Bachelorarbeit

Sebastian Szancer

Ein OMNeT++ Simulationsmodell für SERCOS III mit TSN Interface

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Sebastian Szancer

Ein OMNeT++ Simulationsmodell für SERCOS III mit TSN Interface

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr.-Ing. Franz Korf
Zweitgutachter: Prof. Dr. Martin Becke

Eingereicht am: 2. Oktober 2017

Sebastian Szancer

Thema der Arbeit
Ein OMNeT++ Simulationsmodell für SERCOS III mit TSN Interface

Stichworte
SERCOS III, OMNeT++, Time-Sensitive Networking (TSN), Time-Triggered Ethernet (TTE), Simulation, INET, CoRE4INET

Kurzzusammenfassung
Ob in Industrie-Anlagen, im Auto, im Flugzeug oder im Schiff: überall ist durch den vermehrten Einbau von neuen Komponenten, wie z.B. Sensoren und deren Vernetzung immer mehr Kommunikation, insbesondere Echtzeit-Kommunikation, erforderlich. Eine Technologie mit der die modernen Anforderungen an die Kommunikation erfüllt werden können, ist Echtzeit-Ethernet. SERCOS III ist ein Echtzeit-Ethernet Protokoll zur Realisierung von Echtzeit-Kommunikation über Ethernet. Allerdings unterliegt der Einsatz von SERCOS III bestimmten Einschränkungen, vor allem hinsichtlich der Netzwerktopologie. Diese Arbeit beschäftigt sich mit der Abbildung von SERCOS III auf TSN mit dem Ziel, diese Einschränkungen aufzuheben und den Einsatz von SERCOS III in heterogenen Netzwerken mit komplexen Topologien zu ermöglichen. Realisiert wird dies mit einem OMNeT++ Simulationsmodell. Basierend auf dem Framework CoRE4INET wird ein SERCOS III Simulationsmodell mit TSN Interface erstellt, mit dem die Abbildung von SERCOS III auf TSN mit Hilfe von ausgewählten Fallbeispielen untersucht wird.

Sebastian Szancer

Title of the paper

An OMNeT++ simulation model for SERCOS III with a TSN interface

Keywords

SERCOS III, OMNeT++, Time-Sensitive Networking (TSN), Time-Triggered Ethernet (TTE), Simulation, INET, CoRE4INET

Abstract

Whether in industrial machines, in automobiles, in airplanes or in ships: the increasing use of new components like sensors and their interconnection requires ever more communication, especially real-time communication. One technology to meet the modern demands of communication is real-time ethernet. SERCOS III is a real-time ethernet protocol enabling real-time communication over ethernet. However there are certain restrictions concerning the use of SERCOS III, especially when it comes to the network topology. This work deals with the mapping of SERCOS III to TSN in order to overcome these restrictions and enable the use of SERCOS III in heterogenous networks with complex topologies. This is done using an OMNeT++ simulation model. Based on the CoRE4INET framework a SERCOS III simulation model with a TSN interface is devised to examine the mapping of SERCOS III to TSN using chosen example scenarios.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Ob in Industrie-Anlagen, im Auto, im Flugzeug oder im Schiff: überall ist durch den vermehrten Einbau von Komponenten, wie Sensoren oder modernen Steuergeräten und deren Vernetzung immer mehr Kommunikation, insbesondere Echtzeit-Kommunikation, erforderlich. Die Notwendigkeit hoher und stets steigender Datenübertragungsraten und komplexer Netzwerktopologien zählen zu den Herausforderungen, denen Entwickler moderner Systeme heute und in den nächsten Jahren gegenüberstehen. Eine Technologie mit der die Quality of Service Anforderungen an die Kommunikation erfüllt werden können, ist Echtzeit-Ethernet. Im Vergleich zu Feldbussystemen bietet die Ethernet-Technologie hohe Datendurchsätze (100 Mbits/s und mehr) und erlaubt komplexe Netzwerktopologien. Außerdem erlaubt der Einsatz von Standard Ethernet in vielen Anwendugsfällen auf teure und proprietäre Hardware zu verzichten. Zudem wird durch Ethernet eine Kommunikation über alle Ebenen der Automatisierungspyramide ermöglicht. Als Automatisierungspyramide bezeichnet man die hierarchische Ebenenstruktur der IT-Systeme in der Produktion, von der Office-Software bis hinunter zur Sensor-/Aktor-Ebene von Produktionsmaschinen. Diese vertikale Integration vom Sensor bis zur Buchhaltungssoftware eröffnet neue Möglichkeiten in der System- und Prozess-Optimierung. Peripherie-, Antriebs- und Office-Kommunikation werden auf einem Medium vereint. So wird es möglich das Maschinen-Netz (in dem etwa SERCOS III zum Steuern von Maschinen benutzt wird) mit dem Netz der Fabrik, dem Netz der Firma und darüber sogar mit dem Internet zu verbinden. (Siehe Abbildung 1.1, S. 3). Allerdings bedeutet eine solche Vereinigung diverser Kommunikationsteilnehmer auf einem Medium auch, dass mitunter zeitkritische Anwendungen mit nicht-zeitkritischen Anwendungen um Bandbreite konkurrieren. Dabei darf die Echtzeitfähigkeit nicht durch den nicht-zeitkritischen Cross-Traffic beeinträchtigt werden.

Eines der Echtzeit-Ethernet Protokolle zur Realisierung von Echtzeit-Kommunikation über Ethernet ist SERCOS III (Siehe Kapitel 4), welches den oben genannten Anforderungen genügt und in der Industrie eingesetzt wird. Bei SERCOS III werden Echtzeit-Daten fester Größe von einem Master an eine Menge von Slaves geschickt. Die Devices sind dabei entweder in einem physikalischen Ring oder einer physikalischen Linie

miteinander verbunden. Der Einsatz des SERCOS III Protokolls ist auf Netzwerke beschränkt, deren Topologie entweder ein physikalischer Ring oder eine physikalische Linie ohne Switches ist und in denen jedes Device das SERCOS III Protokoll unterstützt.

Das Ziel dieser Arbeit ist durch die Abbildung von SERCOS III auf TSN (Siehe: Kapitel 2, Abschnitt 2.1) die Einsatzmöglichkeiten von SERCOS III zu erweitern, sodass es auch bei heterogenen Netzwerken mit komplexeren Topologien mit Switches eingesetzt werden kann und nicht nur in einem physikalischen Ring oder einer physikalischen Linie. So können mit einer Abbildung von SERCOS III auf TT die Garantien der Quality of Service bezüglich der Echtzeit selbst bei hohem Cross-Traffic in einem Netzwerk mit Switches eingehalten werden. Das wird mit Hilfe eines OMNeT++ (Siehe: Kapitel 2, Abschnitt 2.2) Simulationsmodells untersucht. Die dafür notwendigen TSN Elemente stellt das genutzte Framework "CoRE4INET" (Siehe: Kapitel 2, Abschnitt 2.2) zur Verfügung.

Der Vorteil eines Simulationsmodells ist nicht nur, dass es günstiger und einfacher in der Handhabung ist als reale Hardware, sondern auch die Analyse erleichtert, da Datenerfassung und Messungen, z.B. Laufzeitmessungen, leichter durchgeführt werden können als in einer realen Umgebung. Ein Event-basiertes Simulationsmodell wird hier einem analytischen Modell vorgezogen, da unterschiedliche spezielle Echtzeit-Traffic Szenarios untersucht werden und nicht nur Worst-Case Laufzeiten ermittelt werden sollen. Damit lässt sich auch vermeiden, dass eine zu unscharfe Obergrenze zu pessimistische (oder eventuell zu optimistische) Ergebnisse liefert.

