest. Die sendende Applikation muss die Beschränkungen der BAG-Accounts einhalten. Andernfalls werden die Nachrichten als ungültig betrachtet und im Switch verworfen.

Da die Weiterleitung von rate-constrained Nachrichten nicht synchronisiert und von niedrigerer Priorität als die Weiterleitung von time-triggered Daten ist, kann die exakte Sende- oder Weiterleitungszeit nicht vorhergesagt werden. Damit haben rate-constrained Nachrichten eine unvorhersagbare Latenz und einen höheren Jitter als time-triggered Nachrichten. Dennoch kann für rate-constrained Nachrichten eine auf Basis der Netzwerk-Konfiguration beweisbare obere Grenze für die Verzögerung berechnet werden (vgl. Charara u. a., 2006). Damit ist die maximale Latenz und der maximale Jitter berechenbar, womit sich rate-constrained Nachrichten insbesondere für die Übertragung von asynchronen Echtzeitdaten eignen.

Best-effort Nachrichten: Der best-effort Datenverkehr in TTEthernet basiert auf Standard-Ethernet-Nachrichten. Er hat die niedrigste Priorität von allen drei Nachrichtenklassen und wird ausschließlich in Leerlaufphasen im Zyklus übertragen. Best-effort Nachrichten können Unicast- und Multicast-Adressen verwenden. Die TTEthernet-Switche unterstützen sowohl die fest konfigurierte Weiterleitung als auch das automatische Lernen von MAC-Address-Tabellen, was dem üblichen Verfahren im Standard-Ethernet entspricht. Die Übertragung von best-effort Nachrichten bietet keinerlei Garantie für die Laufzeit (Latenz) oder sogar die generelle Auslieferung von Nachrichten. Best-effort Nachrichten dürfen jederzeit von jedem Teilnehmer im Netz verworfen werden, sofern sie die zuverlässige Auslieferung von zeitkritischem Datenverkehr gefährden.

Abbildung 2.4 auf der nächsten Seite visualisiert eine Beispiel-Anwendung aus dem Automotive-Umfeld, in der die drei Nachrichtenarten auf demselben physikalischen Link koexistieren. Während time-triggered Nachrichten — beispielsweise für die Übertragung von Fahrwerksinformationen — unter strikter Einhaltung des Übertragungszyklus und mit der höchsten Priorität weitergeleitet werden, werden rate-constrained Pakete in freien Zeitschlitzen übertragen. Die danach noch freie Bandbreite wird für die Übertragung von best-effort Datenverkehr verwendet.

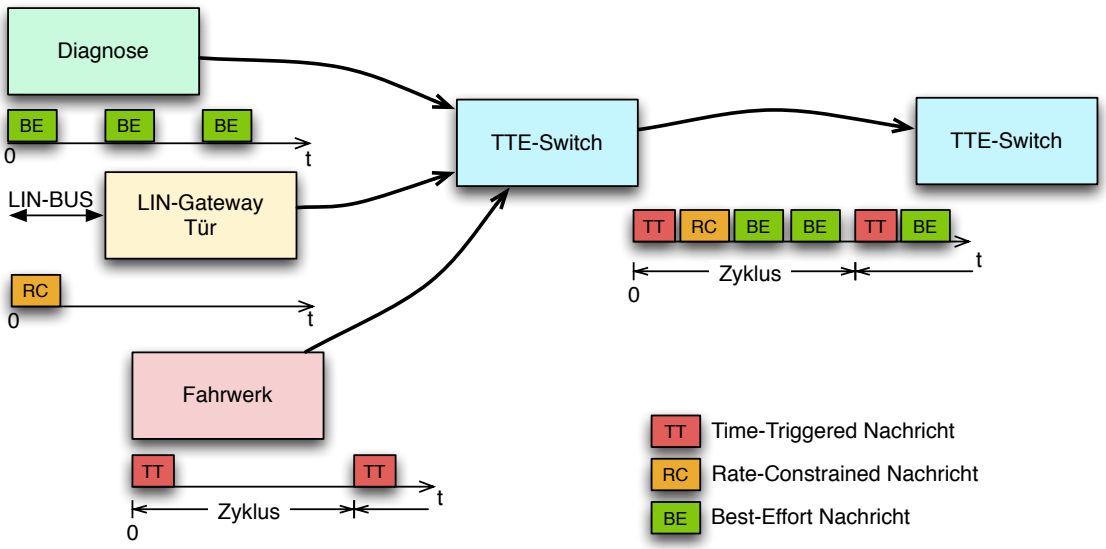

Abbildung 2.4: TTEthernet-Beispielanwendung — Fahrzeugkommunikation mit drei unterschiedlichen Trafficklassen

2.3.2 Komponenten und Werkzeuge

Im folgenden werden die von TTTech angebotenen Komponenten und Werkzeuge vorgestellt.

TTEthernet-Komponenten

TTTech bietet derzeit TTEthernet-Komponenten zur Evaluierung an, darunter Development-Switche und verschiedene Host-Implementierungen. Neben der TTEthernet-Protokollspezifikation dient die Dokumentation und das Verhalten des 100Mbit/s TTEDevelopment-Switch (siehe Abbildung 2.5 auf der nächsten Seite) als Vorlage für die Implementierung. Der Switch ist vollständig in einem FPGA (vgl. Altera) implementiert. Weitere TTEDevelopment-Switch-Modelle, unter anderem mit Gigabit-Ports, werden von TTTech angeboten.
Auf Host-Seite steht neben Netzwerkkarten mit TTEthernet-Unterstützung eine Software-Implementierung des TTEthernet-Protokoll-Stacks für den Einsatz auf COTS-Hardware zur Verfügung. Dieser Software-Stack erlaubt zeitkritischen (time-triggered) und nicht zeitkritischen (best-effort) Datenverkehr. Rate-constrained Nachrichten werden derzeit vom Software-Stack nicht unterstützt. Alle Host-Implementierungen bieten das TTEthernet-Application Programming Interface (API) zur Kommunikation an. Damit kann der Anwendungscode einfach zwischen verschiedenen Systemen ausgetauscht werden.

Abbildung 2.5: 100 Mbit/s TTEDevelopment-Switch (Quelle: TTTech Computertechnik AG)

Das *TTEthernet-API* (vgl. TTTech Computertechnik AG, 2008) ist ein Interface in der Programmiersprache C und dient zum Senden und Empfangen von Nachrichten auf Anwendungsebene; es bildet die Verbindung zwischen Anwendung und Protokoll-Stack. Das TTEthernet-API arbeitet auf Basis von Buffern. Die Anwendung definiert dazu sogenannte Handler, welche den Zugriff auf die Buffer steuern.

TTEthernet-Toolchain

Die TTEthernet-Toolchain gliedert sich momentan in vier Produkte. *TTE-Build* ist ein Werkzeug, um aus der Konfiguration des Gesamtsystems Konfigurationen für die verschiedenen Endsysteme zu generieren. *TTE-Load* lädt die Konfiguration in die Endsysteme. *TTE-Diagnose* ist ein Diagnose- und Netzwerkmanagement-Werkzeug. *TTE-View* ist ein Überwachungs- und Trafficanalysewerkzeug. Für die Erstellung der Konfiguration des Gesamtsystems ist das Werkzeug *TTE-Plan* angekündigt. Abbildung 2.6 auf der nächsten Seite zeigt die verschiedenen Elemente der Toolchain und die verwendeten Formate. Das derzeit noch in der Entwicklung befindliche TTE-Plan hilft bei der Erstellung der Extensible Markup Language (XML)-Konfiguration. Als Eingabe dienen die Anforderungen. TTE-Build wandelt die Konfiguration in ein Image um, welches mit TTE-Load auf die Geräte geladen wird. Mit TTE-View kann ein Netzwerk im Betrieb beobachtet und analysiert werden.
Die Werkzeuge sind überwiegend in der Programmiersprache Pyhon geschrieben und liegen sowohl in Windows- als auch Linux-Paketen vor.

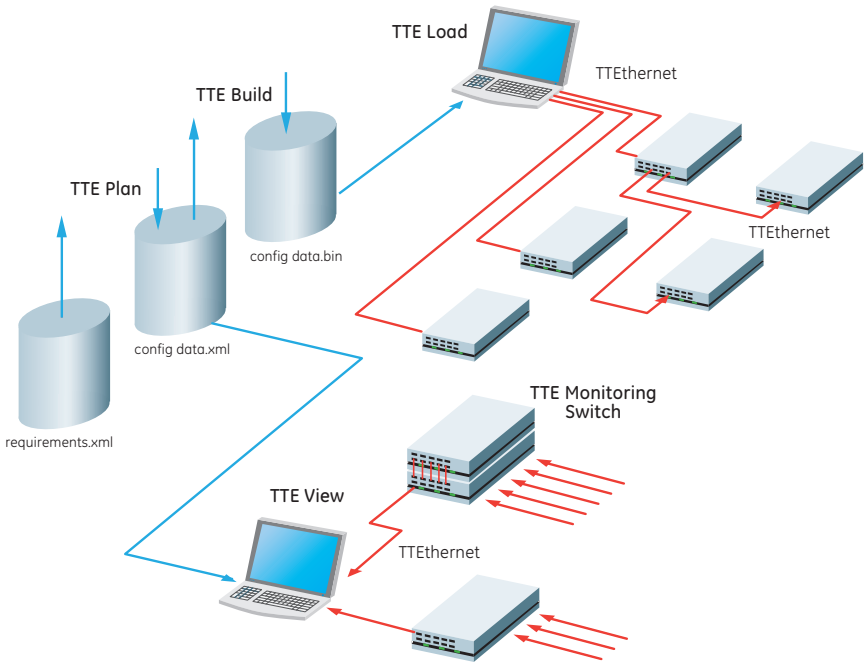

Abbildung 2.6: Überblick über die TTEthernet Toolchain (GE Fanuc Intelligent Platforms, 2009)

Netzwerkkonfiguration: Um ein Netzwerk zu konfigurieren, definiert TTEthernet ein eigenes *XML-Format*, die *Network-Configuration*. Diese Konfiguration soll alle Aspekte des Netzwerkes, von der Topologie bis zum Scheduling, enthalten. Die Network-Configuration basiert auf dem Ecore-Framework aus dem Eclipse Modeling Framework-Projekt (EMF) (vgl. The Eclipse Foundation) und wird von den Werkzeugen der TTEthernet-Toolchain erstellt und verarbeitet. Derzeit ist das Modell der Konfiguration nicht öffentlich. Es werden lediglich die aus dem Modell generierten XML-Schemata an Kunden ausgeliefert. Für diese Arbeit wurde der HAW Hamburg eine Vorabversion des Modells zur Verfügung gestellt.

In seiner derzeitigen Form unterteilt sich das Modell in die Bereiche:

- *System Specification:* Topologieinformationen und High-Level Definitionen zu den Anforderungen
- *Network Configuration:* Netzwerkweite Parameter wie die Konfiguration der Synchronisation
- *Device Specification:* Geräte-spezifische Parameter für jedes Gerät, beispielsweise Schedules und Konformitäts-Überprüfungen
- *Protocol:* Einstellungen für das Standard-Ethernet-Protokoll

- *Device Target Mapping:* Mapping zwischen den abstrakten Definitionen der System-, Device- und Network-Konfigration und der endgültigen Hardware
- *Virtuallink Map:* Mapping zwischen VirtualLink IDs und deren Namen. Mit dem Mapping können symbolische Namen in der Konfiguration verwendet werden

Die Konfiguration wird in mehreren auf diesem Modell basierenden Dateien definiert und mit den TTEthernet-Werkzeugen verarbeitet. Dabei bilden die Dateien einen großen Objektbaum, in dem navigiert werden kann. Durch Referenzen zwischen den Objekten des Baums werden Beziehungen zwischen Konfigurationsobjekten hergestellt.

Ein Vorteil des Konfigurationsschemas ist die vollständige Unabhängigkeit zur verwendeten Hardware. Das Netzwerk wird auf logischer Ebene beschrieben. Erst über das *Device Target Mapping* wird die Beziehung zu echten Geräten hergestellt. Damit ist eine Konfiguration schnell auf verschiedene Geräte adaptierbar. Dies stellt auch einen Vorteil in der Simulation dar, da das Simulationsmodell hardwareunabhängig gehalten werden kann.

TTE-Build: Mit Hilfe des Kommandozeilen-Werkzeugs TTE-Build wird aus der globalen Netzwerk-Konfiguration eine gerätespezifische Konfigurationsdatei erstellt. Für das TTEthernet-Protokoll-Layer sind dies beispielsweise C-Dateien, für die Switche Intel-HEX-Files.

TTE-Load: Um die Konfigurationen in die Endgeräte zu laden, wird TTE-Load eingesetzt. Die Konfiguration besteht aus einem Binärfile, welches mit Hilfe der TTE-Build Software aus einem XML-Dokument generiert wird. Die *TTE-Verify* Software kann diese Konfiguration auf Korrektheit gegenüber der Bitlevel-Spezifikation überprüfen.

TTE-Diagnose: Das Diagnose-Werkzeug hat die Aufgabe, die Diagnoseframes, welche von den Switchen gesendet werden, zu analysieren und kann gleichzeitig mit einem Traffic-Generator Ethernet-Frames erzeugen. Die Diagnosenachrichten vom Switch enthalten globale Werte wie den Zustand, die Version der Konfiguration oder die Summe der verworfenen Pakete genauso wie Diagnoseinformationen für jeden Port. Der interne Traffic-Generator erzeugt Frames mit einer frei wählbaren Quell- und Zieladresse, einer wählbaren Größe und einem definierten Abstand zwischen den Nachrichten.

TTE-View: Die Erweiterung für den Wireshark-Netzwerksniffer (vgl. Combs), TTE-View ermöglicht es, aufgezeichneten TTEthernet-Traffic aufzuschlüsseln und in einer übersichtlichen Weise für einen menschlichen Leser aufzubereiten. Das Plugin stellt Wireshark die Definitionen für das TTEthernet-Protokoll zur Verfügung. Somit können die Informationen beispielsweise aus den Protocol Control Frame (PCF) aufbereitet werden.

2.4 Grundlagen der Simulation

Simulationen unterstützen die Analyse des Verhaltens von dynamischen, nicht trivialen Systemen und Prozessen. Dabei bilden Simulationen die Realität in einem virtuellen Abbild ab. Das Objekt der Simulation wird dabei *System* genannt, das Abbild *Modell* (vgl. Law und Kelton, 1991). Es wird zwischen der Systemzeit des zu simulierenden Systems (Modellzeit) und der der Laufzeit des Simulationsprogramms (Rechenzeit) unterschieden (vgl. Guérin und Peris, 1999, S. 46).
Um die erfolgreiche Umsetzung von Systemen mit komplexer Technologie sicherzustellen, werden oft Simulationen zur Analyse eingesetzt. Mit einem leistungsfähigen Simulationswerkzeug kann bereits in einem frühen Stadium eine Anwendung unter realitätsnahen Bedingungen getestet werden. In Zeiten paralleler Soft- und Hardwareentwicklung ist die Simulation oft die einzige Möglichkeit, erste Integrationstests[10] durchzuführen. Selbst kleine Änderungen an Umsetzung oder Aufbau lassen sich objektiv aufzeichnen und mit früheren Versionen vergleichen. Auch umfangreiche Tests sind ohne großen finanziellen Aufwand möglich (vgl. Kramer, 2008).
Für die Durchführung von Simulationsstudien definiert Schäfer (2004) als Leitfaden das folgende Ablaufschema, das auch als Vorlage für die Umsetzung der Simulationsumgebung in dieser Arbeit dient:

1. Spezifikation des Simulationsobjektes (Abschnitt 2.3 auf Seite 19)
2. Erstellung des konzeptionellen Modells (Abschnitt 4.1 auf Seite 45)
3. Erstellung des Simulationsprogramms (Abschnitt 5.2 auf Seite 63)
4. Durchführen des Simulationsversuches (Abschnitt 6.1 auf Seite 73)
5. Auswertung der Simulationsergebnisse (Abschnitt 6.1.3 auf Seite 74)
6. Validierung des Simulationsmodells (Abschnitt 6.2 auf Seite 77)

2.4.1 Simulationstechniken

Die Art der Zustandsübergänge des zugrunde liegenden konzeptionellen Modells ist ein Klassifizierungsmerkmal für Simulationsmodelle. Es wird zwischen kontinuierlichen und diskreten Modellen unterschieden. Sofern sich die Zustandsgrößen eines Systems über die Zeit kontinuierlich verändern, beispielsweise die Geschwindigkeit bei Beschleunigungsvorgängen, spricht man von einem *kontinuierlichen System*. In *diskreten Systemen* ändern sich

[10] Integrationstests bezeichnen in der Softwareentwicklung Einzeltests, die dazu dienen, voneinander abhängige Komponenten eines Systems im Zusammenspiel zu testen

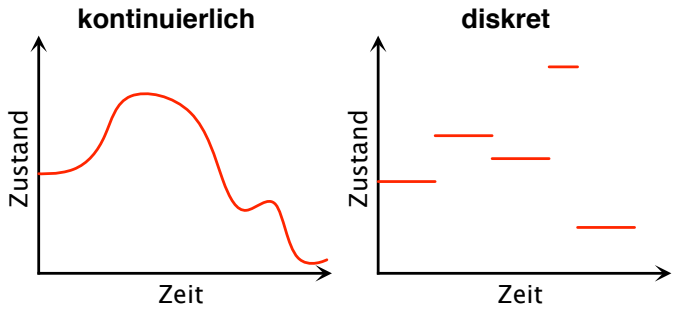

Abbildung 2.7: Zustände in kontinuierlichen und diskreten Systemen

die Zustandsgrößen dagegen *diskontinuierlich*, also sprunghaft (siehe Abbildung 2.7). Die diskreten Zeitpunkte der Zustandsänderung werden dabei *Ereignis* oder *Event* genannt (vgl. Guérin und Peris, 1999, S. 46-47).
In den Naturwissenschaften kommen meist kontinuierliche Systeme vor, die beispielsweise physikalische Phänomene nachbilden. Für die Simulation der Paketweiterleitung in Netzwerken bietet sich dagegen eher ein diskretes, Event-basiertes Modell an, da sich darin die Elemente des Prozesses (Senden, Weiterleiten, Empfangen) besser abbilden lassen (vgl. Liebl, 1995).

2.4.2 Diskrete ereignisbasierte Simulation

Die diskrete ereignisbasierte (Event-basierte) Simulation verwendet Modelle von Systemen, in denen sich die Zustandsvariablen zu verschiedenen Zeitpunkten sofort ändern. Diese Zeitpunkte werden *Events* genannt. Eine diskrete Event-basierte Simulation setzt sich hauptsächlich aus den folgenden Komponenten zusammen:

- *Zustand*: Wert aller Zustandsvariablen, die nötig sind, um das System zu jedem möglichen Zeitpunkt eindeutig zu beschreiben.
- *Takt*: Eine Variable, die die aktuelle Simulationszeit angibt
- *Eventliste*: Eine Liste der folgenden Events und ihrer Zeiten.

Darüber hinaus kann die Simulation zusätzliche Komponenten wie *Zähler* für Statistiken, oder *Report-Generatoren* für die Auswertung enthalten (vgl. Liebl, 1995, S. 10-11).
Der Ablauf einer diskreten Event-basierten Simulation ist in Abbildung 2.8 auf der nächsten Seite dargestellt. Zunächst wird das Modell initialisiert und die Simulationsuhr auf 0 gestellt. Anschließend startet die Simulation im sogenannten *Eventloop* (Simulationsschleife). Zu jedem Simulationsschritt wird das aktuelle Ereignis abgearbeitet. Dabei wird der Systemzustand verändert, und es werden gegebenenfalls neue Folgeereignisse generiert. Danach

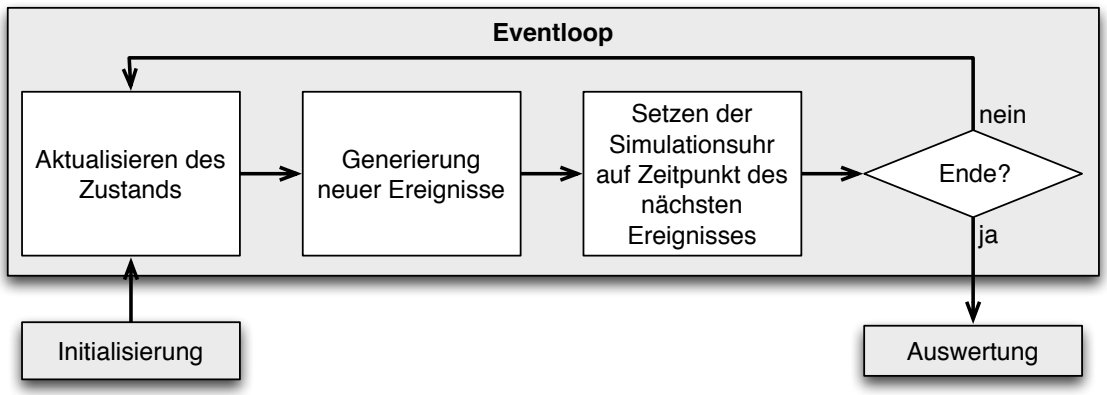

Abbildung 2.8: Ablauf einer diskreten Event-basierten Simulation

wird die Simulationszeit auf den Zeitpunkt des nächsten Events in der Eventliste weiter gestellt. Ist das Simulationsende erreicht, wird der Eventloop abgebrochen und die Auswertung kann gestartet werden.
Die Event-basierte Simulation hat sich de facto als Standard für die Simulation von Netzwerken etabliert. Dies hängt damit zusammen, dass die Paketweiterleitung sehr gut in ein diskretes Event-basiertes Modell überführt werden kann. Nimmt man den Paketfluss zwischen zwei Teilnehmern als Beispiel, sind nur einige wenige Zeitpunkte zwischen Versand und Empfang der Pakete, wie das Eintreffen in den Instanzen zwischen den Teilnehmern, von Interesse. Alle weiteren Zeitpunkte dazwischen können in der Simulation vernachlässigt werden. Im Gegensatz zu physikalischen Modellen, in denen ein bedeutender Rückfluss des Zustandes in das System stattfindet und eine große zeitliche Auflösung zu Ungenauigkeiten führt, hat das System in der Event-basierten Simulation bedeutend weniger Zustände.
Einige verbreitete Werkzeuge für die diskrete Event-basierte Simulation von Netzwerken sind der *NS Network Simulator* (vgl. ns-3 Project), *J-Sim* (vgl. J-Sim Project), *SSFNet* (vgl. Renesys, SSF Project), der *OPNET Modeler* (vgl. OPNET Technologies) und *OMNeT++* (vgl. OMNeT++ Community, b).

2.4.3 Die OMNeT++-Simulationsplattform

Das *Objective Modular Network Testbed in C++ (OMNeT++)* (vgl. OMNeT++ Community, b) ist ein Simulationsframework für Simulationen auf Basis von diskreten Events. Es wird insbesondere in der Event-basierten Simulation von Netzwerken, Mehrprozessor- oder anderen verteilten, parallelen Systemen verwendet.
Die Simulationskomponenten von OMNeT++ werden in der Programmiersprache *C++* entwickelt und sind darauf ausgerichtet, möglichst effizient zu arbeiten. Dadurch können auch

umfangreiche Topologien auf aktueller Computerhardware in akzeptabler Zeit simuliert werden. Die Oberfläche für die Entwicklung, Konfiguration und Auswertung ist in die *Eclipse-Entwicklungsumgebung* (vgl. Eclipse Foundation, a) integriert. Für die Definition von Modellstrukturen verwendet OMNeT++ seine eigene Beschreibungssprache, die Network Description (NED). NED-Dateien können von Hand oder mit Hilfe eines grafischen Editors erstellt und editiert werden. Mit NEDXML steht auch eine XML-konforme Variante der NED zur Verfügung (vgl. Varga, 2001).
Modelle werden in OMNeT++ aus Modulen zusammengestellt, die untereinander über Nachrichtenaustausch kommunizieren. Dabei wird zwischen zwei Arten von Modulen unterschieden:

- *Simple-Modules* enthalten den C++-Quellcode, der mit der OMNeT++-Library entwickelt wird. Sie implementieren das Verhalten des Modells.
- *Compound-Modules* nehmen beliebig viele Simple- und Compound-Modules auf, wobei die Verschachtelungstiefe nicht beschränkt ist. Compound-Modules enthalten keinen Quellcode.

Die Module bieten *Gates* an. Zwischen diesen Gates können gerichtete oder ungerichtete Verbindungen, sogenannte *Connections* bestehen, über die Module referenziert oder Nachrichten versendet werden (siehe Abbildung 2.9 auf der nächsten Seite). Connections über mehrere Hierarchieebenen sind dabei nicht erlaubt, da sie die Wiederverwendbarkeit von Modulen verhindern. Module und Connections können Konfigurationsparameter erhalten. Connections mit spezifischen Eigenschaften nennt man dabei *Channel* (vgl. Varga und Hornig, 2008).
Alle Module sind in der NED-Beschreibungssprache definiert. Die NED verfügt über objektorientierte Techniken wie:

- *Typen*
- *Interfaces*
- *Pakete*
- *Vererbung*
- *Annotationen*

Insbesondere für große Modelle bietet OMNeT++ das Parallel Distributed Simulation-Feature (PDES) (vgl. Şekercioğlu u. a., 2003), welches Simulationen auf einem Cluster verteilen und damit Speicher und CPU-Zeit mehrerer Rechner nutzbar machen kann.
OMNeT++ verfolgt einen Framework-Ansatz. Es bietet nicht direkt Komponenten zur Simulation an, sondern unterstützt den Anwender mit Werkzeugen, um Simulationskomponenten

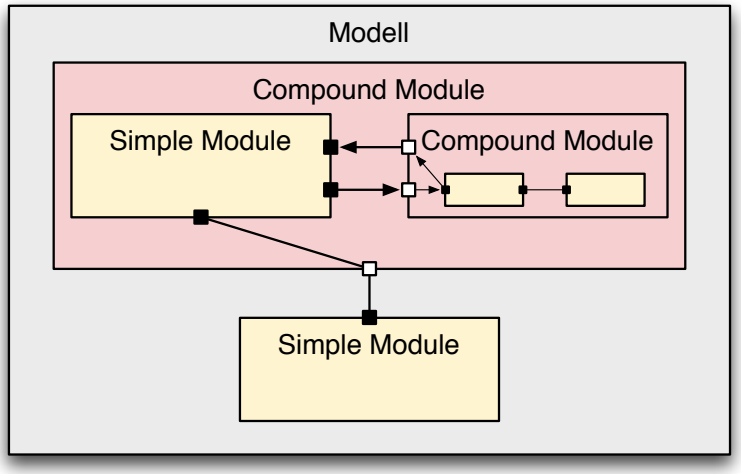

Abbildung 2.9: Struktur von OMNeT++ Modellen — Verschachtelte Module über mehrere Hierarchieebenen

zu erstellen. Frameworks — wie das INET-Framework (siehe Abschnitt 2.4.4) — fügen OMNeT++ Komponenten aus speziellen Anwendungsgebieten hinzu. Diese Frameworks werden unabhängig entwickelt und meist von Dritten herausgegeben.
OMNeT++ ist open-source unter der GNU General Public License (GPL) und für den akademischen Einsatz kostenfrei. Mit OMNEST wird ebenfalls eine kommerzielle Version angeboten. OMNeT++ ist für alle größeren Plattformen verfügbar, darunter Linux, Mac OS X und Windows.
Die Veröffentlichung *An overview of the OMNeT++ simulation environment* (vgl. Varga und Hornig, 2008) zeigt einen Vergleich von OMNeT++ und weiteren Netzwerksimulatoren.

2.4.4 Das INET-Framework

Das INET-Framework (vgl. OMNeT++ Community, a) ist ein Framework für OMNeT++, das die Entwicklung von Simulationen im Bereich der Netzwerkprotokolle vereinfacht. Es besteht aus einer Reihe von Modulen, die die Funktionalität von verschiedenen Elementen — vom physikalischen Layer bis zur Applikationsschicht des OSI-Modells — zur Verfügung stellt.
Das INET-Framework ist modular aufgebaut und setzt intensiv Vererbung ein. Daher ist es besonders gut für die Verwendung in der Simulation von Echtzeit-Ethernet geeignet. Da TTEthernet auf Standard-Ethernet basiert, können große Teile übernommen und um die TTEthernet-Logik erweitert werden. Für die Implementierung von TTEthernet muss sowohl der Ethernet-Host als auch der Ethernet-Switch angepasst und das INET-Framework um das TTE-Protokoll erweitert werden.

Das INET-Framework verwendet dieselben Strukturen wie OMNeT++. Es basiert auf Modulen, welche über Nachrichten kommunizieren. Hosts und Switche sind dabei als Compound-Module realisiert. Diese setzen sich wiederum aus Simple-Modulen zusammen, welche Protokolle, Anwendungen, Algorithmen und weitere funktionale Einheiten implementieren. Die Interfaces, Parameter und Gates werden in NED-Files beschrieben. Ein Netzwerk ist ein spezielles Compound-Modul, welches die Hosts, Switche und die Verbindungen zwischen ihnen beschreibt. Alle INET-Module sind in hierarchischen Paketen organisiert, welche auch den Verzeichnisbaum vorgeben (vgl. Varga, 2010).

2.5 Durchgeführte Vorarbeiten

Im Rahmen des Masterstudiums sind verschiedene Vorarbeiten durchgeführt worden. Als Basis für die Untersuchungen von Ethernet-basierten Vermittlungsinfrastrukturen in Fahrzeugen dient eine Technologie- und Marktübersicht über verschiedene Echtzeit-Ethernet-Protokolle und eine Einführung in die Anforderungen an Automotive-Kommunikationslösungen (vgl. Steinbach, 2008). Aufgrund der Ergebnisse dieser Arbeit wurde TTEthernet als Technologie für weitere Untersuchungen ausgewählt; einerseits wegen seines Fokus auf Fahrzeugnetze, andererseits wegen der verschiedenen angebotenen Traffic-Klassen.
Eine weitere Vorarbeit ist ein Vergleich zwischen TTEthernet und FlexRay auf Basis analytischer Methoden (vgl. Steinbach u. a., 2010; Steinbach, 2009a). Der FlexRay-Bus (vgl. FlexRay Consortium, 2005) ist geeignet für den Einsatz mit harten Echtzeitanforderungen und wird das erste im Automobil eingesetzte Bussystem, welches offline geplanten time-triggered Echtzeitdatenverkehr gemeinsam mit zufällig auftretendem Event-basierten Datenverkehr erlaubt.
Für das analytische Framework sind zunächst die Protokollgrundlagen beider Systeme erarbeitet worden. Sowohl FlexRay als auch TTEthernet basieren auf einem koordinierten TDMA-Konzept. Aus den Spezifikationen können für beide Technologien Modelle erstellt werden, mit deren Hilfe die echtzeitrelevanten Metriken berechnet werden. Anschließend sind für beide Protokolle die Metriken für ausgewählte Topologien verglichen worden. Neben der Ermittlung von grundsätzlichen Eigenschaften ermöglicht das Framework auch die Berechnung von Metriken konkreter Systeme. Damit eignet sich das Framework zur Überprüfung des in dieser Arbeit vorgestellten TTEthernet-Simulationsmodells (siehe Kapitel 6).
Die Ergebnisse des Vergleichs zeigen, dass TTEthernet bei deutlich höherer Bandbreite mit FlexRay vergleichbare Eigenschaften aufweist. Zudem wird deutlich, dass das analytische Modell für die Berechnung von Metriken des time-triggered Datenverkehrs gut geeignet ist, sich für die ungeplanten Abläufe mit event-triggered Datenverkehr jedoch eher nicht eignet.
Weiterhin ist eine Literaturanalyse durchgeführt worden (vgl. Steinbach, 2009b), welche verwandte und ähnliche Veröffentlichungen und Projekte identifiziert und ein Konzept für die

simulationsbasierte Evaluierung von Metriken erarbeitet. Dieses Konzept bildet den Grundstein für den in dieser Arbeit entwickelten simulationsbasierten Evaluierungsprozess.
Als letzter Schritt ist in den Vorarbeiten ein Konzept für ein TTEthernet-Simulationsmodell entwickelt und mit der Implementierung und Dokumentation (vgl. Steinbach u. a., 2011; Steinbach, 2010) begonnen worden. Dabei zeigt es sich, dass die OMNeT++-Plattform zusammen mit dem INET-Framework gut für die Simulation von Echtzeit-Ethernet-Protokollen geeignet ist. Durch die Validierung mit realer Hardware und dem analytischen Modell von TTEthernet lässt sich die geforderte Präzision des Simulationsmodells sicherstellen.
Die durchgeführten Vorarbeiten zeigen die generelle Realisierbarkeit von Fahrzeugkommunikation auf Basis von Echtzeit-Ethernet und die Möglichkeiten der simulationsbasierten Evaluierung der Metriken solcher Netze. Damit haben sie signifikant dazu beigetragen, das Risiko der vorliegenden Arbeit zu reduzieren.

2.6 Verwandte und vergleichbare Arbeiten

Der Einsatz von neuen Vermittlungsinfrastrukturen in Kraftfahrzeugen wird durch verschiedene Institutionen untersucht. Insbesondere ist die BMW-Group zu erwähnen, welche im Jahre 2007 als technische Demonstration einen Internetprotokoll-basierten (IP) Fahrzeug-Prototyp vorstellte und den Einsatz von Ethernet für Anwendungen im Fahrzeug anstrebt. Als prototypische Anwendung nennt BMW die Videoübertragung der Kameras für den Parkassistenten im kommenden BMW X5-Modell m Jahr 2013 (vgl. Bruckmeier, 2010). Große Vorteile werden dabei in der hohen Verbreitung von Ethernet gesehen. Es werden heute sogar mehr Ethernet-Komponenten als Einheiten der klassischen Automotive-Technologien verkauft (siehe auch Abbildung 2.1 auf Seite 14). Dies erhöht die Qualität der Komponenten. Bruckmeier (2010): „The larger the market for a technology, the more knowledge available, the better tested, maintained and developed.“ Als langfristiges Ziel für einen hoch integrierten Ethernet-basierten Backbone wird das Jahr 2020 genannt.
Insbesondere in Zusammenarbeit zwischen der Technischen Universität München und BMW (CAR@TUM) entstanden verschiedene Forschungsprojekte und Veröffentlichungen zu Grundlagenthemen wie dem Design von vereinheitlichten Kommunikationsinfrastrukturen (vgl. Müller-Rathgeber u. a., 2008b,c), der Abbildung von Kommunikationsstrukturen aktueller Fahrzeuge auf Ethernet-basierte Vermittlungsinfrastrukturen (vgl. Müller-Rathgeber und Michel, 2009) oder der Koexistenz von Traffic-Klassen mit verschiedenen Eigenschaften in geswitchten Netzwerken (vgl. Hillebrand u. a., 2007). Die Arbeiten basieren fast ausschließlich auf Standard-Ethernet und grenzen sich damit von den Arbeiten der CoRE-Arbeitsgruppe (vgl. CoRE RG, a) an der Hochschule für Angewandte Wissenschaften Hamburg ab.
Aus den Ergebnissen der Untersuchungen der TU München mit BMW lassen sich erste Erkenntnisse zu Metriken von Ethernet-basierten Vermittlungsinfrastrukturen gewinnen. Die

Versuche zu den Schedulingverfahren zeigen, dass die synchronisierten Protokolle gegenüber ausschließlich prioritätengesteuerten Protokollen insbesondere beim Jitter im Vorteil sind. Dies unterstützt die Entscheidung für TTEthernet, welches mit den time-triggered Nachrichten auch die synchrone Kommunikation beherrscht.
Als Fazit aus den genannten Arbeiten lässt sich ziehen, dass die Entwicklung im Bereich der Kommunikationsinfrastrukturen bisher nicht mit der Entwicklung im Softwarebereich schritthält. Neue Verfahren sind daher dringend notwendig (vgl. Müller-Rathgeber u. a., 2008b).
Im Gegensatz zu den Arbeiten an der TU München steht in dieser Arbeit mit Time-Triggered Ethernet (TTEthernet) (vgl. Steiner, 2008), einem Produkt der Firma TTTech (vgl. TTTech Computertechnik AG), eine zeitgesteuerte Ethernet-Variante im Zentrum. Die Grundlagen zu TTEthernet stammen aus der Real-Time Systems Group des Instituts für Technische Informatik an der Technischen Universität Wien (vgl. Kopetz u. a., 2005). Später wurden die Ideen durch die Wiener Firma TTTech aufgegriffen und mit Partnern (vgl. z.B. Honeywell International) als kommerzielles Produkt — Time-Triggered Ethernet (TTEthernet) — weiterentwickelt. Das in dieser Arbeit eingesetzte TTEthernet von TTTech ist nicht identisch mit der Arbeit an der TU Wien, es beruht aber im Grundsatz auf denselben Prinzipien (vgl. Kopetz u. a., 2005; Kopetz, 2008). TTEthernet wird derzeit unter der Kennung AS 6802 in der Arbeitsgruppe „Time Triggered Systems and Architecture Committee“ (AS-2D) der Society of Automotive Engineers (SAE) standardisiert (SAE - AS-2D Time Triggered Systems and Architecture Committee, 2009).
In der Prozessautomatisierung sind die Evaluierungen von Echtzeit-Ethernet-Protokollen schon weit fortgeschritten. Die Dissertation von Jürgen Jasperneite (vgl. Jasperneite, 2002) beschäftigt sich mit der Leistungsbewertung von Ethernet für die prozessnahe Echtzeitkommunikation. Hier konnten viele Anregungen aus dem Bereich der Modellierung und der Simulation übernommen werden. Impulse zur Evaluierung von Zuverlässigkeitsmetriken mit Hilfe der diskreten Event-basierten Simulation liefert die Dissertation von Andreas Schäfer zur Verlässlichkeitsanalyse in paketvermittelnden Netzwerken (vgl. Schäfer, 2004).
Im Bereich der Simulationsmodelle ist das Masterprojekt von Taweewit Pensawat (Halmstad University) (vgl. Pensawat, 2006) eine nah verwandte Arbeit. Hier wurde Earliest-Deadline-First-Scheduling (EDF), ein weiterer Ansatz für eine echtzeitfähige Ethernet-Erweiterung, exemplarisch mit dem Simulationswerkzeug OMNeT++ analysiert. Im Gegensatz zur vorliegenden Arbeit geht es jedoch in dem Projekt weniger um die Bewertung von Echtzeitsystemen, als um die Umsetzung des EDF-Algorithmus in ein zeitdiskretes Simulationsmodell. Ein einfaches Simulations-Modell des RTnet Echtzeit-Ethernet-Protokolls (vgl. Real-Time Systems Group) wurde von Chongwei Ling und Zhonglei Wang (vgl. Ling und Wang, 2008) in OMNET++ entwickelt. Der Fokus der Arbeit lag dabei auf der Umsetzung des RTnet-Protokolls in die Simulation und nicht wie in vorliegender Arbeit auf der Erstellung eines Evaluierungsprozesses.
Da TTEthernet eine Echtzeit-Ethernet-Erweiterung ist, welche verschiedene Traffic-Arten auf derselben Layer 2 Infrastruktur vereint, sind Erfahrungen mit der Simulation der Koexistenz

verschiedener Echtzeit-Ethernet-Protokolle (vgl. Ferrari u. a., 2008) für diese Arbeit relevant. Für die Modellierung des Schedulings werden zudem Anregungen aus einem IEEE 1588 (vgl. Institute of Electrical and Electronics Engineers, 2002) Protokollmodell in OMNeT++ (vgl. Praus u. a., 2007) übernommen.
Auch im Automotive-Bereich werden OMNeT++ basierte Simulationsansätze verwendet, so z.B. bei der Verifizierung von Kernanforderungen an Switch-basierte Kommunikation im Fahrzeug (vgl. Müller-Rathgeber u. a., 2008a) oder der Simulation von MOST-Strecken (vgl. Schramm u. a., 2008). Für bestehende Automotive-Bussysteme werden Simulationsumgebungen wie z.B. CANoe (vgl. Vector Informatik GmbH) oder chronSIM (vgl. INCHRON GmbH) angeboten.
Nach aktuellem Kenntnisstand sind keine vollständigen Implementationen von time-triggered Echtzeit-Ethernet-Erweiterungen für OMNeT++ öffentlich verfügbar. Für andere Simulationswerkzeuge wie den *OPNET Modeler* gibt es Modelle für Protokolle aus der Anlagensteuerung (vgl. Seno und Zunino, 2008; Ferrari u. a., 2006). Da die Protokolle jedoch von TTEthernet abweichen, sind diese Modelle nicht als Vorlage für das Simulationsmodell geeignet.