Diese Arbeit ist wie folgt aufgebaut: in Kapitel 2 werden die Grundlagen vermittelt, welche für das Verständnis dieser Arbeit nötig sind. In Kapitel 3 werden verwandte Arbeiten vorgestellt und gezeigt, was diese Arbeit neues einbringt. Kapitel 4 behandelt das SERCOS III Protokoll. In Kapitel 5 wird das Simulationsmodell für SERCOS III mit TSN Interface vorgestellt. Es beginnt mit einer Analyse des SERCOS III Protokolls hinsichtlich der Zielsetzung dieser Arbeit. Basierend darauf werden die Anforderungen an das Simulationsmodell abgeleitet. Es folgen das resultierende Konzept des Simulationsmodells und die Umsetzung. Kapitel 6 befasst sich mit der Qualitätssicherung und der Durchführung der Tests und deren Auswertung. Kapitel 7 enthält eine abschließende Zusammenfassung dieser Arbeit und bietet einen kurzen Ausblick auf mögliches weiteres Vorgehen.

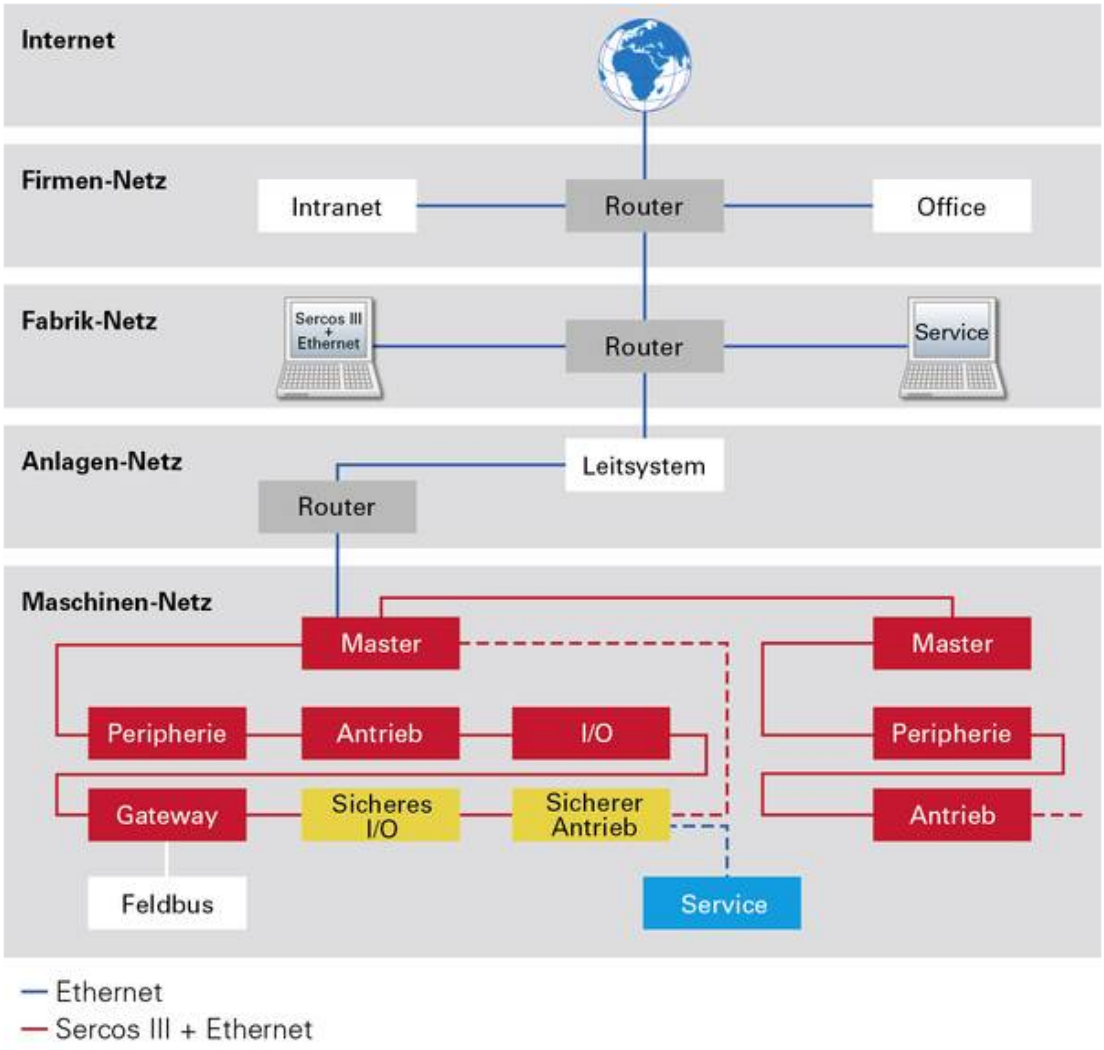

Abbildung 1.1: Peripherie-, Antriebs- und Office-Kommunikation werden durch Ethernet auf einem Medium vereint. (Quelle: http://www.sercos.de/technologie/warum-ethernetechtzeit-ethernet/)

2 Grundlagen

In diesem Kapitel werden die für das Verständnis der Arbeit nötigen Grundlagen vermittelt. Dazu zählen die Technologien "Time-Triggered Ethernet" (TTE) (vgl. [TTTech a] und [TTTech b]) und "Time-Sensitive Networking" (TSN) (vgl. [TTTech c]) sowie die Simulationsumgebung OMNeT++ (vgl. [OMNeT++ Community a]) und die zugehörigen Bibliotheken "INET" (vgl. [OMNeT++ Community b]) und "CoRE4INET" (vgl. [Steinbach u.a. 2011]).

2.1 Time-Triggered Ethernet und TSN

Time-Triggered Ethernet

Time-Triggered Ethernet (TTE) ist eine von der Firma TTTech Computertechnik AG entwickelte Netzwerktechnologie zur Realisierung von Echtzeit-Kommunikation über Ethernet. TTE liegt das Prinzip des "Time Division Multiple Access" (TDMA) zugrunde. Es werden 3 Typen von Nachrichten unterstützt: Time-Triggered (TT), Rate-Constrained (RC) und Best-Effort (BE) Nachrichten. Die gesamte zur Verfügung stehende Bandbreite wird auf diese 3 Nachrichtentypen aufgeteilt.

TT-Nachrichten werden zu vordefinierten Zeitpunkten versendet und haben somit die höchste Priorität im Netzwerk. Durch das Versenden zu festen Zeitpunkten wird eine Garantie für Latenz und Jitter realisiert. Allerdings muss bei der Konfiguration darauf geachtet werden, dass sich zwei verschiedenen TT-Frames nicht überschneiden. Da die TT-Frames zu vordefinierten Zeitpunkten versendet und empfangen werden müssen, ist eine Synchronisation aller Devices des Netzwerks, also eine globale Zeit nötig (Siehe: "Synchronisation", S.5).

Für **RC-Nachrichten** wird offline die nutzbare Bandbreite konfiguriert. RC-Nachrichten werden zwar nicht zu einem garantierten Zeitpunkt versendet, aber es wird garantiert, dass sie ihre konfigurierte Bandbreite verwenden. Die Latenz und der Jitter hängen al-

so von der Priorität und der Menge der zu versendenden Frames ab, bspw. ob gerade TT-Frames zu versenden sind, oder nicht.

BE-Nachrichten entsprechen dem Standard Ethernet. Es gibt keine Garantien für Latenz, Jitter und ob ein Frame den Empfänger überhaupt erreicht. Der Best-Effort Verkehr nutzt die restliche, nicht für TT und RC verwendete Bandbreite.

Die **Synchronisation** der Netzwerkteilnehmer bei TTE basiert auf der Verteilung eines Mittelwerts der Zeiten der Netzwerkteilnehmer. Für die Synchronisation wird jedem Netzwerkteilnehmer eine der 3 möglichen Rollen "Synchronisation Master" (SM), "Synchronisation Client" (SC) oder "Compression Master" (CM) zugewiesen. Ein CM ist in der Regel, aber nicht notwendigerweise, ein Switch. Die Zeit wird in zwei Schritten synchronisiert. Im ersten Schritt bekommt der CM von jedem SM einen „Protocol Control Frame“ (PCF) mit der Zeit des jeweiligen SM. Der CM errechnet dann einen Mittelwert dieser Zeiten. Im zweiten Schritt wird diese berechnete neue Uhrzeit über PCFs an die SM's und SC's gesendet. Diese stellen dann ihre Uhren auf die empfangene neue Uhrzeit.

Time-Sensitive Networking

Time-Sensitive Networking (TSN) ist die Bezeichnung für eine Reihe von Standards die Echtzeit-Kommunikation über Ethernet ermöglichen. TSN ist eine Erweiterung des Audio/Video Bridging (AVB), mit dem sich Philipp Meyer in seiner Arbeit [Meyer 2013], welche hier in Kapitel 3 "Verwandte Arbeiten" Abschnitt 3.2 vorgestellt wird, ausführlicher beschäftigt hat. Mit AVB ist es möglich Daten mit einer garantierten Latenz (maximale Übertragungslatenz von 2ms) über ein Netzwerk (mit Switches) zu streamen. Diese Garantie gilt nur innerhalb einer begrenzten "Wolke" aus Devices, die AVB unterstützen. Eine zeitliche Synchronisation der Netzwerkteilnehmer wird nur auf Anwendungsebene benötigt, im Gegensatz zu TT, wo die zeitliche Synchronisation der Netzwerkteilnehmer auf Protokollebene benötigt wird.

Im Rahmen dieser Arbeit werden die Protokolle TT, RC und AVB zur Abkürzung unter dem Begriff TSN zusammengefasst.

2.2 OMNeT++

OMNeT++ ist ein erweiterbares, modulares, Komponenten-basiertes Framework zum Erstellen von Netzwerksimulationen. OMNet++ ist Event-basiert. Es ermöglicht das Erstellen von Modulen mit einer so genannten "Network Description Language" (NED). Das Verhalten dieser Module wird in C++ programmiert. Ein Modul kann bspw. ein Port, eine Anwendung oder ein Buffer sein. Aus diesen Modulen lassen sich beliebige Netzwerkkomponenten wie z.B. Switches, Sensoren oder Steuergeräte zusammensetzen. Auf diese Weise können beliebige Netzwerke erstellt und simuliert werden. Deswegen eignet sich OMNet++ auch zum Untersuchen unterschiedlicher Netzwerkprotokolle. Die Nutzung von OMNeT++ ist durch die "Academic Public License" kostenslos. Daneben gibt es auch eine kommerzielle Version "OMNEST" (vgl. [Simulcraft, Inc.]). Es gibt bereits viele Frameworks für OMNeT++ die unterschiedliche Netzwerkgeräte und Netzwerkprotokolle implementieren. Dazu zählen die beiden im Rahmen dieser Arbeit verwendeten Frameworks INET und CoRE4INET die anschließend kurz vorgestellt werden.

2.2.1 INET

Das INET Framework ist eine für OMNeT++ erstellte Open-Source Bibliothek. Es implementiert bereits viele wireless und kabelgebundene Switches und Netzwerkteilnehmer. Diese unterstützen Protokolle wie z.B.: IP, IPv6, TCP, UDP, SCTP, Ethernet und viele weitere. Für das dieser Arbeit zugrunde liegende Simulationsmodell wurde die Ethernet Komponente genutzt. Diese bietet eine MAC Implementierung und u.a. ein Switch Modell.

2.2.2 CoRE4INET

Das CoRE4INET Framework ist eine Erweiterung des INET Frameworks für Real-time Ethernet. Es stellt u.a. Komponenten für TTE und AVB zur Verfügung, welche für das Simulationsmodell dieser Arbeit genutzt wurden. Neben speziell TTE- und AVB-Modulen bietet CoRE4INET auch Module für Anwendungen und Scheduling allgemein. Auch diese bildeten eine Grundlage für das Simulationsmodell.

3 Verwandte Arbeiten

Dieses Kapitel gibt eine Übersicht über ausgewählte Arbeiten, die sich mit SERCOS III oder dem Einsatz von TSN/Time-Triggered Ethernet befasst haben und bettet diese Arbeit in den Kontext bisheriger Arbeiten ein. Dabei wird gezeigt, was diese Arbeit neues einbringt.

3.1 Arbeiten zu SERCOS III

Die beiden in diesem Abschnitt vorgestellten Arbeiten haben SERCOS III behandelt. Insbesondere hat die erste Arbeit - [Chongyuan u.a.2010], ähnlich wie diese, ein SERCOS III Simulationsmodell hervorgebracht. In der zweiten Arbeit - [Abel u.a.2011] kam SERCOS III praktisch zur Anwendung.

Research on Implementing Real Time Ethernet for Ship Power System

In ihrer Arbeit befassen sich Hou Chongyuan, Jiang Hanhong, Yang Yuan, Rui Wanzhi und Hu Lianwu [Chongyuan u.a.2010] mit dem Einsatz von SERCOS III zur Realisierung der internen Kommunikation eines Schiffsantriebssystems.
Als Anforderungen an ein modernes Kommunikationssystem eines Schiffsantriebssystems nennen sie 1) eine hohe Datenübertragungsrate (100 Mbit/s und mehr), 2) Echtzeitfähigkeit, 3) Redundanz (besonders bei Kriegsschiffen) und 4) Synchronisation. Mit Hilfe eines Opnet Simulationsmodells zeigen sie, dass mit SERCOS III alle diese Anforderungen erfüllt werden können.

Application of RT-Preempt Linux and Sercos III for Real-time Simulation

In der Arbeit von Michael Abel, Luis Contreras und Peter Klemm [Abel u.a.2011] kommt SERCOS III zur Kommunikation zwischen einem Echtzeit-Simulationsmodell auf einem PC, das bspw. die Hardware einer Produktionsmaschine simuliert und einem Controller

(PLC: Programmable Logic Controller) zum Einsatz. Das Ziel der Arbeit war es, den Entwicklungsprozess von neuen, besonders spezialisierten, Maschinen zu optimieren, indem die Software auf simulierter Hardware (dem Echtzeit-Simulationsmodell), statt auf der konkreten Hardware getestet wird.
Im Vordergrund stand hier der Austausch von konkreten Anwendungsdaten in Echtzeit zwischen dem Controller und dem Simulationsmodell auf dem PC. Eine Maschine wurde erfolgreich auf einem PC emuliert und das Modell über den Controller mit den nötigen Daten versorgt. Dafür wurde SERCOS III verwendet, da es ein Protokoll ist, welches die Anforderung der Echtzeit an die Kommunikation erfüllt.