Kapitel 3

Metriken der Automotive-Kommunikation

> If you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind.
>
> *Lord Kelvin 1889*

Metrik, (griechisch $\mu\varepsilon\tau\rho\iota\kappa\eta$ Zählung, Messung) ist in der Physik und Technik ein Begriff für Kennzahlen. Metriken legen fest, in welchen Indikatoren ein Objekt gemessen bzw. angegeben werden kann oder soll.
Dieses Kapitel beschreibt zunächst die Bedeutung von Metriken für die Evaluierung von Echtzeit-Systemen (Abschnitt 3.1). Anschließend wird gezeigt, mit welchen Techniken Metriken ermittelt werden können (3.2). Abschließend wird eine Auswahl echtzeitrelevanter Metriken, welche sich für die simulationsbasierte Evaluierung eignen, erarbeitet und erklärt (Abschnitt 3.3).

3.1 Bedeutung von Metriken

Metriken ordnen Systemzuständen reale Zahlenwerte zu. Damit sind sie geeignet als Qualitätsangabe (Absolutbewertung) sowie für die Vergleichbarkeit verschiedener Systeme (Relativbewertung) (vgl. Schäfer, 2004, S. 29).
Für die Bewertung von Kommunikationstechnologien und ihrer Konfiguration haben Metriken eine große Bedeutung (vgl. Jasperneite, 2002). Sie können die Grundlage für Aussagen zum technologischen und wirtschaftlichen Erfolg von zukünftigen Vermittlungsinfrastrukturen in Fahrzeugen bilden. Insbesondere für einen objektiven, kritischen Blick auf Optimierungen und Lösungsvorschläge und die Bewertung von Protokoll-Verbesserungen sind Metriken wichtig. Auch für das Festlegen von Anforderungen sind Metriken bedeutsam. Werden Anforderungen in definierten Metriken formuliert, sind sie allgemein verständlich, mess- und

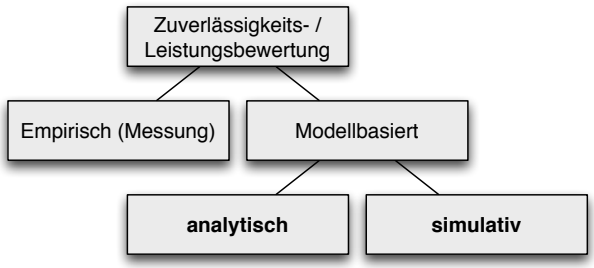

Abbildung 3.1: Verschiedene Methoden der Ermittlung von Metriken (nach Haas und Zorn, 1995)

verifizierbar. Damit eignen sich Metriken als gemeinsame Sprache für die in der Automobilbranche übliche verteilte Entwicklung (vgl. McCabe, 2007).
Metriken stehen im Zentrum der Leistungsbewertung von Systemen. Bei der Leistungsbewertung wird die Leistungsfähigkeit eines Systems untersucht. Dabei wird das System unter Arbeitslast betrieben und die entsprechenden Metriken ermittelt (vgl. Jasperneite, 2002). Im Bereich der Bewertung von Netzwerk-Komponenten gibt es derzeit sowohl bei der Auswahl als auch bei den Messverfahren der Verlässlichkeits- und Leistungsmetriken keine Standardisierung (vgl. Schäfer, 2004, S. 29). Daher werden im Folgenden Techniken zur Ermittlung von Metriken vorgestellt und die Metriken für den simulationsbasierten Evaluierungsprozess dieser Arbeit ausgewählt und erklärt.

3.2 Ermittlung von Metriken

Die Ermittlung von Metriken im Rahmen der Leistungs- und Verlässlichkeitsbewertung kann sowohl durch empirische Untersuchung im Betrieb als auch am Modell vorgenommen werden. Die empirische Untersuchung wird anhand der Messung von Metriken am realen System vorgenommen. Oft sind empirische Untersuchungen aufwendiger als Versuche am Modell oder können gar nicht durchgeführt werden, weil das reale System noch nicht existiert. Insbesondere bei großen und komplexen Systemen kann eine empirische Untersuchung unwirtschaftlich sein. Zudem müssen empirische Studien, insbesondere für die Verlässlichkeitsbewertung, über lange Zeit durchgeführt werden, um statistisch verwertbare Aussagen zu erhalten. Eine Alternative zur Messung am realen System können hier Untersuchungen am Modell sein.
Modellierungstechniken dienen zur Ermittlung von Metriken auf Basis abstrakter Modelle. Die Ermittlung kann dabei entweder durch mathematische Methoden (analytisch) oder durch Nachbildung realer Abläufe (Simulation) erfolgen (vgl. Jeruchim u. a., 2000, S. 2) (siehe Abbildung 3.1).

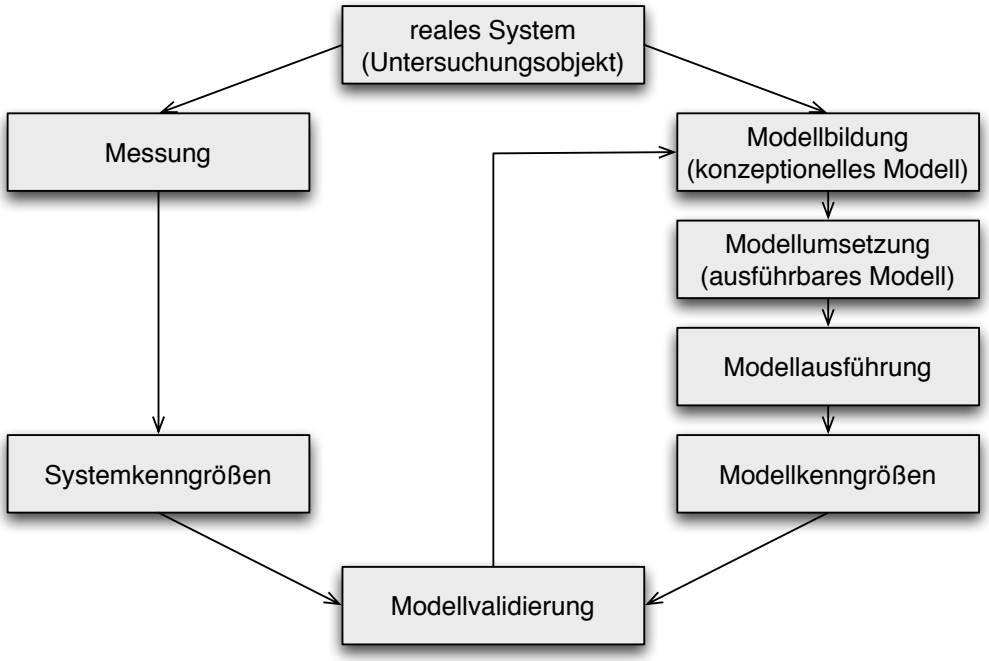

Abbildung 3.2: Arbeitsschritte bei der Modellierung nach Jasperneite (2002) und Haas und Zorn (1995)

Im Gegensatz zu Messungen erfolgt bei der Modellierung (sowohl analytisch als auch in der Simulation) die Untersuchung nicht am realen System. Zunächst werden beim Modellierungsprozess die relevanten Systemeigenschaften des Evaluierungsobjektes ermittelt. Daraus entsteht ein abstraktes Modell des Systems, das die Abläufe nachbildet. Durch analytische oder simulative Verfahren werden anschließend die Modellkenngrößen bestimmt. Durch den Vergleich mit den Systemeigenschaften kann das Modell evaluiert werden. (vgl. Jasperneite, 2002, S. 44-46) (siehe auch Abbildung 3.2).
Die Abbildungsgenauigkeit eines Modells wird bestimmt durch die konkurrierenden Größen Effizienz und Detaillierungsgrad. Je genauer ein Modell das System abbildet, desto ineffizienter ist es in der Ausführung. Daher ist bei der Modellbildung wichtig, dass nur die Teile des Systems abgebildet werden, welche Einfluss auf die gesuchten Größen haben (vgl. Schäfer, 2004, S. 37).
In einem analytischen Modell wird die Realität auf eine abstrakte mathematische Struktur abgebildet. In der Simulation wird dagegen versucht, mit einem Modell die Abläufe und Gegebenheiten der Realität nachzubilden (vgl. Solomon, 1980). Analytische Lösungen sind besser geeignet, wenn die Beziehungen und Zusammenhänge innerhalb des Modells einfach strukturiert sind. Viele reale Prozesse sind jedoch komplex und daher nicht für die ana-

lytische Evaluierung geeignet. Solche Modelle sollten mit Hilfe von Simulationen betrachtet werden (vgl. Law und Kelton, 1991).
Bei der Auswahl der Modellierungsmethode sollten nach Schäfer (2004) die folgenden Kriterien beachtet werden:

- Abbildungsgenauigkeit
- Detaillierungsgrad
- Verallgemeinerbarkeit
- Entwicklungsaufwand
- Rechenaufwand

In den Vorarbeiten zu dieser Abschlussarbeit sind sowohl analytische- als auch Simulationsverfahren für die Ermittlung von Metriken verwendet worden. Im folgenden werden die Eigenschaften beider Modellierungsansätze vorgestellt.

3.2.1 Analytische Evaluierung von Metriken

Analytische Modelle werden insbesondere in der Entwurfsphase von Rechensystemen eingesetzt. Sie dienen der Steigerung der Leistung und der Verlässlichkeit, indem verschiedene Systementwürfe miteinander verglichen werden. Vorteile von analytischen Modellen sind die gute Skalierbarkeit, die einfache Möglichkeit, Auswirkungen von Eingabeparametern zu quantifizieren, und die Geschwindigkeit der analytischen Algorithmen. Ein wesentlicher Nachteil ist die oft notwendige Vereinfachung des Systems und damit eine geringere Abbildungsgenauigkeit. Analytische Verfahren weisen in der Regel einen höheren Abstraktionsgrad auf als Simulationen (vgl. Schäfer, 2004, S. 37).
Im Bereich der Netzwerke sind analytische Modelle insbesondere für die Analyse einfacher Strukturen geeignet. Je komplexer die Topologien und Konfigurationen werden, desto aufwändiger wird die Berechnung der Metriken. Insbesondere die Metriken von event-triggered Datenverkehr erfordern großen Aufwand bei der Berechnung.
In den Vorarbeiten ist gezeigt worden, wie mit Hilfe eines analytisch-mathematischen Modells, Metriken verschiedener Echtzeit-Datenübertragungssysteme miteinander verglichen werden können. Zu diesem Zweck ist ein Framework (vgl. Steinbach u. a., 2010) erstellt worden, welches die Metriken Latenz, Jitter und Bandbreite für die time-triggered Kommunikation von FlexRay und TTEthernet gegenüberstellt. Es wird deutlich, dass das analytische Modell für die Berechnung von Metriken des time-triggered Datenverkehrs gut geeignet ist, sich für die komplexen Abläufe des event-triggered Datenverkehrs jedoch eher nicht eignet.

3.2.2 Evaluierung von Metriken in der Simulation

Im Gegensatz zur analytischen Evaluierung basiert die Evaluierung in der Simulation auf Versuchen am Modell. Dabei wird das Verhalten des Untersuchungsobjektes und der Umgebung durch Simulation der Abläufe und Prozesse in einem Simulationsprogramm nachgestellt. Es wird zwischen der kontinuierlichen und der diskreten ereignisorientierten Simulation unterschieden (siehe Abschnitt 2.4 auf Seite 27).
Ein bedeutender Vorteil der Simulation, verglichen mit analytischen Verfahren, ist der Detaillierungsgrad. Mit simulationsbasierten Ansätzen lässt sich ein System in nahezu jeder gewünschten Tiefe betrachten. Der Aufwand, der dafür in die Modellierung gesteckt werden muss, hängt vom zu erzielenden Detailgrad ab. Zudem ist die Komplexität der Modelle nicht durch die mathematische Berechenbarkeit beschränkt. Die Grenze der Komplexität einer Simulation liegt in erster Linie in der vom Benutzer tolerierten Laufzeit und dem im Simulationssystem verfügbaren Speicher(vgl. Jasperneite, 2002).
Somit stellt vor allem die Rechenzeit einen bedeutenden Nachteil der Simulation dar. Sie liegt weit über der der analytischen Evaluierung. Mit einer sorgfältigen Auswahl der Modellierungs- und Simulationstechniken kann die Rechenzeit jedoch auf ein notwendiges Mindestmaß reduziert werden (vgl. Jeruchim u. a., 2000). Darüber hinaus liegen die Entwicklungskosten des Simulationsmodells oft deutlich über denen eines analytischen Modells (vgl. Schäfer, 2004, S. 36).
Um die Präzision des Simulationsmodells zu verbessern, können Messungen am Simulationsobjekt zur Gewinnung realitätsnaher Einflussgrößen hinzugezogen werden. Dies gilt insbesondere für die Modellierung der Arbeitslast (vgl. Jasperneite, 2002).

3.3 Auswahl von Metriken

Grundsätzlich können Metriken in verschiedene Klassen unterteilt werden. Möglich ist beispielsweise eine Einteilung in folgende Klassen:

- *Ökonomische Metriken* bewerten Größen aus dem wirtschaftlichen Bereich wie die Anschaffungs-, Installations- oder Unterhaltskosten.
- *Verlässlichkeits-Metriken* bewerten die Zuverlässigkeit, mit der ein System eine Aufgabe erfüllt.
- *Leistungs-Metriken* bewerten die Leistungsfähigkeit eines Systems bei der Erbringung einer Aufgabe.

Die Tabelle 3.1 auf der nächsten Seite zeigt Metriken und ihre Klassifizierung innerhalb dieser Kategorien.

Ökonomische Metriken	Verlässlichkeits-Metriken	Leistungs-Metriken
Anschaffungskosten	Zuverlässigkeit	Latenz
Entwicklungskosten	Wartbarkeit	Jitter
Installationskosten	Verfügbarkeit	Bandbreite
Unterhaltskosten	Paketverlustrate	Zykluslänge
		Anzahl Botschaften
		Energieverbrauch
		Rechenaufwand
		Start-Up Geschwindigkeit
		EMV
		Gewicht
		Abmessung

Tabelle 3.1: Klassifizierung von Metriken für Kommunikationsdienste

Zu einer Metrik gehört neben einer genauen Definition auch die Beschreibung des Blickwinkels der Betrachtung. So kann die Verfügbarkeit beispielsweise als 99,99% angegeben werden und bewertet in diesem Fall den Intaktzustand, oder als mittlere Ausfalldauer von 52 Minuten/Jahr und damit als Aussage über den Defektzustand (vgl. Schäfer, 2004, S. 31).
Nicht alle Metriken sind geeignet, um in einem simulationsbasierten Evaluierungsprozess ermittelt zu werden. Für die Bewertung der Qualität von Kommunikationsdiensten sind — neben den großen Drei: *Reliability, Maintainability, Availability (RMA)* — die wichtigsten Metriken die *Latenz*, die *Varianz der Latenz (Jitter)* und die nutzbare *Bandbreite* (vgl. Guérin und Peris, 1999). Im folgenden werden die Metriken, die im Evaluierungsprozess dieser Arbeit berücksichtigt werden sollen, vorgestellt.

Zuverlässigkeit (Reliability), Wartbarkeit (Maintainability), Verfügbarkeit (Availability)

Die Zuverlässigkeit ist ein statistischer Faktor, der die Wahrscheinlichkeit angibt, ob ein System zu einem gegebenen Zeitpunkt t funktioniert. Unter der Annahme, dass das System zum Zeitpunkt t_0 funktionsfähig war, und das System eine Fehlerrate von λ Ausfällen pro Stunde besitzt, kann die Zuverlässigkeit $R(t)$ nach Kopetz (2004) mit der Formel 3.1 berechnet werden:

$$R(t) \quad = \quad \mathrm{e}^{-\lambda \cdot (t - t_0)} \tag{3.1}$$

Die Fehlerrate gibt die Klasse der Zuverlässigkeit an. Ab einer Fehlerrate niedriger 10^{-9} wird von *ultrahigh reliability* gesprochen. Das folgende Beispiel gibt einen Eindruck von der

Größenordnung der ultrahigh reliability. Eine Fehlerrate von 10^{-9} würde verlangen, dass bei einer durchschnittlichen Nutzung von einer Stunde bei einer Million Fahrzeugen nur ein sicherheitskritischer Fehler pro Jahr auftreten darf.
Für Komponenten werden die Fehlerraten meist in Parts per Million (ppm) angegeben. Da die Lebensdauer von Systemen im Automobil bei mehreren tausend Betriebsstunden über einen Zeitraum von über 10 Jahren liegt und für viele Anwendungen mehrere Elemente benötigt werden, werden im Automobilbereich für einige Komponenten Fehlerraten im einstelligen ppm Bereich gefordert (vgl. Reif, 2009, S. 76). Das bedeutet, dass über die komplette Lebensdauer bei einer Million ausgelieferten Teilen weniger als 10 einen Defekt aufweisen dürfen (vgl. Kopetz, 2004, S. 9-11).
Mit Wartbarkeit wird üblicherweise die Zeit beschrieben, die benötigt wird, um das System aus dem Fehlerzustand in den betriebsbereiten Zustand zu überführen. Wenn eine sicherheitskritische Komponente im Fahrzeug ausfällt, ist üblicherweise das Fahrzeug nicht mehr betriebsbereit. Die Dauer der Reparatur ist insbesondere unter ökonomischen Gesichtspunkten von Interesse, für die simulationsbasierte Evaluierung jedoch eher nicht geeignet.
Die Verfügbarkeit ist die Relation zwischen der Ausfallzeit und der Laufzeit. Sie kann daher durch Verbesserung der Zuverlässigkeit oder Verkürzung der Wartung verbessert werden (vgl. McCabe, 2007).

Latenz

Die Latenz oder Nachrichtenlaufzeit beschreibt die zeitliche Verzögerung zwischen dem Aussenden eines Paketes beim Sender und dem Eingang des Paketes beim Empfänger (siehe Abbildung 3.3). Sie setzt sich aus verschiedenen Komponenten wie der *Bearbeitungsverzögerung*, der *Warteschlangenverzögerung*, der *Sendeverzögerung* und der *Signallaufzeit* zusammen (vgl. Guérin und Peris, 1999).
Die Latenz ist eine wichtige Metrik für die zeitkritische Kommunikation. Insbesondere bei der Regelung von Prozessen spielt die Latenz eine wichtige Rolle. Sie hat direkten Einfluss auf die Totzeit[11] und ist damit eine maßgebende Größe für die Regelgüte einer verteilten Regelung. Insbesondere bei Prozessen im Automobil wie Antiblockiersystem (ABS), Antriebsschlupfregelung (ASR) oder Elektronisches Stabilitätsprogramm (ESP) ist eine geringe Totzeit und damit auch eine geringe Latenz wichtig. Ein allgemeiner Richtwert für die maximale Totzeit in Regelprozessen kann nicht festgelegt werden. Er hängt vor allem von der Geschwindigkeit des zu regelnden Prozesses ab und ist daher stets in Relation zur Anwendung zu betrachten (vgl. Kopetz, 2004, S. 6).
Die Latenz kann von Sender zu Empfänger (*Ende-zu-Ende Latenz*) oder in beide Richtungen (*Roundtrip Latenz*) gemessen und angegeben werden (vgl. McCabe, 2007). In zeitkritischen

[11] Die Totzeit (auch Laufzeit oder Transportzeit) bezeichnet die Zeitspanne zwischen Änderung am Systemeingang und Antwort am Systemausgang einer Regelstrecke

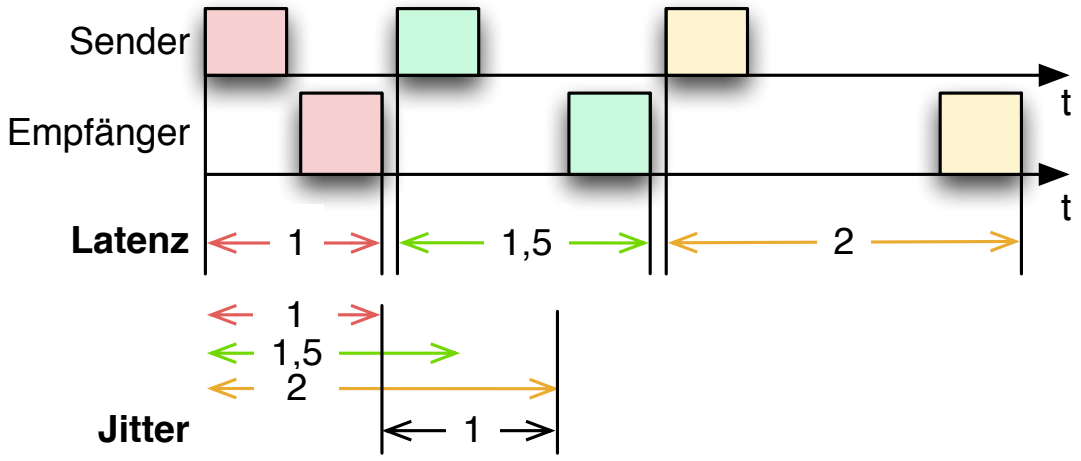

Abbildung 3.3: Latenz (Laufzeit) und Jitter (Differenz der maximalen und minimalen Laufzeit) in der Paketvermittlung

Systemen wird üblicherweise nur die Ende-zu-Ende Latenz angegeben, da Verbindungen unidirektional oder asymmetrisch konfiguriert sein können.
Neben der Nachrichtenlaufzeit können noch weitere Faktoren die Latenz bestimmen. Dazu gehört beispielsweise die *Error Detection Latency*, die Dauer der Fehlererkennung. Dies betrifft Systeme, die Fehler, beispielweise über Redundanz, kompensieren können, zur Fehlererkennung jedoch eine gewisse Zeit benötigen(vgl. Kopetz, 2004, S. 9).

Jitter

Der Nachrichten-Jitter beschreibt die Varianz der Nachrichten-Laufzeit. In Netzwerken wird der Jitter meist über eine definierte Anzahl von Paketen angegeben (vgl. Tanenbaum und Wetherall, 2011). Für Echtzeit-Systeme ist es wichtig, den maximalen Jitter, also die Differenz zwischen maximaler und minimaler Paketlaufzeit, zu kennen. Daher ist es in diesen Systemen gängig, den Jitter als die Differenz zwischen oberer und unterer Grenze der Nachrichtenlaufzeit zu definieren (vgl. Kopetz, 2004, S. 9) (siehe Abbildung 3.3).
Ein übliches Verfahren für verteilte Regler ist die zyklische Übertragung von Informationen. Der Sender sendet einen Wert in regelmäßigen Abständen an die Empfänger. Der Nachrichten-Jitter führt dazu, dass der Zyklus beim Empfänger nicht mehr sauber eingehalten werden kann. In synchronisierten Systemen kann durch die Verzögerung der Verarbeitung von eingehenden Nachrichten ein leichter Jitter zuungunsten der Latenz ausgeglichen werden. Ein starker Jitter führt dabei jedoch zu instabilen Regelprozessen. Eine übliche obere Schranke für den Jitter ist 10% der Latenz (vgl. Kopetz, 2004).

Bandbreite

Die Bandbreite, oft auch als Kapazität oder Durchsatz bezeichnet, beschreibt die maximale Menge an Informationen, die pro Zeiteinheit übertragen werden können. Damit ist die Bandbreite ein Kriterium der Leistungsfähigkeit eines Übertragungsmediums (vgl. Tanenbaum und Wetherall, 2011). Die nutzbare Bandbreite sollte stets netto, ohne die Protokollinformationen, angegeben werden. Da jede Technologie einen anderen Protokoll-Overhead besitzt, ist nur so ein gerechter Vergleich möglich. Bei Übertragungstechniken mit einer zyklischen Übertragung ist zudem die maximale Anzahl verschiedener Botschaften in einem Zyklus von Interesse. Sie hängt neben der Bandbreite des Mediums auch von der Länge der Nachrichten ab.

Weitere Metriken wie die Zykluslänge oder die Anzahl möglicher Botschaften lassen sich gut mit analytischen Methoden durch Analyse der Scheduling-Konfiguration ermitteln. Andere Metriken wie der Energieverbrauch, die Start-Up Geschwindigkeit, die Elektromagnetische Verträglichkeit (EMV), das Gewicht oder die Abmessungen erfordern reale Hardware für empirische Untersuchungen oder zumindest Modelle der Implementierungen.

Kapitel 4

Konzept und Architektur

In diesem Kapitel wird das Konzept des Evaluierungsprozesses vorgestellt (Abschnitt 4.1) und ein Überblick über die Architektur und Komponenten (Abschnitt 4.2) gegeben.

4.1 Konzept des Evaluierungsprozesses

Aufgrund der Erfahrungen aus den Vorarbeiten mit der analytischen und der simulativen Evaluierung (siehe Abschnitt 2.5 auf Seite 32) wird ein simulationsbasierter Evaluierungsprozess gewählt. Dies liegt insbesondere am benötigten Detaillierungsgrad, der in einem analytischen Modell nur mit großen Aufwand erreicht werden könnte.
Abbildung 4.1 auf der nächsten Seite zeigt die Elemente des Evaluierungsprozesses, sowie die Ein- und Ausgangsgrößen. Die Simulation wird durch eine Netzwerkkonfiguration gespeist, die zuvor manuell oder mit Werkzeugunterstützung aus dem Fahrzeugmodell erstellt wurde. Zusätzlich definieren die Anwendungen dieses Modells die Anforderungen an die Vermittlungsinfrastruktur, die in der Evaluierung analysiert werden.
Der Evaluierungsprozess selbst ist in drei Phasen unterteilt. Zunächst wird die Simulation auf Basis der Netzwerkkonfiguration und Anwendungsanforderungen konfiguriert. Anschließend wird in der Simulationsphase das abgeleitete Modell in der Simulationsumgebung vermessen und dokumentiert. Abschließend werden die Ergebnisse in der Auswertungsphase mit den zuvor erstellten Anforderungen verglichen und bewertet. In einer übersichtlichen Darstellung wird dem Nutzer gezeigt, an welcher Stelle das Modell nicht den Anforderungen genügt. Diese Information soll dem Nutzer helfen, in einem iterativen Vorgehen die Konfiguration anzupassen, bis sie alle Vorgaben erfüllt. Als Ergebnis erhält der Nutzer eine Übersicht über alle für ihn relevanten Metriken zusammen mit einer Netzwerkkonfiguration, welche direkt in vorhandene Hardware eingespielt werden kann.
Im folgenden werden mögliche Einsatzszenarien, die Arbeitsabläufe im Evaluierungsprozess und das Konzept für die Modellierung beschrieben.

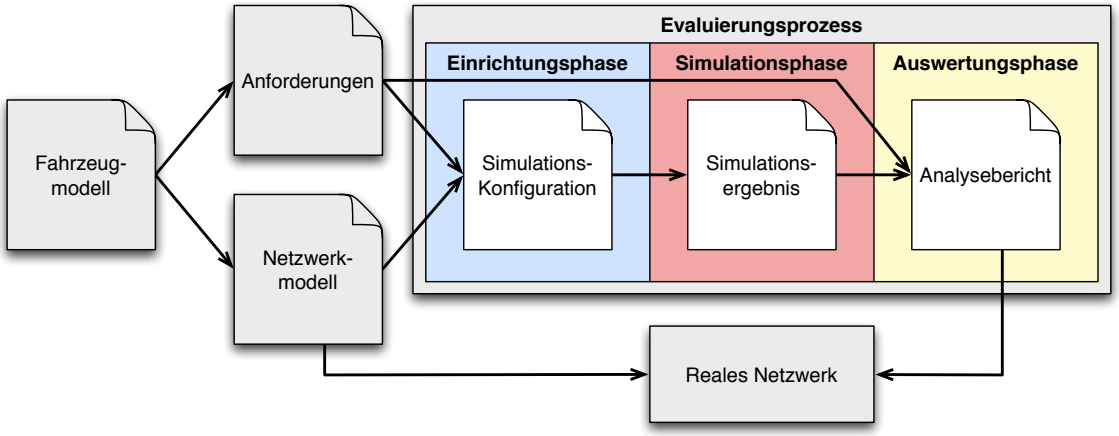

Abbildung 4.1: Überblick über die Phasen des Evaluierungsprozesses und seine Ein- und Ausgabegrößen

4.1.1 Mögliche Einsatzszenarien

Der in dieser Arbeit beschriebene Evaluierungsprozess soll die Entwicklung von Echtzeit-Ethernet basierten Netzwerken, insbesondere im Bereich der Automotive-Anwendungen, an verschiedenen Stellen unterstützen und begleiten.
Angefangen bei ersten Proof-of-Concept[12] Untersuchungen, kann die Simulation verwendet werden, um grobe Aussagen über die zu erwartenden Eigenschaften einer Echtzeit-Ethernet basierten Vermittlungsinfrastruktur zu treffen. Damit kann das Risiko beim Einsatz dieser neuen Technologie von Anfang an minimiert werden.
Anschließend soll der Evaluierungsprozess die Entwicklung bis zum realen Prototyp begleiten. Während der Planungsphase können mit Hilfe des Prozesses Eigenschaften verschiedener Topologien und Konfigurationen getestet werden. Damit kann schon zum frühest möglichen Zeitpunkt überprüft werden, ob die durch die Anwendung festgelegten Anforderungen erfüllt werden können. Diese frühe Analyse ermöglicht, eine ökonomische Lösung zu finden und entspricht der Idee von *AUTOSAR* (vgl. AUTOSAR Development Cooperation), einer Architektur für Automotive-Steuergeräte, die eine flexible Verschiebung von Funktionen zwischen Komponenten vorsieht. Anschließend begleitet die Simulation die Entwicklung der Konfiguration des Netzwerkes. In einer verteilten Entwicklung, wie sie im Automobilbau üblich ist, ist dieser Schritt besonders wertvoll, da er spätere Änderungen und damit Probleme bei den Integrationstests reduzieren oder verhindern kann.

[12] Proof-of-Concept (deutsch: Machbarkeitsnachweis) bezeichnet einen Schritt in der Entwicklung, bei dem die prinzipielle Durchführbarkeit bewiesen wird

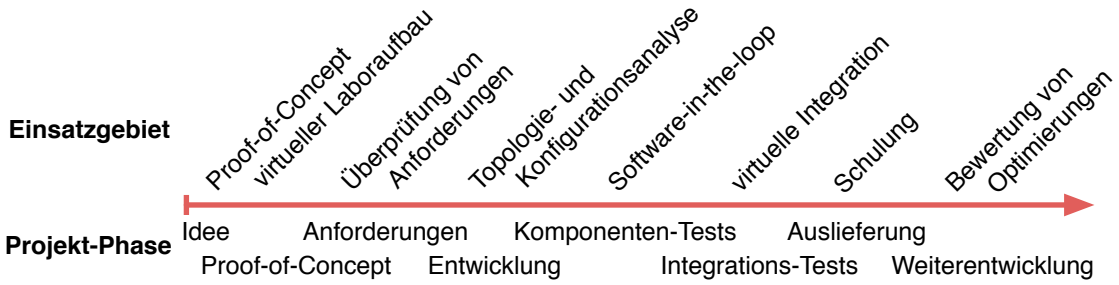

Abbildung 4.2: Einsatzszenarien des Evaluierungsprozesses in verschiedenen Projektphasen

Während der Entwicklung von Steuergeräten kann der Evaluierungsprozess für sogenannte *Software-in-the-Loop-Tests*[13] eingesetzt werden. Dabei werden die Programme und Algorithmen eines Steuergerätes in die Simulation eingebettet und dort mit Nachrichten stimuliert. Durch die präzise Abbildung der Vermittlungsinfrastruktur im Simulationsmodell werden dabei auch die Einflüsse der Datenübertragung auf die Ausführung berücksichtigt. Später kann ein virtueller Integrationstest in der Simulation durchgeführt werden, indem die Software mehrerer Steuergeräte eingebettet wird. Dadurch können, bereits Hardware-unabhängig, Probleme im Zusammenwirken mehrerer Komponenten aufgedeckt werden.
Über diese Anwendungsfälle hinaus ist der Einsatz des Evaluierungsprozesses auch in anderen Bereichen denkbar. Dadurch, dass das Protokollmodell in abstrakten Algorithmen definiert und frei von Hardware-spezifischem Programmcode ist, ist die Simulationsumgebung geeignet, um objektiv den Einfluss von Änderungen am Protokoll auf die Leistungsfähigkeit zu protokollieren. Ein weiterer Einsatzbereich ist die Schulung in der time-triggered Ethernet-Technologie. Durch die enge Anbindung an die TTEthernet-Toolchain kann die Konfiguration von Netzwerken auch ohne entsprechende Hardware ausprobiert und geübt werden. Dies betrifft insbesondere die Konfiguration von großen Topologien.
Abbildung 4.2 zeigt die möglichen Einsatzszenarien in unterschiedlichen Projektphasen.

4.1.2 Geplante Arbeitsabläufe

Die Basis für den Evaluierungsprozess ist eine Netzwerkkonfiguration im Format von TTTech (siehe Abschnitt 2.3.2 auf Seite 24). Diese mit den TTEthernet-Werkzeugen erstellte XML-Datei enthält bereits die wichtigsten Informationen für die Simulation. Diese XML-Netzwerkkonfiguration wird vom Evaluierungs-Framework eingelesen und in das Format der Simulationsumgebung übersetzt. An dieser Stelle kann die Simulationsumgebung bereits

[13] Bei der Methode Software-in-the-Loop (SiL) wird der entwickelte Programmcode auf dem Entwicklungsrechner zusammen mit dem simulierten Modell ausgeführt

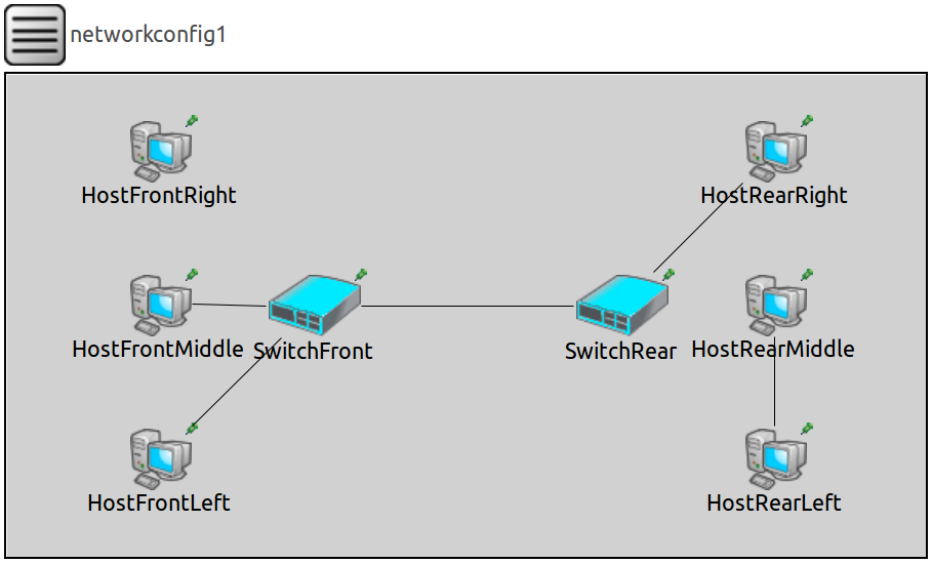

Abbildung 4.3: Fehlersuche durch grafische Visualisierung von Komponenten und Topologie — fehlende oder falsche Verbindungen sind leicht erkennbar

zum ersten Mal verwendet werden, um die Konfiguration zu überprüfen. Die Simulationsumgebung zeigt die Komponenten und die Topologie des in der Netzwerkkonfiguration definierten Netzwerkes. Fehlende Links oder falsche Verbindungen können durch die grafische Visualisierung schnell gefunden werden (siehe Abbildung 4.3).
Als weitere Eingabe in den Evaluierungsprozess wird eine Beschreibung der Datenflüsse und ihrer Anforderungen definiert. Diese Beschreibung ist notwendig, um in der Simulation für die spätere Anwendung charakteristische Datenflüsse zu generieren. Nur so ist ein gutes Simulationsergebnis, welches die späteren realen Eigenschaften wiedergibt, erzielbar.
Im nächsten Schritt werden die globalen Parameter der Simulation, beispielsweise die Simulationsdauer oder die zeitliche Auflösung, festgelegt. Anschließend wird die Simulation — je nach Anforderung unterschiedlich — durchgeführt. Um eine Konfiguration zu überprüfen, das Verhalten des Netzwerkes zu untersuchen oder den Einfluss von Konfigurationsparametern auf die Weiterleitung zu beobachten, bietet sich eine grafische Simulationsoberfläche an. Hier kann detailliert der Nachrichtenfluss beobachtet werden. Fehler, die im Switch erkannt werden, werden direkt an den Nutzer gemeldet. Für Simulationen über lange Zeiträume bietet sich dagegen eine Ausführung ohne grafische Oberfläche oder auch eine Verteilung über mehrere Rechner an. Dadurch kann die Simulationsdauer signifikant reduziert werden.
Im letzten Schritt generiert das System einen Bericht über die simulierten Metriken. Als Eingabe dient hier, neben den Ergebnissen der Simulation, die Beschreibung der Datenflüsse und Anforderungen. Erfüllt die Konfiguration alle Anforderungen, kann der Nutzer auf Basis

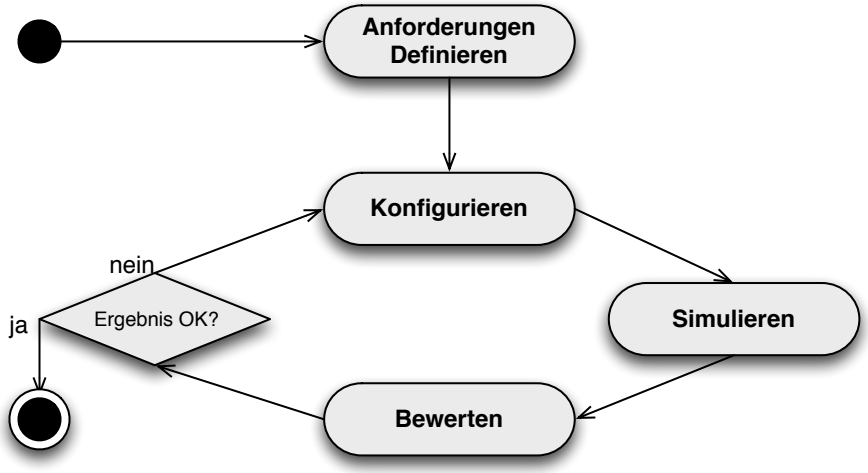

Abbildung 4.4: Zyklischer Arbeitsablauf im Evaluierungsprozess

des Berichtes entscheiden, ob die evaluierte Konfiguration in dieser Form eingesetzt werden soll. Werden Anforderungen nicht erfüllt, muss die Konfiguration angepasst und erneut evaluiert werden. Die Evaluierung unterstützt den Nutzer nun so lange in einem zyklischen Prozess, bis das gewünschte Ergebnis erzielt wird (siehe Abbildung 4.4). Dabei kann die Simulationsumgebung Hinweise auf Probleme geben. Das Know-how zur Konfiguration muss beim Nutzer vorhanden sein.

4.1.3 Grundlagen der Modellierung

Bei der Modellierung kann auf die umfangreichen Vorarbeiten zurückgegriffen werden. Der Modellierungsprozess orientiert sich an Jasperneite (2002) und Haas und Zorn (1995) (siehe Abschnitt 3.2 auf Seite 37 und Abbildung 3.2 auf Seite 38).
Abbildung 4.5 auf der nächsten Seite zeigt das Modellierungskonzept für den Evaluierungsprozess. Das konzeptionelle Modell wurde bereits für die analytische Evaluierung in den Vorarbeiten (vgl. Steinbach u. a., 2010) erstellt. Es kann für das Simulationsmodell verwendet werden. In die Validierung des Simulationsmodells gehen sowohl Messungen mit TTEthernet-Hardware (vgl. Bartols u. a., 2011) als auch die Ergebnisse aus der analytischen Evaluierung ein. Dies ermöglicht eine sehr genaue Validierung des Modells und reduziert das Risiko systematischer Fehler.
Die Abbildungsgenauigkeit des Modells wird bewusst reduziert, um von den Implementierungsdetails zu abstrahieren. Als Vorlage für das Modell dient die Spezifikation und nicht die Hardware-Implementierung der TTEthernet-Komponenten. Dies ist auch der Tatsache geschuldet, dass die Implementierung Closed-Source ist. Bei einer Modellierung anhand der

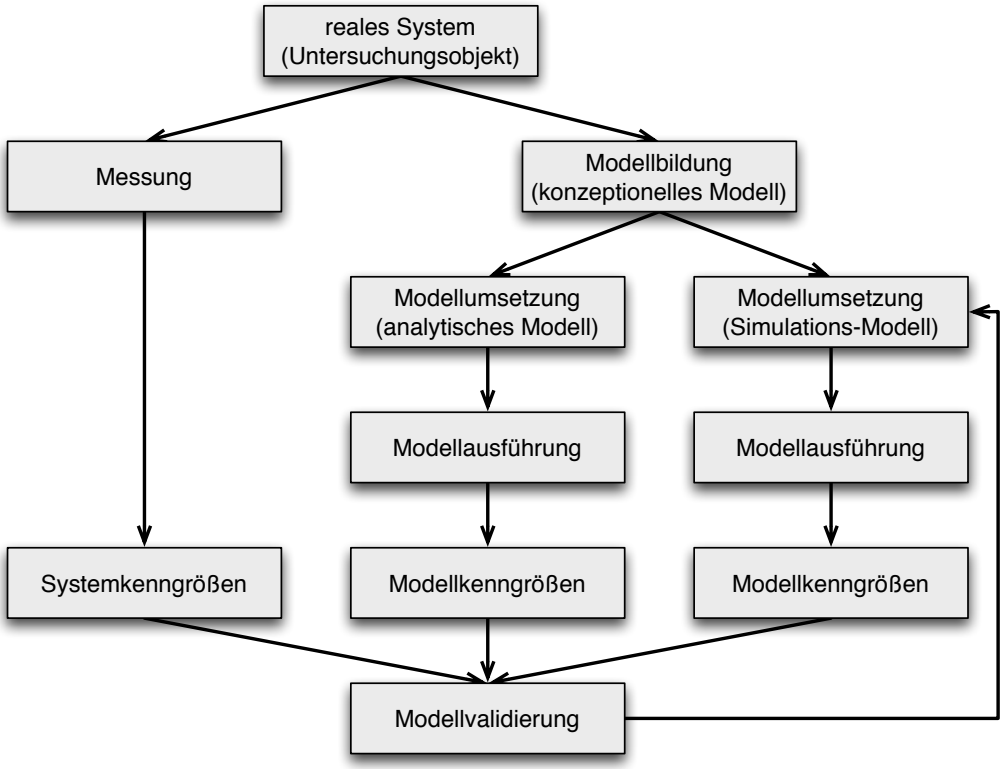

Abbildung 4.5: Konzept der Modellierung des Systems im Evaluierungsprozess

Spezifikation wird das Modell so umgesetzt, dass es den vorgegebenen Regeln entspricht. Damit abstrahiert es von der genauen Umsetzung in der Hardware.
Ein Vorteil dieser Abstraktionsebene ist, dass die Implementierung ohne die sonst bei Hardware-Entwürfen notwendigen Kompromisse umgesetzt werden kann. Der Nachteil ist eine mögliche Abweichung der Simulationsergebnisse von der späteren Implementierung. Solche Abweichungen lassen sich jedoch durch Korrekturfaktoren im Modell aufheben. Auf verschiedene Varianten vom Realsystem kann dann durch zielsystemspezifische Korrekturfaktoren eingegangen werden.

4.1.4 Generierung von charakteristischen Verkehrsflüssen

Insbesondere für die Analyse von event-triggered Datenverkehr sind charakteristische Verkehrsflüsse in der Simulation wichtig. Da die Verkehrsmodellierung in Kommunikationsnetzwerken für sich selbst bereits ein umfangreiches Thema ist, wird sie in dieser Arbeit nicht tiefer betrachtet. In der CoRE-Arbeitsgruppe hat sich Hermand Dieumo Kenfack bereits mit Verkehrsmodellierung beschäftigt (vgl. Dieumo Kenfack, 2010a). Für zukünftige Simulationen sind weitere Untersuchungen notwendig. Für die Fallstudien in dieser Arbeit werden einfache Verkehrsmodelle verwendet. Der time-triggered Datenverkehr benötigt ohnehin kein eigenes Modell, da er zyklisch gesendet wird. Im Folgenden werden einige Konzepte zur Verkehrsmodellierung in Kommunikationsnetzwerken vorgestellt, welche die Modellierung von event-triggered Datenverkehr ermöglichen. Alle Modelle basieren auf der Wahrscheinlichkeitsrechnung.

Poisson Modelle: Modelle, die auf der Poissonverteilung basieren, werden eingesetzt, um eintreffende Ereignisse innerhalb eines bestimmten Zeitintervalls zu modellieren. Beispiele für den Einsatz sind die Anzahl eintreffender Daten-Pakete innerhalb einer Minute. Die Poissonverteilung ist eine Modellierung mit wenigen Parametern und dadurch relativ einfach einzusetzen. Die Modellierung von komplexen Zusammenhängen ist mit ihr allerdings nicht möglich. Somit eignet sie sich nur begrenzt für die Modellierung von Ethernet-Verkehrsflüssen (vgl. Paxson und Floyd, 1995).

Selbstähnliche (self-similar) Modelle: Ein wichtiger Schritt in der Modellierung von Verkehrsflüssen ist die Entdeckung von selbstähnlichen Verkehrsmustern in paketvermittelnden Netzwerken (vgl. Schäfer, 2004, S. 68f). Heute haben sich selbstähnliche Verkehrsmodelle für die Darstellung von Verkehr in Datennetzen durchgesetzt. Durch eine Reihe von Analysen wurde die Selbstähnlichkeit von Verkehrsflüssen belegt (vgl. Erramilli u. a., 2002).
Die Besonderheit von selbstähnlichen Verkehrsflüssen ist, dass sich Muster darin, die innerhalb eines Zeitraums auftreten, in größeren Zeitabschnitten stetig wiederholen.

Markow Modelle: Markow-Verkehrsflussmodelle basieren auf der Markow-Eigenschaft (vgl. Behrends, 2000, S. 5ff), die besagt, dass ein Zustand eines Markow-Prozesses zu einem beliebigen Zeitpunkt t_x nur bis zu einem vorherigen Zustand t_n abhängig ist, nicht aber von den Zuständen vor t_n (vgl. Gaede, 1977, S. 86).
Anders als Modelle mit Poissonverteilung können Markow-Modelle daher die Struktur des Verkehrsstroms mit der Zeit fortentwickeln (vgl. Schäfer, 2004, S. 68).

4.2 Architektur und Komponenten

Damit der Evaluierungsprozess möglichst optimal in die Werkzeugkette von TTEthernet integriert werden kann, wird angestrebt, das Netzwerk in einer sogenannten Network-Configuration zu beschreiben. Die Network-Configuration ist ein XML-Format zur Beschreibung von TTEthernet-Konfigurationen (siehe Abschnitt 2.3.2 auf Seite 24). Die Simulation in OMNeT++ selbst wird in NED-Files beschrieben. NEDs enthalten die Struktur der Simulation. In einem automatisierten Prozess soll die NED aus der Network-Configuration generiert werden. Dadurch kann sichergestellt werden, dass die Simulation der gewünschten Hardwarekonfiguration entspricht. Darüber hinaus kann die Simulation in OMNeT++ optimal durch die Werkzeuge für TTEthernet (siehe Abschnitt 2.3.2 auf Seite 24) konfiguriert werden.
Im Zentrum des Evaluierungs-Frameworks steht das Simulationsmodell. Daran angeschlossen sind verschiedene Werkzeuge, die die Simulation vorbereiten oder auswerten. Abbildung 4.6 auf der nächsten Seite zeigt die Interaktion mit der TTEthernet-Toolchain (siehe Abschnitt 2.3.2 auf Seite 24) und die verschiedenen Zwischenschritte im Evaluierungsprozess. Durch die enge Anbindung an die TTEthernet-Werkzeuge kann sichergestellt werden, dass die Simulation der gewünschten Hardwarekonfiguration entspricht. Darüber hinaus erleichtern die Werkzeuge die Konfiguration der Simulation in OMNeT++.
Die Basis für den Evaluierungsprozess ist das Fahrzeugmodell. Es wird manuell oder werkzeuggestützt in eine Netzwerk-Konfiguration im Format von TTTech (siehe Abschnitt 2.3.2 auf Seite 24) übersetzt. Die Regeln für diesen Übersetzungsprozess sind bisher noch nicht erarbeitet und sind Teil zukünftiger Aktivitäten der CoRE-Arbeitsgruppe (vgl. CoRE RG, a). Die so erstellte XML-Datei enthält bereits die wichtigsten Informationen für die Simulation. Zudem kann sie direkt zur Konfiguration von TTEthernet-Hardware verwendet werden. Das XML-Dokument wird vom Evaluierungs-Framework eingelesen und zu einer OMNeT++-NED übersetzt. Dazu bedarf es eines Übersetzungswerkzeugs, das die Konfiguration aus dem TTTech-Konfigurationsmodell in das OMNeT++-NED-Konfigurationsmodell übersetzt (Modelltransformation). Das nach der Simulation vorliegende Ergebnis nutzt das OMNeT++-Format. Mit Hilfe eines Auswertungswerkzeugs wird es in einen vom Menschen lesbaren Analysebericht übersetzt.
Im folgenden werden die verschiedenen Komponenten und Werkzeuge des Evaluierungs-Frameworks vorgestellt.

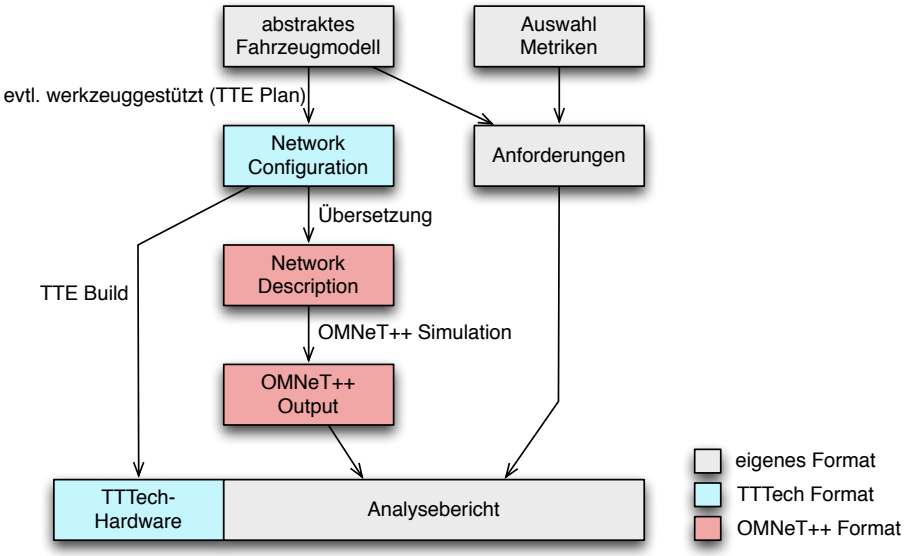

Abbildung 4.6: Evaluierungsprozess — Formate und Anbindung an externe Werkzeuge

4.2.1 Generierung der Simulationskonfiguration

Die OMNeT++-Simulationsumgebung wird durch NED-Dateien konfiguriert. Das Werkzeug zur NED-Generierung übernimmt die Übersetzung der TTEthernet-Network-Configuration zu NED-Dateien. Die Herausforderung bei der Umsetzung des Werkzeugs liegt im Einlesen der Konfiguration und dem Generieren des NED-Codes.
Um den Implementierungs-Aufwand möglichst gering zu halten, muss bei der Wahl der Programmiersprache für die Umsetzung die Verfügbarkeit von zusätzlichen Bibliotheken berücksichtigt werden. Da die Network-Configuration nicht auf den Standard-XML-Datentypen, sondern auf den Datentypen des Ecore-Metamodells (vgl. Eclipse Foundation, b) basiert, kann kein normaler XML-Parser für das Einlesen verwendet werden. Für das Einlesen von in XML persistent gehaltenen Ecore-Modellen, sogenanntem *XML Metadata Interchange (XMI)*, bietet sich die Programmiersprache Java an. Ecore ist Teil des Eclipse Modeling Framework (EMF) (vgl. The Eclipse Foundation), welches komplett in Java implementiert ist. Für die Ausgabe in das NED-Format kann ein Code-Generator entweder selbst implementiert oder auf Werkzeuge wie *Xpand* (vgl. Eclipse Foundation, c) zurückgegriffen werden. Xpand ist eine auf die Code-Generierung aus EMF-Modellen spezialisierte Sprache.
Eine Alternative zur Implementierung in Java und der Verwendung der Code-Generatoren aus dem EMF ist der Einsatz der *libnedxml* (vgl. OMNeT++ Community, b). Die libnedxml ist die NED-Parser- und NED-Generatorkomponente von OMNeT++ für den Einsatz in C++. Mit Hilfe der libnedxml kann die Topologie in einem Dokumenten-Baum angelegt und dann wahl-

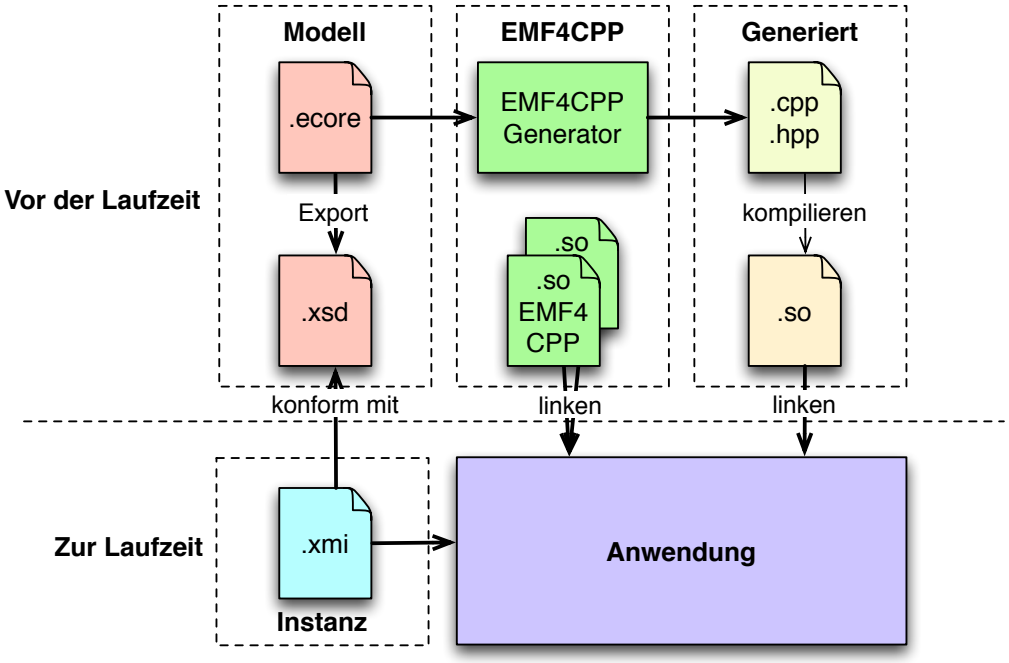

Abbildung 4.7: EMF4CPP Entwicklungsprozess — Quellcode-Generierung zur Entwicklungszeit, XML-Parsen zur Laufzeit

weise nach NED oder NED-XML serialisiert werden. Um die libnedxml einzusetzen, muss auch der Programmteil, der die XML Network-Configuration einliest in C++ umgesetzt werden. Das EMF bietet keine eigenen C++-Libraries für die Codegenerierung und das Parsen von Ecore-Modellen an. Das Projekt *EMF4CPP* (früher Ecore2CPP)(vgl. Senac und Sevilla) der Universität Murcia schließt diese Lücke.
EMF4CPP besteht aus zwei Teilen. Einem Sourcecode-Generator, um aus Ecore-Metamodellen C++-Code zu generieren, und den Bibliotheken für das Parsen und Serialisieren der Modelle im XMI-Format. Abbildung 4.7 zeigt den Entwicklungsprozess mit EMF4CPP.
Aus dem Ursprungsmodell werden mit Hilfe des Codegenerators C++-Klassen generiert und zu einer Bibliothek kompiliert. Die Endanwendung wird gegen diese und die EMF4CPP-Libraries gelinkt. XMI-Dokumente die konform zum Modell sind, können nun von der Anwendung zur Laufzeit eingelesen werden (vgl. Senac u. a., 2010).
Da für die Konfiguration der Komponenten in der Simulation ohnehin das Ecore-XMI in C++ eingelesen werden muss, wird entschieden EMF4CPP, für das Werkzeug zur NED-Generierung zu verwenden und die Entwicklung in C++ umzusetzen.

4.2.2 Konfiguration der Metriken und Anforderungen

Für die Definition der Anforderungen wird ein einfaches XML-Format verwendet. XML ist hier ein Kompromiss zwischen guter Unterstützung für das maschinelle Einlesen und guter Lesbarkeit für den Benutzer.
Die Anforderungsdefinition wird neben der Analyse des Simulationsergebnisses auch für die Parametrisierung der Simulationsumgebung verwendet. So wird auf Basis der Anforderungen entschieden, welche Größen während der Simulation aufgezeichnet werden, um die Metriken von Interesse zu evaluieren. Die dafür notwendigen Informationen werden aus der Anforderungsdatei extrahiert und in die OMNeT++-Konfiguration (omnetpp.ini-Datei) übertragen. Darüber hinaus werden in der Anforderungskonfiguration die Verkehrsstöme definiert, die nicht durch einen festen Zeitplan definiert sind (RC- und BE-Nachrichten).
Teilweise können Anforderungen bereits in der Network-Configuration definiert sein. Listing 4.1 zeigt die Definition von Anforderungen an einen VirtualLink für PCF. Der maximale Jitter ist mit 1000ns, die maximale Ende-zu-Ende Latenz mit 434656ns angegeben.

Listing 4.1: Anforderungen für Virtual-Links in der System-Specification der Network-Configuration

```
1  ...
2  <virtualLinks>
3      <virtualLink xsi:type="sys:PCFVirtualLink"
4                   destMacAddressCF="03:04:05:06"
5                   vlid="VL_PCF_switch_0_sm_IN_0815"
6                   jitter="1000ns"
7                   maxE2eLatency="434656ns"
8                   refSender="#//@devices/@device[name='switch_0']">
9          <refReceiver>#//@devices/@device[name='ecu_0']</refReceiver>
10         <refReceiver>#//@devices/@device[name='ecu_1']</refReceiver>
11         ...
12     </virtualLink>
13     ...
14 </virtualLinks>
15 ...
```

Diese Anforderungen sind nicht präzise genug für den Evaluierungsprozess, denn sie beinhalten nicht die Perspektive, aus der die Metrik betrachtet wird. So kann beispielsweise die Ende-zu-Ende Latenz eines Virtual-Links bei jedem Empfänger unterschiedlich sein. In der Network-Configuration können daher nur globale Anforderungen, die für alle Empfänger gelten, definiert werden. Die Werte, die in der Network-Configuration definiert sind, sollten dennoch in die Anforderungsdefinition übernommen werden, damit sie an einer eindeutigen Stelle für den Benutzer abrufbar sind. Dies kann auch automatisiert durchgeführt werden.

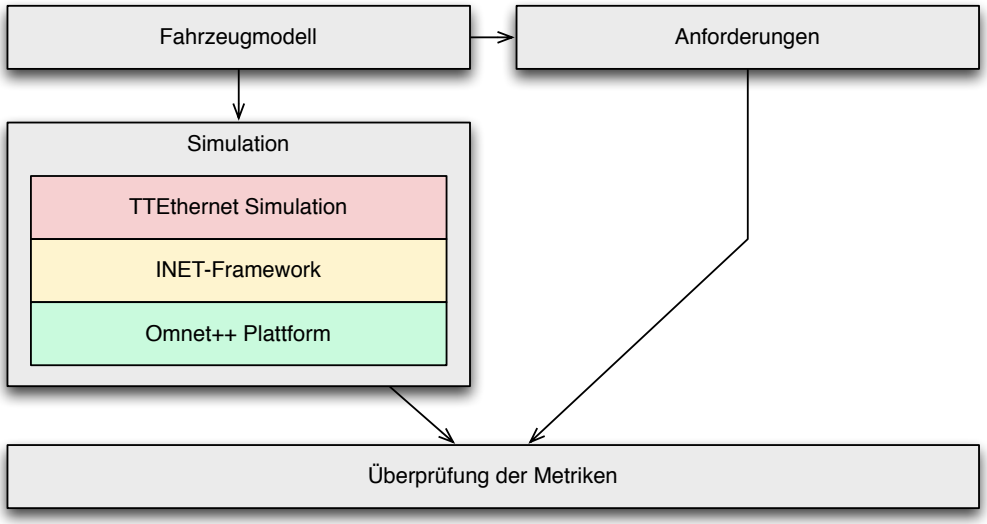

Abbildung 4.8: Elemente des Evaluierungsprozesses

Dabei sollte überprüft werden, ob Anforderungen der Network-Configuration und der XML-Anforderungsdefinition kollidieren.

4.2.3 Elemente des Simulationsmodells

Abbildung 4.8 zeigt einen Überblick über die relevanten Elemente des Simulationsmodells. Die Simulation setzt sich aus drei übereinander gelagerten Schichten zusammen. Die Basis bildet die OMNeT++-Simulationsumgebung (siehe Abschnitt 2.4.3 auf Seite 29).
Basierend auf OMNeT++ wird das INET-Framework (siehe Abschnitt 2.4.4 auf Seite 31) eingesetzt. Das INET-Framework ist besonders gut für die Verwendung in einer Simulation von Echtzeit-Ethernet geeignet, da es modular aufgebaut ist und intensiv Vererbung verwendet. Da TTEthernet auf Standard-Ethernet basiert, kann durch Vererbung die grundlegende Netzwerkfunktionalität vom Physikalischen- und Linklayer aus dem INET-Framework abgeleitet werden.
Auf Basis des INET-Frameworks wird die TTEthernet-Simulationsschicht implementiert. Sie erweitert das Modell um die Echtzeiteigenschaften des TTEthernet-Protokolls. Dabei muss besonders auf eine präzise Abbildung des Zeitverhaltens geachtet werden. Nur so lassen sich die Ergebnisse der Simulation zuverlässig auf ein echtes Szenario übertragen.
Abbildung 4.9 auf der nächsten Seite zeigt die wichtigsten Komponenten des TTEthernet-Modells und ihre Integration in die Schichten der Simulationsplattform. Dadurch, dass TTEthernet auf Standard-Ethernet basiert, müssen ausschließlich Komponenten des INET-Frameworks oberhalb der MAC-Ebene erweitert oder angepasst werden. Für das Modell

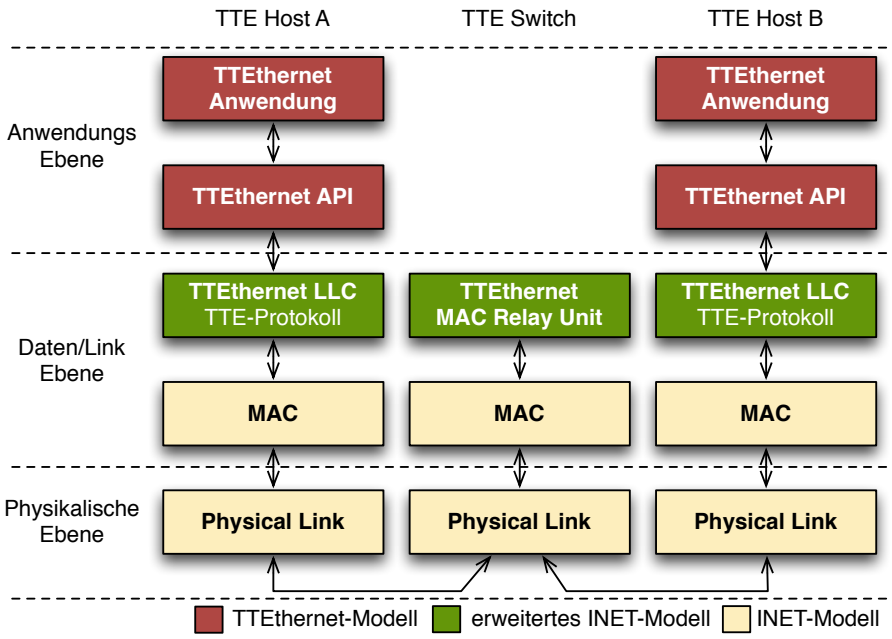

Abbildung 4.9: TTEthernet-Integration in das INET-Framework

des Ethernet-Hosts wird der Protokollstack in einer Erweiterung des Ethernet-Logical-Link-Control (LLC) implementiert. Darüber liegen das TTEthernet-API und die Anwendungen in neuen Modulen. Für den TTEthernet-Switch kann die Protokoll-Implementierung direkt als Erweiterung der MACRelayUnit aus dem INET-Framework umgesetzt werden.

Uhrenmodell

Die global synchronisierte Zeit und das Scheduling von jeder Komponente sind die wichtigsten Unterschiede von TTEthernet gegenüber Standard-Ethernet. Daher benötigt jeder Teilnehmer des Netzwerks ein eigenes Uhrenmodell, das über ein fehlergesichertes Synchronisationsprotokoll mit den Uhren der anderen Teilnehmer synchronisiert wird (siehe Abschnitt 2.3.1 auf Seite 20). Die kleinste Einheit der Uhr in TTEthernet ist ein *Tick*. Dabei ist die Zeit des Ticks konfigurierbar.
Bei der Modellierung der Uhr muss beachtet werden, dass jeder Oszillator eine gewisse Ungenauigkeit besitzt, welche als *Clock-Drift* bezeichnet wird (vgl. Sullivan u. a., 1990). Diese Clock-Drift hat signifikanten Einfluss auf das Verhalten des Protokolls und muss daher überlegt in das Uhrenmodell integriert werden. Aus offensichtlichen Gründen kann die Clock-Drift nicht simuliert werden, indem jeder Tick als ein separates Event ausgeführt wird. Dies würde die Simulation über die Maßen verlangsamen.

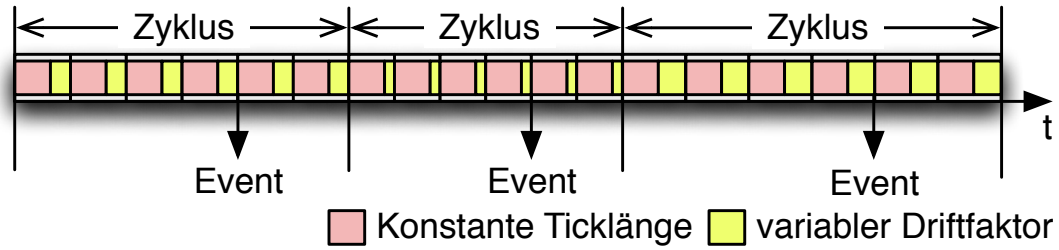

Abbildung 4.10: Berechnung der Uhr mit Clock-Drift — konstanter- und Clock-Drift-Anteil

Im vorgeschlagenen Konzept werden mehrere Ticks simultan in einem Schritt simuliert. Dabei wird ein *Clock-Drift-Faktor* eingeführt, welcher die Ungenauigkeit der Uhr nachbildet. Dieser Faktor wird für ein konfigurierbares Intervall als konstant angenommen. Dadurch entfallen diverse Simulationsschritte. Die Gleichung 4.1 zeigt die Formel für die Berechnung des aktuellen Wertes der Uhr.

$$t' = t + \delta \cdot (\Delta t_{Tick} + \Delta t_{Drift}) \tag{4.1}$$

Dabei ist t' die Zeit für das nächste geschedulte Event, t ist die aktuelle Simulationszeit, δ ist die Anzahl von Ticks im Schedule für das nächste Event, Δt_{Tick} ist die konfigurierbare Zeit für einen Tick und Δt_{Drift} ist die aktuelle mittlere Clock-Drift. Dieses Modell der Uhren (siehe Abbildung 4.10) ist eine zulässige und ausreichend genaue Vereinfachung des Verhaltens von echten Uhren, da während eines Zyklus die Varianz der mittleren Clock-Drift in echten Uhren nur gering ist (vgl. Sullivan u. a., 1990).
Während die Clock-Drift Uhren ungenau werden lässt, sorgen die zyklischen Synchronisationsnachrichten dafür, dass die Uhr stets auf einen bekannten Punkt im Zyklus zurückgesetzt wird.

Switchmodell

TTEthernet benötigt eine Switch-Erweiterung, welche das time-triggered Protokoll verarbeitet (siehe Abschnitt 2.3 auf Seite 19). Der TTEthernet-Switch kombiniert ein Standard-Switched-Ethernet-Gerät für die Weiterleitung von best-effort Daten mit einem Modul, welches das Synchronisierungsprotokoll und das Weiterleiten von zeitkritischen Daten (time-triggered und rate-constrained) unterstützt. Dieses Modul enthält weiterhin die Schedules und die lokale Uhr.

Für die Implementierung des TTEthernet-Switches wird das MACRelayUnit Interface aus dem INET-Framework verwendet. Die Logik für die Weiterleitung des zeitkritischen Datenverkehrs verwendet das beschriebene Uhrenmodell (siehe Abschnitt 4.2.3 auf Seite 57).
Wie durch die TTEthernet-Spezifikation vorgegeben, werden ankommende Pakete anhand ihrer Zieladresse klassifiziert und dann, basierend auf ihrer Nachrichtenart, weitergeleitet oder bearbeitet:

- Synchronisationsnachrichten werden überprüft und an das Synchronisationsmodul des Gerätes weitergeleitet.
- Time-triggered Nachrichten werden in einem Puffer bis zu ihrem geplanten Weiterleitungszeitpunkt zwischengespeichert und dann abgesendet.
- Rate-constrained Nachrichten werden so schnell wie möglich weitergeleitet (jedoch mit geringerer Priorität als time-triggered Nachrichten, um diese nicht zu verzögern).
- Best-effort Nachrichten werden in der übrigen Zeit weitergeleitet.

Direkt nach dem Empfang eines Ethernet-Frames überprüft der TTEthernet-Switch die Nachricht auf Konformität mit den vorkonfigurierten Regeln. Für zeitkritischen Datenverkehr sind die Konformitätsüberprüfungen in der Critical-Traffic-Conformance-Tabelle (CTC) abgelegt. Der Switch kann unter anderem überprüfen, ob die Nachricht auf dem richtigen Port und zum korrekten Zeitpunkt angekommen ist. Während für rate-constrained Nachrichten die Einhaltung des Bandwidth Allocation Gap (BAG) überprüft wird, wird für time-triggered Daten geprüft, ob die Nachricht zur geplanten Zeit eintraf (siehe Abbildung 4.11 auf der nächsten Seite). Diese Analyse ist wichtig, um das System vor der Störung durch fehlerhafte oder bösartige Sender zu schützen. So können Nachrichten von Geräten, die vorgeben, time-triggered Sender zu sein, verworfen werden, sofern sie an einem falschen Port eintreffen. Best-effort Frames sind von der Überprüfung ausgeschlossen. Sie dürfen zu jedem Zeitpunkt des Schedules an jedem Port des TTEthernet-Switches eintreffen. Damit können auch Geräte in das Netzwerk integriert werden, die das time-triggered Protokoll nicht unterstützen.

Hostmodell

Für die Simulation von TTEthernet-Endsystemen muss der Standard-Ethernet-Protokollstack um die time-triggered Funktionalität erweitert werden. Dafür muss das Nachrichten-Scheduling und der zugehörige Scheduler, die Klassifizierung von eingehenden Nachrichten und das Synchronisationsmodul implementiert werden. Das Nachrichten-Scheduling legt fest, wann eine time-triggered Nachricht gesendet oder empfangen wird (siehe Abschnitt 2.3.1 auf Seite 20). Es wird offline konfiguriert. Für unterschiedliche Events, darunter das Versenden oder Empfangen von Nachrichten, aktiviert der Scheduler sogenannte *Tasks*.

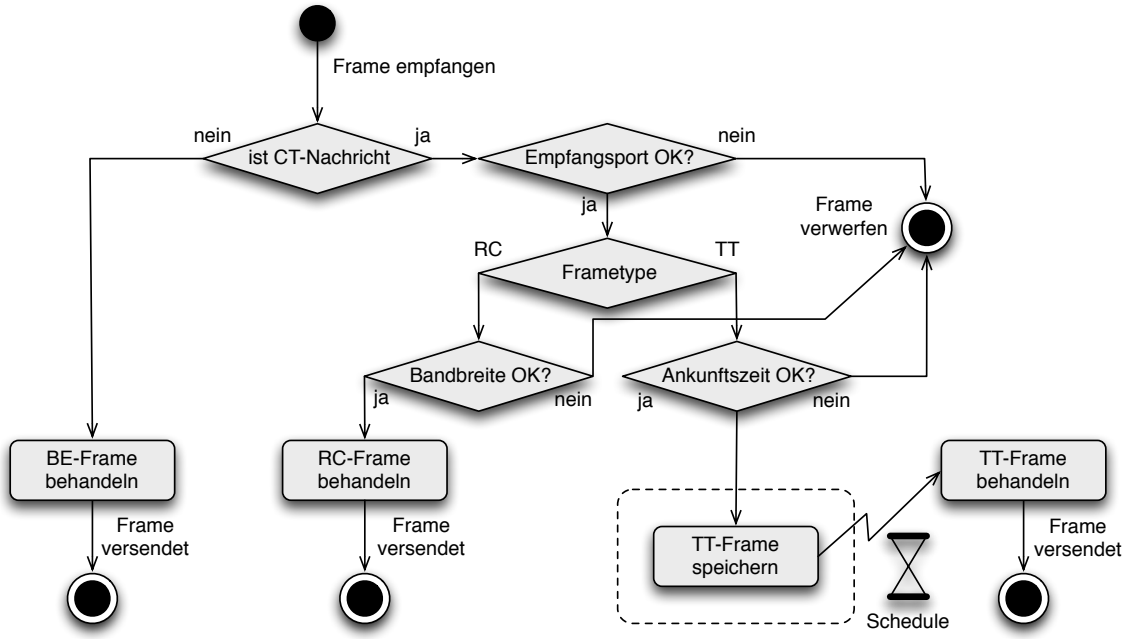

Abbildung 4.11: Vereinfachtes Aktivitätsdiagramm der Verarbeitung im Switch

Für die direkte Integration von Anwendungen in das Simulationsmodell wird die TTEthernet-API für das Simulationsmodell implementiert. Über Code-Integration können darüber Anwendungen, die gegen das API programmiert wurden, direkt in der Simulation verwendet werden. Dabei verwendet das Hostmodell die gleichen Datenstrukturen wie die reale Implementierung. Alle Nachrichten werden in Buffern zwischen Anwendungs- und Protokoll-Layer ausgetauscht.