3.2 Arbeiten zu TSN/Time-Triggered Ethernet

Die beiden in diesem Abschnitt vorgestellten Arbeiten befassen sich mit TSN/Time-Triggered Ethernet. In der ersten Arbeit - [Meyer 2013] werden unterschiedliche Echtzeit-Protokolle (in dem Fall AVB und TT) auf demselben Medium verwendet, was auch in dieser Arbeit der Fall ist. In der zweiten Arbeit - [Steinbach u.a. 2015] wird der Einfluss von Cross-Traffic auf Echtzeit-Traffic untersucht, was auch für diese Arbeit relevant ist. Wie auch diese Arbeit, verwenden beide OMNeT++ Simulationsmodelle.

Simulationsbasierte Analyse der Integration von TDMA basierter Kommunikation in Ethernet AVB

In der Arbeit von Philipp Meyer [Meyer 2013] wird das Verhalten des Echtzeit-Ethernet Protokolls Audio-Video-Bridging (AVB) mit einer Time-Triggered-Ethernet (TTE) Erweiterung untersucht.
Dazu wurde ein OMNeT++ Simulationsmodell verwendet. Die Bibliothek CoRE4INET (vgl. [Steinbach u.a. 2011]), welche bereits Time-Triggered Ethernet unterstützte, wurde um AVB erweitert. In das AVB Protokoll wurde die Nachrichtenklasse TT (Time-Triggered) des Time-Triggered Ethernet eingebunden. Ziel war es, dadurch die Genauigkeit und Latenz von AVB zu verbessern. Dieses Ziel wurde über die Nachrichtenklasse TT erreicht, aber auf Kosten einer höheren Latenz bei der AVB Nachrichtenklasse A und eines hohen offline Aufwands für die statische Konfiguration der TT-Komponenten.

Beware of the Hidden! How Cross-traffic Affects Quality Assurances of Competing Real-time Ethernet Standards for In-Car Communication

In ihrer Arbeit untersuchen Till Steinbach et al. [Steinbach u.a. 2015] den negativen Einfluss von Cross-Traffic im Automobil auf die Latenz und den Jitter von Echtzeit-Traffic bei Verwendung von AVB oder Time-Triggered Ethernet (TTE) und mit welchen Ansätzen sich dieser negative Effekt minimieren lässt. Dazu wurde ein OMNeT++ Simulationsmodell verwendet.
Bei einer Ethernet-basierten Kommunikation in einem physikalischen Netzwerk im Automobil gibt es einerseits hochgradig zeitkritische Daten (Echtzeit-Traffic, z.B. Fahrzeugsteuerung) und andererseits eine große Menge Best-Effort Cross-Traffic (z.B. Firmware-Updates oder Multimedia). Dieser Cross-Traffic führt zu höheren Latenzen und mehr Jitter bei dem Echtzeit-Traffic (AVB oder TTE). Um die Erhöhung der Latenzen und Jitter möglichst gering zu halten, wurden mehrere Ansätze präsentiert: 1) Beschränkung der MTU (Frame-Größe), 2) Einschränkung des Cross-Traffic, 3) Erhöhung der Bandbreite, 4) Optimierung der Topologie, 5) Frame-Preemption. Die Umsetzbarkeit und Kosten der einzelnen Ansätze wurden diskutiert. Es wurde gezeigt, dass auch bei hohem Cross-Traffic die aktuellen Echtzeit-Anforderungen im Automobil eingehalten werden können. Frame-Preemption wurde als der Ansatz mit dem größten Potential identifiziert, wobei hier die Umsetzbarkeit diskutiert werden muss, da Änderungen an der Ethernet MAC und Unterstützung von Preemption auf mehreren Schichten benötigt werden.

3.3 Beitrag dieser Arbeit

In den in Abschnitt 3.1 vorgestellten Arbeiten wird das SERCOS III Protokoll für konkrete Anwendungsfälle umgesetzt und zwar unter den für den Einsatz von SERCOS III vorgesehenen Bedingungen. Dabei wird SERCOS III als einziges Protokoll auf OSI-Layer 2 eingesetzt. Diese Arbeit hingegen befasst sich nicht mit einem konkreten Anwendungsszenario, sondern allgemein mit der Abbildung von SERCOS III auf TSN, wodurch die Einsatzmöglichkeiten von SERCOS III erweitert werden. Es wird also ein Zusammenspiel von SERCOS III und anderen Echtzeit-Ethernet Protokollen, auch bei Vorhandensein von Cross-Traffic, untersucht.

Im Gegensatz zu der in Abschnitt 3.2 vorgestellten Arbeit von Philipp Meyer [Meyer 2013] wird in dieser Arbeit das SERCOS III Protokoll nicht erweitert, um etwa eine Verbesserung der Quality of Service bezüglich der Latenz bei gleich bleibenden Einsatzmöglichkeiten zu erreichen, sondern das SERCOS III Protokoll wird auf TSN abgebildet, damit bestimmte Einschränkungen für den Einsatz von SERCOS III aufgehoben werden. Konkret soll die Topologie des Netzwerks nicht mehr nur auf einen physikalischen Ring oder eine physikalische Linie beschränkt sein, sondern komplexere Netzwerke mit Switches sollen zulässig sein. Außerdem sollen auch Devices in dem Netzwerk erlaubt sein, welche das SERCOS III Protokoll nicht unterstützen. Ein wichtiger Aspekt ist hierbei der nicht-Echtzeit Cross-Traffic im Netzwerk. Ähnlich wie in der unter 3.2 vorgestellten Arbeit von Till Steinbach et al [Steinbach u.a. 2015] wird hier der Einfluss des Cross-Traffic auf Latenz und Jitter des Echtzeit-Traffics untersucht. Allerdings werden in dem Simulationsmodell, welches dieser Arbeit zugrunde liegt, die in [Steinbach u.a. 2015] präsentierten Strategien, die gezielt Latenz und Jitter minimieren, nicht implementiert, sondern es wird ermittelt, wie groß Latenz und Jitter bei der Abbildung von SERCOS III auf die entsprechenden Ethernet-Protokolle sind und welche Konsequenzen diese Werte haben.

4 SERCOS III

In diesem Kapitel wird das Echtzeit-Ethernet Protokoll SERCOS III, wie es in den IEC Standards 61158, und 61784-2 definiert ist, vorgestellt.
Nach einer allgemeinen Übersicht wird auf die für diese Arbeit relevanten Aspekte von SERCOS III eingegangen: Physikalischer- und Data-Link Layer, Netzwerktopologie, Definition der Echtzeit-Telegramme, Telegramm Timing und Quality of Service.

4.1 SERCOS III - eine Übersicht

SERCOS III ("**Se**rial **R**eal-time **Co**mmunication **S**ystem") ist ein Protokoll für ethernetbasierte Echtzeitkommunikation in einem Netzwerk, welches neben der Echtzeitkommunikation auch Standard-Ethernet Kommunikation erlaubt.
SERCOS III liegen ein Master-Slave-Prinzip und Zeitschlitzverfahren (TDMA) zugrunde. Die gesamte zur Verfügung stehende Bandbreite wird in zwei Kanäle unterteilt: einen Echtzeit-Kanal (Real-time Channel: RTC) und einen Kanal für die Standard-Ethernet Kommunikation (Unified Communication Channel: UCC).
Im UC-Kanal werden die Standard-Ethernet-Frames versendet. Der UC-Kanal kann, falls dies erwünscht ist, deaktiviert werden und so die gesamte Bandbreite für Echtzeitkommunikation genutzt werden. Im Echtzeit-Kanal werden die Frames mit den Echtzeit-Daten, auch Telegramme genannt, versendet.

Das Versenden der Daten erfolgt in Zyklen. In jedem Kommunikationszyklus wird im Echtzeit-Kanal eine feste Anzahl von Telegrammen gesendet. Danach können im UC-Kanal die Standard-Ethernet-Frames gesendet werden (Siehe: Abbildung 4.1, S. 12).

Die Dauer eines Kommunikationszyklus kann ein beliebiges n-faches von 250 μs sein, mit n gleich "0,125", "0,25", "0,5" oder einer ganzen Zahl zwischen "1" und "260.000", was eine Dauer zwischen 31,25 μs und 65 ms bedeutet. Die zu wählende Dauer hängt von der Menge der zu übertragenden Daten ab.

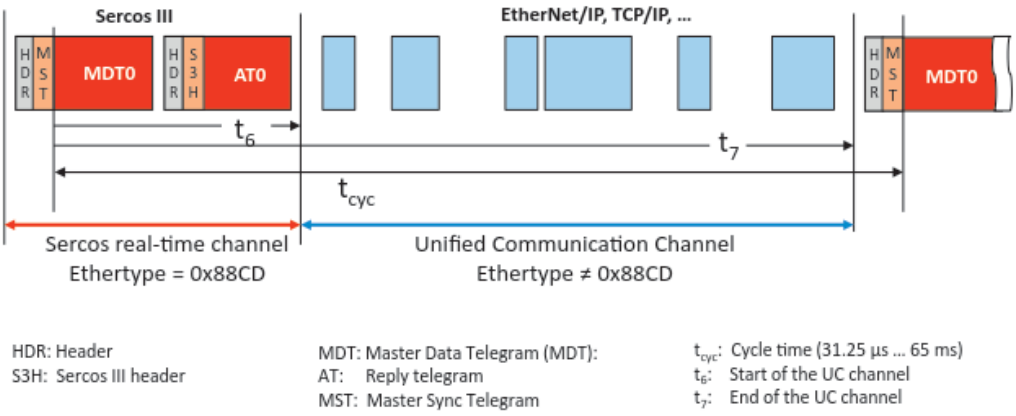

Abbildung 4.1: SERCOS III Kommunikationszyklus (Quelle: Sercos News 01/2015, S.6)

Bei SERCOS III ist genau ein Device Master. Alle anderen Devices sind Slaves. Die Devices sind entweder in einem physikalischen Ring oder einer physikalischen Linie miteinander verbunden. Der Master erstellt die Telegramme und sendet sie an den ersten Slave im Ring oder in der Linie. Dieser sendet die Telegramme nach Verarbeitung weiter an den nächsten Slave usw. bis der letzte Slave die Telegramme erhält. Der letzte Slave sendet die Telegramme (über den Ring oder die Linie) an den Master zurück.

Der Kommunikationsablauf in einem SERCOS III Netzwerk ist durch 5 klar definierte Kommunikationsphasen festgelegt (CP0 bis CP4 (CP = Communication Phase)). In den ersten 4 Phasen wird das Netzwerk schrittweise für die Echtzeitkommunikation konfiguriert und im laufenden Betrieb befindet sich das Netzwerk in der letzten Phase (CP4).

Bei SERCOS III ist es über so genanntes "Hot-plugging" möglich, im laufenden Betrieb Kommunikationsteilnehmer hinzu zu schalten.

Eine zeitliche Synchronisation der Netzwerkteilnehmer wird bei SERCOS III nur auf Anwendungsebene benötigt.

4.2 Physikalischer Layer und Data-Link Layer

Auf dem physikalischen Layer (OSI-Layer 1) wird Standard Ethernet Hardware nach ISO/IEC 8802-3 vorgeschrieben (100BASE-TX oder 100BASE-FX). Für SERCOS III Devices wird eine Datenübertragungsrate von 100 Mbits/s festgelegt. Alle Verbindungen zwischen den SERCOS III Devices müssen voll-duplex Verbindungen sein.

Auf dem Data-Link Layer (OSI-Layer 2) werden Standard Ethernet ISO/IEC 8802-3 Frames mit dem SERCOS III Ether-Type ”0x88CD” verwendet. Jedes Device besitzt genau 2 Ports: Port1 (P1) und Port2 (P2).

4.3 Netzwerktopologie

Bei SERCOS III muss die Netzwerktopologie entweder ein physikalischer Ring oder eine physikalische Linie sein. Der Master kann bis zu 511 Slave-Devices kontrollieren. Diese Beschränkung ergibt sich aus der maximalen Größe eines Standard-Ethernet-Frames (Siehe: Abschnitt 4.4).

Ein Device kann sich entweder im Modus "Fast-Forwarding" (FF) oder im Modus "Loopback with Forward" (LF) befinden. Beim "Fast-Forwarding" wird jedes über einen der beiden Ports erhaltene Telegramm nach der Verarbeitung schnellstmöglich über den jeweils anderen Port versendet. Beim "Loopback with Forward" wird jedes über einen Port erhaltene Telegramm nach der Verarbeitung schnellstmöglich über denselben Port versendet. (Beispiel: FF und LF in Abbildung 4.2).

Netzwerktopologie: Linie

Im Fall der Linie (Siehe: Abbildung 4.2) liegt nur ein Channel im Netzwerk vor: der Primary-Channel (P Channel): der Master sendet die Telegramme an den ersten Slave usw. und der letzte Slave sendet die erhaltenen Telegramme zurück an seinen Vorgänger und dieser sendet sie wiederum an seinen Vorgänger zurück, bis die Telegramme wieder beim Master ankommen. Beim Master und beim letzten Slave ist im Fall der Linie jeweils nur ein Port aktiviert. Bis auf den letzten Slave befinden sich alle Slaves im Modus "Fast-Forwarding" und nur der letzte Slave in der Linie befindet sich im Modus "Loopback with Forward".

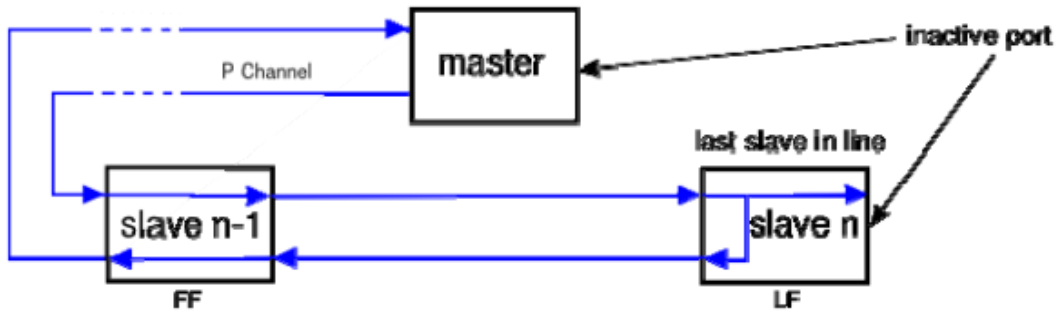

Abbildung 4.2: Topologie: Linie (Quelle: DIN EN 61158-4-19, Figure 14, S.92)

Netzwerktopologie: Ring

Im Fall des Rings (Siehe: Abbildung 4.3) liegen zwei Channel im Netzwerk vor: der Primary-Channel (P Channel) und der Secondary-Channel (S Channel): der Master erstellt jedes Telegramm zweimal und sendet eine Kopie an den ersten Slave im Ring, der sie dann an seinen Nachfolger weitersendet, usw. bis sie wieder beim Master ankommt (Primary-Channel) und die andere an den letzten Slave im Ring, der sie dann an seinen Vorgänger weitersendet, usw. bis sie wieder beim Master ankommt (Secondary-Channel). Bis auf die Information ob es sich bei der Kopie um ein Primary-Channel oder Secondary-Channel Telegramm handelt, sind beide identisch. Auf diese Weise werden beide Kopien jedes Telegramms einmal durch den kompletten Ring geschickt. Alle Slaves befinden sich im Modus "Fast-Forwarding" (FF).