Konfiguration der Simulationskomponenten

Ein wesentliches Ziel bei der Umsetzung ist die enge Integration des Simulationsprozesses in die TTEthernet-Toolchain von TTTech (siehe Abschnitt 2.3.2). Um dieses Ziel zu erreichen, wird die XML-Network-Configuration verwendet, um die Komponenten in der Simulation zu konfigurieren.
Die Herausforderung beim Einlesen der Konfiguration liegt im Format. Die XMI-Dateien basieren auf den Ecore-Datentypen. Daher kann kein normaler C++ XML-Parser verwendet werden. Auch die Referenzen werden in einem speziellen EMF-konformen Format, sogenannten Fragment-based-URIs, gespeichert. Diese Form der Referenzen weicht von der normalen Syntax in XML-Dokumenten, die in der XML Path Language (Xpath) (vgl. World Wide Web Consortium, 1999) definiert sind, ab (siehe Listing 4.2).

Listing 4.2: Syntax von Referenzen in EMF Fragment-based-URIs und Xpath

```
1 <!-- EMF Fragment-based URI -->
2 <Student studiengang="#//@departments/@department.0/@studiengang[name='
    Informatik']"/>
3 <!-- Xpath -->
4 <Student studiengang="//departments/department[1]/studiengang[@name='
    Informatik']"/>
```

Da das EMF auf Java basiert, stehen hierfür Parser zur Verfügung. Da die Simulation aber in C++ implementiert ist, ist eine Java-Lösung nicht möglich. Grundsätzlich ist es möglich, aus C++ auf Java zuzugreifen — hierzu dient das Java Native Interface (JNI) — jedoch würde dies den Code nicht nur extrem verkomplizieren, sondern auch stark verlangsamen.
Wie bei der NED-Generierung (siehe Abschnitt 4.2.1 auf Seite 53) wird daher auch für die Konfiguration das EMF4CPP-Framework eingesetzt. Dadurch kann das Einlesen der Konfiguration direkt in der C++ Simulationsumgebung erfolgen. Der Einsatz nur eines Parsers für alle Werkzeuge des Evaluierungsprozesses reduziert zudem die Komplexität des Gesamtprojektes.

4.2.4 Automatisierte Auswertung der Simulationsergebnisse

Die Auswertung verwendet die Ausgabe der Simulationsergebnisse. OMNeT++ liefert verschiedene Ausgabeformate (vgl. Varga, 2010):

- Output Vectors: Die Ausgabevektoren sind Datenserien, die von Modulen oder den Verbindungen zwischen den Modulen aufgezeichnet werden. Aus den Ausgabevektoren kann beispielsweise die Ende-zu-Ende Latenz eines Paketes ausgelesen werden.
- Output Scalars: Die Ausgabeskalare summieren Ereignisse und eignen sich für statistische Zusammenfassungen wie der Anzahl verlorener Pakete.
- Eventlog: Das Eventlog zeichnet zu einem Modul jedes eintreffende Event auf. Mit dem Eventlog lässt sich der Ablauf zu jedem Zeitpunkt der Simulation nachvollziehen.

Um die Größe der Ausgabedateien zu reduzieren, wird vor der Simulation konfiguriert, welche Elemente aufgezeichnet werden. Hier müssen insbesondere die Elemente konfiguriert sein, die für die Auswertung der Anforderungen (siehe Abschnitt 4.2.2 auf Seite 55) wichtig sind.
Als Ausgabeformat wird erneut XML gewählt, da es gut lesbar ist und darüber hinaus mit der Extensible Stylesheet Language Transformation (XSLT) (vgl. World Wide Web Consortium, 2007) in diverse Formate umgewandelt werden kann. Beispiele für solche Zielformate sind einfache Textdateien oder Berichte in HTML-Format oder PDF. Über eine XSL-Transformation in das TeX-Format (vgl. Knuth, 1998) können dabei ansprechende Layouts mit geringem Aufwand realisiert werden (vgl. Hufflen, 2009; Parashchenko, 2010).

Kapitel 5

Umsetzung des Evaluierungsprozesses

Dieses Kapitel zeigt die Umsetzung des Konzeptes in einen Evaluierungsprozess. Es werden Implementierungs-Details der Simulationskonfiguration (Abschnitt 5.1), des Simulationsmodells (5.2) und des Auswertungswerkzeugs (5.3) gezeigt.

5.1 Werkzeug zur Generierung der Network-Description

Für die automatisierte NED-Generierung wird das TTE-Config2NED-Werkzeug entwickelt. Das Programm wird in C++ auf Basis von EMF4CPP (vgl. Senac und Sevilla) und der libnedxml (vgl. OMNeT++ Community, b) implementiert (siehe Abschnitt 4.2.1 auf Seite 53).
Das Werkzeug parst mit Hilfe der EMF4CPP-Bibliothek die Konfiguration, bildet die Konfigurations-Objekte auf NED-Objekte ab und serialisiert diese anschließend zu einer NED-Datei, welche in OMNeT++ simuliert werden kann.
Abbildung 5.1 auf der nächsten Seite zeigt die Implementierung der Transformation. Die XMI-Dateien, die konform zum XML-Schema des Ecore-Modells der Network-Configuration definiert wurden, werden durch das Werkzeug in einen Objektbaum eingelesen. Dieser Ecore-Document-Tree basiert auf den durch EMF4CPP erzeugten C++-Klassen. Ein internes Mapping auf einen auf libnedxml basierenden NED-Document-Tree (siehe Abbildung 5.1 auf der nächsten Seite) stellt die Verknüpfung zwischen den Objekten der Network-Configuration und der OMNeT++-NED her. Dieser NED-Document-Tree kann nun mit der libnedxml-Bibliothek zu NED- oder NEDXML-Dateien serialisiert werden.
Die Network-Configuration enthält erheblich mehr Informationen als die NED-Datei. Nicht alle Elemente können oder müssen in die NED abgebildet werden. Da die NED durch den grafischen Editor gut geeignet ist, um Fehler in der Network-Configuration zu finden, werden dennoch möglichst viele Informationen übernommen. So behalten beispielsweise die Geräte dieselben Bezeichnungen, und Elemente wie Links werden in der NED mit Kommentaren versehen, die Informationen der Konfiguration wie beispielsweise den Link-Namen oder die

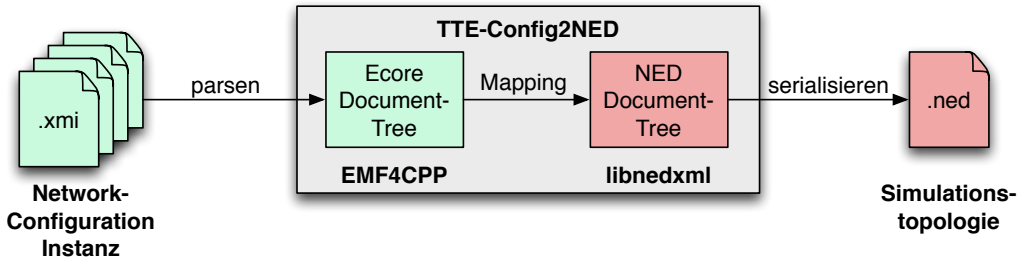

Abbildung 5.1: Mapping zwischen Network-Configuration und NED

Namen der verbundenen Ports enthalten. Auf diese Weise kann ein Fehler unmittelbar auf das entsprechende Element der Konfiguration zurückgeführt werden.
Bei der Implementierung zeigt sich, dass EMF4CPP grundsätzlich das richtige Werkzeug für das Einlesen der Konfiguration ist, jedoch an vielen Stellen die Qualität der verwendeten Version (0.0.2) noch nicht ausreicht. Hier kann durch Beiträge zum EMF4CPP-Projekt die Stabilität des Werkzeugs weiter erhöht werden.
Das Listing C.1 auf Seite 104 zeigt eine System-Specification und die dazu gehörige generierte NED-Datei (Listing C.2 auf Seite 105).

5.2 Komponenten des Simulationsmodells

Bei der Implementierung der Komponenten für die Simulation werden die Konzepte des INET-Frameworks so genau wie möglich übernommen. Wenn möglich wird Vererbung verwendet, um existierende Strukturen des Frameworks zu erweitern. Neue Module werden nur dort eingeführt, wo Erweiterungen die speziellen Anforderungen der time-triggered Kommunikation nicht erfüllen können.

5.2.1 Der TTEthernet-Switch

Die Funktionalität des TTEthernet-Switches kann in zwei logische Weiterleitungsmodule (*RelayUnits*) unterteilt werden (siehe Abschnitt 4.2.3 auf Seite 58). Die eine RelayUnit ist für die Weiterleitung von best-effort Nachrichten verantwortlich, die andere leitet den zeitkritischen Datenverkehr (rate-constrained und time-triggered) weiter. Beide Module teilen sich dieselben physikalischen Anschlüsse (Ports). Spezielle Module zwischen diesen RelayUnits und dem MAC-Layer überprüfen die Einhaltung der Prioritäten und der Regeln für den Medienzugriff.
Für das Simulationsmodell wird diese Sichtweise auf eine TTEthernet-Komponente aufgegriffen, da es die Benutzung der INET-Implementierung für den best-effort Teil des

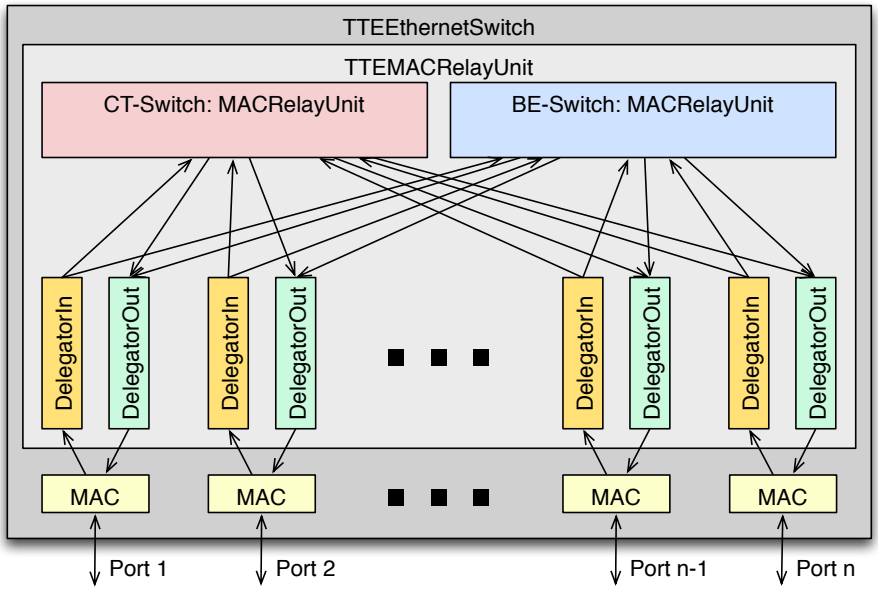

Abbildung 5.2: Switch-Design mit zwei eingebetteten RelayUnits und zwei Delegatoren pro Port

TTEthernet-Switches erlaubt. Dies reduziert den Implementierungsaufwand und erlaubt von späteren Erweiterungen im INET-Switch direkt zu profitieren.
Wie alle Ethernet-Switche im INET-Framework basiert auch der TTEthernet-Switch auf dem MACRelayUnit-Interface. Die Implementierung — *TTEMACRelayUnit* — enthält selbst wiederum zwei MACRelayUnits, eine für die Weiterleitung von best-effort Daten und eine für die zeitkritischen Nachrichten (siehe Abbildung 5.2).
Um beiden RelayUnits den Zugriff auf die physikalischen Ports des Switches zu ermöglichen, werden die Module des MAC-Layers über jeweils zwei *Delegator-Module*[14] mit den RelayUnits verbunden. Der *Input-Delegator* leitet die ankommenden Nachrichten vom EtherMAC-Modul zur richtigen RelayUnit weiter. Er überprüft, ob der eingehende Frame zeitkritische Daten enthält, indem er die Zieladresse aus dem Ethernet-Header liest. Dann wird der Frame zur verantwortlichen RelayUnit weitergeleitet. Der *Output-Delegator* empfängt Frames von den Ausgängen der eingebetteten RelayUnits und leitet sie unter Einhaltung der Prioritäten (siehe Abschnitt 2.3.1 auf Seite 20) an das EtherMAC-Modul weiter (siehe Abbildung 5.2).
In TTEthernet darf eine time-triggered Nachricht nicht durch einen anderen Frame, der den ausgehenden Port belegt, verzögert werden, es sei denn, dies ist explizit in der Konfiguration erlaubt. Daher muss der Output-Delegator sicherstellen, dass alle Nachrichten mit der rich-

[14]Das Delegation-Pattern ist ein Design-Element aus dem Software Engineering, bei dem ein Objekt eine Aufgabe an ein Helper-Objekt delegiert, anstatt sie selbst auszuführen

tigen Priorität und zum richtigen Zeitpunkt abgesendet werden. Bevor eine rate-constrained oder best-effort Nachricht abgesendet wird, überprüft der Delegator, ob die Nachricht in die Leerlauflücke bis zur nächsten time-triggered Nachricht passt. Damit kann sichergestellt werden, dass das MAC-Layer nicht durch Nachrichten blockiert ist, wenn eine time-triggered Nachricht abgeschickt werden soll. Best-effort und rate-constrained Nachrichten, die nicht in diese Lücke passen, werden in separaten Buffern im Delegator gespeichert. Sobald die time-triggered Nachricht versendet wurde, werden die Nachrichten, die in diesen Buffern gespeichert wurden, weitergeleitet.
Abbildung 5.3 auf der nächsten Seite zeigt einen exemplarischen Ablauf beim Versand von Nachrichten unterschiedlicher Traffic-Klassen. Das Sequenzdiagramm zeigt, wie der Delegator vor dem Senden von event-triggered Nachrichten (rate-constrained oder best-effort) überprüft, ob das Interface frei zu halten ist.
Ein weiteres Problem für die Weiterleitung von time-triggered Nachrichten ist der Buffer des INET-EtherMAC-Moduls. Trotz der Überprüfung des Delegators, ob niedrig priorisierte Frames eine time-triggered Nachricht verzögern würden, könnten Frames im Buffer der MAC-Ebene gespeichert sein, welche das Versenden der Nachricht verzögern würden (siehe Abbildung 5.4 auf Seite 67). Dieses Problem wurde gelöst, indem die Nutzung des Buffers im EtherMAC-Modul durch den Delegator verhindert wird. Der Delegator sendet den nächsten Frame erst dann an das MAC-Layer, wenn der Buffer dort komplett leer ist. Um dieses Verhalten umzusetzen, wird eine Nachricht vom EtherMAC-Modul zum Delegator benötigt, die das vollständige Absenden eines Frames — also eine freie Datenleitung — meldet. Die Standardimplementierung des EtherMAC-Moduls erlaubt solche Nachrichten nicht. Es gibt jedoch zwei Möglichkeiten, diese Funktionalität umzusetzen. Die erste Möglichkeit ist eine Erweiterung des EtherMAC-Moduls um die Funktionalität einer *Idle-Nachricht*. Die zweite Möglichkeit ist die Nutzung des *INET-Notification-Boards*.
Das Notification-Board ist ein Publisher-Subscriber-System. Jedes Modul darf Events an das Board senden oder sich für den Empfang von Events anmelden. Die Events werden durch Callback-Methoden ausgeliefert. Das EtherMAC-Modul sendet verschiedene Events zu seinem Zustand an das Notification-Board, darunter auch das *NF_PP_TX_END-Event*, welches das Ende der Auslieferung eines Frames angibt. Der Delegator empfängt das Event, überprüft, ob der Buffer auf MAC-Ebene komplett leer ist, und sendet dann, unter Einhaltung der Prioritäten, den nächsten Frame. Der große Vorteil dieser Implementierung ist, dass das INET-EtherMAC-Modul ohne Änderung weiter genutzt werden kann. Abbildung B.1 (Anhang auf Seite 102) zeigt das Sequenzdiagramm 5.3 auf der nächsten Seite unter Verwendung des Notification-Boards.
Abbildung 5.5 auf Seite 67 zeigt einen Screenshot der Switch-Implementierung in OMNeT++, auf dem die drei Ebenen des Gerätes zu sehen sind. Das Basis-Layer enthält neben den EtherMAC-Modulen das Notification-Board und die RelayUnit, welche den TTEthernet-Switch implementiert. Diese RelayUnit besteht wiederum aus den eingebetteten RelayUnits für kritischen und best-effort Datenverkehr und den Delegatoren, die sich um die Zuweisung

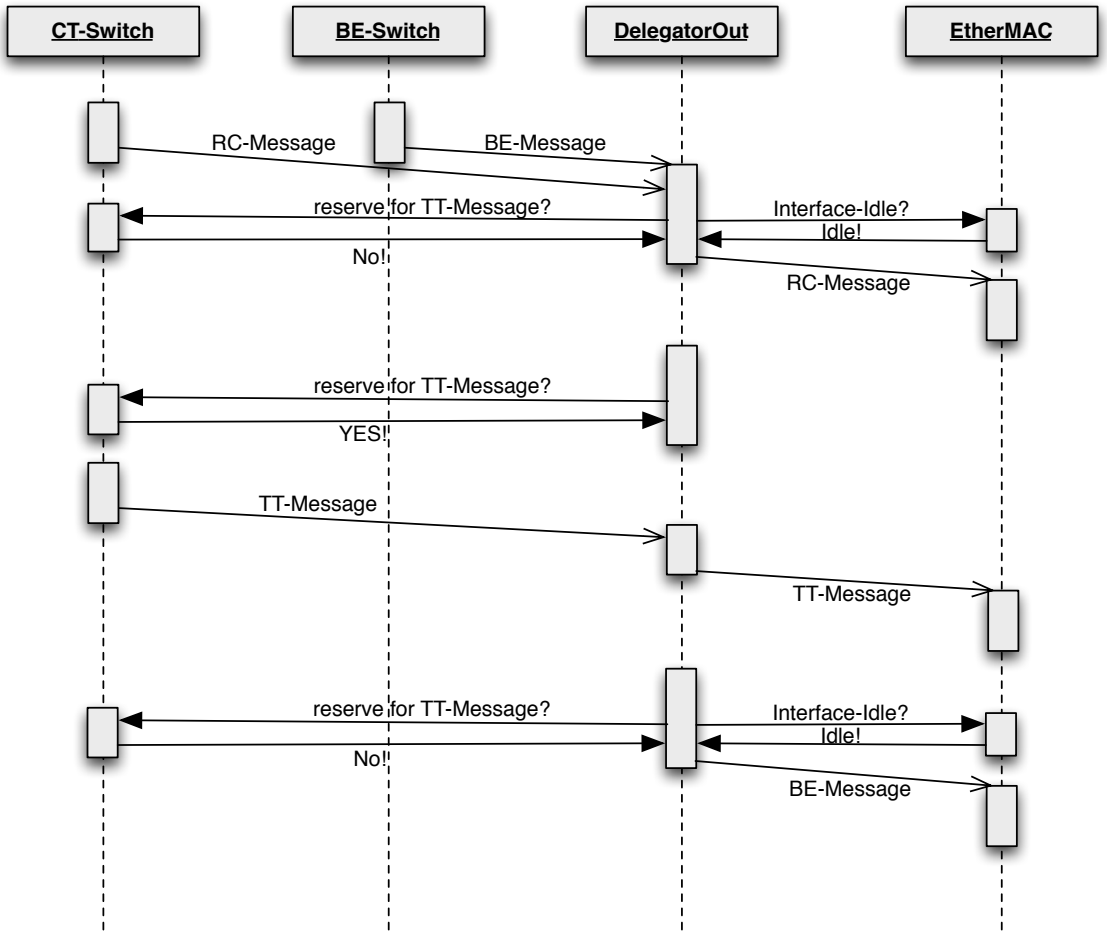

Abbildung 5.3: Exemplarische Nachrichtensequenz zwischen den Modulen beim Nachrichtenversand

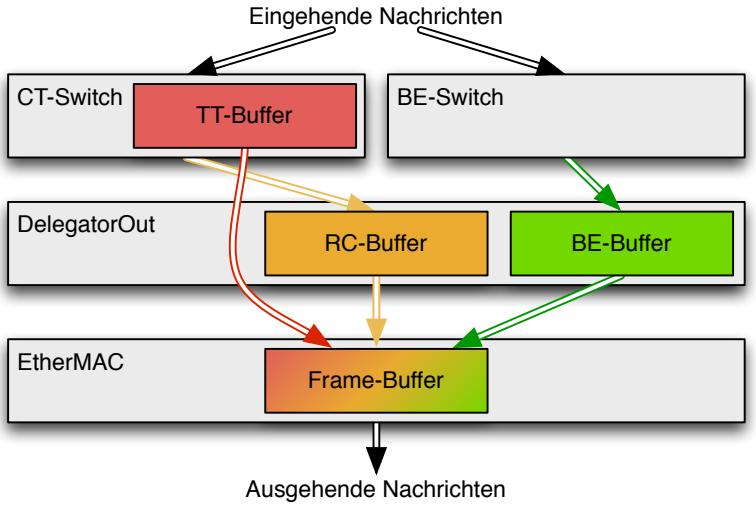

Abbildung 5.4: Verteilung der Buffer verschiedener Traffic-Klassen auf die Module — gemeinsamer Buffer im EtherMAC-Modul

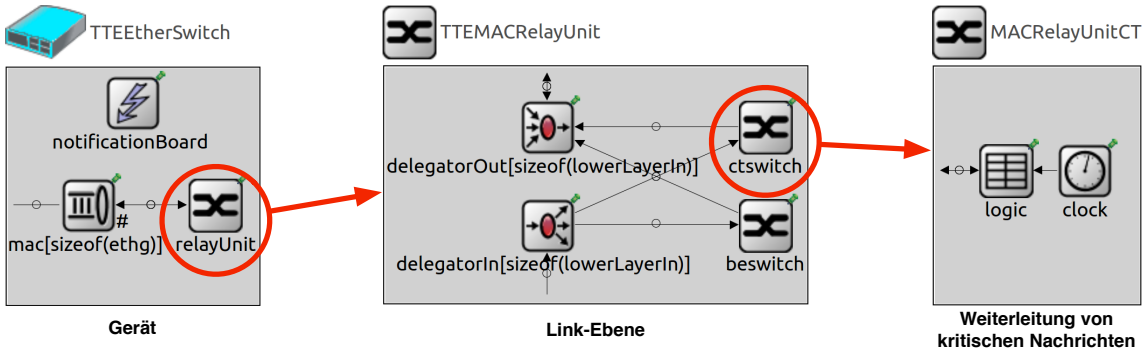

Abbildung 5.5: Überblick über die verschiedenen Ebenen der Implementierung des TTEthernet-Switchmodells

von Nachrichten kümmern. Die RelayUnit für den kritischen Datenverkehr enthält die Logik für die Weiterleitung und die lokale Uhr.
Untersuchungen zur Performance zeigen, dass die Implementierung mit dem Notification-Board nicht gut skaliert, weil unnötig viele Events zu allen Delegator-Modulen weitergeleitet werden. Daher wurde die Simulations-Plattform in einer späteren Version auf ein zugeschnittenes EtherMAC-Modul, welches direkt Idle-Nachrichten an den Delegator sendet, umgestellt.

5.2.2 Der TTEthernet-Host

Das Modul des TTEthernet-Hosts wurde im Wesentlichen durch den Masterstudenten Hermand Dieumo Kenfack erarbeitet. Daher werden hier nur die wichtigsten Teile des Moduls und die durchgeführten Erweiterungen beschrieben. Details können dem Projektbericht von Hermand Dieumo Kenfack (vgl. Dieumo Kenfack, 2010b) entnommen werden.
Um den TTEthernet-Protokollstack in die Implementierung des Hosts aus dem INET-Framework zu integrieren, müssen zwei wesentliche Erweiterungen am INET-Ethernet-Stack vorgenommen werden (siehe Abbildung 4.9 auf Seite 57). Diese betreffen das INET LLC und die Erweiterung des Simulations-Hosts um die TTEthernet-API (vgl. TTTech Computertechnik AG, 2008).

TTEthernet Logical-Link-Control Ebene

Das neu eingeführte TTEthernet-LLC erweitert die LLC-Ebene des INET-Frameworks um die TTEthernet-Funktionalität (siehe Abbildung 5.6 auf der nächsten Seite). Es ist verbunden mit einem Scheduler, der die time-triggered Kommunikation steuert. Für jede time-triggered Nachricht wird im LLC ein Transmission- oder Receive-Buffer reserviert. Die Addressierung erfolgt dabei über die CT-ID der Nachricht. Der Scheduler koordiniert die Übertragung der Nachrichten. Er entscheidet, wann eine Nachricht im TTEthernet-LLC aus dem Buffer gelesen und versandt wird. Das folgende Beispiel (siehe auch Sequenzdiagramm B.2 auf Seite 103 im Anhang) zeigt den Ablauf für eine ausgehende time-triggered Nachricht: Zum geplanten Zeitpunkt des Versandes sendet der Scheduler ein *TTOutgoing-Traffic-Event* mit der entsprechenden CT-ID der Nachricht an das LLC-Layer. Dort wird die Nachricht aus dem Buffer gelesen, in einen INET-Ethernet-Frame verpackt und anschließend an das MAC-Layer weitergeleitet.
Zusätzlich zum Nachrichtenversand werden auch die time-triggered Anwendungen, also die Anwendungen, die zeitgesteuert kommunizieren, durch den Scheduler synchronisiert. Solche durch Events ausgelösten Verarbeitungen in Anwendungen werden im TTEthernet-Schedule als *Tasks* bezeichnet. Durch Tasks können Anwendungen die zu sendenden Daten sehr kurz vor dem zeitgesteuerten Versand berechnen.

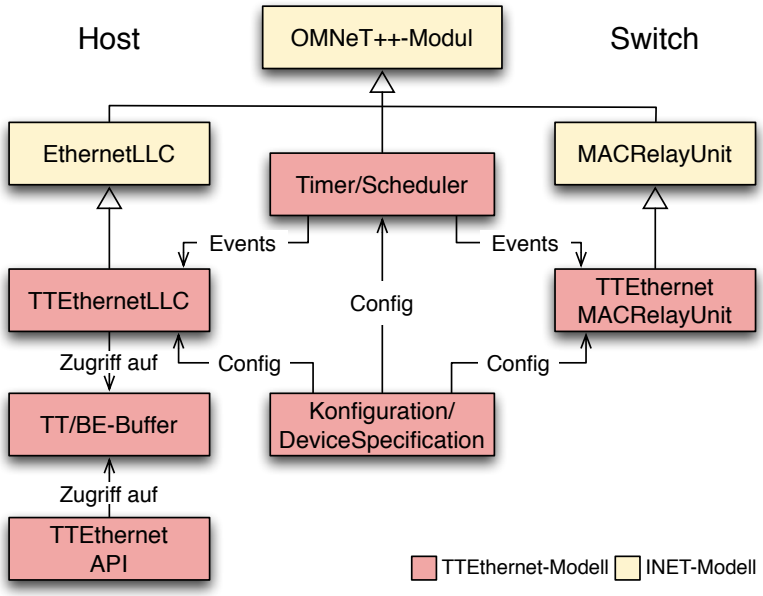

Abbildung 5.6: Objektmodell des TTEthernet-Protokoll-Layers

TTEthernet-API

Das TTEthernet-API (vgl. TTTech Computertechnik AG, 2008) wurde von TTTech definiert und liefert die Schnittstelle, um Nachrichten auf Anwendungsebene zu versenden und zu empfangen. Es verbindet die Anwendung mit dem LLC (siehe Abbildung 4.9 auf Seite 57). Das TTEthernet-API arbeitet mit Buffern. Die Anwendung allokiert Buffer zum Senden von Nachrichten und liest beim Empfangen Referenzen auf Buffer mit neuen Nachrichten aus. Dadurch kann auf die Nachricht, sehr effizient, auch Byte-weise zugegriffen werden. Der Zugriff auf die Buffer in der Anwendung wird in einem oder mehreren sogenannten *Handlern* durchgeführt, welche durch den Scheduler aktiviert werden.
Generell erlaubt die Implementierung des TTEthernet-API die Integration von realen Anwendungen während der Entwicklung und den ersten Tests in der Simulation. Da die Einbettung von Anwendungen in Simulationen komplex ist und für diese Arbeit nur von untergeordneter Bedeutung, wird sie an dieser Stelle nur erwähnt. Es gibt diverse Techniken für die Einbettung, beispielsweise über Sockets, Shared-Libraries oder direkte Quellcode-Integration (vgl. Mayer und Gamer, 2008). Für diese Arbeit reicht die direkte Code-Integration aus. Die Einbettung von Applikationen ist dennoch ein interessantes Feld und sollte in weiteren Arbeiten näher betrachtet werden.

5.2.3 Die Simulationskonfiguration

Auch für die Konfiguration der Simulationskomponenten wird die EMF4CPP-Bibliothek eingesetzt. Da hier die komplette Network-Configuration (ca. 200 verschiedene Klassen) eingelesen werden muss, kommen die erwähnten Qualitätsdefizite und fehlenden Features der verwendeten Version (0.0.2) (siehe Abschnitt 5.1 auf Seite 62) noch deutlicher zum Tragen. Um die komplette Konfiguration einzulesen, muss insbesondere die Funktionalität des EMF4CPP-Parsers für Referenzen im Modell erweitert werden. Bisher unterstützt der Parser nur Links auf Elemente innerhalb derselben Datei. Da die TTEthernet Network-Configuration jedoch auf mehrere Dateien aufgeteilt ist, muss der Parser so erweitert werden, dass er rekursiv auch Referenzen in andere Dateien auflöst. Darüber hinaus muss auch die Syntax der Pfade in den *EMF Fragment-based Uniform Resource Identifiers (URIs)* erweitert werden. Standardmäßig unterstützt EMF4CPP nur Pfade mit Indizes, also Pfade, in denen die Position des referenzierten Elements im Pfad angegeben ist. Die Network-Configuration erfordert jedoch eine erweiterte Syntax der URIs, die auch sogenannte Constraints[15] unterstützt. Bei Constraint-basierten Referenzen wird das referenzierte Objekt durch die Auswertung einer Bedingung gefunden.
Listing 5.1 zeigt Beispiele für die zu implementierenden erweiterten EMF Fragment-based URIs.

Listing 5.1: Erweiterte Syntax von Referenzen in EMF Fragment-based URIs

```
1 <!-- EMF Fragment-based URI mit Link in andere Datei-->
2 <Student studiengang="studiengaenge.xmi#//@studiengang.2" />
3 <!-- EMF Fragment-based URI mit Indizes-->
4 <Student pruefungsordnung="#//@studiengang.2/@pruefungsordnung.2" />
5 <!-- EMF Fragment-based URI mit Constraints-->
6 <Student pruefungsordnung="#//@studiengang[name='Informatik']/
    @pruefungsordnung[jahr='2008']" />
```

Nach der Erweiterung der Syntax kann die Konfiguration in das Simulationsmodell eingebaut werden. Die Simulationsmodule (TTEthernet-Host und -Switch) erhalten die Network-Configuration und ihre Bezeichnung im Modell über die NED-Datei als Parameter übergeben. Diese Informationen reichen aus, um innerhalb des Modells die korrekte Komponente zu finden. In der Simulation kann nun durch den kompletten Objektbaum navigiert werden.
Damit die Konfiguration nicht für jede Komponente erneut geparst wird und um Speicher zu sparen, wird EMF4CPP um eine *Model-Repository-Komponente* erweitert. Das Modell-Repository speichert den Objektbaum zu einer geparsten Datei und gibt von bereits verarbeiteten XML-Dateien die gespeicherte Referenz zurück. Da die Komponenten ausschließlich

[15] Constraint (deutsch: Zwangsbedingung) bezeichnet Bedingungen, welche eingehalten werden müssen, damit ein Wert in ein System übernommen wird

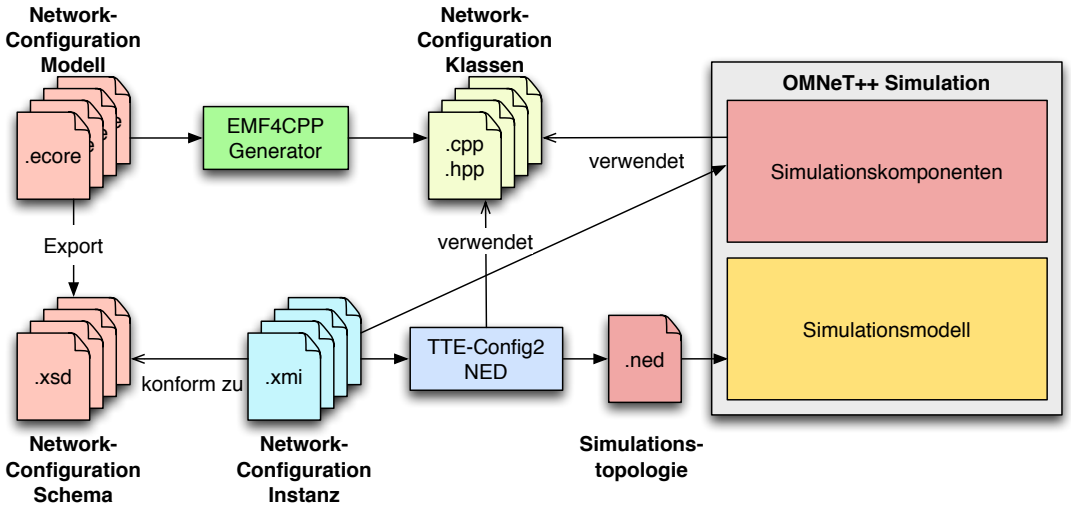

Abbildung 5.7: Überblick über den Prozess der Simulationskonfiguration

lesend auf das Modell zugreifen, dürfen alle Komponenten auf demselben Objektbaum arbeiten.
Abbildung 5.7 zeigt die Komponenten und den kompletten Verarbeitungsprozess bei der Simulationskonfiguration. Dabei werden zur Laufzeit die XMI-Dateien der Network-Configuration eingelesen. Das TTE-Config2NED-Werkzeug (siehe Abschnitt 5.1 auf Seite 62) erstellt aus ihnen die NED für die Simulation; die Simulationskomponenten verwenden die XMI-Dateien als Konfiguration.

5.3 Werkzeuge zur Auswertung

Für das Einlesen der Anforderungen und die Ausgabe des Analyse-XML kann ein einfacher Parser wie *Xerces-C++* (vgl. Apache Software Foundation) verwendet werden. Xerces ist eine Bibliothek, die das Parsen, Generieren, Manipulieren und Validieren von XML-Dokumenten erlaubt. Dabei werden APIs für das *Document Object Model (DOM)* und die *Simple API for XML (SAX)* bzw *SAX2* angeboten.
Für das Parsen und Extrahieren von Daten in Simulation-Result-Dateien bietet OMNeT++ das *Scave Tool*. Das Scave Tool ist ein Kommandozeilenprogramm, das auf der *oppscave-Bibliothek* basiert. Um die Funktionalität des Werkzeugs direkt verwenden zu können, wird die liboppscave in das Analysewerkzeug eingebunden. Dies reduziert den Entwicklungsaufwand beim Parsen der OMNeT++-Simulationsausgabe.
Im Auswertungsprozess wird festgestellt, ob die Anforderungen erfüllt sind. Dafür werden die aufgezeichneten Werte gegen die definierten Bedingungen der Anforderungen überprüft.

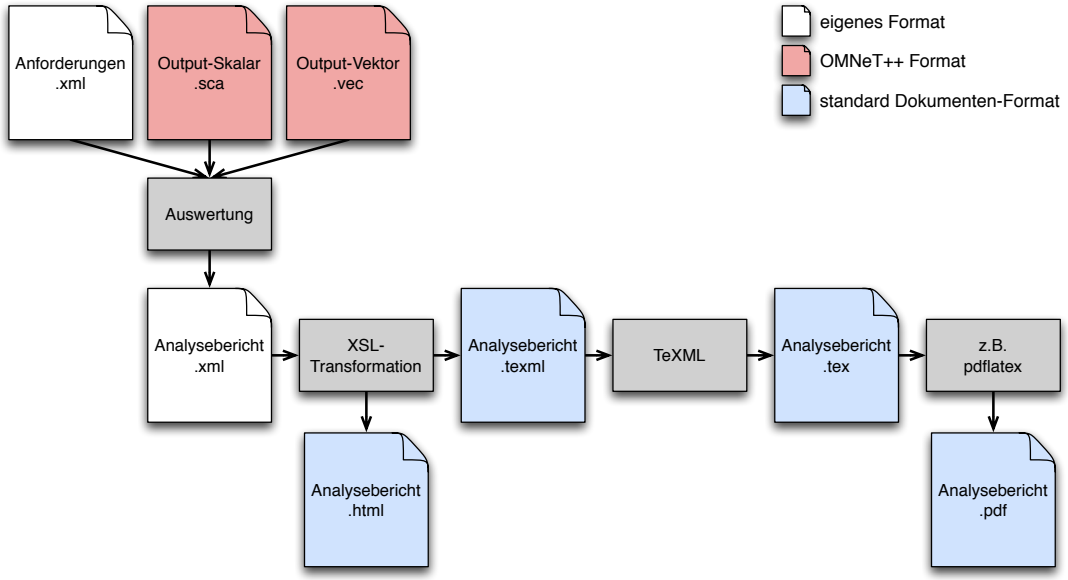

Abbildung 5.8: Komponenten und Formate im Auswertungsprozess

Das Zielformat des Analyseberichtes ist das *PDF*, da es ein verbreitetes Format ist, das sich gut zur Dokumentation eignet. Für die Ausgabe im PDF gibt es bereits diverse C++-Bibliotheken. Um dem Nutzer jedoch Freiheiten bei der Gestaltung der Berichte zu geben, wird ein auf TeX (vgl. Knuth, 1998) basierendes Dokument erstellt. Dafür muss das XML-Format in die TeX-Syntax übersetzt werden.
Mit der *Extensible Stylesheet Language Transformation (XSLT)* gibt es bereits ein standardisiertes Verfahren für die Übersetzung von XML. Da XSLT jedoch ineffizient ist, wenn das Ergebnis der Transformation selbst nicht XML ist, wird ein Zwischenschritt für die Übersetzung eingefügt. Das Dokument wird zunächst per XSLT in die *TeX Markup Language (TeXML)* übersetzt. Dabei kann der Nutzer durch Manipulation der XSLT-Beschreibung das Layout des späteren Dokuments komplett anpassen.
TeXML (vgl. Parashchenko, 2011) ist eine XML-basierte Beschreibungssprache für LaTeX-Dokumente. Durch den TeXML-Prozessor wird das TeXML-Dokument zu einer Tex-Datei übersetzt. Anschließend kann mit den standard LaTeX-Werkzeugen beispielsweise ein PDF erstellt werden.
Abbildung 5.8 zeigt die Komponenten im Auswertungswerkzeug. Der XML-Analysebericht (Beispiel Listing C.4 auf Seite 107) wird per XSL-Transformation direkt in einen formatierten Bericht (beispielsweise in HTML), oder in ein TeXML-Dokument (Listings C.5 auf Seite 108 und C.6) übersetzt. Mit dem TeXML-Werkzeug kann diese zu einem LaTeX-Dokument übersetzt werden (Listing C.4 auf Seite 107). Aus dem LaTeX-Dokument können beliebige Formate wie z.B. PDF erzeugt werden (Beispiel-Dokument C.1 auf Seite 114).