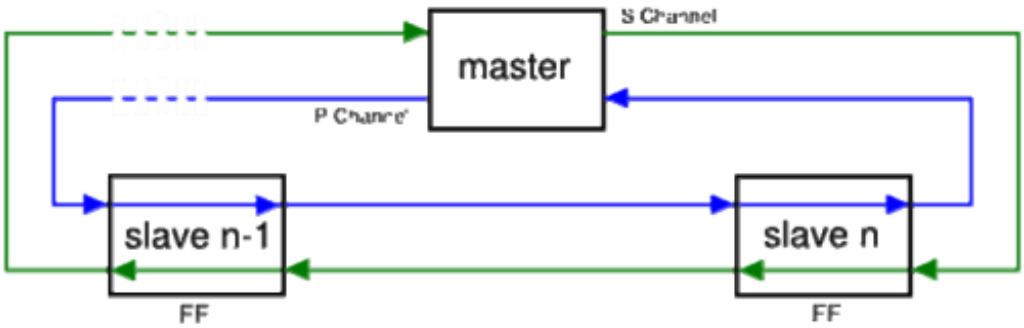

Abbildung 4.3: Topologie: Ring (Quelle: DIN EN 61158-4-19, Figure 13, S.92.)

Durch den Ring wird Redundanz erreicht, was im Fall eines Abbruchs einer Verbindung (bspw. durch Kabelbruch) das Aufrechterhalten der Kommunikation erlaubt. Der Ring zerfällt dann in zwei Linien (Siehe Abbildung 4.4, S. 16), oder, wenn die Verbindung zwischen Master und erstem oder letztem Slave abbricht, in eine Linie (Siehe Abbildung 4.5, S. 16). Der Ring ist deshalb der Linie vorzuziehen. Generell ist mit dem Ring auch eine geringere Roundtrip-Time (Siehe: Abschnitt 4.6) für die Telegramme möglich, als im Fall der Linie, da die Telegramme beim Ring vom letzten Slave über einen Link direkt an den Master gesendet werden, während sie bei der Linie vom letzten Slave über alle seine Vorgänger an den Master geschickt werden.

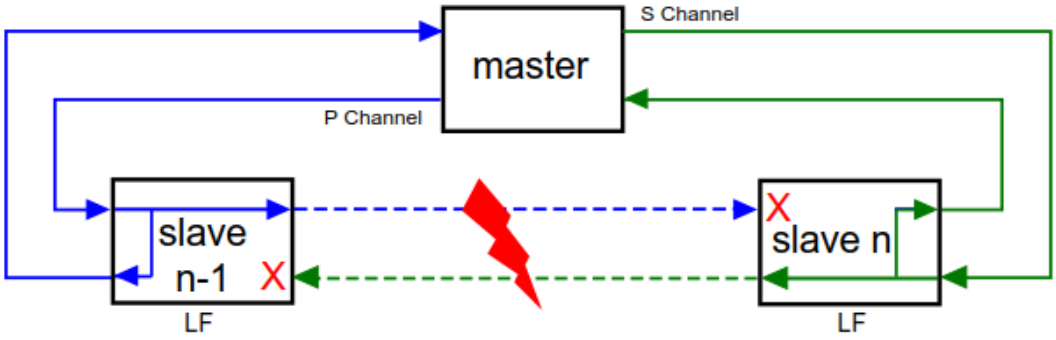

Abbildung 4.4: Ringbruch (Quelle: DIN EN 61158-4-19, Figure 25, S.102.)

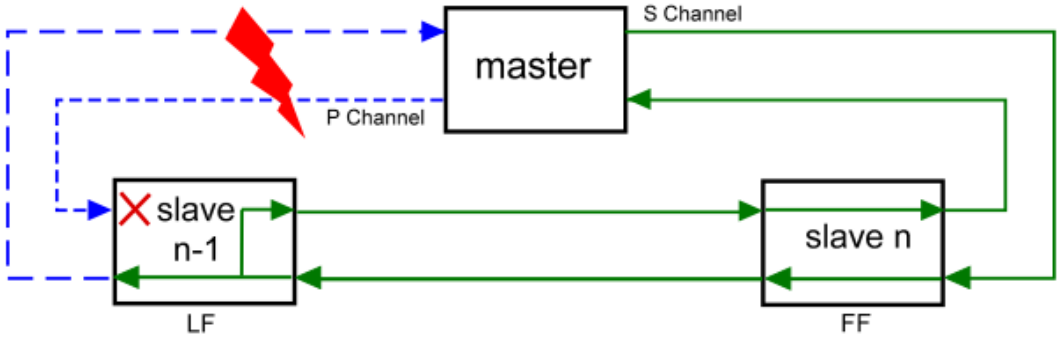

Abbildung 4.5: Ringbruch am Master (Quelle: DIN EN 61158-4-19, Figure 26, S.102.)

4.4 Definition der Telegramme

Bei SERCOS III unterscheidet man zwischen zwei Arten der Telegramme: den "Master-Data Telegrammen" (MDTs) in die der Master Daten für die Slaves schreibt und den "Answer Telegrammen" (ATs) in die die Slaves Daten für den Master und andere Slaves schreiben.
Auf die MDTs haben die Slaves also nur lesenden Zugriff, während sie auf die ATs lesenden und schreibenden Zugriff haben. Der Master hat auf die MDTs schreibenden Zugriff und auf die ATs nur lesenden Zugriff.
Der Master kann, abhängig von der Anzahl der Slaves, entweder jeweils 2 MDTs und ATs (MDT0/AT0 bis MDT1/AT1) oder jeweils 4 MDTs und ATs (MDT0/AT0 bis MDT3/AT3) verschicken (Siehe: Abschnitt 4.5). Bei 256 bis 511 Slaves müssen jeweils 4 MDTs und ATs verschickt werden, bei bis zu 255 Slaves sind beide Konfigurationen zulässig.

Telegramm Aufbau

Der Aufbau der Telegramme ist im Prinzip immer gleich. Im Header (DLPDU Header) wird u.a. festgelegt, ob es sich bei dem Telegramm um ein MDT oder AT handelt und um welches (MDT0, MDT1, ...). Hauptbestandteil eines Telegramms sind die beiden für die Kommunikation relevanten Felder:
Das Servicekanalfeld und das Echtzeitdatenfeld (Siehe: Abbildung 4.6, S. 18). Das Servicekanalfeld enthält für jeden Slave einen Service Channel (SVC). Über diesen Service Channel werden nicht-anwendungs-relevante Daten ausgetauscht, bspw. findet die Konfiguration der Slaves über den Service Channel statt.
Das Echtzeitdatenfeld enthält die Connections, über welche die eigentliche Echtzeit-Kommunikation läuft. Eine Connection hat immer einen Producer und einen oder mehrere Consumer. So kann bspw. der Master der Producer einer Connection sein und alle Slaves die Consumer dieser Connection. Welche Devices wie miteinander kommunizieren, wird also durch die Connections festgelegt. Die Connections werden zu Beginn konfiguriert und können nicht verändert werden.
Daneben gibt es noch für jeden Slave ein Device-Control Feld (u.a. für den Modus ("Fast-Forwarding" oder "Loopback with Forward") des Slaves).