Kapitel 6

Ergebnisse und Fallstudie

In diesem Kapitel werden ausgewählte Simulationsergebnisse, sowie eine Evaluierung der Simulationsperformance aufgezeigt (Abschnitt 6.1). Weiterhin wird die vorgestellte Simulationsumgebung realen Hardwaremessungen und Ergebnissen aus dem analytischen Modell (siehe Abschnitt 3.2.1 auf Seite 39) gegenübergestellt (Abschnitt 6.2). Das Kapitel schließt mit einer Fallstudie ab, die den Einsatz des Evaluierungsprozesses anhand einer realen Automotive-Anwendung bewertet (Abschnitt 6.3).

6.1 Simulationsergebnisse und Performance

Im Folgenden wird die Leistungsfähigkeit des Simulationsmodells untersucht. Dabei wird analysiert, ob das Modell spezifikationskonform Pakete weiterleitet und die Simulation für Anwendungen im Automotive-Bereich ausreichend skaliert.

6.1.1 Simulations-Setup und Methodik

Die entwickelte TTEthernet-Implementierung wird zunächst auf Basis künstlich generierter Topologien für Fahrzeug-Backbones getestet.
Um zu überprüfen, ob das Modell den Anforderungen an die Skalierbarkeit gerecht wird, werden Topologien innerhalb der erwarteten maximalen Backbone-Größe generiert. Basierend auf aktuellen Designs kann vorläufig mit einer Topologie von maximal 70 Hosts und 7 Switchen gerechnet werden.
Um die Metriken und Protokollkonformität des TTEthernet-Modells zu zeigen, wird weiterhin die Latenz und der Jitter eines Beispiel-Netzwerkes mit TTEthernet- und INET-Standard-Ethernet-Switchen verglichen. Die dafür verwendete Topologie besteht aus zwei verbundenen Switchen, an denen zwei TTEthernet-Hosts für time-triggered und zehn Standard-Etehrnet-Hosts für die Erzeugung von best-effort Hintergrunddatenverkehr hängen (siehe Abbildung 6.2 auf Seite 75). Alle Links im simulierten Netzwerk haben eine Bandbreite von

100 Mbit/s. Die Clock-Drift wird mit 200 ppm konfiguriert. Die Verzögerung des Kabels beträgt 100 ns auf jedem Link.
Um den Einfluss des best-effort Datenverkehrs auf die Übertragung über den Link zwischen den beiden Switchen zu messen, wird dieser zusätzlich zu den time-triggered Nachrichten zwischen 0 % und 100 % mit best-effort Frames ausgelastet. Während dieser Simulation wird die minimale und maximale Ende-zu-Ende Latenz über 10.000 Nachrichten gemessen. Die Extremwerte der Ende-zu-Ende Latenz sind die wichtigste Metrik für die Echtzeit-Fahrzeugkommunikation (siehe Abschnitt 3.3 auf Seite 40).

6.1.2 Overhead des TTEthernet-Simulationsmodells

Der Overhead, der durch die Erweiterung des INET-Frameworks um die TTEthernet-Funktionalität entsteht, wird durch den Vergleich von CPU-Zeit und Speicherverbrauch im INET-Ethernet- und TTEthernet-Modell bestimmt. Um ein vergleichbares Ergebnis zu erhalten wird dabei dieselbe Topologie eingesetzt. Das TTEthernet-Protokoll führt diverse Konformitätsüberprüfungen bei eingehenden Nachrichten durch, die im Standard-Ethernet nicht vorhanden sind (siehe Abschnitt 4.2.3 auf Seite 58). Darüber hinaus generiert das TTEthernet-Modell wegen der lokalen Uhren (siehe Abschnitt 4.2.3 auf Seite 57) ca. 3 % mehr Events. Dadurch entsteht eine ungefähr 10 % höhere CPU-Zeit für die Simulation in TTEthernet gegenüber der INET-Standard-Ethernet-Implementierung. Durch die neu eingeführten Buffer steigt der Speicherbedarf um ca. 60 %.
Die Simulation läuft auf standard PC-Hardware mit zwei Kernen à 2,4 GHz und 2 GB Arbeitsspeicher. Experimente zeigen eine lineare Abhängigkeit zwischen dem Speicherverbrauch und der Netzwerkgröße sowie der Anzahl von time-triggered Botschaften im Zyklus. Abbildung 6.1 auf der nächsten Seite zeigt die benötigte CPU-Zeit und den Speicherverbrauch bei Simulationen unterschiedlicher Größe. Dabei wird jeweils über 30 Echtzeitsekunden simuliert. Auf der X-Achse ist die Topologiegröße (Anzahl Switche und Anzahl Hosts) aufgetragen. Es wird dabei eine synthetisch generierte Topologie ausgemessen. Simulationen auf verschiedenen Topologien gleicher Größe zeigen, dass die Rechenzeit und der Speicherverbrauch ausschließlich von der Größe und nicht von der Struktur der Topologie abhängt.
Empirische Untersuchungen innerhalb der maximal erwarteten Netzwerkgröße von 70 Hosts und 7 Switchen für einen zukünftigen Fahrzeug-Backbone zeigen, dass die Simulation für die Experimente ausreichend performant ist.

6.1.3 Exemplarische Simulationsergebnisse

Die Ende-zu-Ende Latenz zwischen den time-triggered Hosts wird für die INET-Ethernet-Switch-Implementierung und das TTEthernet-Modell gemessen. Die Ergebnisse (siehe Abbildung 6.3 auf Seite 76) zeigen die maximale und minimale Nachrichtenlatenz für die Bei-

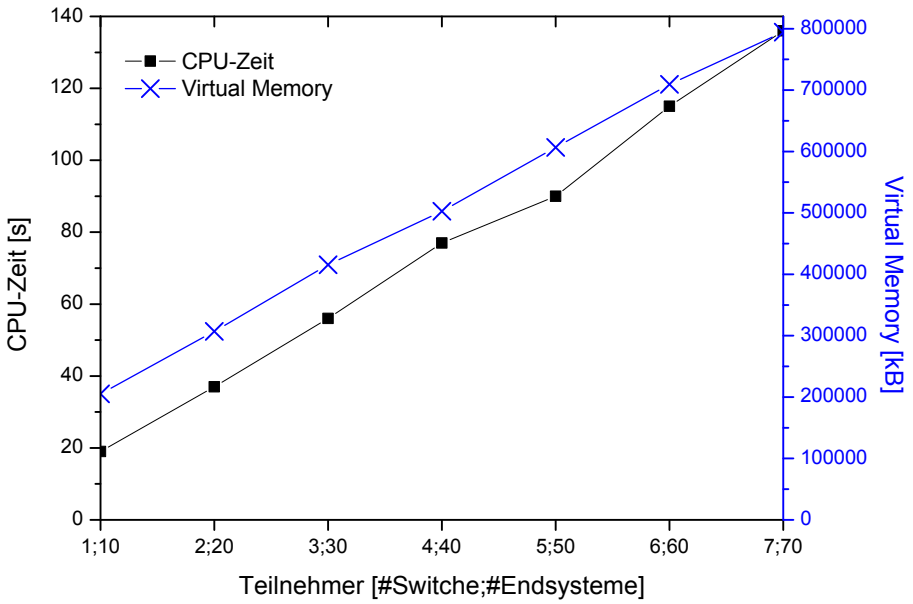

Abbildung 6.1: Lineares Skalierungsverhalten des Simulationsmodells bei abstrakten Topologien verschiedener Größe

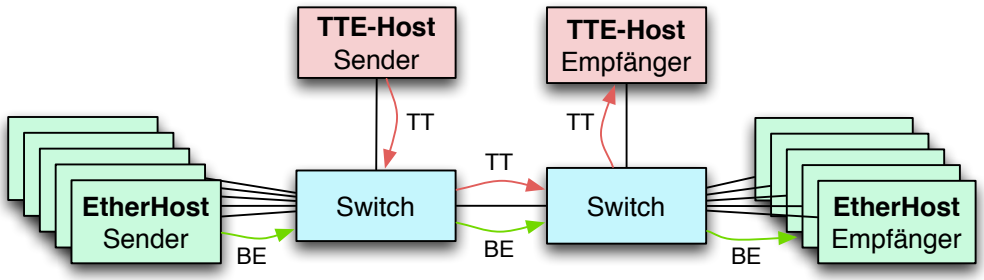

Abbildung 6.2: Simulierte Topologie für den Nachweis der Protokollkonformität — Parallele Übertragung von time-triggered und best-effort Nachrichten

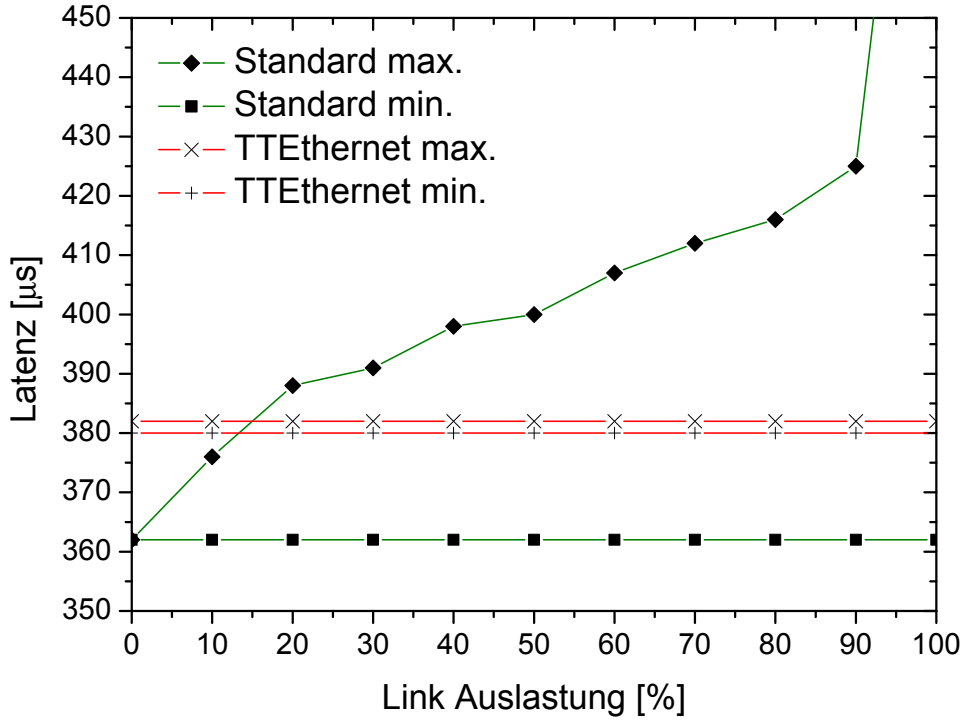

Abbildung 6.3: Vergleich der Ende-zu-Ende Latenz — Standard INET-Switch und TTEthernet-Switch

spieltopologie (Abbildung 6.2 auf der vorherigen Seite) in einer Messung über 10.000 Nachrichten, während das Netzwerk zusätzlich mit einer variablen Menge von best-effort Frames belastet wird.
TTEthernet verspricht zuverlässige Kommunikation innerhalb definierter zeitlicher Grenzen. Abbildung 6.3 zeigt, dass durch das synchronisierte Protokoll, die Priorisierung und die Mechanismen zur deterministischen Medienreservierung (siehe Abschnitt 2.3.1 auf Seite 20) eine gleichförmige, von der Linkauslastung unabhängige Latenz erreicht wird. Die geringen Schwankungen zwischen minimaler und maximaler Latenz entstehen durch die Clock-Drift der lokalen Uhren (siehe Abschnitt 4.2.3 auf Seite 57). Der Einfluss der Clock-Drift ist in den Vorarbeiten bereits analytisch verifiziert worden (vgl. Steinbach u. a., 2010).
Dem gegenüber steigt beim INET-Ethernet-Switch die Differenz zwischen minimaler und maximaler Latenz mit zunehmender Link-Auslastung. Dieser Effekt entsteht durch das Aufstauen von Nachrichten an den ausgehenden Ports der Switche. Wird der Link voll ausgelastet, ist die Kommunikation nicht mehr zuverlässig. Es werden Nachrichten verworfen, da die Übertragungskapazität nicht balanciert ist.
Eine weitere wichtige Analyse, welche die Topologie und Konfiguration eines Netzwerkes charakterisiert, ist die *Latenzverteilung*. Graphen zur Latenzverteilung können die Entschei-

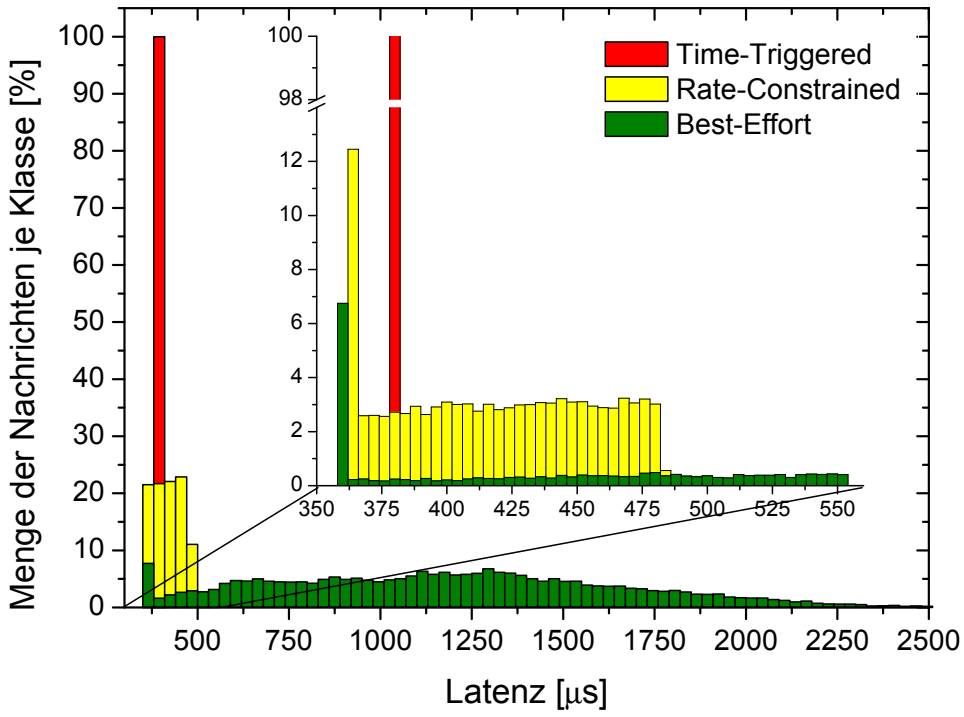

Abbildung 6.4: Latenzverteilung der drei Traffic-Klassen von TTEthernet bei einem voll ausgelasteten Link

dung unterstützen, ob ein Netzwerk die Anforderungen der Applikationen an die Weiterleitung von event-triggered Datenverkehr erfüllt (vgl. Van Mieghem, 2006). Abbildung 6.4 zeigt die Latenz für die drei Nachrichtenarten von TTEthernet auf demselben voll ausgelasteten TTEthernet Netzwerk. Wie gefordert haben die time-triggered Nachrichten eine nahezu konstante Latenz. Die rate-constrained Nachrichten werden mit einer variablen, jedoch unter 500 µs begrenzten Verzögerung ausgeliefert. Durch das Aufstauen und die Mechanismen zur Linkreservierung ist im Vergleich dazu beim best-effort Datenverkehr der Jitter hoch.

6.2 Verifikation des Simulationsmodells

Um zu überprüfen, ob das TTEthernet-Simulationsmodell das Verhalten des Protokolls korrekt abbildet, werden die Simulationsergebnisse mit Ergebnissen aus dem analytischen Modell (siehe Abschnitt 3.2.1 auf Seite 39) und Messungen mit echter TTEthernet-Hardware verglichen. Da zum Zeitpunkt der Messungen für die Überprüfung mit echter Hardware nur ein Switch zur Verfügung stand, ist eine einfachere Topologie für die Verifikation genutzt worden. Die Verifikation kann wie hier beschrieben jedoch auf jegliche Topologie angewendet werden.

Die Evaluierung nutzt eine Topologie mit einem Switch und mehreren Hosts. Dabei läuft die Kommunikation zwischen den Teilnehmern stets mit einem Switching-Hop. Während der Evaluierung werden im Switch verschiedene Schedules mit unterschiedlichen Weiterleitungsverzögerungen eingesetzt.
Die Simulation läuft über 10.000 Nachrichten jeweils für einen Switch-Schedule mit 350 µs Verzögerung zwischen Empfang und Weiterleitung des Frames und für einen Schedule mit 9 µs. Eine Verzögerung von 9 µs ist zur Zeit das Minimum für eine zuverlässige Weiterleitung. Der Schedule mit 9 µs führt zu einer simulierten Ende-zu-Ende Latenz von 19,5 µs für einen Frame mit minimalem Payload und 252,0 µs für einen Frame mit maximalem Payload. Für den Schedule mit 350 µs Weiterleitungsverzögerung beträgt die Ende-zu-Ende Latenz, abhängig von der Payload-Größe, 360,5 µs bis 593,0 µs. Im Folgenden werden diese Simulationsergebnisse mit dem mathematischen Modell und der Hardware-Messung verglichen.

6.2.1 Vergleich mit mathematischem Modell

Das mathematische Modell (siehe Abschnitt 3.2.1 auf Seite 39) der TTEthernet-Komponenten ist in einer Vorarbeit für einen theoretischen Vergleich der Leistungsmerkmale von TTEthernet und FlexRay entwickelt und veröffentlicht worden (vgl. Steinbach u. a., 2010).
Die folgenden Gleichungen sind ein Teil dieses Frameworks. Mit ihrer Hilfe lässt sich das zeitliche Verhalten einer TTEthernet-Komponente berechnen: Die Latenz t_L wird dabei für minimalen und maximalen Payload berechnet. Sie setzt sich zusammen aus der physikalischen Signallaufzeit auf dem Kabel $t_{WD} \cdot l_W$, der Zeit für die Übertragung der Daten — jeweils beim Sender und dem weiterleitenden Switch — $2 \cdot l_F \cdot t_b$ und der Verzögerung durch den Switch-Schedule t_{SD}:

$$t_L(t_{SD}) = t_{WD} \cdot l_W + 2 \cdot l_F \cdot t_b + t_{SD} \tag{6.1}$$

$$t_{L_{min}}(t_{SD}) = 10\,\frac{\text{ns}}{\text{m}} \cdot 0{,}5\,\text{m} + 2 \cdot 5{,}12\,\mu\text{s} + t_{SD} \tag{6.2}$$

$$t_{L_{max}}(t_{SD}) = 10\,\frac{\text{ns}}{\text{m}} \cdot 0{,}5\,\text{m} + 2 \cdot 121{,}44\,\mu\text{s} + t_{SD} \tag{6.3}$$

Wie erwartet besteht eine lineare Abhängigkeit zur Frame-Länge l_F. Für den Schedule mit $t_{SD} = 350$ µs liegt die berechnete Latenz je nach Payload-Größe zwischen 360,245 µs und 592,885 µs. Dieselbe Berechnung für den Schedule mit der minimalen Verzögerung von $t_{SD} = 9$ µs ergibt eine Latenz zwischen 19,245 µs und 251,885 µs.
Die Ergebnisse des mathematischen Frameworks sind geringfügig ($<$300 ns) niedriger als die Simulationsergebnisse. Der Grund sind Vereinfachungen im mathematischen Modell bei der Abbildung von Hardware-Verzögerungen. Diese Verzögerungen werden in der Simulation präzise berücksichtigt.

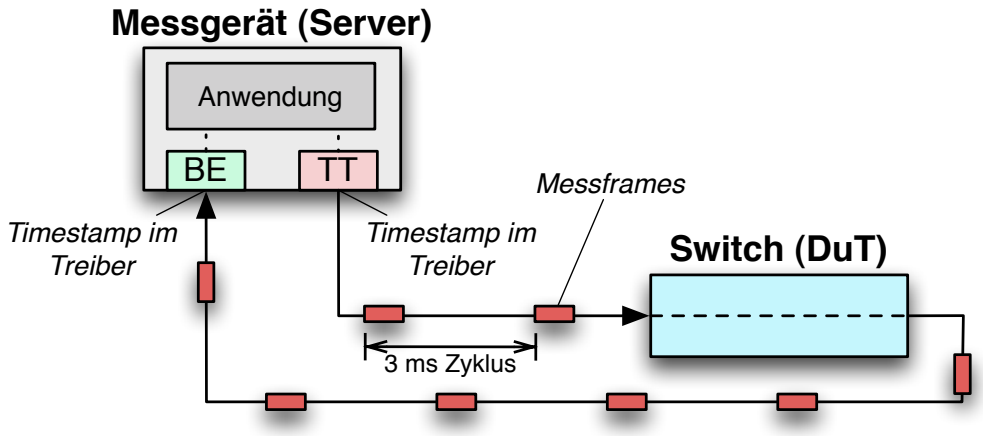

Abbildung 6.5: Hardware-Messung nach Bartols u. a. (2011)

6.2.2 Vergleich mit Messungen auf TTEthernet-Hardware

Um die Qualität des Simulationsmodells besser bewerten zu können, werden die Ergebnisse mit dem Verhalten echter TTEthernet-Hardware verglichen. Dazu wird eine von Florian Bartols (vgl. Bartols u. a., 2011) entwickelte Messmethode, die auf sogenannten *Components-of-the-shelf (COTS)* basiert, verwendet. Sie ist auf einem eingebetteten System mit zwei Ethernet-Ports und einem Echtzeit-Linux-Kernel umgesetzt. Das *Device-under-Test (DuT)* — also das auszumessende Netzwerk — wird zwischen den beiden Ports des Systems angeschlossen. Für die Messung werden Nachrichten mit Zeitstempel von einem Port abgesendet, durchlaufen das Netzwerk und erreichen den zweiten Port des Systems. Beim Empfang wird ein zweiter Zeitstempel aufgezeichnet (siehe Abbildung 6.5). Diese Methode erlaubt die Messung von Latenz und Jitter des angeschlossenen Netzwerkes mit einer Genauigkeit im einstelligen Mikrosekunden-Bereich.
Für das simulierte Netzwerk mit einem Schedule von 350 µs Verzögerung ergibt sich eine gemessene Latenz zwischen 360 µs und 592 µs je nach Framegröße. Bei Vernachlässigung der niedrigeren Präzision der Messmethode gegenüber der Simulation sind die Ergebnisse als vergleichbar anzusehen. Der Schedule mit 9 µs Verzögerung konnte nicht gemessen werden, da die Messmethode auf einer Softwareimplementierung des TTEthernet-Stacks basiert, dessen Präzision der Synchronisierung nicht ausreicht. Infolgedessen ergeben sich bei diesem Schedule partielle Paketverluste bei der Software-basierten Messmethode. Durch eine Hardware-nahe Implementierung der Messung auf einem ARM9-System-on-a-Chip (SoC) soll die Messung jedoch zukünftig ohne Paketverlust möglich sein.

Weiterleitungsverzögerung	Payload	Simulation	Theorie	Messung
350 µs Schedule	minimum	360,5 µs	360,245 µs	360 µs
	maximum	593,0 µs	592,885 µs	592 µs
9 µs Schedule	minimum	19,5 µs	19,245 µs	-
	maximum	252,0 µs	251,885 µs	-

Tabelle 6.1: Vergleich der Latenzgrenzwerte bei minimalem und maximalem Payload für verschiedene Switch-Schedules

6.2.3 Bewertung der Verifikation

Das Simulationsmodell implementiert das erwartete Verhalten eines TTEthernet-Systems mit einer hohen Genauigkeit. Die Ergebnisse der Experimente zeigen, dass das Modell konform mit der TTEthernet-Spezifikation, dem mathematisch-analytischen Modell und der Hardware-Messung arbeitet. Die Latenz der time-triggered Nachrichten ist unabhängig vom simultan gesendeten best-effort Datenverkehr. Tabelle 6.1 zeigt die Ergebnisse der verschiedenen Messungen und Experimente für die Schedules mit 9 µs und 350 µs im Vergleich. Da alle verglichenen Verfahren Ungenauigkeiten haben, kann die Qualität des Simulationsmodells mit der Hardware-Messung nur bis zu einer Genauigkeit im Mikrosekundenbereich, mit dem analytischen Modell im Bereich von ca. 500ns überprüft werden. Diese Genauigkeit reicht jedoch für die Anwendungen im Automobil aus. Um dennoch genauere Hardware-Messergebnisse für den Vergleich zu erhalten, ist die Implementierung der Messmethode ohne Betriebssystem auf einem ARM9-SoC geplant. Dieses kann eingehende Frames mit einer Genauigkeit von 10^{-8} Sekunden mit Zeitstempeln versehen.
Da in der Simulation im Gegensatz zum analytischen Modell die Auflösung der Zeit begrenzt ist, ist auch die Simulationsgenauigkeit begrenzt. Sie muss daher bei der Bewertung der Verifikation mit beachtet werden. Die Simulationsgenauigkeit entscheidet, ob zwei zeitlich nah beieinander liegende Ereignisse auf denselben Zeitpunkt oder auf verschiedene Zeitpunkte abgebildet werden. Momentan ist die Simulation in Pikosekunden aufgelöst und damit in kleinere Zeiteinheiten als die Prozessortakte aktueller Computersysteme. Die nichtdeterministischen Einflussgrößen, beispielsweise durch die Clock-Drift, sind um ein vielfaches größer als die Ungenauigkeit durch die Simulationsauflösung. Daher kann dieser Einflussfaktor als gering eingestuft werden.
Eine weitere Einflussgröße für die Qualität der Verifikation ist die Testabdeckung. Hier sollte in zukünftigen Arbeiten überprüft werden, wie die Clock-Drift noch umfassender simuliert werden kann. Eine Verbesserung ist dabei eine künstlich synchronisierte Drift aller Uhren im Modell, um sicherzustellen, dass die maximalen Abweichungen in der Synchronisation während der Simulation erreicht werden.

6.3 Fallstudie zum Automotive-Einsatz

Um den Evaluierungsprozess zu überprüfen, wird eine Fallstudie durchgeführt. Die Fallstudie zur kamerabasierten Fahrzeugumfelderfassung beschreibt die Simulation einer realen Automotive-Anwendung, die auf Realtime-Ethernet übertragen wird.

6.3.1 Kamerabasierte Fahrzeugumfelderfassung

Um die Anzahl der Verletzten und Toten im Verkehr weiter zu reduzieren, erhalten aktive Sicherheitssysteme die das Fahrzeugumfeld bei Aktionsentscheidungen einbeziehen, größere Bedeutung. Dabei sollen Situationen mit erhöhter Gefahr für die Verkehrsteilnehmer vor deren Eintritt erkannt und geeignete Gegenmaßnahmen ergriffen werden (vgl. Kämpchen u. a., 2005).
Im Bereich solcher Fahrerassistenzsysteme haben Kameras in den vergangenen Jahren stark an Bedeutung gewonnen. In aktuellen Oberklassefahrzeugen werden bereits bis zu fünf Kameras eingesetzt, um — in Verbindung mit weiteren Sensoren — das Fahrzeugumfeld zu erfassen und durch Steuergeräte, sogenannte Electronic Control Units (ECUs), auszuwerten.
Um die hohen Datenmengen von den Kameras zur verarbeitenden ECU zu transportieren, werden heute proprietäre Protokolle eingesetzt, welche über Low Voltage Differential Signaling (LVDS), einem Schnittstellenstandard für Hochgeschwindigkeitsdatenübertragung, transportiert werden. Heutige Automotive-Bussysteme wie der MOST-Bus bewältigen die anfallenden Datenmengen nicht (vgl. Kämpchen u. a., 2005). Die Kompression der Daten ist aufgrund der dabei entstehenden Unschärfe bei der Kantenerkennung in den Kamerabildern problematisch.
Auch BMW plant bis 2013 bei der Fahrzeugumfelderfassung die LVDS-Strecken durch Ethernet zu ersetzen. Dies soll im Gegensatz zu geschirmten LVDS-Verbindungen eine signifikante Kostenersparnis ermöglichen (vgl. Bruckmeier, 2010).
Um für die Evaluierung des Simulationsprozesses ein möglichst realistisches Beispiel aus dem Bereich der Automotive-Anwendungen zu verwenden, wird die Fallstudie in Zusammenarbeit mit der Ingenieurgesellschaft Auto und Verkehr (IAV) (vgl. Ingenieurgesellschaft Auto und Verkehr GmbH) durchgeführt. Dadurch kann auf reale Anforderungen, Topologien und Verkehrsmuster zurückgegriffen werden.

Versuchsszenario und Anforderungen

Simuliert wird die Datenübertragung einer kamerabasierten Fahrzeugumfelderfassung. Die Anwendung umfasst vier Kameras — Front und Heck, Spiegel links, Spiegel rechts — und eine oder mehrere ECUs, welche die Bilddatenströme empfangen.

Aktuell liegen die Kameras in der VGA-Klasse (640x480 Pixel). Für diese Untersuchung wird vorerst davon ausgegangen, dass die Daten unkomprimiert übertragen werden müssen, um eine zuverlässige Bildverarbeitung zu gewährleisten. Daher entspricht die geforderte Bandbreite dem Worst-Case. Die geforderte Bildrate (Zyklus) liegt bei 30 Bildern pro Sekunde. Die Latenz darf ein Bild nicht übersteigen. Ein Jitter, der maximal 10 % dieser Zykluszeit beträgt, ist akzeptabel.

Konzept und Durchführung der Fallstudie

Im Folgenden werden die Anforderungen der kamerabasierten Fahrzeugumfelderfassung und das Konzept des Simulationsversuchs sowie seiner Durchführung beschrieben.

Anforderungen: Zunächst werden die Anforderungen in konkrete Metriken übersetzt. Die Auflösung von 640x480 Pixel entspricht 307.200 Pixeln. Je nach Farbkodierung werden 2 Byte (YUV-Farbmodell) oder 3 Byte (RGB-Farbmodell) je Pixel benötigt. In der Simulation wird vorerst von der YUV-Kodierung ausgegangen. Damit kann das Datenvolumen eines Bildes berechnet werden:

$$\begin{aligned} Datenvolumen &= 640 \cdot 480 \cdot 2\,\text{Byte} \\ &= 307.200 \cdot 2\,\text{Byte} = 614.400\,\text{Byte} \end{aligned} \tag{6.4}$$

Aufgrund der sehr hohen Datenmenge wird in Ethernet-Frames mit maximalem Payload (1500 Byte) übertragen. Dadurch ergibt sich die Anzahl nötiger Frames pro Bild:

$$\begin{aligned} Frames\,pro\,Bild &= \left\lceil \frac{614.400\,\text{Byte}}{1500\frac{\text{Byte}}{\text{Frame}}} \right\rceil \\ &= \lceil 409,6\,\text{Frames} \rceil = 410\,\text{Frames} \end{aligned} \tag{6.5}$$

Übertragen werden 30 Bilder pro Sekunde; dies entspricht einer Zykluszeit von $t_{Cycle} \approx$ 33,3 ms. Die dabei benötigte Bandbreite jeder Kamera wird nun für den vollen Ethernet-Frame mit Header (1518 Byte) berechnet:

$$\begin{aligned} Bandbreite &= 30\frac{\text{Bilder}}{\text{s}} \cdot 410\frac{\text{Frames}}{\text{Bild}} \cdot 1518\frac{\text{Byte}}{\text{Frame}} \\ &\approx 17,8\frac{\text{MByte}}{\text{s}} \approx 142,5\frac{\text{MBit}}{\text{s}} \end{aligned} \tag{6.6}$$

Hier zeigt sich bereits, dass Links mit 100 Mbit/s nicht ausreichen, um die Daten der Kameras zu übertragen. Es müssen Links mit 1 Gbit/s eingesetzt werden. Tabelle 6.2 auf der nächsten Seite zeigt die aus der Anwendung abgeleiteten Anforderungen.

Anforderungen	
Zykluszeit	33,3 ms
Bandbreite	142,5 Mbit/s je Kamera
maximale Latenz	33,3 ms
maximaler Jitter	3,3 ms
Weitere Parameter	
Links	1 Gbit/s
Abweichung des Quarz	200 ppm

Tabelle 6.2: Abgeleitete Anforderungen der Fahrzeugumfelderfassung

Topologie: Als Topologie wird ein Stern konfiguriert, an dem die 4 Kameras und zwei ECUs hängen, welche die Ströme der Kameras empfangen. Dabei werden die Bildströme der Kameras im Switch zusammengefügt und nach Bedarf zu neuen Bildströmen an die ECUs zusammengestellt. Anschließend wird auch der Einfluss auf die Latenz in Topologien mit mehr als einem Switch simuliert.

Konfiguration: Bei der Konfiguration wird der Zyklus des Netzwerkes auf die Bildwiederholrate (33,3 ms) konfiguriert. Damit wird je Kamera ein Bild pro Netzwerk-Zyklus übertragen. Durch die netzwerkweite Synchronisation können alle Kameras zum exakt gleichen Zeitpunkt das Bild aufnehmen. Anschließend werden die Bilder aller Kameras parallel zum Switch übertragen und dort zu neuen Bildströmen für die ECUs zusammengestellt. Abbildung 6.6 auf der nächsten Seite zeigt einen Überblick über das zeitliche Verhalten. Jedes Bild ist in 410 Frames geteilt. Damit muss eine ECU, die die Bilder aller Kameras verarbeitet, 1.640 Frames pro Zyklus empfangen.
Beim Erstellen der Konfiguration wird deutlich, dass ein Werkzeug zur Erstellung der XML-Network-Configuration (siehe Abschnitt 2.3.2 auf Seite 24) wie TTE-Plan dringend notwendig ist. Derzeit steht ein solches Werkzeug nicht zur Verfügung, sodass die Konfiguration von Hand, bzw. mit Bash-Scripten erstellt werden muss. Dieser Prozess ist allerdings in hohem Maße fehleranfällig. Das Netzwerk definiert das zeitliche Verhalten von 1.640 verschiedenen Nachrichten im Zyklus. Die Konfiguration des Zeitplans im Switch umfasst daher mehr als 35.000 Zeilen XML-Code.
Für die Konfiguration der Nachrichten wird je Frame (Bildfragment) eine unterschiedliche CT-ID gewählt (siehe Abschnitt 2.3.1 auf Seite 20). Dies hat den Vorteil, dass beim Empfang bereits anhand der CT-ID erkannt wird, um welchen Teil des Bildes es sich handelt. Ein weiterer Vorteil liegt in der Behandlung von verlorenen Frames. Da die Links nicht redundant ausgelegt sind, können — sehr selten — einzelne Frames durch Paketverlust, beispielsweise

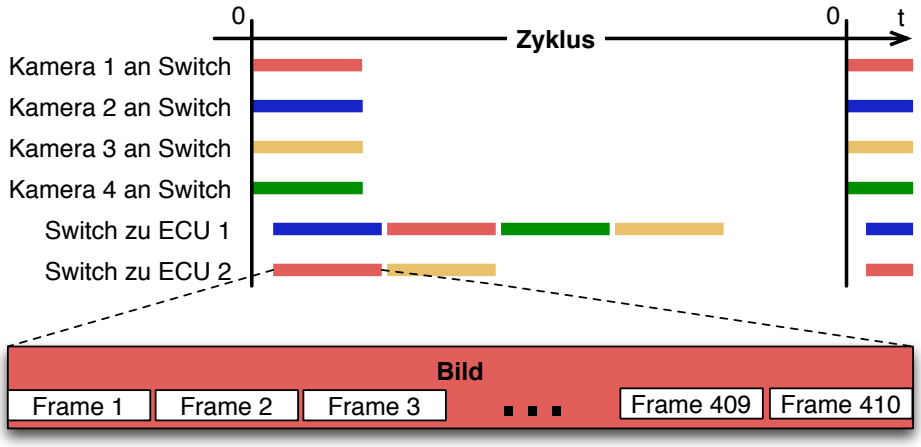

Abbildung 6.6: Überblick Bildströme und konfiguriertes Verhalten des Switches

bei Bitfehlern, verloren gehen. In diesem Fall wird automatisch der alte Wert im Empfangs-Buffer beibehalten. Auf das Bild übertragen bedeutet dies, dass der Teil des Bildes, der im verlorenen Frame liegt, durch den gleichen Bereich im vorherigen Bild ersetzt wird. Unter der Annahme, dass sich zwei aufeinander folgende Bilder ähnlich sind, kann sich dadurch die Genauigkeit anschließender Bildverarbeitungsprozesse verbessern.

Simulation: Simuliert wird eine Echtzeit-Sekunde, also 30 Zyklen, was bereits ca. 50.000 Ethernet-Paketen entspricht. Es werden verschiedene Simulationsdauern getestet. Die Ergebnisse ändern sich jedoch mit weiter zunehmender Simulationsdauer nur noch marginal. Abbildung 6.7 auf der nächsten Seite zeigt die grafische Darstellung während der Simulation. Hier kann die konfigurierte Topologie kontrolliert und die Weiterleitung der Pakete beobachtet werden.
Die grafische Darstellung und das Event-Log der Simulation unterstützen bei der Behebung von Fehlern in der Konfiguration.

Simulationsergebnisse

Abbildung 6.8 auf der nächsten Seite zeigt die Ende-zu-Ende Latenz für die Übertragung der Bilder der verschiedenen Kameras. Die Plots werden durch die OMNeT++-IDE generiert. Gemessen wird je Kamera die Zeit zwischen Versenden des ersten Bildframes und Empfangen des letzten Bildframes. Hier ist die gewählte Konfiguration (Abbildung 6.6) gut wiederzuerkennen. Trotz des zeitgleichen Empfangs aller Bilder im Switch, werden die Bilder nacheinander zur ECU weitergeleitet. Dadurch haben die Bilder beim Empfang eine unterschiedliche — jedoch in jedem Zyklus konstante — Latenz.