Telegramm Größe

Die Größen der MDTs und ATs werden fest konfiguriert und können nicht verändert werden. Sie sind abhängig von der Anzahl der Slaves und der Größe des Connections-Felds. Unterschiedliche Größen für die MDTs und ATs sind zulässig, also jedes der Telegramme kann eine andere Größe haben.

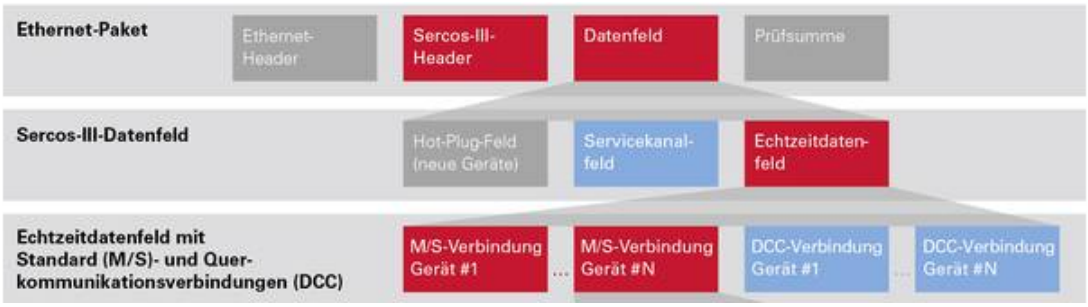

Abbildung 4.6: Aufbau eines SERCOS III Telegramms (Quelle: http://www.sercos.de /technologie/sercos-iii/protokollstruktur/)

Da das Datenfeld eines Standard-Ethernet-Frames nicht größer als 1500 Byte sein darf, kann das Datenfeld eines SERCOS III Telegramms, welches sowohl Protokoll- als auch Anwendungsdaten (die Connections) enthält und mindestens eine Länge von 40 Byte haben muss, maximal 1494 Bytes groß sein (1500 Byte - 6 Byte (SERCOS III Header) = 1494). Da die Menge an Protokolldaten mit steigender Anzahl an Slaves zunimmt, ist die Anzahl der maximal zulässigen Slaves auf 511 beschränkt. Für SERCOS III Telegramme ergibt sich somit eine zulässige Größe von 72 bis 1526 Bytes (Siehe: Abbildung 4.7).

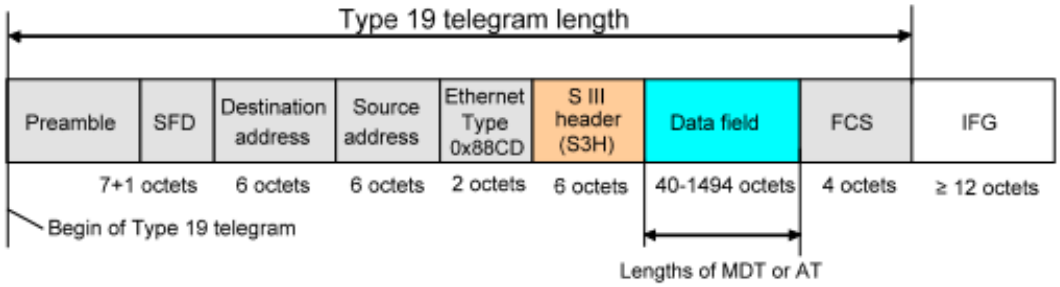

Abbildung 4.7: Größe und Zusammensetzung eines SERCOS III Telegramms (Quelle: DIN EN 61158-4-19, Figure 51, S.151)

4.5 Telegramm Timing

Die Telegramme und Standard-Ethernet-Frames werden in ihren jeweiligen Zeitfenstern (RTC und UCC) gesendet. Dabei werden immer zuerst die MDTs und danach die ATs in fester Reihenfolge (MDT0, MDT1, ..., AT0, AT1, ...) gesendet (Siehe: Abbildung 4.8). Erhält ein Device einen Frame wird dieser schnellstmöglich weitergesendet. Standard-Ethernet-Frames dürfen nur innerhalb des UC-Kanals gesendet werden. Ein Standard-Ethernet-Frame, der nicht mehr vor Ende des UC-Kanals gesendet werden kann, weil er zu groß für das verbleibende Zeitfenster ist, wird auch nicht mehr gesendet.
Die Position des Zeitfensters des UC-Kanals innerhalb des Kommunikationszyklus ist konfigurierbar: entweder der UC-Kanal kommt nach dem Zeitfenster des gesamten Echtzeit-Kanals (Methode 1), oder der UC-Kanal kommt nach dem MDT-Block, aber vor dem AT-Block (Methode 2), unterteilt also den Echtzeit-Kanal in zwei Hälften (Siehe: Abbildung 4.8).

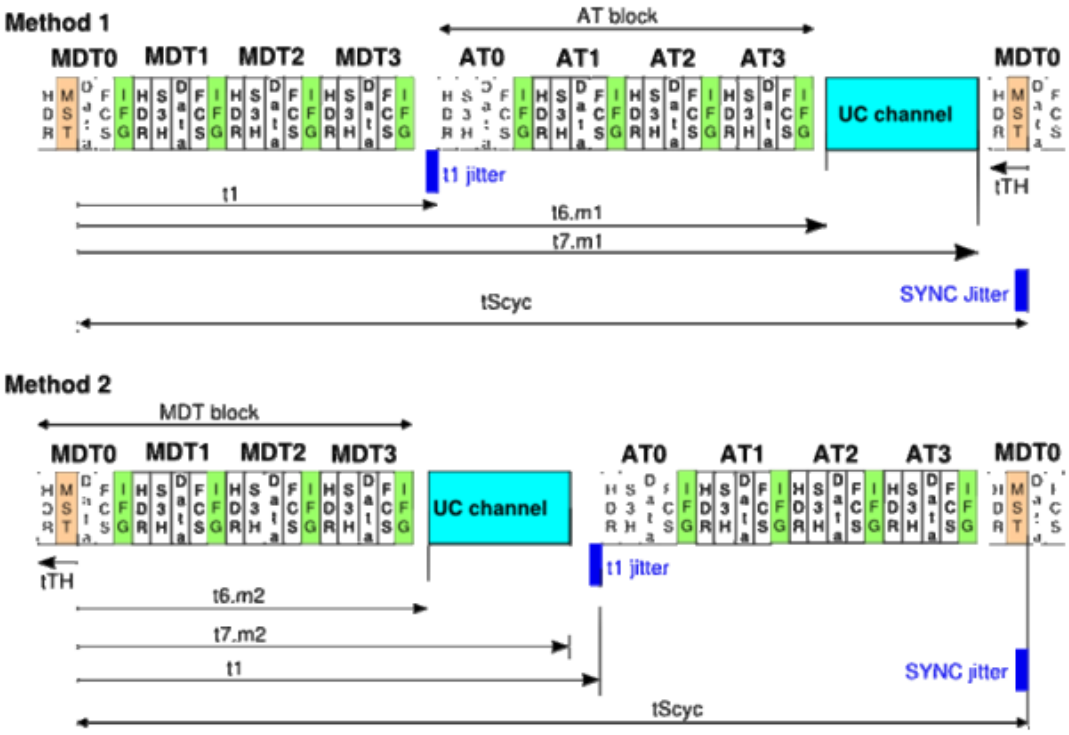

Abbildung 4.8: Position des UC-Kanal Zeitfensters im Kommunikationszyklus
(Quelle: DIN EN 61158-4-19, Figure 53, S.154)