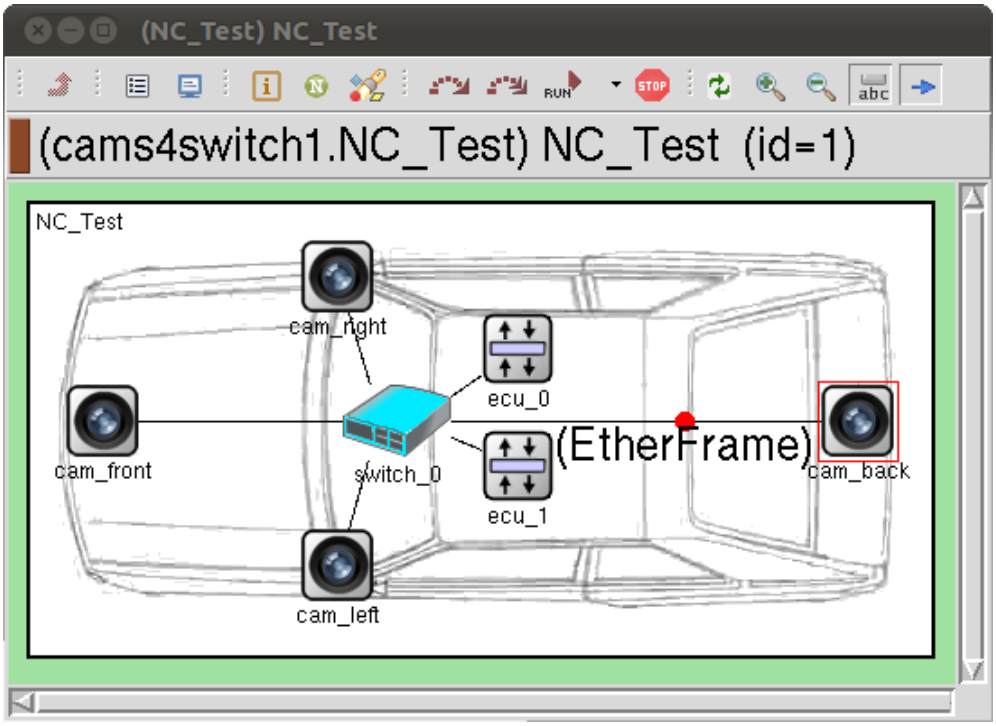

Abbildung 6.7: Grafische Darstellung der Anwendung in der Simulation

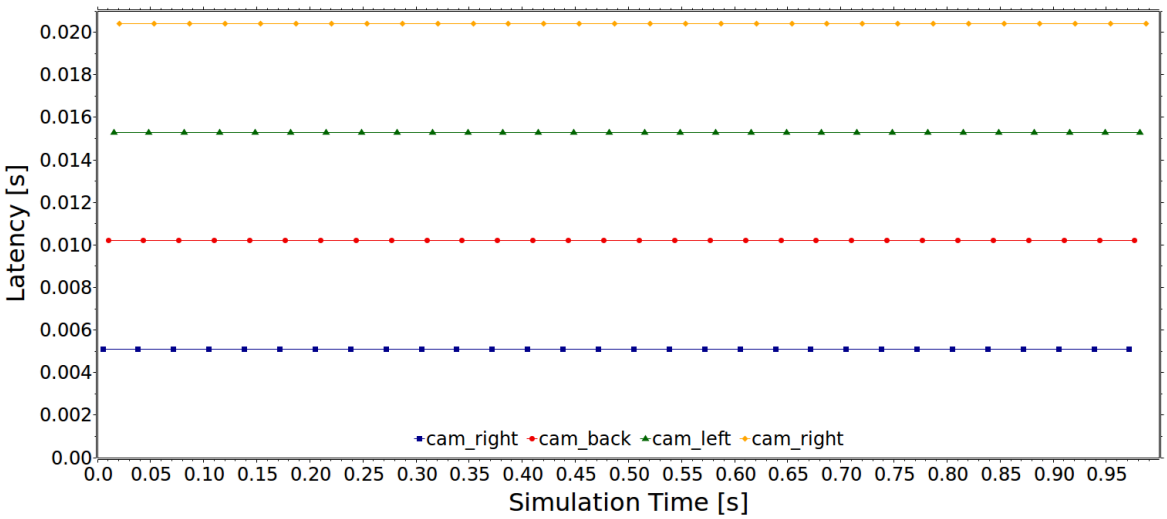

Abbildung 6.8: Ende-zu-Ende Latenz: Komplettes Bild je Kamera über time-triggered Nachrichten (OMNeT++-Plot)

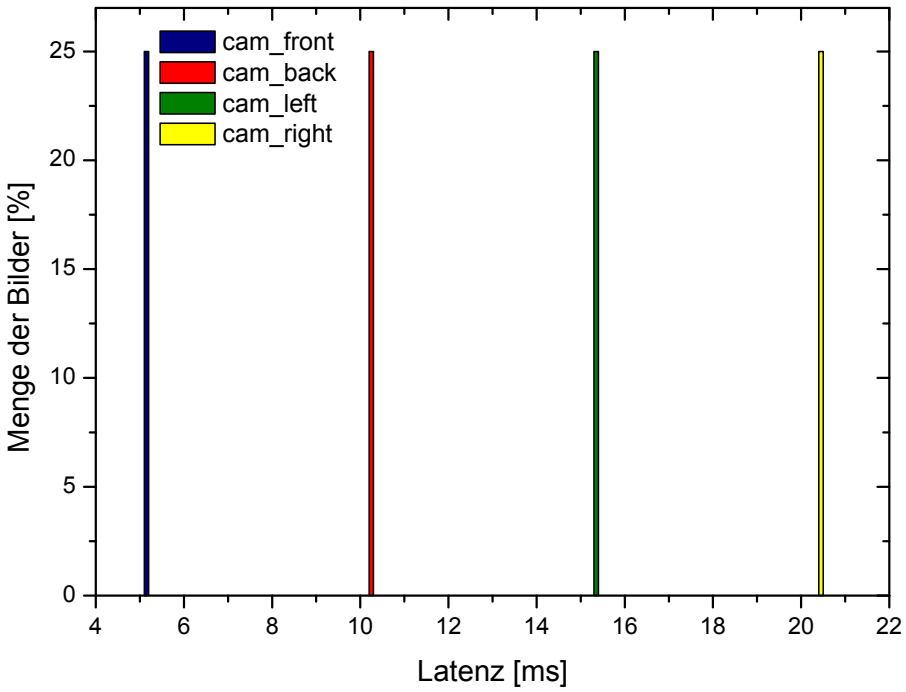

Abbildung 6.9: Latenzverteilung komplettes Bild je Kamera über time-triggered Nachrichten

Die Anforderungen für Latenz und Jitter werden in dieser Konfiguration vollständig erfüllt. Die Latenz liegt bei dem Bild mit der längsten Übertragungsdauer bei maximal 20,4 ms und damit unter den geforderten 33,3 ms (entspricht 61,2 % der maximal erlaubten Latenz). Der Jitter liegt mit maximal 9,68 µs deutlich unter den geforderten 3,3 ms (entspricht 0,29 % des maximal erlaubten Jitters), sodass dieser auf dem Graphen (Abbildung 6.8) kaum erkennbar ist. Die Latenzverteilung (Abbildung 6.9) zeigt, dass die Bilder je nach Kamera eine genau definierte Latenz aufweisen. Weitere Switche haben im Verhältnis zur Zykluslänge nur geringen Einfluss auf die Latenz. So erhöht sich die Latenz je Switch auf der Strecke zwischen Kamera und ECU um ungefähr 25 µs.
Auch rate-constrained Nachrichten eignen sich für die Echtzeit-Übertragung. Abbildung 6.10 auf der nächsten Seite zeigt die Ende-zu-Ende Latenz für dieselbe Konfiguration mit rate-constrained Nachrichten. Dadurch, dass die Übertragung der rate-constrained Nachrichten nicht geplant ist, werden in dieser Konfiguration die Frames aller Bilder gleichberechtigt im Switch weitergeleitet. Dadurch haben alle Bilder ungefähr die gleiche Latenz. Der Vorteil bei dieser Konfiguration ist, dass alle Bilder parallel verarbeitet werden können. Der Nachteil ist, dass die Verarbeitung erst relativ spät begonnen werden kann. Für eine serielle Bearbeitung der Bilder in der ECU ist sie daher ungeeignet.

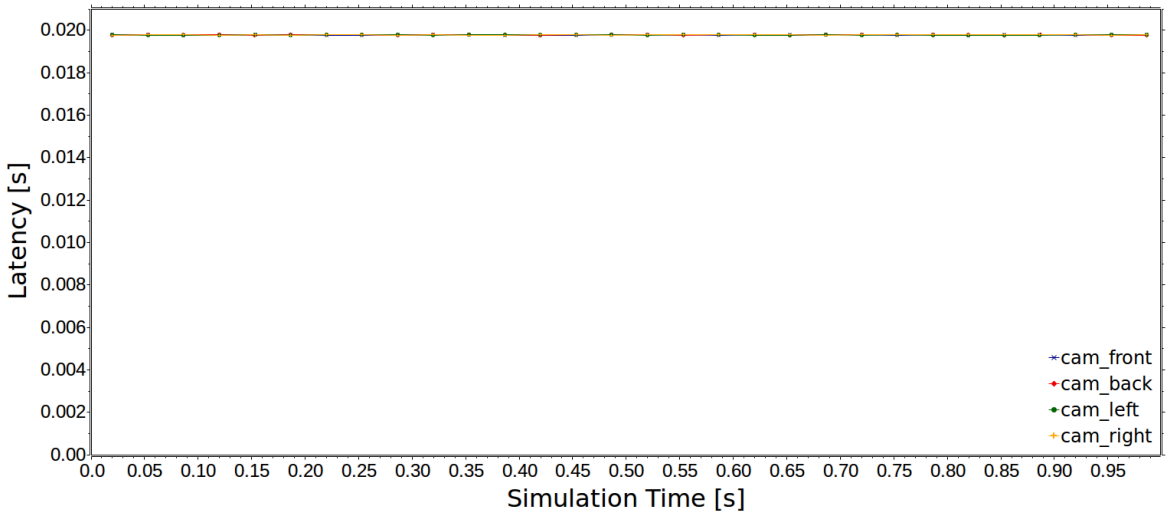

Abbildung 6.10: Ende-zu-Ende Latenz: Komplettes Bild je Kamera über rate-constrained Nachrichten (OMNeT++-Plot)

Auch bei der rate-constrained Übertragung werden die Anforderungen eingehalten. Die Latenz liegt mit maximal 19,8 ms bei einem Jitter von 39,85 µs im geforderten Bereich. Abbildung 6.11 auf der nächsten Seite zeigt eine anders skalierte Version der Ende-Zu-Ende Latenz, in der der Jitter zu erkennen ist.
Im Gegensatz dazu können mit unsynchronisiertem Standard-Ethernet die Anforderungen nicht erfüllt werden. Abbildung 6.12 auf der nächsten Seite zeigt die Ende-zu-Ende Latenz bei der Übertragung mit Standard-Ethernet-Komponenten in einer Simulation über 35 Sekunden (ca. 1,7 Milionen Frames / 4250 Bilder). Hier zeigt sich, dass durch unterschiedliche Takte das Versenden der Bilder auseinander läuft. Durch die unterschiedlichen Frequenzen entstehen Überlagerungen, die zu sehr langen und sehr kurzen Übertragungen führen. Dieses Verhalten ist nichtdeterministisch.
Mit maximal 20 ms Latenz liegt diese Konfiguration zwar in den Anforderungen für die Latenz (entspricht 60 % der erlaubten Latenz), jedoch führt der im Vergleich hohe Jitter von 14,67 ms (entspricht 440,1 % des erlaubten Jitters) dazu, dass diese Konfiguration unbrauchbar wird. Verantwortlich hierfür sind die Queues in den Switchen, welche die Nachrichten unpriorisiert behandeln und daher bei einzelnen Frames zu einer extrem hohen Verzögerung führen. Das nichtdeterministische Verhalten wird in der Latenzverteilung (Abbildung 6.13 auf Seite 89) deutlich. Obwohl die meisten Bilder (ca. 40 %) mit 5 - 6 ms Latenz übertragen werden, ist nahezu jeder Bereich bis zu 20 ms vorhanden.

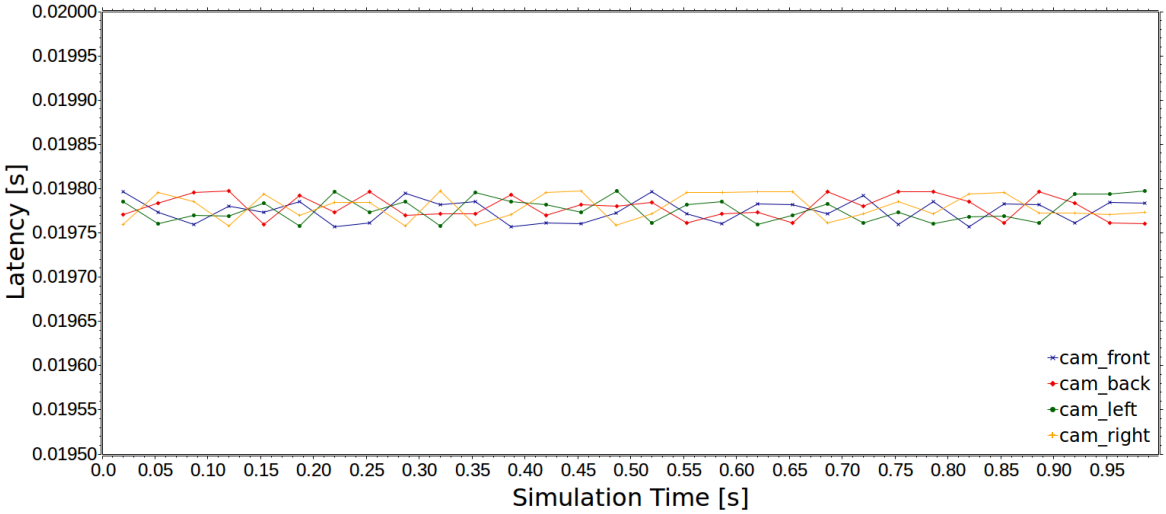

Abbildung 6.11: Skalierter Ausschnitt — Ende-zu-Ende Latenz: Komplettes Bild je Kamera über rate-constrained Nachrichten (OMNeT++-Plot)

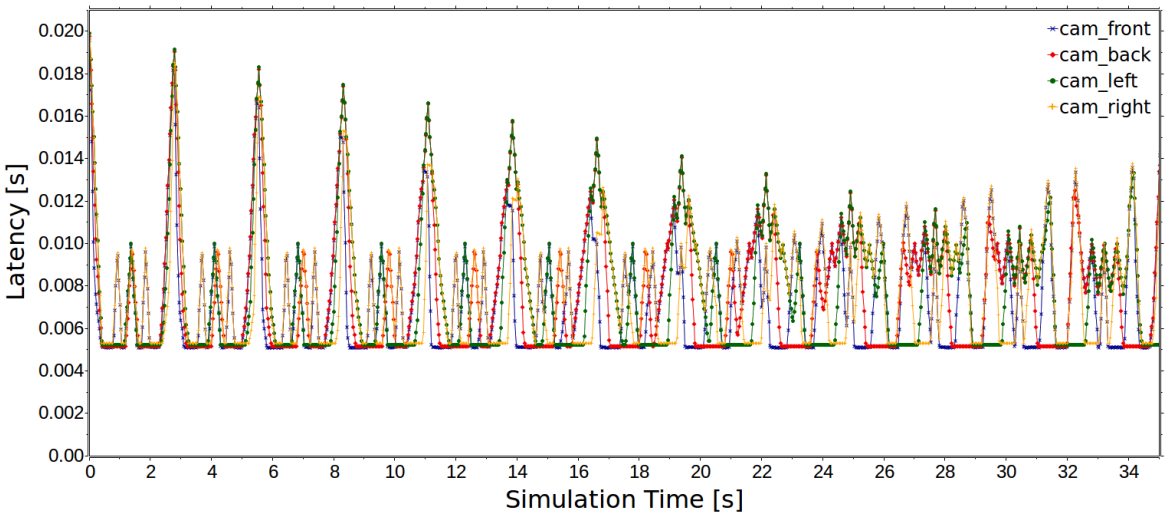

Abbildung 6.12: Ende-zu-Ende Latenz: Komplettes Bild je Kamera über Standard-Ethernet (OMNeT++-Plot)

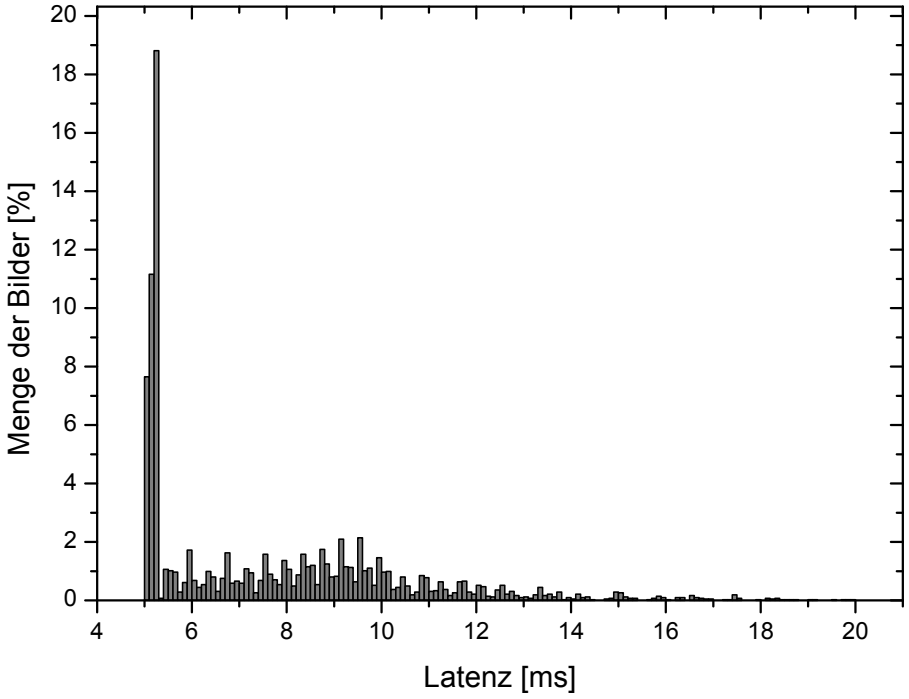

Abbildung 6.13: Latenzverteilung: Komplettes Bild über Standard-Ethernet

Bewertung der Simulationsergebnisse

Auch wenn diese Untersuchung vornehmlich der Überprüfung des Simulationsprozesses und seiner Eignung für die Simulation von Automotive-Anwendungen dient, können mit diesen Ergebnissen bereits erste Aussagen zum Einsatz von time-triggered Ethernet in der kamerabasierten Fahrzeugumfelderfassung getroffen werden.
Die Analyse zeigt, dass es möglich ist, die heute üblichen Bildauflösungen zu übertragen. Zusätzlich ist ausreichend Kapazität im Schedule vorhanden, um eine weitere Kamera, beispielsweise für die Fahrerüberwachung, einzusetzen. Auch die Änderung der Farbkodierung in das RGB-Farbmodell mit 3 Byte je Pixel und damit einer um 50 % höheren Bandbreite ist möglich. Der Vorteil der TTEthernet-basierten Lösung liegt in der Reduzierung von Kabeln im Gegensatz zur direkten Verkabelung zwischen Kamera und ECU wie beispielsweise bei LVDS-Lösungen. Darüber hinaus ist mit ihr die Verteilung eines Videostroms auf mehrere Endsysteme einfach möglich. Durch die Synchronisierung können zudem alle Bilder zum exakt gleichen Zeitpunkt aufgenommen werden.
Es zeigt sich, dass beide Echtzeit-Traffic-Klassen von TTEthernet für die Übertragung geeignet sind. Dabei kann zwischen Konfigurationen mit versetzter Übertragung, also der Übertragung der Bilder verschiedener Kameras nacheinander, und gleichzeitiger Übertragung, in der Frame-weise zwischen den Bildern gewechselt wird, unterschieden werden. Die versetz-

Traffic-Klasse	Konfiguration Übertragungsfolge	Switch-Hops	Latenz Bild erste Kamera	Latenz Bild letzte Kamera	maximaler Jitter
TT	Bilder versetzt	1	5,1 ms	20,4 ms	9,68 µs
TT	Bilder versetzt	3	5,2 ms	20,5 ms	10,04 µs
TT	Bilder gleichzeitig	1	20,0 ms	20,0 ms	36,98 µs
RC	Bilder gleichzeitig	1	19,8 ms	19,8 ms	39,85 µs
BE	Bilder gleichzeitig	1	20,0 ms	20,0 ms	14.671,16 µs
	Anforderungen		33,3 ms	33,3 ms	3.333,33 µs

Tabelle 6.3: Kamerabasierte Fahrzeugumfelderfassung — ausgewählte Simulationsergebnisse im Überblick

te Übertragung eignet sich dabei eher für die serielle, die gleichzeitige Übertragung eher für die parallele Verarbeitung der Bilder in der ECU. Die Simulation mit größeren Topologien zeigt, dass der Einsatz weiterer Switche kein Problem für die Erfüllung der Anforderungen darstellt. Die unsynchronisierte Übertragung mit Standard-Ethernet eignet sich wegen des dabei entstehenden hohen Jitters nicht. Tabelle 6.3 fasst die Ergebnisse zusammen.
Zukünftige Systeme werden eine deutlich höhere Auflösung im Megapixel-Bereich haben. Je Kamera würde dies, sofern keine geeignete Komprimierung eingesetzt wird, zu einer Nettobandbreite von ca. 480 Mbit/s bei YUV- und 720 Mbit/s bei RGB-Farbmodellverwendung führen. In diesem Fall können die Datenströme mehrerer Kameras nicht mehr im Switch zusammenlaufen, da die Bandbreite des Links zur ECU nicht ausreicht. Derzeit werden noch keine Switche mit einer Bandbreite über 1 GBit/s angeboten. Bis Hardware mit höherer Bandbreite angeboten wird, muss die Übertragung der hoch aufgelösten Bilder als Punkt-zu-Punkt Verbindung zwischen Kamera und ECU umgesetzt werden. Dies entspricht den heutigen Lösungen auf Basis von LVDS mit dem bereits erwähnten Mehraufwand in der Verkabelung. Auch bei der Punkt-zu-Punkt Verbindung kann der Videostrom einer Kamera auf mehrere ECUs verteilt werden, indem Switche ihn verteilen.

Bewertung des entwickelten Evaluierungsprozesses

Die Fallstudie zeigt, dass der Evaluierungsprozess bereits gute Ergebnisse erzielt. Nach der Erstellung des Netzwerkmodelles und seiner Konfiguration kann die Simulation mit geringem Aufwand durchgeführt werden. Der größte Zeitaufwand liegt in der Erstellung der Konfiguration. Dafür werden dringend geeignete Werkzeuge benötigt, die den Nutzer bei der Umsetzung des Anwendungsmodells in eine Konfiguration unterstützen.
Mit über 1.600 konfigurierten Nachrichten und einer XML-Konfiguration von zusammen nahezu 200.000 Zeilen (bei sechs Endsystemen und vier Switchen) ist die Netzwerkkonfiguration sehr groß. Es zeigt sich, dass diese extrem großen Schedules sich negativ auf die Si-

mulationsgeschwindigkeit auswirken. Hier sollte die Performance der Simulationsumgebung sowohl beim Parsen von großen Konfigurationsdateien als auch bei der Simulation selbst verbessert werden. Während der Simulation ist insbesondere die Navigation innerhalb des Objektbaumes problematisch; Lookup-Tables können an dieser Stelle die Simulationsgeschwindigkeit signifikant verbessern. Derzeit gibt es in den OMNeT++- und INET-Modellen keine Komponenten, die eine annähernd ähnlich umfangreiche Konfiguration erfordern. Daher können daraus keine geschwindigkeitsverbessernden Konzepte übernommen werden.

Kapitel 7

Zusammenfassung und Fazit

Ziel dieser Arbeit ist es, einen simulationsbasierten Evaluierungsprozess für Untersuchungen an Realtime-Ethernet für den Automotive-Einsatz zu entwickeln und geeignete Metriken, die in diesem Prozess untersucht werden festzulegen.
Dieses abschließende Kapitel fasst die Ergebnisse dieser Arbeit zusammen (Abschnitt 7.1), bewertet die Ergebnisse (7.2) und gibt einen Ausblick auf weiterführende Projekte (7.3).

7.1 Zusammenfassung der Ziele und Ergebnisse

Durch die Zunahme von elektronischen Systemen im Automobil sind die Anforderungen an Kommunikationsverbindungen im Automobil stark gestiegen. Dies motiviert neue Konzepte für den Datenaustausch zwischen den verteilten Steuergeräten. Ein solcher neuer Architekturansatz ist ein Backbone-Netzwerk, das als zentrales Element aller Datenverbindungen hoher Last gewachsen sein muss und dabei eine vorhersagbare, deterministische Nachrichtenübertragung garantiert. Zudem bringen neue Fahrerassistenzsysteme wie die Fahrzeugumfelderkennung, welche mithilfe kamerabasierter Objekterkennung den Fahrer unterstützt, heutige Automotive-Bussysteme an ihre Grenzen. Eine Zunahme der Datenmenge in diesem Segment ist zudem bereits absehbar.
Eine mögliche Technologie für zukünftige Fahrzeugnetzwerke ist Echtzeit-Ethernet. Ethernet ist ein flexibles, hoch skalierbares Netzwerkprotokoll, welches durch Echtzeiterweiterungen um die Funktionalität der deterministischen, zuverlässigen und zeitlich hoch präzisen Paketvermittlung erweitert wird. Um frühzeitig Aussagen zur Eignung von Ethernet-basierter Fahrzeugkommunikation machen zu können, spielt die Simulation von konkreten Anwendungsszenarien eine wichtige Rolle für OEMs und Zulieferer. Ein flexibler Evaluierungsprozess unterstützt dabei die Entwicklung kontinuierlich.
Im Abschnitt Hintergrund (Kapitel 2) werden die Anforderungen an die Datenübertragung im Automobil gezeigt. Sie sind umfangreich und in dieser Kombination einzigartig in der Industrie. Die speziellen Automotive-Bussysteme, welche heute verbreitet eingesetzt werden,

kommen mit den neuen Anforderungen an ihre Grenzen. Die verschiedenen vorgestellten Echtzeit-Ethernet-Technologien versprechen, die Anforderungen zu erfüllen. Das in dieser Arbeit eingesetzte TTEthernet zielt auf die Verwendung in Flug- und Fahrzeugen ab. Durch die Verwendung und Anlehnung an bewährte Technologien wie TTP, FlexRay und AFDX werden Konzepte verwendet, die sich in der Automotive-Branche bereits etabliert haben.
Kapitel 3 erarbeitet die Grundlagen zu den im Evaluierungsprozess betrachteten Metriken. Es zeigt sich, dass Metriken eine hohe Bedeutung für die Bewertung von Kommunikations-Technologien haben. Auf ihrer Basis können Aussagen zum technologischen und wirtschaftlichen Erfolg getroffen werden. Darüber hinaus bilden sie die Kenngrößen für die Bewertung von Konfigurationsparametern und Optimierungen.
Grundsätzlich können Metriken auf verschiedene Art ermittelt werden. Neben der empirischen Untersuchung sind dies vor allem die modellbasierten Techniken. Dabei wird grundsätzlich zwischen analytischen Modellen, welche auf mathematischen Methoden basieren, und Simulationsmodellen unterschieden.
Vorteile bei den analytischen Modellen sind die gute Skalierbarkeit, die einfache Möglichkeit, Auswirkungen von Eingabeparametern zu quantifizieren, und die Geschwindigkeit der analytischen Algorithmen. Damit das Modell berechenbar bleibt, sind jedoch meist Vereinfachungen des Systems notwendig, die sich negativ auf die Abbildungsgenauigkeit auswirken. Der Vorteil der Simulation ist, dass ein System in nahezu jeder gewünschten Tiefe betrachtet werden kann. Zudem ist die Komplexität des Modells nicht wie in den analytischen Modellen durch die mathematische Berechenbarkeit beschränkt. Der Nachteil der Simulation ist dagegen die im Vergleich zu analytischen Modellen hohe Rechenzeit und der erhöhte Entwicklungsaufwand. Da sich bereits in den Vorarbeiten gezeigt hat, dass sich ein analytisches Modell gut für die Abbildung von synchroner Kommunikation eignet, jedoch asynchrone Kommunikation sehr kompliziert zu berechnen ist, ist für den Evaluierungsprozess in dieser Arbeit die Simulation gewählt worden.
Nicht alle Metriken eignen sich für einen simulationsbasierten Evaluierungsprozess. So sind für diese Arbeit die für die Echtzeit-Kommunikation wichtigsten Verlässlichkeits- und Leistungs-Metriken identifiziert und verwendet worden. Diese umfassen neben der Zuverlässigkeit, der Wartbarkeit und der Verfügbarkeit vor allem die Leistungsmetriken Latenz, Jitter und Bandbreite.
Die Einsatzgebiete eines simulationsbasierten Evaluierungsprozesses sind umfangreich und reichen von Proof-of-Concept-Untersuchungen über Software-in-the-loop- und Integrationstests bis zur Schulung. Dabei ist ein möglichst einfacher Ablauf, der den Nutzer in jedem Schritt unterstützt, wichtig. Der in dieser Arbeit entwickelte Prozess basiert auf drei Phasen. In der Einrichtungsphase wird die Simulation aus vorhandenen Konfigurationsformaten eingerichtet. In der Simulationsphase wird die Simulation auf Basis dieser Konfigurationen stimuliert. In der Auswertungsphase bekommt der Nutzer die Simulationsergebnisse in Form eines Analyseberichtes aufbereitet. Dabei wird darauf Wert gelegt, dass die Simulations-

ergebnisse direkt auf die Konfiguration und damit auf die Eigenschaften realer Hardware zurückführbar sind.
Die Herausforderung beim Simulationsmodell liegt insbesondere in der präzisen Umsetzung des Zeitverhaltens. Da TTEthernet ein synchron arbeitendes Protokoll ist, muss eine geeignete Abbildung der lokalen Uhren in jedem Gerät und der in jedem Zeitgeber vorhandenen Clock-Drift gefunden werden. Damit sich das Simulationsmodell möglichst gut in die vorhandenen Strukturen der Simulationsplattform OMNeT++ und dem INET-Framework integriert, ist genau überprüft worden, an welcher Stelle bereits vorhandene Strukturen durch Vererbung erweitert werden können.
Bei der Umsetzung (Kapitel 5) sind insbesondere Probleme, die durch die enge Interaktion mit den TTEthernet-Werkzeugen entstanden sind, gelöst worden. Diese betreffen insbesondere das Einlesen der Konfiguration, welche mit dem Ecore-Format vom standard XML abweicht. Dies macht den Einsatz der Ecore C++ Bibliothek EMF4CPP notwendig, welche noch im Anfangsstadium der Entwicklung ist und daher nicht alle benötigten Funktionen anbietet. Eine Erweiterung der Bibliothek ist daher unumgänglich gewesen.
Bei der Umsetzung der Auswertung des Prozesses wird vollständig auf Standard-Formate wie XML, TeX und das PDF gesetzt. Die Einbindung des Textsatzsystems TeX ermöglicht dabei eine ansprechende Gestaltung der automatisiert generierten Berichte.
Die erzielten Ergebnisse (Kapitel 6) zeigen, dass das Simulationsmodell spezifikationskonform arbeitet. Die Performance-Untersuchung an Modellen innerhalb der erwarteten Topologiegröße für Automotive-Anwendungen zeigt, dass das reine Modell der Komponenten ausreichend gut skaliert. Die benötigte Rechenzeit und der Speicherverbrauch liegen dabei über der Standard-Ethernet-Implementierung des INET-Frameworks und wachsen linear mit der Modellgröße.
In der Verifikation wird auf Basis einer synthetischen Topologie gezeigt, dass das Simulationsmodell eine hohe zeitliche Präzision aufweist. Dazu werden die Simulationsergebnisse mit den Ergebnissen aus dem analytischen Modell und Messungen mit echter TTEthernet-Hardware verglichen.
In der abschließenden Fallstudie wird der Evaluierungsprozess mit einer realen Automotive-Anwendung durchgeführt. Es wird gezeigt, dass sich TTEthernet für die Übertragung von Bilddaten in der kamerabasierten Fahrzeugumfelderkennung eignet. Dazu werden Metriken bei verschiedenen Konfigurationen und Topologien vermessen und mit den Anforderungen verglichen. Es zeigt sich, dass sich die Anforderungen mit best-effort Datenverkehr wegen der fehlenden Synchronisierung nicht erreichen lassen. Die Übertragung als time-triggered oder rate-constrained Nachrichten erfüllen dagegen die Anforderungen. Dabei erreicht die Konfiguration auf Basis von time-triggered Frames wie erwartet die besten Werte für Latenz und Jitter.
Die Fallstudie deckt auf, dass die Simulationskonfiguration bei Netzwerken mit sehr vielen Nachrichten nicht ausreichend skaliert. Hier muss insbesondere der Parser aus dem EMF4CPP-Projekt verbessert werden. Zudem wird dringend ein Werkzeug für die Unterstüt-

zung bei der Erstellung der Konfiguration benötigt. Bei umfangreichen Netzwerken können Konfigurationen mit insgesamt über 200.000 Zeilen XML-Code entstehen. Diese lassen sich derzeit nur mit Scripten erstellen. Dieser Prozess ist jedoch fehleranfällig.

7.2 Bewertung des Evaluierungsprozesses

Diese Arbeit zeigt ein präzises, gut skalierendes Simulationsmodell für Echtzeit-Ethernet-basierte Kommunikation in Fahrzeugen. Durch die sorgfältige Validierung auf Basis von mathematischen Berechnungen in einem analytischen Modell und Messungen auf echter TTEthernet-Hardware kann sichergestellt werden, dass die Ergebnisse der Simulation auf reale Probleme anwendbar sind. Durch die Implementierung der TTEthernet-API kann die Simulation auf die Analyse des Verhaltens von Anwendungen ausgedehnt werden.
Insbesondere während der Untersuchung im Rahmen der Fallstudie sind die Stärken und Schwächen des entwickelten simulationsbasierten Evaluierungsprozesses aufgedeckt worden. So ermöglicht der Prozess durch die Integration der TTEthernet-Werkzeuge ein extrem schnelles Setup. Nachdem eine TTEthernet-Netzwerk-Konfiguration erstellt worden ist, werden nur noch wenige Schritte benötigt, um eine Simulation zu starten. Die dafür benötigten Konfigurationen erstellt der Prozess selbständig aus den eingegebenen Konfigurationsdaten. Während der Simulation erhält der Nutzer sofort Rückmeldung über Konfigurationsfehler. Nach der Anpassung der Konfiguration ist die Simulation sofort wieder einsetzbar. Diese extrem kurzen Zyklen zwischen Konfiguration und Test sind mit realer Hardware nicht möglich. Hier müssen mit jeder Änderung diverse Geräte mit neuen Firmware-Images programmiert werden. Darüber hinaus ist die reale Hardware trotz des möglichen Einsatzes von Netzwerk-Sniffern wie TTE-View eine Black-Box. Die Simulation dagegen ist für den Benutzer vollständig transparent.
Eine derzeit noch bestehende Schwäche ist die Performance der Simulationskonfiguration. Es dauert sehr lange, bis umfangreiche Konfigurationen geparst sind, und auch die Ausführung der Simulation wird durch das verwendete Simulationsmodell deutlich verlangsamt. Bisher sind diese Elemente noch nicht auf Geschwindigkeit optimiert worden, sodass hier deutliche Leistungssteigerungen möglich sind.
Die freie C++-Ecore-Bibliothek EMF4CPP ist umfangreich erweitert und verbessert worden. Diese Änderungen können nun in das Projekt übernommen werden. Damit trägt diese Arbeit auch dazu bei, dieses OpenSource-Projekt zu verbessern, und gibt damit ein Stück der in Anspruch genommenen Leistung der Autoren zurück.
Das Vorgehen in dieser Arbeit entspricht weitgehend den Planungen und erzielt so das gewünschte Ergebnis. Die während des gesamten Masterstudiums durchgeführten Vorarbeiten haben dabei diese Abschlussarbeit optimal vorbereitet. Damit ist das Risiko, welches bei jedem Projekt besteht, auf ein Minimum reduziert worden. Durch die Partitionierung der Problemstellung in kleine Einheiten ist die Komplexität des Problems geringer. Ein Hilfsmittel

der Arbeit ist dabei ein kontinuierlich geführtes „Entwicklungstagebuch“ gewesen, welches die Arbeit begleitet hat.
Um die Nutzbarkeit der Ergebnisse sicherzustellen, ist die abschließende Fallstudie ein wichtiges Element. Sie konnte dank des Kontaktes mit der IAV als Unternehmen aus der Automobilbranche anhand einer realen Automotive-Anwendung durchgeführt werden.

7.3 Ausblick auf zukünftige Arbeiten

Trotz aller Sorgfalt kann eine Abschlussarbeit wie diese kein verkaufsfähiges Produkt hervorbringen. Das Ziel dieser Arbeit ist zu zeigen, dass sich Echtzeit-Ethernet-basierte Vermittlungsstrukturen im Automobil mit simulationsbasierten Evaluierungsprozessen analysieren lassen. Als Ergebnis liefert diese Arbeit einen Proof-of-Concept, welcher auf das Fallbeispiel angewendet bereits aussagekräftige Ergebnisse liefert. Dennoch ist dieser Prozess noch nicht fehlerfrei und sollte um zusätzliche Funktionen erweitert werden.
Fortführende Arbeiten sollten an dieser Stelle ansetzen und Einschränkungen des aktuellen Prozesses beseitigen. Dies betrifft insbesondere die Verbesserung der Skalierbarkeit, welche durch Änderungen im EMF4CPP-Parser und in den Speichermechanismen der Simulationskomponenten erreicht werden kann. Um den Prozess in sich zu überprüfen, wäre neben weiteren Fallstudien auch die spätere Untersuchung der Usability[16] von Interesse.
Konkrete technische Verbesserungen sind beispielsweise die direkte Integration der Konfigurations- und Auswertungswerkzeuge in die OMNeT++-IDE auf Basis von Eclipse-Plugins, die Stimulierung der Simulation mit realistischen Verkehrsflüssen oder die Anbindung von aktuellen Automotive-Bussystemen für die Simulation von Gateways.
Die Veröffentlichung des Simulationsmodells als Open-Source-Komponente ist bereits in Vorbereitung (vgl. CoRE RG, b). Darüber hinaus ist auch die Veröffentlichung des gesamten Evaluierungsprozesses denkbar. Hier sollte jedoch zunächst abgewogen werden, ob das zu erwartende Interesse den Aufwand rechtfertigt.
Auch das in den Vorarbeiten verwendete analytische Modell sollte weiterentwickelt werden. Ein attraktives Einsatzgebiet für dieses Modell ist ein Planungswerkzeug für die Netzwerkkonfiguration. Mit Hilfe des analytischen Modells könnten hier bereits Vorhersagen über die Laufzeiten berechnet werden, welche den Nutzer bei der Erstellung von Schedules unterstützen.
Zukünftige Simulationsarbeiten auf Basis konkreter Automobiltopologien und realistischer Anwendungsdaten begleiten die Analyse, wie time-triggered Ethernet-Systeme in einem zukünftigen Fahrzeug-Backbone umgesetzt werden können. Dabei unterstützt der in dieser Arbeit vorgestellten Evaluierungsprozess und die darin enthaltene Simulationsumgebung die

[16] Die Usability (deutsch: Brauchbarkeit, (Be-)Nutzbarkeit, Bedienbarkeit) bezeichnet die messbare Nutzungsqualität eines Systems bei der Interaktion mit dem Nutzer

Entwicklung von prototypischen Umsetzungen und konkreten Lösungen wie der Konsolidierung aktueller Automotive-Bussysteme zu einer intelligenten Ethernet-basierten Vermittlungsinfrastruktur.

Danksagung

Ich danke der Ingenieurgesellschaft Auto und Verkehr (IAV), insbesondere Herrn Dipl.-Ing. René Röllig und Herrn Dipl.-Ing. Udo Wehner für die Unterstützung bei der in dieser Arbeit durchgeführten Fallstudie zur Echtzeit-Ethernet basierten Datenübertragung in der kamera-basierten Fahrzeugumfelderfassung.
Durch die Unterstützung der IAV konnte der Evaluierungsprozess anhand eines realen Beispiels aus der Automobilindustrie überprüft und gezeigt werden, dass das entwickelte Modell den Anforderungen von Analysen im Automobilbereich genügt.

Weiterhin danke ich der Firma TTTech, dort insbesondere Herrn Dr. Markus Plankensteiner und Herrn Dr. Astrit Ademaj, für die Unterstützung bei Fragen zu TTEthernet und für die unbürokratische Zurverfügungstellung der Konfigurations-Modelle.

Ganz besonders möchte ich meinen Betreuern Herrn Prof. Dr. Franz Korf und Herrn Prof. Dr. Thomas C. Schmidt für die Förderung, die konstruktive Kritik und ihren großen Zeitaufwand danken. Durch ihre Unterstützung wurden die Ergebnisse und Erfolge der Arbeiten in den vergangenen Jahren erst möglich.

Anhang A

Echtzeit-Ethernet-Systeme im Vergleich

Die Tabelle A.1 auf der nächsten Seite basiert auf Hersteller- oder Entwicklerangaben und kann daher vor allem im Bereich der Zeiten nur als Richtwert angesehen werden.

System	Hardware Busteilnehmer	Hardware Switch	min. Zykluszeit	max. Jitter	Zugänglichkeit
Ethernet	standard Controller mit IEEE1588	managed Switch	2-6 ms	ca. 20 µs	offen
Profinet	spezieller Controller (ASIC)	proprietärer Switch	1 ms	1 µs	proprietär
TTEthernet	optional spezieller Controller	FPGA basierter Switch	k.A.	$<$10 µs	offen
SynqNet	spezieller Controller (FPGA)	Linien/Ring Topologie	$<$25 µs	$<$1 µs	offen
Ethercat	spezieller Controller (FPGA/ASIC)	standard Switch	125-250 µs	1 µs	offen
RTnet	standard Controller	Switch oder Hub	k.A.	k.A.	offen
Powerlink	optional spezieller Controller	Hub	400 µs	1 µs	offen
SERCOS III	spezieller Controller (FPGA)	Linien/Ring Topologie	125-250 µs	1 µs	offen

Tabelle A.1: Überblick über Echtzeit-Ethernet-Systeme — Angaben laut Hersteller/Entwickler

Anhang B

Ausschnitte aus dem Modell

In diesem Abschnitt werden Diagramme zum Simulationsmodell gezeigt.

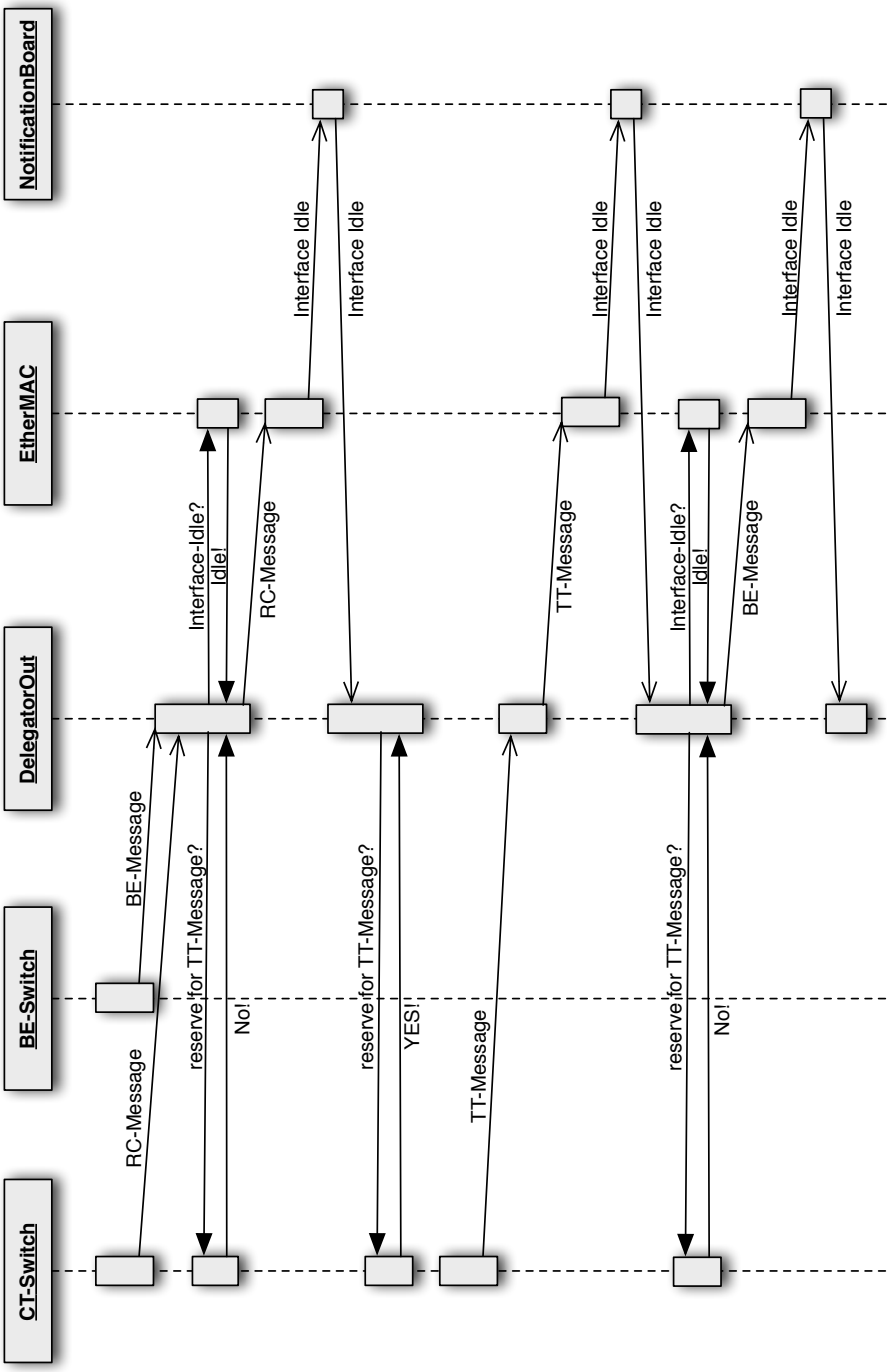

Abbildung B.1: Exemplarische Nachrichtensequenz zwischen den Modulen beim Nachrichtenversand mit NotificationBoard

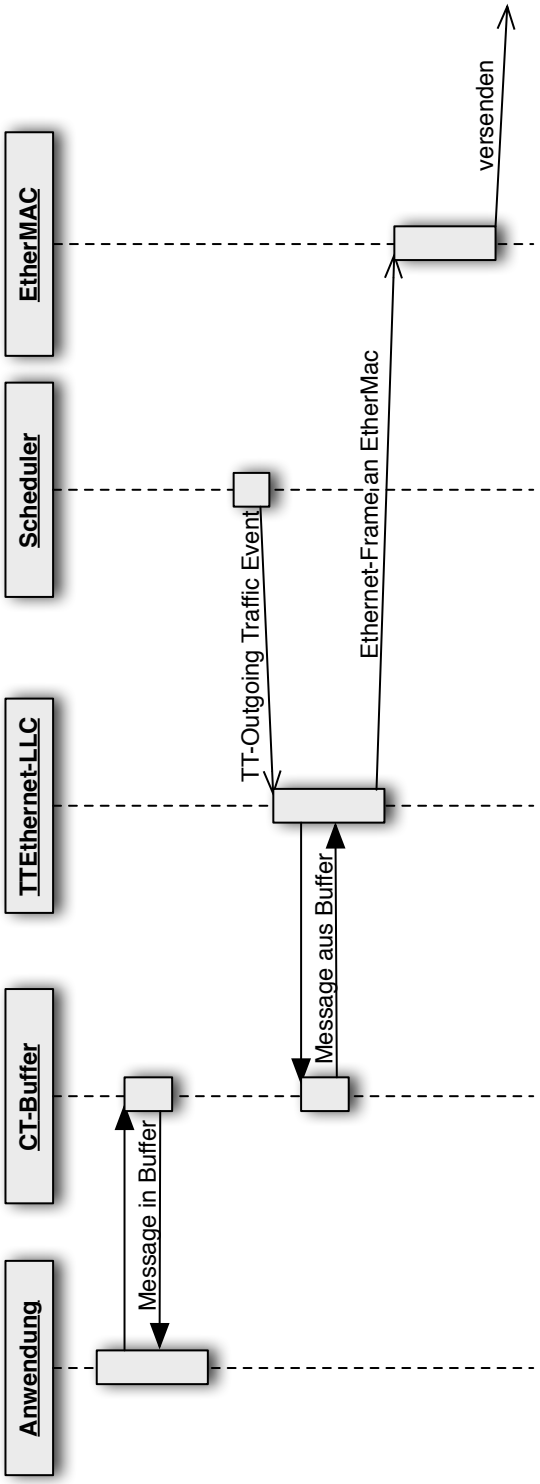

Abbildung B.2: Exemplarische Nachrichtensequenz zwischen den Modulen beim Absenden einer Nachricht im TTEthernet-Host

Anhang C

Beispiel-Listings und Dokumente

Listing C.1: Ausschnitt XML-System-Specification

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <sys:SystemSpecification
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xmlns:sys="http://www.tttech.com/Schema/TTEthernet/System_Specification_V0"
5      xsi:schemaLocation="http://www.tttech.com/Schema/TTEthernet/System_Specification_V0 platform:/
           plugin/com.tttech.tte.models/model/System_Specification_V0.xsd">
6      <metaInformation creationToolVersion="MANUAL" description="Tesfile for Syntactic Tests"/>
7  ...
8      <devices>
9          <!-- switch (TTE_Dev_Switch_8port_100M_V2) -->
10         <device name="switch_0">
11             <port name="P_switch_0_P0"
12                 transmissionSpeed="100Mbps"
13                 staticRxDelay = "0us"
14                 staticTxDelay = "0us"
15                 mediaType     = "default"/>
16             <port name="P_switch_0_P1"
17                 transmissionSpeed="100Mbps"
18                 staticRxDelay = "0us"
19                 staticTxDelay = "0us"
20                 mediaType     = "default"/>
21 ...
22             <port name="P_switch_0_P8"
23                 transmissionSpeed="100Mbps"
24                 staticRxDelay = "0us"
25                 staticTxDelay = "0us"
26                 mediaType     = "default"/>
27             <port name="P_switch_0_PSYNC"
28                 transmissionSpeed="100Mbps"
29                 staticRxDelay = "0us"
30                 staticTxDelay = "0us"
31                 mediaType     = "default"/>
32             <port name="P_switch_0_PMGMT"
33                 transmissionSpeed="100Mbps"
34                 staticRxDelay = "0us"
35                 staticTxDelay = "0us"
36                 mediaType     = "default"/>
37             <capability>managementModule</capability>
```

```
38          </device>
39          <!-- videoserver (TTE_Prot_Layer_100M) -->
40          <device name="videoserver">
41              <port name="P_videoserver_P1"
42                    transmissionSpeed="100Mbps"
43                    staticRxDelay = "0us"
44                    staticTxDelay = "0us"
45                    mediaType     = "default"/>
46  ...
47              <port name="P_videoclient_PHOST"
48                    transmissionSpeed="100Mbps"
49                    staticRxDelay = "0us"
50                    staticTxDelay = "0us"
51                    mediaType     = "default"/>
52              <capability>syncMaster</capability>
53              <capability>managementModule</capability>
54          </device>
55          <!-- videoclient (TTE_Prot_Layer_100M) -->
56          <device name="videoclient">
57              <port name="P_videoclient_P1"
58                    transmissionSpeed="100Mbps"
59                    staticRxDelay = "0us"
60                    staticTxDelay = "0us"
61                    mediaType     = "default"/>
62  ...
63              <port name="P_videoclient_PHOST"
64                    transmissionSpeed="100Mbps"
65                    staticRxDelay = "0us"
66                    staticTxDelay = "0us"
67                    mediaType     = "default"/>
68              <capability>managementModule</capability>
69          </device>
70      </devices>
71      <links>
72          <link name="LNK_P_switch_0_P6->P_videoclient_P1"
73                refSender    ="#//@devices/@device[name='switch_0']/@port[name='P_switch_0_P6']"
74                refReceiver  ="#//@devices/@device[name='videoclient']/@port[name='P_videoclient_P1']"
75                cableLength="1"
76                maxSpeed="1"/>
77          <link name="LNK_P_videoserver_P1->P_switch_0_P4"
78                refSender    ="#//@devices/@device[name='videoserver']/@port[name='P_videoserver_P1']"
79                refReceiver  ="#//@devices/@device[name='switch_0']/@port[name='P_switch_0_P4']"
80                cableLength="1"
81                maxSpeed="1"/>
82          <link name="LNK_P_switch_0_P4->P_videoserver_P1"
83                refSender    ="#//@devices/@device[name='switch_0']/@port[name='P_switch_0_P4']"
84                refReceiver  ="#//@devices/@device[name='videoserver']/@port[name='P_videoserver_P1']"
85                cableLength="1"
86                maxSpeed="1"/>
87          <link name="LNK_P_videoclient_P1->P_switch_0_P6"
88                refSender    ="#//@devices/@device[name='videoclient']/@port[name='P_videoclient_P1']"
89                refReceiver  ="#//@devices/@device[name='switch_0']/@port[name='P_switch_0_P6']"
90                cableLength="1"
91                maxSpeed="1"/>
92      </links>
93  ...
94  </sys:SystemSpecification>
```

Listing C.2: Aus System-Specification (Listing C.1) generierte NED-Datei

```
import ttethernet.nodes.ethernet.TTEEtherSwitch;
import ttethernet.nodes.ethernet.TTEEtherHost;
//
// auto-generated module
//
network NC_Test
{
    @display("bgb=597,357");
    submodules:
        switch_0: TTEEtherSwitch {
            parameters:
                network_configuration = "NC_Test.network_config";
                device_name = "switch_0";
                @display("p=257,156");
            gates:
                ethg[8];
        }
        videoserver: TTEEtherHost {
            parameters:
                network_configuration = "NC_Test.network_config";
                device_name = "videoserver";
                @display("p=80,53;b=115,75");
        }
        videoclient: TTEEtherHost {
            parameters:
                network_configuration = "NC_Test.network_config";
                device_name = "videoclient";
                @display("p=488,276;b=125,83");
        }

    connections allowunconnected:
        // Link: LNK_P_switch_0_P6->P_videoclient_P1 & LNK_P_videoclient_P1->P_switch_0_P6
        switch_0.ethg[6] <--> {datarate=100Mbps; delay=10ns;} <--> videoclient.ethg;
        // Link: LNK_P_videoserver_P1->P_switch_0_P4 & LNK_P_switch_0_P4->P_videoserver_P1
        videoserver.ethg <--> {datarate=100Mbps; delay=10ns;} <--> switch_0.ethg[4];
}
```

Listing C.3: Beispielhafte XML-Datei für Anforderungen

```
<Requirement network="NC_Test" author="Till␣Steinbach" date="2010-02-01␣13:12:01">
        <Requirements>
                <TTRequirement id="1" ctid="100" metric="e2elatency" lowerBound="12us" upperBound="
                    17500us">
                    <MeasureAt>ecu_0</MeasureAt>
                    <Description>Die Latenz von Sensorwert 5 (Time-triggered Nachricht CT-ID:
                        100) gemessen an ecu_0 darf 17,5ms nicht überschreiten und 12us nicht
                        unterschreiten</Description>
                </TTRequirement>
                <TTRequirement id="2" ctid="100" metric="jitter" lowerBound="-1" upperBound="9us">
                        <MeasureAt>ecu_0</MeasureAt>
                        <Description>Der Jitter von Sensorwert 5 (Time-triggered Nachricht CT-ID:
                            100) gemessen an ecu_0 darf 9 Mikrosekunden nicht überschreiten</
                            Description>
                </TTRequirement>
        </Requirements>
        <Statistics>
                <TTBandwidthStatistics id="1"  dir="rx">
                        <MeasureAt>ecu_0/0</MeasureAt>
                        <Description>Eingehende Bandbreite der TT-Nachrichten an ecu_0, Port 0</
                            Description>
                </TTBandwidthStatistics>
        <Statistics/>
</Requirement>
```

Listing C.4: Beispielhafter XML-Analysebericht

```
<Analysis network="NC_Test" author="Till␣Steinbach" date="2010-08-13␣18:52:13">
        <Requirements>
                <TTRequirement id="1" ctid="100" metric="e2elatency" lowerBound="12us" upperBound="
                    17500us">
                    <MeasureAt>ecu_0</MeasureAt>
                    <Description>Die Latenz von Sensorwert 5 (Time-triggered Nachricht CT-ID:
                        100) gemessen an ecu_0 darf 17,5ms nicht überschreiten und 12us nicht
                        unterschreiten</Description>
                    <RequirementResult passed="true" lowerBound="24158ns" upperBound="14696743ns"
                        />
                    <SimPlot file="e2e_ct100_ecu0.png" caption="Ende-zu-Ende␣Latenz␣Sensorwert␣5␣
                        gemessen␣an␣ecu_0"/>
                </TTRequirement>
                <TTRequirement id="2" ctid="100" metric="jitter" upperBound="9us">
                        <MeasureAt>ecu_0</MeasureAt>
                        <Description>Der Jitter von Sensorwert 5 (Time-triggered Nachricht CT-ID:
                            100) gemessen an ecu_0 darf 9 Mikrosekunden nicht überschreiten</
                            Description>
                        <RequirementResult passed="false" value="9032ns"/>
                </TTRequirement>
        </Requirements>
        <Statistics>
                <TTBandwidthStatistics id="1" dir="rx">
                        <MeasureAt>ecu_0/0</MeasureAt>
                        <Description>Durchschnittliche eingehende Bandbreite der TT-Nachrichten an
                            ecu_0, Port 0</Description>
                        <StatisticResult value="1.578Mbit/s" />
                </TTBandwidthStatistics>
        </Statistics>
</Analysis>
```

Listing C.5: XSL-Transformation Analysebericht zu TeXML

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<!-- convert "document": create header and continue -->
<xsl:template match="Analysis">
        <TeXML>
                <!-- create header -->
                <TeXML escape="0">
\documentclass[13pt,german,a4]{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[unicode]{hyperref}
\usepackage{color}
\usepackage{a4wide}
\usepackage{graphicx}
\usepackage[absolute,verbose,overlay]{textpos}
                </TeXML>
                <!-- process content -->
                <env name="document">
                    <TeXML escape="0">
\begin{textblock}{1}(2,0.7)
      \includegraphics[width=6cm]{CoRE_logo.pdf}
\end{textblock}
                    </TeXML>
                        <cmd name="title"><parm>Analysis Report for Network: <xsl:apply-templates
                            select="@network"/></parm></cmd>
                        <cmd name="date"><parm><xsl:apply-templates select="@date"/></parm></cmd>
                        <cmd name="author"><parm><xsl:apply-templates select="@author"/></parm></cmd>
                        <cmd name="maketitle" gr="0" />
                        <xsl:apply-templates/>
                </env>
        </TeXML>
</xsl:template>

<xsl:template match="Requirements">
        <cmd name="section"><parm>Requirements</parm></cmd>
        <xsl:apply-templates/>
</xsl:template>

<xsl:template match="Statistics">
        <cmd name="section"><parm>Statistics</parm>
        </cmd><xsl:apply-templates/>
</xsl:template>

<xsl:template match="TTRequirement">
  <TeXML>
        <cmd name="subsection"><parm>Requirement No: <xsl:apply-templates select="@id"/></parm></cmd>
        <env name="itemize">
                <cmd name="item" gr="0" /><cmd name="textbf"><parm>Metric: </parm></cmd><
                    xsl:apply-templates select="@metric"/>
                <xsl:choose>
                        <xsl:when test="./@lowerBound">
                        <cmd name="item" gr="0" /><cmd name="textbf"><parm>lower Bound: </parm></cmd>
                            <xsl:apply-templates select="@lowerBound"/>
                        </xsl:when>
                </xsl:choose>
                <xsl:choose>
                        <xsl:when test="./@upperBound">
```

```
55                          <cmd name="item" gr="0" /><cmd name="textbf"><parm>upper Bound: </parm></cmd>
                                <xsl:apply-templates select="@upperBound"/>
56                          </xsl:when>
57                  </xsl:choose>
58                  <xsl:apply-templates/>
59          </env>
60    </TeXML>
61  </xsl:template>
62
63  <xsl:template match="Description">
64          <cmd name="item" gr="0" /><cmd name="textbf"><parm>Description: </parm></cmd><
                xsl:apply-templates/>
65  </xsl:template>
66
67  <xsl:template match="MeasureAt">
68          <cmd name="item" gr="0" /><cmd name="textbf"><parm>Measured At: </parm></cmd><
                xsl:apply-templates/>
69  </xsl:template>
70
71  <xsl:template match="SimPlot">
72          <cmd name="item" gr="0" /><cmd name="textbf"><parm>Chart: </parm></cmd>see figure <cmd name="
                ref"><parm>fig:<TeXML escape="0"><xsl:apply-templates select="@file"/></TeXML></parm></
                cmd>
73          <env name="figure">
74                  <opt>!h</opt>
75                  <cmd name="centering" gr="0" />
76                  <cmd name="includegraphics"><opt><TeXML escape="0">width=1\columnwidth</TeXML></opt><
                        parm><TeXML escape="0"><xsl:apply-templates select="@file"/></TeXML></parm></cmd
                        >
77                  <cmd name="caption"><parm><xsl:apply-templates select="@caption"/></parm></cmd>
78                  <cmd name="label"><parm>fig:<TeXML escape="0"><xsl:apply-templates select="@file"/></
                        TeXML></parm></cmd>
79          </env>
80  </xsl:template>
81
82  <xsl:template match="RequirementResult">
83          <xsl:choose>
84                  <xsl:when test="./@passed = 'true'">
85                  <cmd name="item" gr="0"/><cmd name="textbf"><parm>Requirement complied: <cmd name="
                        textcolor"><parm>green</parm><parm>yes</parm></cmd></parm></cmd>
86                  </xsl:when>
87                  <xsl:otherwise>
88                  <cmd name="item" gr="0"/><cmd name="textbf"><parm>Requirement complied: <cmd name="
                        textcolor"><parm>red</parm><parm>no</parm></cmd></parm></cmd>
89                  </xsl:otherwise>
90          </xsl:choose>
91          <xsl:choose>
92                  <xsl:when test="./@value">
93                      <xsl:choose>
94                              <xsl:when test="./@passed = 'true'">
95                              <cmd name="item" gr="0" /><cmd name="textbf"><parm>Simulated Value
                                    was: </parm></cmd><xsl:apply-templates select="@value"/>
96                              </xsl:when>
97                              <xsl:otherwise>
98                              <cmd name="item" gr="0" /><cmd name="textbf"><parm>Simulated Value
                                    was: <cmd name="textcolor"><parm>red</parm><parm><
                                    xsl:apply-templates select="@value"/></parm></cmd></parm></cmd>
99                              </xsl:otherwise>
100         </xsl:choose>
```

```
101                 </xsl:when>
102         </xsl:choose>
103         <xsl:choose>
104                 <xsl:when test="./@upperBound">
105                 <cmd name="item" gr="0" /><cmd name="textbf"><parm>Simulated upper bound was: </parm>
                        </cmd><xsl:apply-templates select="@upperBound"/>
106                 </xsl:when>
107         </xsl:choose>
108         <xsl:choose>
109                 <xsl:when test="./@lowerBound">
110                 <cmd name="item" gr="0" /><cmd name="textbf"><parm>Simulated lower bound was: </parm>
                        </cmd><xsl:apply-templates select="@lowerBound"/>
111                 </xsl:when>
112         </xsl:choose>
113 </xsl:template>
114
115 <xsl:template match="TTBandwidthStatistics">
116   <TeXML>
117         <cmd name="subsection"><parm>Statistic No: <xsl:apply-templates select="@id"/></parm></cmd>
118         <env name="itemize">
119                 <xsl:apply-templates/>
120         </env>
121   </TeXML>
122 </xsl:template>
123
124 <xsl:template match="StatisticResult">
125         <xsl:choose>
126                 <xsl:when test="./@value">
127                 <cmd name="item" gr="0" /><cmd name="textbf"><parm>Simulated Value was: </parm></cmd>
                        <xsl:apply-templates select="@value"/>
128                 </xsl:when>
129         </xsl:choose>
130         <xsl:choose>
131                 <xsl:when test="./@upperBound">
132                 <cmd name="item" gr="0" /><cmd name="textbf"><parm>Simulated upper bound was: </parm>
                        </cmd><xsl:apply-templates select="@upperBound"/>
133                 </xsl:when>
134         </xsl:choose>
135         <xsl:choose>
136                 <xsl:when test="./@lowerBound">
137                 <cmd name="item" gr="0" /><cmd name="textbf"><parm>Simulated lower bound was: </parm>
                        </cmd><xsl:apply-templates select="@lowerBound"/>
138                 </xsl:when>
139         </xsl:choose>
140 </xsl:template>
141
142 </xsl:stylesheet>
```

Listing C.6: Beispielhafter XML-Analysebericht in TeXML-Syntax

```
1  <?xml version="1.0"?>
2  <TeXML><TeXML escape="0">
3  \documentclass[13pt,german,a4]{article}
4  \usepackage[T1]{fontenc}
5  \usepackage[utf8]{inputenc}
6  \usepackage[unicode]{hyperref}
7  \usepackage{color}
8  \usepackage{a4wide}
9  \usepackage{graphicx}
10 \usepackage[absolute,verbose,overlay]{textpos}
11                 </TeXML><env name="document"><TeXML escape="0">
12 \begin{textblock}{1}(2,0.7)
13       \includegraphics[width=6cm]{CoRE_logo.pdf}
14 \end{textblock}
15                     </TeXML><cmd name="title"><parm>Analysis Report for Network: NC_Test</parm></cmd
                           ><cmd name="date"><parm>2010-08-13 18:52:13</parm></cmd><cmd name="author"><
                           parm>Till Steinbach</parm></cmd><cmd name="maketitle" gr="0"/>
16         <cmd name="section"><parm>Requirements</parm></cmd>
17                 <TeXML><cmd name="subsection"><parm>Requirement No: 1</parm></cmd><env name="itemize"
                         ><cmd name="item" gr="0"/><cmd name="textbf"><parm>Metric: </parm></cmd>
                         e2elatency<cmd name="item" gr="0"/><cmd name="textbf"><parm>lower Bound: </parm
                         ></cmd>12us<cmd name="item" gr="0"/><cmd name="textbf"><parm>upper Bound: </parm
                         ></cmd>17500us
18                           <cmd name="item" gr="0"/><cmd name="textbf"><parm>Measured At: </parm></cmd>
                               ecu_0
19                           <cmd name="item" gr="0"/><cmd name="textbf"><parm>Description: </parm></cmd>
                               Die Latenz von Sensorwert 5 (Time-triggered Nachricht CT-ID: 100)
                               gemessen an ecu_0 darf 17,5ms nicht überschreiten und 12us nicht
                               unterschreiten
20                           <cmd name="item" gr="0"/><cmd name="textbf"><parm>Requirement complied: <cmd
                               name="textcolor"><parm>green</parm><parm>yes</parm></cmd></parm></cmd><
                               cmd name="item" gr="0"/><cmd name="textbf"><parm>Simulated upper bound
                               was: </parm></cmd>14696743ns<cmd name="item" gr="0"/><cmd name="textbf"
                               ><parm>Simulated lower bound was: </parm></cmd>24158ns
21                           <cmd name="item" gr="0"/><cmd name="textbf"><parm>Chart: </parm></cmd>see
                               figure <cmd name="ref"><parm>fig:<TeXML escape="0">e2e_ct100_ecu0.png</
                               TeXML></parm></cmd><env name="figure"><opt>!h</opt><cmd name="centering"
                                gr="0"/><cmd name="includegraphics"><opt><TeXML escape="0">width=1\
                               columnwidth</TeXML></opt><parm><TeXML escape="0">e2e_ct100_ecu0.png</
                               TeXML></parm></cmd><cmd name="caption"><parm>Ende-zu-Ende Latenz
                               Sensorwert 5 gemessen an ecu_0</parm></cmd><cmd name="label"><parm>fig:<
                               TeXML escape="0">e2e_ct100_ecu0.png</TeXML></parm></cmd></env>
22                 </env></TeXML>
23                 <TeXML><cmd name="subsection"><parm>Requirement No: 2</parm></cmd><env name="itemize"
                         ><cmd name="item" gr="0"/><cmd name="textbf"><parm>Metric: </parm></cmd>jitter<
                         cmd name="item" gr="0"/><cmd name="textbf"><parm>upper Bound: </parm></cmd>9us
24                           <cmd name="item" gr="0"/><cmd name="textbf"><parm>Measured At: </parm></cmd>
                               ecu_0
25                           <cmd name="item" gr="0"/><cmd name="textbf"><parm>Description: </parm></cmd>
                               Der Jitter von Sensorwert 5 (Time-triggered Nachricht CT-ID: 100)
                               gemessen an ecu_0 darf 9 Mikrosekunden nicht überschreiten
26                           <cmd name="item" gr="0"/><cmd name="textbf"><parm>Requirement complied: <cmd
                               name="textcolor"><parm>red</parm><parm>no</parm></cmd></parm></cmd><cmd
                               name="item" gr="0"/><cmd name="textbf"><parm>Simulated Value was: <cmd
                               name="textcolor"><parm>red</parm><parm>9032ns</parm></cmd></parm></cmd>
27                 </env></TeXML>
28
29         <cmd name="section"><parm>Statistics</parm></cmd>
```

```
30                  <TeXML><cmd name="subsection"><parm>Statistic No: 1</parm></cmd><env name="itemize">
31                          <cmd name="item" gr="0"/><cmd name="textbf"><parm>Measured At: </parm></cmd>
                                ecu_0/0
32                          <cmd name="item" gr="0"/><cmd name="textbf"><parm>Description: </parm></cmd>
                                Durchschnittliche eingehende Bandbreite der TT-Nachrichten an ecu_0,
                                Port 0
33                          <cmd name="item" gr="0"/><cmd name="textbf"><parm>Simulated Value was: </parm
                                ></cmd>1.578 Mbit/s
34                  </env></TeXML>
35
36  </env></TeXML>
```

Listing C.7: Beispielhafter XML-Analysebericht in TeX-Syntax

```
\documentclass[13pt,german,a4]{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[unicode]{hyperref}
\usepackage{color}
\usepackage{a4wide}
\usepackage{graphicx}
\usepackage[absolute,verbose,overlay]{textpos}
\begin{document}
\begin{textblock}{1}(2,0.7)
      \includegraphics[width=6cm]{CoRE_logo.pdf}
\end{textblock} \title{Analysis Report for Network: NC\_Test}\date{2010-08-13 18:52:13}\author{Till
    Steinbach}\maketitle
\section{Requirements} \subsection{Requirement No: 1}
\begin{itemize}
\item \textbf{Metric: }e2elatency\item \textbf{lower Bound: }12us\item
\textbf{upper Bound: }17500us \item \textbf{Measured At: }ecu\_0
\item \textbf{Description: }Die Latenz von Sensorwert 5 (Time-triggered Nachricht CT-ID: 100)
    gemessen an ecu\_0 darf 17,5ms nicht \"{u}berschreiten und 12us nicht unterschreiten
\item \textbf{Requirement complied: }\textcolor{green}{yes}}\item
\textbf{Simulated upper bound was: }14696743ns\item \textbf{Simulated lower bound was:
}24158ns \item \textbf{Chart: }see figure \ref{fig:e2e_ct100_ecu0.png}
\begin{figure}[!h]
\centering \includegraphics[width=1\columnwidth]{e2e_ct100_ecu0.png}\caption{Ende-zu-Ende Latenz
    Sensorwert 5 gemessen an ecu\_0}\label{fig:e2e_ct100_ecu0.png}
\end{figure}
\end{itemize}
\subsection{Requirement No: 2}
\begin{itemize}
\item \textbf{Metric: }jitter\item \textbf{upper Bound: }9us \item
\textbf{Measured At: }ecu\_0 \item \textbf{Description: }Der Jitter von Sensorwert 5 (Time-triggered
    Nachricht CT-ID: 100) gemessen an ecu\_0 darf 9 Mikrosekunden nicht \"{u}berschreiten
\item \textbf{Requirement complied: }\textcolor{red}{no}}\item \textbf{Simulated Value was:
\textcolor{red}{9032ns}}
\end{itemize}
\section{Statistics} \subsection{Statistic No: 1}
\begin{itemize}
\item \textbf{Measured At: }ecu\_0/0 \item \textbf{Description:
}Durchschnittliche eingehende Bandbreite der TT-Nachrichten an ecu\_0, Port 0
\item \textbf{Simulated Value was: }1.578Mbit/s
\end{itemize}
\end{document}
```

Analysis Report for Network: NC_Test

Till Steinbach

2010-08-13 18:52:13

1 Requirements

1.1 Requirement No: 1

- **Metric:** e2elatency
- **lower Bound:** 12us
- **upper Bound:** 17500us
- **Measured At:** ecu_0
- **Description:** Die Latenz von Sensorwert 5 (Time-triggered Nachricht CT-ID: 100) gemessen an ecu_0 darf 17,5ms nicht überschreiten und 12us nicht unterschreiten
- **Requirement complied:** yes
- **Simulated upper bound was:** 14696743ns
- **Simulated lower bound was:** 24158ns
- **Chart:** see figure 1

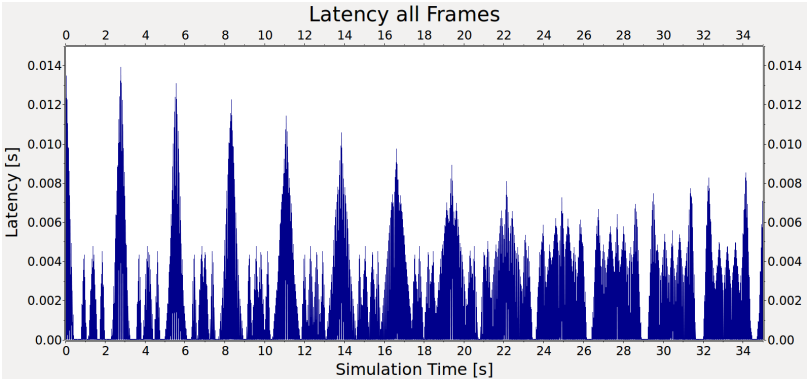

Figure 1: Ende-zu-Ende Latenz Sensorwert 5 gemessen an ecu_0

Abbildung C.1: Beispielhafter Analyse-Bericht als PDF (Seite 1 von 2)

1.2 Requirement No: 2

- **Metric:** jitter
- **upper Bound:** 9us
- **Measured At:** ecu_0
- **Description:** Der Jitter von Sensorwert 5 (Time-triggered Nachricht CT-ID: 100) gemessen an ecu_0 darf 9 Mikrosekunden nicht überschreiten
- **Requirement complied:** no
- **Simulated Value was:** 9032ns

2 Statistics

2.1 Statistic No: 1

- **Measured At:** ecu_0/0
- **Description:** Durchschnittliche eingehende Bandbreite der TT-Nachrichten an ecu_0, Port 0
- **Simulated Value was:** 1.578Mbit/s

Abbildung C.2: Beispielhafter Analyse-Bericht als PDF (Seite 2 von 2)

Literaturverzeichnis

[Aeronautical Radio Incorporated 2002] AERONAUTICAL RADIO INCORPORATED: Aircraft Data Network / ARINC. Annapolis, Maryland, 2002 (664). – Standard

[Aeronautical Radio Incorporated 2009] AERONAUTICAL RADIO INCORPORATED: Aircraft Data Network, Part 7, Avionics Full-Duplex Switched Ethernet Network / ARINC. 2009 (ARINC Report 664P7-1). – Standard

[Altera] ALTERA: *Cyclone III FPGA*. – URL `http://www.altera.com/products/devices/cyclone3/`. – Zugriffsdatum: 2011-01-19

[Apache Software Foundation] APACHE SOFTWARE FOUNDATION: *Xerces-C++ XML-Parser*. – URL `http://xerces.apache.org/xerces-c/`

[AUDI AG, TTTech Automotive GmbH 2009] AUDI AG, TTTECH AUTOMOTIVE GMBH: *FlexRay-Based Optimization of Data Communication: Audi A8 - The Sportiest Sedan in the Luxury Class*. Dezember 2009. – URL `http://www.tttech-automotive.com/fileadmin/content/pdf/Audi-Casestudy/TTTech-Audi-Casestudy-A8.pdf`. – Zugriffsdatum: 2011-01-18

[AUTOSAR Development Cooperation] AUTOSAR DEVELOPMENT COOPERATION: *AUTomotive Open System ARchitecture*. – URL `http://www.autosar.org`

[Badstübner 2008] BADSTÜBNER, Jens: Kollaps im Bordnetz: Schluss mit Can, Lin und Flexray. In: *KFZ-Betrieb* (2008), Nr. 17, S. 68–70

[Bartols 2010] BARTOLS, Florian: *Leistungsmessung von Time-Triggered Ethernet Komponenten unter harten Echtzeitbedingungen mithilfe modifizierter Linux-Treiber*. Hamburg, HAW Hamburg, Bachelorthesis, Juli 2010. – Bachelorthesis

[Bartols u. a. 2011] BARTOLS, Florian ; STEINBACH, Till ; KORF, Franz ; SCHMIDT, Thomas C.: Performance Analysis of Time-Triggered Ether-Networks Using Off-The-Shelf-Components. In: *14th IEEE International Symposium on Object/Component/Service-oriented Real-time Distributed Computing*, 2011. – to appear

[Behrends 2000] BEHRENDS, Ehrhard: *Introduction to Markov chains. With special emphasis on rapid mixing.* Braunschweig : Vieweg, 2000. – ISBN 3-528-06986-4

[Belschner u. a. 2000] BELSCHNER, Ralf ; BERWANGER, Josef ; BRACKLO, Claas ; EBNER, Christian ; HEDENETZ, Bernd ; KUFFNER, Walter ; LOHRMANN, Peter ; MINUTH, Jürgen ; PELLER, Martin ; SCHEDL, Anton ; SEEFRIED, Volker: Anforderungen an ein zukünftiges Bussystem für fehlertolerante Anwendungen aus Sicht Kfz-Hersteller. In: *VDI-Berichte* 1547 (2000), S. 23–41. – ISBN 3-18-091547-1

[Benz 2004] BENZ, Stefan: *Eine Entwicklungsmethodik für sicherheitsrelevante Elektroniksysteme im Automobil*. Karlsruhe, Universität Karlsruhe (TH), Dissertation, April 2004

[Bruckmeier 2010] BRUCKMEIER, Robert: *Ethernet for Automotive Applications.* Vortrag. Juni 2010. – URL `http://www.freescale.com/files/ftf_2010/Americas/WBNR_FTF10_AUT_F0558.pdf`. – Zugriffsdatum: 2010-12-10

[Charara u. a. 2006] CHARARA, Hussein ; SCHARBARG, Jean-Luc ; ERMONT, Jérôme ; FRABOUL, Christian: Methods for bounding end-to-end delays on an AFDX network. In: *18th Euromicro Conference on Real-Time Systems*, 2006. – ISSN 1068-3070

[Combs] COMBS, Gerald: *Wireshark*. – URL `http://www.wireshark.org`. – Zugriffsdatum: 2011-01-23

[CoRE RG a] CORE RG: *Communication over Real-time Ethernet*. – URL `http://www.informatik.haw-hamburg.de/core.html`

[CoRE RG b] CORE RG: *TTE for INET*. – URL `http://www.informatik.haw-hamburg.de/tte4inet.html`

[Şekercioğlu u. a. 2003] ŞEKERCIOĞLU, Ahmet Y. ; VARGA, András ; EGAN, Gregory K.: Parallel simulation made easy with OMNeT++. In: *ESS 2003: 15th European Simulation Symposium*, 2003

[D'Ambrosia 2010] D'AMBROSIA, John: 100 gigabit Ethernet and beyond [Commentary]. In: *Communications Magazine, IEEE* 48 (2010), März, Nr. 3, S. S6 – S13. – ISSN 0163-6804

[Decotignie 2005] DECOTIGNIE, Jean-Dominique: Ethernet-Based Real-Time and Industrial Communications. In: *Proceedings of the IEEE* 93 (2005), Juni, Nr. 6, S. 1102 – 1117. – ISSN 0018-9219

[Dieumo Kenfack 2010a] DIEUMO KENFACK, Hermand: *Erzeugung von charakteristischen Last-Profilen zur simulationsgestützten Analyse von TTEthernet*. August 2010. – Anwendungen 1 Ausarbeitung

[Dieumo Kenfack 2010b] DIEUMO KENFACK, Hermand: *TTEthernet Protokollstack für OMNeT++*. November 2010. – Projekt 1 Bericht

[Dohmke 2002] DOHMKE, Thomas: Bussysteme im Automobil: CAN, FlexRay und MOST / Technische Universität Berlin, DaimlerChrysler AG. 2002. – Forschungsbericht

[Eclipse Foundation a] ECLIPSE FOUNDATION: *Eclipse IDE*. – URL `http://www.eclipse.org/`. – Zugriffsdatum: 2011-01-29

[Eclipse Foundation b] ECLIPSE FOUNDATION: *Eclipse Modeling Framework*. – URL `http://www.eclipse.org/modeling/emf/`. – Zugriffsdatum: 2011-01-29

[Eclipse Foundation c] ECLIPSE FOUNDATION: *Xpand*. – URL `http://wiki.eclipse.org/Xpand`. – Zugriffsdatum: 2011-01-29

[Elektronik 2007] ELEKTRONIK: Offenes Netzwerk: Zeitgesteuertes Ethernet für Industrie, Auto und Avioni. In: *Elektronik* 56 (2007), Nr. 24, S. 8–9

[Erramilli u. a. 2002] ERRAMILLI, Ashok ; ROUGHAN, Matthew ; VEITCH, Darryl ; WILLINGER, Walter: Self-similar traffic and network dynamics. In: *Proceedings of the IEEE* 90 (2002), Mai, Nr. 5, S. 800–819. – ISSN 0018-9219

[EtherCAT Technology Group] ETHERCAT TECHNOLOGY GROUP: *EtherCAT*. – URL `http://www.ethercat.org`

[Ethernet POWERLINK Standardization Group] ETHERNET POWERLINK STANDARDIZATION GROUP: *Powerlink*. – URL `http://www.ethernet-powerlink.org/`. – Zugriffsdatum: 2011-02-07

[Fachinformationszentrum Karlsruhe] FACHINFORMATIONSZENTRUM KARLSRUHE: *FIZ AutoDoc*. – URL `http://www.fiz-karlsruhe.de/fiz_autodoc.html`. – Zugriffsdatum: 2011-02-23

[Felser 2005] FELSER, Max: *Real-Time Ethernet: Standards and PROFINET*. Bd. IFAC Summer School in Prague 2005. International Federation of Automatic Control, 2005

[Ferrari u. a. 2008] FERRARI, P. ; FLAMMINI, A. ; RINALDI, S. ; GADERER, G.: Evaluation of clock synchronization accuracy of coexistent Real-Time Ethernet protocols. In: *IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, September 2008, S. 87–91

[Ferrari u. a. 2006] FERRARI, P. ; FLAMMINI, A. ; VITTURI, S.: Performance analysis of PROFINET networks. In: *Computer Standards & Interfaces* 28 (2006), April, S. 369–385. – ISSN 0920-5489

[FlexRay Consortium 2005] FLEXRAY CONSORTIUM: Protocol Specification / FlexRay Consortium. Stuttgart, Dezember 2005 (2.1). – Specification

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

[GE Fanuc Intelligent Platforms 2009] GE FANUC INTELLIGENT PLATFORMS: *TTEthernet - A Powerful Network Solution for Advanced Integrated Systems*. August 2009. – URL `http://www.ge-ip.com/library/detail/12014`. – Zugriffsdatum: 2011-01-18. – GFT-751

[Greifeneder und Frey 2007] GREIFENEDER, Jürgen ; FREY, Georg: Analyse netzbasierter Automatisierungssysteme. In: *VDI-Berichte* 1980 (2007), S. 23–33

[Grillinger u. a. 2006] GRILLINGER, Petr ; ADEMAJ, Astrit ; STEINHAMMER, Klaus ; KOPETZ, Hermann: Software Implementation of Time-Triggered Ethernet Controller. In: *Workshop on Factory Communication Systems*, 2006, S. 145–150. – ISBN 1-4244-0379-0

[Guérin und Peris 1999] GUÉRIN, Roch ; PERIS, Vinod: Quality-of-service in packet networks: basic mechanisms and directions. In: *Computer Networks* 31 (1999), Februar, S. 169–189. – ISSN 1389-1286

[Haas und Zorn 1995] HAAS, Martin ; ZORN, Werner: *Handbuch der Informatik*. Bd. 2: *Methodische Leistungsanalyse von Rechensystemen*. München : Oldenbourg, 1995. – ISBN 3-486-20779-2

[Hammerschmidt 2007] HAMMERSCHMIDT, Christoph: BMW brings Internet protocol under the hood. In: *EE Times Europe* (2007). – URL `http://www.eetimes.com/showArticle.jhtml?articleID=204300325`. – Zugriffsdatum: 2010-12-10

[Hillebrand u. a. 2007] HILLEBRAND, Joachim ; RAHMANI, Mehrnoush ; BOGENBERGER, Richard ; STEINBACH, Eckehard: Coexistence of Time-Triggered and Event-Triggered Traffic in Switched Full-Duplex Ethernet Networks. In: *International Symposium on Industrial Embedded Systems, 2007. SIES '07.*, Juli 2007, S. 217–224

[Honeywell International] HONEYWELL INTERNATIONAL: . – URL `http://www.honeywell.com`. – Zugriffsdatum: 2011-01-23

[Hufflen 2009] HUFFLEN, Jean-Michel: Processing Computed Texts. In: *ArsTeXnica* 8 (2009), Oktober, S. 102–109

[INCHRON GmbH] INCHRON GMBH: *chronSIM*. – URL `http://www.inchron.de/chronsim.html`. – Zugriffsdatum: 2011-01-17

[Ingenieurgesellschaft Auto und Verkehr GmbH] INGENIEURGESELLSCHAFT AUTO UND VERKEHR GMBH: . – URL `http://www.iav.de`. – Zugriffsdatum: 2011-02-12

[Institute of Electrical and Electronics Engineers 1997] INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS: IEEE 802.5: Token Ring Access Method and Physical Layer Specifications / IEEE. 1997 (IEEE 802.5-1997). – Standard

[Institute of Electrical and Electronics Engineers 2002] INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS: IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems / IEEE. 2002 (IEEE Std. 1588). – Standard. – ISBN 0-7381-3369-8

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

[International Organization for Standardization 2003] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION: Road vehicles – Controller area network (CAN) / ISO. Genf, 2003 (11898). – ISO

[International Organization for Standardization 2004] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION: Road vehicles – Controller area network (CAN) - Part 4: Time-triggered communication / ISO. Genf, 2004 (11898-4:2004). – ISO

[J-Sim Project] J-SIM PROJECT: *J-Sim*. – URL `http://sites.google.com/site/jsimofficial/`

[Jasperneite 2002] JASPERNEITE, Jürgen: *Leistungsbewertung eines lokalen Netzwerkes mit Class-of-Service Unterstützung für die prozessnahe Echtzeitkommunikation*. Aachen : Shaker Verlag, Oktober 2002 (Kommunikationstechnik). – ISBN 978-3-8322-0832-5

[Jeruchim u. a. 2000] JERUCHIM, Michel C. ; BALABAN, Philip ; SHANMUGAN, K. S.: *Simulation of Communication Systems*. 2. New York : Kluwer Academic / Plenum Publishers, 2000 (Applications of communications theory). – ISBN 0-306-46267-2

[Kämpchen u. a. 2005] KÄMPCHEN, N. ; CLAUSS, M. ; GUENTER, Y. ; SCHREIER, R. M. ; STIEGELER, M. ; TISCHLER, K. ; DIETMAYER, K. ; GROSSMANN, H. P. ; KABZA, H. ; NEUMANN, H. ; ROTHERMEL, A. ; STILLER, C.: Vernetzte Fahrzeug-Umfelderfassung für zukünftige Fahrerassistenzsysteme. In: MAURER, Markus (Hrsg.) ; STILLER, Christoph (Hrsg.): *Proceedings of 3. Workshop Fahrerassistenzsysteme*. Walting, Altmühltal : Freundeskreis Mess- und Regelungstechnik Karlsruhe e.V., April 2005, S. 139–150

[Kiszka und Schwebel 2004] KISZKA, Jan ; SCHWEBEL, Robert: Alternative: RTnet. In: *A&D Newsletter* 10 (2004), S. 67–69

[Kiszka und Wagner 2005] KISZKA, Jan ; WAGNER, Bernado: RTnet – a flexible hard real-time networking framework. In: *10th IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2005)* 1 (2005), S. 456–463. – URL `http://www.rts.uni-hannover.de/rtnet/download/RTnet-ETFA05.pdf`

[Knuth 1998] KNUTH, Donald E.: *Digital Typography*. Stanford : Center for the Study of Language and Information (CSLI), Juni 1998. – ISBN 978-1575860107

[Koller u. a. 2001] KOLLER, Gerald ; RAUSCHER, Thomas ; SAUTER, Thilo: Der Einfluss von Feldbussen auf verteilte Regelungen. In: *e&i Elektrotechnik und Informationstechnik* 118 (2001), S. 556–563. – URL `http://dx.doi.org/10.1007/BF03158993`. – 10.1007/BF03158993. – ISSN 0932-383X

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

[Kopetz 2008] KOPETZ, Hermann: The Rationale for Time-Triggered Ethernet. In: *Real-Time Systems Symposium, 2008*, Dezember 2008, S. 3–11. – ISSN 1052-8725

[Kopetz u. a. 2005] KOPETZ, Hermann ; ADEMAJ, Astrit ; GRILLINGER, Petr ; STEINHAMMER, Klaus: The time-triggered Ethernet (TTE) design. In: *Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, 2005. ISORC 2005.*, Mai 2005, S. 22–33

[Kramer 2008] KRAMER, Tapio: Wechsel von CAN zu FlexRay. In: *AUTOMOBIL-ELEKTRONIK* electronica - Sonderausgabe 2008 (2008), S. 54–55

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

[Leen und Heffernan 2002] LEEN, Gabriel ; HEFFERNAN, Donal: Expanding Automotive Electronic Systems. In: *Computer* 35 (2002), Nr. 1, S. 88–93. – ISSN 0018-9162

[Liebl 1995] LIEBL, Franz: *Simulation: problemorientierte Einführung*. 2. München : Oldenbourg, 1995. – ISBN 3-486-23373-4

[LIN-Administration] LIN-ADMINISTRATION: *Local Interconnect Network*. – URL `http://www.lin-subbus.org/`. – Zugriffsdatum: 2011-01-06

[Ling und Wang 2008] LING, Chongwei ; WANG, Zhonglei: *Performance Analysis of A Real-Time Ethernet Using OMNeT++*. München, TU München, Masterthesis, September 2008

[Malek 2001] MALEK, Wolfgang: Zukünftige Architektur Fahrzeug Bordnetze / RUETZ TECHNOLOGIES. München, November 2001. – Forschungsbericht

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

[Mayer und Gamer 2008] MAYER, Christoph P. ; GAMER, Thomas: Integrating real world applications into OMNeT++ / Universität Karlsruhe (TH), Institut für Telematik. Karlsruhe, 2008. – Forschungsbericht

[McCabe 2007] MCCABE, James D.: *Network Analysis, Architecture, and Design*. 3. Amsterdam : Elsevier/Morgan Kaufmann, 2007 (The Morgan Kaufmann series in networking). – ISBN 978-0-12-370480-1

[MOST Cooperation] MOST COOPERATION: *Media Oriented Systems Transport*. – URL `http://www.mostcooperation.com/`. – Zugriffsdatum: 2011-01-06

[Müller-Rathgeber u. a. 2008a] MÜLLER-RATHGEBER, Bernd ; EICHHORN, Michael ; MICHEL, Hans-Ulrich: A unified Car-IT Communication-Architecture: Network switch design guidelines. In: *IEEE International Conference on Vehicular Electronics and Safety*, September 2008, S. 16–21

[Müller-Rathgeber u. a. 2008b] MÜLLER-RATHGEBER, Bernd ; EICHHORN, Michael ; MICHEL, Hans-Ulrich: A unified Car-IT Communication-Architecture: Design guidelines and prototypical implementation. In: *IEEE Intelligent Vehicles Symposium, 2008*, Juni 2008, S. 709–714. – ISSN 1931-0587

[Müller-Rathgeber u. a. 2008c] MÜLLER-RATHGEBER, Bernd ; EICHHORN, Michael ; MICHEL, Hans-Ulrich: A unified Car-IT Communication-Architecture: Network switch design guidelines. In: *IEEE International Conference on Vehicular Electronics and Safety, 2008*, September 2008, S. 16–21

[Müller-Rathgeber und Michel 2009] MÜLLER-RATHGEBER, Bernd ; MICHEL, Hans-Ulrich: Automotive network planning - A genetic approach. In: *IEEE Intelligent Vehicles Symposium, 2009*, Juni 2009, S. 1088–1092. – ISSN 1931-0587

[Navet u. a. 2005] NAVET, Nicolas ; SONG, Yeqiong ; SIMONOT-LION, Françoise ; WILWERT, Cédric: Trends in Automotive Communication Systems. In: *Proceedings of the IEEE* 93 (2005), Juni, Nr. 6, S. 1204–1223. – ISSN 0018-9219

[ns-3 Project] NS-3 PROJECT: *The ns-3 network simulator*. – URL `http://www.nsnam.org/`

[NYFEGA Elektro-Garage AG] NYFEGA ELEKTRO-GARAGE AG: *Phaeton Bordnetz*. – URL `http://www.nyfega.ch/bilder/phaeton_bordnetz.jpg`. – Zugriffsdatum: 2011-02-23

[OMNeT++ Community a] OMNET++ COMMUNITY: *INET Framework for OMNeT++ 4.0*. – URL `http://inet.omnetpp.org/`

[OMNeT++ Community b] OMNET++ COMMUNITY: *OMNeT++ 4.0*. – URL `http://www.omnetpp.org`

[OPNET Technologies] OPNET TECHNOLOGIES: *OPNET Modeler*. – URL `http://www.opnet.com/solutions/network_rd/modeler.html`

[Parashchenko 2010] PARASHCHENKO, Oleg: XML to paper publishing with manual intervention. In: SIMÕES, Alberto (Hrsg.) ; CRUZ, Daniela da (Hrsg.) ; RAMALHO, José C. (Hrsg.): *XATA 2010 XML : associated technologies and applications*. Vila do Conde, Mai 2010. – ISBN 978-972-99166-9-4

[Parashchenko 2011] PARASHCHENKO, Oleg: *TeXML 2.0.2*. Januar 2011. – URL `http://getfo.org/texml/`. – Zugriffsdatum: 2011-02-13

[Paret 2007] PARET, Dominique: *Multiplexed networks for embedded systems: CAN, LIN, Flexray, Safe-by-Wire ... (Réseaux multiplexes pour systemés embarqués)*. Chichester : Wiley, 2007. – ISBN 0-470-03416-5

[Paxson und Floyd 1995] PAXSON, Vern ; FLOYD, Sally: Wide area traffic: the failure of Poisson modeling. In: *IEEE/ACM Transactions on Networking* 3 (1995), Juni, Nr. 3, S. 226–244. – ISSN 1063-6692

[Pensawat 2006] PENSAWAT, Taweewit: *Real-Time Ethernet Networks Simulation Model*. Halmstad, Sweden, Halmstad University, Masterthesis, Dezember 2006

[Praus u. a. 2007] PRAUS, Fritz ; GRANZER, Wolfgang ; GADERER, Georg ; SAUTER, Thilo: A simulation framework for fault-tolerant clock synchronization in industrial automation networks. In: *IEEE Conference on Emerging Technologies and Factory Automation 2007*, September 2007, S. 1465–1472

[PROFIBUS & PROFINET International] PROFIBUS & PROFINET INTERNATIONAL: *Profinet*. – URL `http://www.profibus.com/technology/profinet/`. – Zugriffsdatum: 2011-02-07

[Real-Time Systems Group] REAL-TIME SYSTEMS GROUP: *RTnet*. – URL `http://www.rts.uni-hannover.de/rtnet/`. – Zugriffsdatum: 2010-12-10

[Real Time Systems Group (RTS)] REAL TIME SYSTEMS GROUP (RTS): *TTEthernet*. – URL `http://ti.tuwien.ac.at/rts`. – Zugriffsdatum: 2010-12-10

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

[Renesys, SSF Project] RENESYS, SSF PROJECT: *Scalable Simulation Framework, SSF Network Models (SSFNet)*. – URL `http://www.ssfnet.org/`

[Rokosch 2002] ROKOSCH, Uwe: *Airbag und Gurtstraffer*. Würzburg : Vogel, 2002. – ISBN 978-3802318832

[RTAI Team 2010] RTAI TEAM: *RTAI - the RealTime Application Interface for Linux from DIAPM*. 2010. – URL `http://www.rtai.org`. – Zugriffsdatum: 2010-11-10

[SAE - AS-2D Time Triggered Systems and Architecture Committee 2009] SAE - AS-2D TIME TRIGGERED SYSTEMS AND ARCHITECTURE COMMITTEE: *Time-Triggered Ethernet (AS 6802)*. 2009. – URL `http://www.sae.org`. – Zugriffsdatum: 2010-12-11

[Schäfer 2004] SCHÄFER, Andreas: *Verlässlichkeitsanalyse paketvermittelnder Netzwerke mittels diskreter ereignisorientierter Simulation*. Karlsruhe, Universität Fridericiana zu Karlsruhe, Dissertation, Juli 2004

[Schäuffele und Zurawka 2010] SCHÄUFFELE, Jörg ; ZURAWKA, Thomas: *Automotive Software Engineering*. Wiesbaden : Vieweg und Teubner, 2010. – ISBN 978-3-8348-0364-1

[Schedl 2007] SCHEDL, Anton: *Goals and Architecture of FlexRay at BMW*. Vector FlexRay Symposium 2007, Stuttgart. März 2007. – URL `https://www.vector-worldwide.com/portal/medien/cmc/speeches/FlexRay_Symposium_2007/FRS07_02_Schedl.pdf`

[Schramm u. a. 2008] SCHRAMM, Andreas ; HEFFERNAN, Donal ; WURM, Richard: MOST Simulation Framework. In: *MOST Forum, International MOST Conference & Exhibition*. Starnberg, Deutschland, September 2008, S. 29–31

[Senac und Sevilla] SENAC, Andrés ; SEVILLA, Diego: *EMF4CPP*. – URL `http://www.catedrasaes.org/trac/wiki/EMF4CPP`. – Zugriffsdatum: 2011-01-29

[Senac u. a. 2010] SENAC, Andrés ; SEVILLA, Diego ; MARTÍNEZ, Gregorio: EMF4CPP: a C++ Ecore Implementation. In: AVILA-GARCÍA, Orlando (Hrsg.) ; CABOT, Jordi (Hrsg.) ; NOZ, Javier M. (Hrsg.) ; ROMERO, Jose R. (Hrsg.) ; VALLECILLO, Antonio (Hrsg.): *Actas del VII Taller sobre Desarrollo de Software Dirigido por Modelos* Bd. 4. San Sebastián : SISTEDES, September 2010, S. 98–106

[Seno und Zunino 2008] SENO, Lucia ; ZUNINO, Claudio: A simulation approach to a Real-Time Ethernet protocol: EtherCAT. In: *IEEE Int. Conference on Emerging Technologies and Factory Automation 2008*, September 2008, S. 440–443

[Solomon 1980] SOLOMON, Susan L.: Building Modelers: Teaching the Art of Simulation. In: *Interfaces* 10 (1980), Nr. 2, S. 65–72

[Steinbach 2008] STEINBACH, Till: *Ethernet als Bus für Echtzeitanwendungen im Automobil*. Dezember 2008. – URL `http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master08-09-aw1/steinbach/bericht.pdf`. – Zugriffsdatum: 2011-01-08. – Bericht

[Steinbach 2009a] STEINBACH, Till: *Ethernet als Bus für Echtzeitanwendungen im Automobil - Projektbericht*. September 2009. – URL `http://papers.till-steinbach.de/s-ebeap-09.pdf`. – Zugriffsdatum: 2011-01-08. – Bericht

[Steinbach 2009b] STEINBACH, Till: *Time-Triggered Ethernet in Fahrzeugnetzwerken – Related Work*. Juni 2009. – URL `http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2009-aw2/steinbach/bericht.pdf`. – Bericht

[Steinbach 2010] STEINBACH, Till: *Eine Plattform für die eventbasierte Simulation von time-triggered Ethernet Netzwerken*. Juli 2010. – URL `http://papers.till-steinbach.de/s-peste-10.pdf`. – Bericht

[Steinbach u. a. 2011] STEINBACH, Till ; DIEUMO KENFACK, Hermand ; KORF, Franz ; SCHMIDT, Thomas: An Extension of the OMNeT++ INET Framework for Simulating Real-time Ethernet with High Accuracy. In: *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*, 2011. – to appear

[Steinbach u. a. 2010] STEINBACH, Till ; KORF, Franz ; SCHMIDT, Thomas C.: Comparing Time-Triggered Ethernet with FlexRay: An Evaluation of Competing Approaches to Real-time for In-Vehicle Networks. In: *8th IEEE Intern. Workshop on Factory Communication Systems*. Piscataway, New Jersey : IEEE Press, Mai 2010, S. 199–202

[Steiner 2008] STEINER, Wilfried: *TTEthernet Specification*. TTTech Computertechnik AG. November 2008. – URL `http://www.tttech.com`

[Sullivan u. a. 1990] SULLIVAN, D. B. (Hrsg.) ; ALLAN, D. W. (Hrsg.) ; HOWE, D. A. (Hrsg.) ; WALLS, F. L. (Hrsg.): *Characterization of Clocks and Oscillators*. Boulder, Colorado : National Institute of Standards and Technology, 1990. – technical note 1337

[SynqNet Interest Group] SYNQNET INTEREST GROUP: *SynqNet*. – URL `http://www.synqnet.org/`

[Tanenbaum und Wetherall 2011] TANENBAUM, Andrew S. ; WETHERALL, David J.: *Computer Networks*. 5. Boston : Pearson, 2011. – ISBN 978-0-13-255317-9

[The Eclipse Foundation] THE ECLIPSE FOUNDATION: *Eclipse Modeling Framework Project (EMF)*. The Eclipse Foundation. – URL `http://www.eclipse.org/modeling/emf/`

[TTTech Computertechnik AG] TTTECH COMPUTERTECHNIK AG: . – URL `http://www.tttech.com`. – Zugriffsdatum: 2011-01-17

[TTTech Computertechnik AG 2008] TTTECH COMPUTERTECHNIK AG: *TTEthernet Application Programming Interface*. TTTech Computertechnik AG. Dezember 2008. – URL `http://www.tttech.com`

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

[Van Mieghem 2006] VAN MIEGHEM, Piet: *Performance Analysis of Communications Networks and Systems*. Cambridge : Cambridge University Press, 2006. – ISBN 0-521-85515-2

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

[Varga 2010] VARGA, Andras: *INET Framework for OMNeT++ - Manual*, Oktober 2010

[Varga und Hornig 2008] VARGA, András ; HORNIG, Rudolf: An overview of the OMNeT++ simulation environment. In: *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*. Brussels : ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008 (Simutools '08), S. 60:1–60:10. – URL `http://portal.acm.org/citation.cfm?id=1416222.1416290`. – ISBN 978-963-9799-20-2

[Vector Informatik GmbH] VECTOR INFORMATIK GMBH: *CANoe*. – URL `http://www.vector.com/vi_canoe_de.html`. – Zugriffsdatum: 2011-01-17

[World Wide Web Consortium 1999] WORLD WIDE WEB CONSORTIUM ; CLARK, James (Hrsg.) ; DEROSE, Steve (Hrsg.): *XML Path Language (XPath)*. W3C Recommendation. November 1999. – URL `http://www.w3.org/TR/xpath/`. – Zugriffsdatum: 2011-01-29

[World Wide Web Consortium 2007] WORLD WIDE WEB CONSORTIUM ; KAY, Michael (Hrsg.): *XSL Transformations (XSLT) Version 2.0*. W3C Recommendation. Januar 2007. – URL `http://www.w3.org/TR/xslt20/`. – Zugriffsdatum: 2011-02-13

[WTI-Frankfurt eG] WTI-FRANKFURT EG: *TecFinder*. – URL `http://tecfinder.fiz-technik.de`. – Zugriffsdatum: 2011-02-23

Glossar

adaptives Fahrwerk
: Fahrwerk, welches elektronisch je nach Fahrsituation die Federrate und Dämpfung anpasst.

Arbitrierung
: Die Arbitrierung oder Arbitration ist ein Zugangsverfahren, welches gleichberechtigten Teilnehmern durch gegenseitige Vereinbarung das Zugangsrecht zu einer geteilten Ressource erteilt.

automotive
: Automotive ist ein Oberbegriff für maschinenangetriebene Fahrzeuge.

Backbone
: Backbone (engl. für Rückgrat, Hauptstrang, Basisnetz) bezeichnet den zentralen Kernbereich eines Telekommunikationsnetzes.

best-effort
: Best-effort (deutsch: größte Bemühung) bezeichnet die geringste Dienstgütezusicherung in Netzwerken. Übermittlungsanfragen werden im Rahmen der zur Verfügung stehenden Ressourcen bedient.

Build-to-Order (BTO)
: Build-to-Order (BTO) (deutsch: Fertigung nach Auftrag) bezeichnet eine Kombination aus Serien- und Auftragsfertigung. Details des Produktes werden dabei vom Kunden definiert, anschließend wird nach diesen Vorgaben gefertigt.

Constraint
: Constraint (deutsch: Zwangsbedingung) bezeichnet Bedingungen, welche eingehalten werden müssen, damit ein Wert in ein System übernommen wird.

Delegator
: Das Delegation-Pattern ist ein Design-Element aus dem Software Engineering, bei

dem ein Objekt eine Aufgabe an ein Helper-Objekt delegiert, anstatt sie selbst auszuführen.

Integrationstest
Integrationstests bezeichnen in der Softwareentwicklung Einzeltests, die dazu dienen, voneinander abhängige Komponenten eines Systems im Zusammenspiel zu testen.

Jitter
Mit Jitter (deutsch: Fluktuation/Schwankung) wird in der Nachrichtentechnik die Varianz der Laufzeit von Datenpaketen bezeichnet.

Proof-of-Concept
Proof-of-Concept (deutsch: Machbarkeitsnachweis) bezeichnet einen Schritt in der Entwicklung, bei dem die prinzipielle Durchführbarkeit bewiesen wird.

Redundanz
Redundanz (lat. redundare – im Überfluss vorhanden sein) bezeichnet in der Technik das Vorhandensein zusätzlicher, funktional gleicher oder vergleichbarer Ressourcen, wenn diese bei einem störungsfreien Betrieb nicht benötigt werden.

Software-in-the-Loop
Bei der Methode Software-in-the-Loop (SiL) wird der entwickelte Programmcode auf dem Entwicklungsrechner zusammen mit dem simulierten Modell ausgeführt.

Topologie
Mit Topologie wird in Computernetzen die Struktur der Verbindungen mehrerer Teilnehmer untereinander bezeichnet.

Totzeit
Die Totzeit (auch Laufzeit oder Transportzeit) bezeichnet die Zeitspanne zwischen Änderung am Systemeingang und Antwort am Systemausgang einer Regelstrecke.

Usability
Die Usability (deutsch: Brauchbarkeit, (Be-)Nutzbarkeit, Bedienbarkeit) bezeichnet die messbare Nutzungsqualität eines Systems bei der Interaktion mit dem Nutzer.

X-by-Wire
X-by-Wire bezeichnet den Ersatz von mechanischen Verbindungen zur Steuerung durch die Leitung elektrischer, elektronischer, optoelektronischer oder optischer Steuersignale vom Bediener zum Aktor.

Abkürzungsverzeichnis

ABS	Antiblockiersystem
AFDX	Avionics Full DupleX Switched Ethernet
API	Application Programming Interface
ASIC	Application Specific Integrated Circuit
ASR	Antriebsschlupfregelung
BAG	Bandwidth Allocation Gap
BMW	Bayerische Motoren Werke AG
BTO	Build-to-Order
CAN	Controller Area Network
CDMA	Collision-Detection-Multiple-Access
CoRE	Communication over Real-time Ethernet
COTS	Components-of-the-shelf
CPU	Central Processing Unit
CT	Critical-Traffic
CTC	Critical-Traffic-Conformance
DOM	Document Object Model
DuT	Device-under-Test
ECU	Electronic Control Unit
EDF	Earliest-Deadline-First
EMF	Eclipse Modeling Framework
EMV	Elektromagnetische Verträglichkeit
EPSG	Ethernet POWERLINK Standardization Group
ESP	Elektronisches Stabilitätsprogramm
FPGA	Field Programmable Gate Array

GB	Gigabyte
Gbit	Gigabit
GHz	Gigahertz
GPL	GNU General Public License
HAW Hamburg	Hochschule für Angewandte Wissenschaften Hamburg
HTML	Hypertext Markup Language
IAV	Ingenieurgesellschaft Auto und Verkehr
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IP	Internetprotokoll
IRT	Isochronous Real Time
ISO	International Organization for Standardization
IT	Informationstechnik
JNI	Java Native Interface
LAN	Local Area Network
LIN	Local Interconnect Network
LLC	Logical-Link-Control
LVDS	Low Voltage Differential Signaling
LWL	Lichtwellenleiter
MAC	Media Access Control
Mbit	Megabit
MOST	Media Oriented System Transport
NED	Network Description
NRT	Non Real Time
ns	Nanosekunde
OEM	Original Equipment Manufacturer
OMNeT++	Objective Modular Network Testbed in C++
OPNET	Optimized Network Evaluation Tool
OSI	Open Systems Interconnection
PCF	Protocol Control Frame
PDES	Parallel Distributed Simulation

PDF	Portable Document Format
ppm	Parts per Million
RC	rate-constrained
RMA	Reliability, Maintainability, Availability
RT	Real Time
RTAI	Real Time Application Interface
SAE	Society of Automotive Engineers
SAX	Simple API for XML
SoA	Start of Asynchron
SoC	Start of Cycle
SoC	System-on-a-Chip
SSFNet	Scalable Simulation Framework Network Models
TDMA	Time Division Multiple Access
TeXML	TeX Markup Language
TT	Time-triggered
TTCAN	Time-triggered Controller Area Network
TTP	Time-triggered Protocol
URI	Uniform Resource Identifier
WAN	Wide Area Network
XMI	XML Metadata Interchange
XML	Extensible Markup Language
Xpath	XML Path Language
XSL	Extensible Stylesheet Language
XSLT	Extensible Stylesheet Language Transformation

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Inhalt der beigelegten DVD

- **Digitale Version dieser Arbeit** im Portable Document Format (PDF).
  - Till_Steinbach-Echtzeit-Ethernet_fuer_Anwendungen_im_Automobil.pdf
- **Sourcecode:** Der Quellcode zu den selbst entwickelten und darin verwendeten Projekten
- **Programme und Tools:** Verwendete Programme und Werkzeuge
- **Literatur (digital):** Kopie von Seiten und Dokumenten, die ausschließlich im Internet veröffentlicht sind.
  - /Literatur (digital)/[Allmann und Voigtländer 2009]
  - /Literatur (digital)/[Altera]
  - /Literatur (digital)/[Apache Software Foundation]
  - /Literatur (digital)/[AUDI AG, TTTech Automotive GmbH 2009]
  - /Literatur (digital)/[AUTOSAR Development Cooperation]
  - /Literatur (digital)/[Bruckmeier 2010]
  - /Literatur (digital)/[Combs]
  - /Literatur (digital)/[CoRE RG a]
  - /Literatur (digital)/[CoRE RG b]
  - /Literatur (digital)/[Eclipse Foundation a]
  - /Literatur (digital)/[Eclipse Foundation b]
  - /Literatur (digital)/[Eclipse Foundation c]
  - /Literatur (digital)/[EtherCAT Technology Group]
  - /Literatur (digital)/[Ethernet POWERLINK Standardization Group]
  - /Literatur (digital)/[GE Fanuc Intelligent Platforms 2009]
  - /Literatur (digital)/[Hammerschmidt 2007]

- /Literatur (digital)/[Honeywell International]
- /Literatur (digital)/[INCHRON GmbH]
- /Literatur (digital)/[Ingenieurgesellschaft Auto und Verkehr GmbH]
- /Literatur (digital)/[J-Sim Project]
- /Literatur (digital)/[Kiszka und Wagner 2005]
- /Literatur (digital)/[LIN-Administration]
- /Literatur (digital)/[MOST Cooperation]
- /Literatur (digital)/[ns-3 Project]
- /Literatur (digital)/[OMNeT++ Community a]
- /Literatur (digital)/[OMNeT++ Community b]
- /Literatur (digital)/[OPNET Technologies]
- /Literatur (digital)/[Parashchenko 2011]
- /Literatur (digital)/[PROFIBUS & PROFINET International]
- /Literatur (digital)/[Real-Time Systems Group]
- /Literatur (digital)/[Real Time Systems Group (RTS)]
- /Literatur (digital)/[Renesys, SSF Project]
- /Literatur (digital)/[RTAI Team 2010]
- /Literatur (digital)/[SAE - AS-2D Time Triggered Systems and Architecture Committee 2009]
- /Literatur (digital)/[Schedl 2007]
- /Literatur (digital)/[Senac und Sevilla]
- /Literatur (digital)/[Steinbach 2008]
- /Literatur (digital)/[Steinbach 2009a]
- /Literatur (digital)/[Steinbach 2009b]
- /Literatur (digital)/[Steinbach 2010]
- /Literatur (digital)/[Steiner 2008]
- /Literatur (digital)/[SynqNet Interest Group]
- /Literatur (digital)/[The Eclipse Foundation]
- /Literatur (digital)/[TTTech Computertechnik AG]
- /Literatur (digital)/[TTTech Computertechnik AG 2008]
- /Literatur (digital)/[Vector Informatik GmbH]
- /Literatur (digital)/[World Wide Web Consortium 1999]
- /Literatur (digital)/[World Wide Web Consortium 2007]

Sachregister

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

Hamburg, 28. Februar 2011 Till Steinbach