Der Master berechnet aus den Größen der MDT- und AT-Blocks und aus der konfigurierten Dauer des UC-Kanals für die jeweilige Methode die Startzeit des AT-Blocks (*t1*) und den Anfang und das Ende des UC-Kanals (*t6* und *t7*), alles relativ zum Anfang des Headers des MDT0 (auch als "MST" bezeichnet). Die Zeiten *t1*, *t6* und *t7* sind also Offsets zum MST.
Diese Zeiten werden während der Konfiguration an alle Slaves verteilt. So kann jeder Slave den Start und das Ende des UC-Kanals für den aktuellen Kommunikationszyklus bestimmen, sobald er das MDT0, das den Anfang des Kommunikationszyklus markiert, erhalten hat.
Jedes Device ist also in der Lage die Zeitfenster einzuhalten und für ein korrektes Timing, sowohl der Telegramme, als auch der Standard-Ethernet-Frames zu sorgen.

Ist die konfigurierte Dauer des Kommunikationszyklus < 250 μs, muss das Zeitfenster für den UC-Kanal "0s" sein. Ansonsten ist die Aufteilung zwischen Echtzeit-Kanal und UC-Kanal frei konfigurierbar, solange das Zeitfenster für den Echtzeit-Kanal für alle Echtzeit-Telegramme (inklusive Inter-Frame-Gap) ausreicht.

4.6 Quality of Service

In diesem Abschnitt wird auf die Echtzeit-Garantie von SERCOS III eingegangen. Dafür wird zuerst die Roundtrip-Time betrachtet und danach der Jitter.

Roundtrip-Time

Als Roundtrip-Time wird hier die Zeit definiert, die ein Telegramm benötigt, um vom Master aus den kompletten Ring oder die komplette Linie zu durchlaufen und wieder beim Master anzukommen. Generell setzt sich diese Zeit aus den Bearbeitungszeiten des Telegramms durch alle Slave-Devices und aus der Zeit die das Telegramm benötigt, um über die Links des Netzwerks zu wandern zusammen. Die Zeit, welche ein Device benötigt, um ein Telegramm zu verarbeiten, wird hier als Processing Delay bezeichnet. Die Zeit, welche ein Telegramm benötigt, um über einen Link von einem Device zum nächsten zu wandern, wird hier als Latenz bezeichnet und setzt sich zusammen aus dem Transmission Delay und dem Propagation Delay. Dabei fällt der Propagation Delay bei den meisten praktischen Anwendungen allerdings nur minimal ins Gewicht, da er bei der für SERCOS III vorgesehenen Ethernet Hardware (100BASE-TX oder 100BASE-FX) pro Meter Kabel ca. $5ns$ beträgt. Der Transmission Delay für einen Frame der maximalen Größe (1500 Byte + Preamble + SFD + Destination- und Source MAC-Adressen + Ether-Type + FCS = 1526 Byte) beträgt bei einer Datenübertragungsrate von 100 Mbits/s ca. 122 us.

$$Latenz = TransmissionDelay + PropagationDelay.$$

Die Roundtrip-Time hängt bei SERCOS III auch von der Netzwerktopologie ab und lässt sich, je nachdem ob ein physikalischer Ring oder eine physikalische Linie vorliegt, durch folgende Formeln beschreiben:

Ring:

$$Roundtrip\text{-}Time\ _{Ring} = \sum_{1}^{n} Latenz\ _{i} + \sum_{2}^{n} ProcessingDelay\ _{i}$$

Dabei ist n die Anzahl der Devices im Netzwerk. Die zweite Summe beginnt mit "2", da der Processing Delay des Masters nicht zur Roundtrip-Time gezählt wird. Die Anzahl der Links entspricht n und das Telegramm läuft 1-mal über jeden Link.

Linie:

$$Roundtrip\text{-}Time_{\,Linie} = 2 \cdot \Big(\sum_{1}^{n-1} Latenz_{\,i} + \sum_{2}^{n-1} ProcessingDelay_{\,i} \Big) + ProcessingDelay_{\,n}$$

Dabei ist n die Anzahl der Devices im Netzwerk. Die zweite Summe beginnt mit "2", da der Processing Delay des Masters nicht zur Roundtrip-Time gezählt wird. Die Anzahl der Links entspricht $n-1$. Bei der Linie wird das Telegramm von jedem Device bis auf den letzten Slave (und den Master) 2-mal bearbeitet und läuft 2-mal über jeden Link.

Die Latenzen sind bei einer festgelegten Datenübertragungsrate von 100 MBits/s abhängig von der Kabellänge des jeweiligen Links, welche konstant ist und von der Größe des Telegramms. Da bei SERCOS III die Größen der Telegramme konstant sind, sind für ein gegebenes Netzwerk auch alle Latenzen konstant.
Der Processing Delay ist für jedes Device von der Hardware und von der Echtzeit-Anwendung (Menge der aktuell zu verarbeitenden Anwendungsdaten für das Telegramm) abhängig und muss nicht für jedes Device konstant sein (wenn die Menge der zu verarbeitenden Daten der Telegramme variiert). Jedes Device verarbeitet ein Telegramm bei Erhalt sofort und sendet es ohne Verzögerung weiter (Forwarding). Deshalb kann für jedes Device leicht ein konstanter "Worst Case" Processing Delay ermittelt werden, der dem Fall mit der maximal möglichen Menge an zu verarbeitenden Anwendungsdaten für ein Telegramm entspricht.
Es resultiert eine vorhersehbare, konstante ("Worst Case") Roundtrip-Time, welche in jedem Fall nur von der Hardware, der Anzahl der Devices und der Echtzeit-Anwendung abhängig ist, also von Faktoren, die von vornherein bekannt und konstant sind. SERCOS III garantiert also, dass diese Roundtrip-Time in jedem Fall eingehalten wird. Dass der Processing-Delay für jedes Device nur von der Hardware und der Menge der zu verarbeitenden Echtzeit-Daten abhängig ist, liegt daran, dass die Echtzeit-Kommunikation aufgrund von TDMA durchgehend kollisionsfrei ist. Dafür müssen alle Devices im Netzwerk SERCOS III unterstützen und es muss eine einfache Ring- oder Linien-Topologie ohne Switches vorliegen.

Jitter

Durch die kollisionsfreie Echtzeit-Kommunikation durch das Zeitschlitzverfahren (TDMA) und die einfache Ring- oder Linien-Topologie ohne Switches und durch feste Telegramm-Größen, wird der Jitter bei SERCOS III auf den Hardware-Jitter der Devices reduziert. Der Jitter ist somit nur abhängig von der Hardware und der Anzahl der Devices.

5 Simulationsmodell für SERCOS III mit TSN Interface

In diesem Kapitel wird das Simulationsmodell für SERCOS III mit TSN Interface vorgestellt. Es beginnt mit einer Analyse, aus der die Anforderungen an das Simulationsmodell abgeleitet werden. Es folgen das resultierende Konzept des Simulationsmodells und dessen Umsetzung.

5.1 Analyse und Anforderungen

Analyse

SERCOS III garantiert, dass in jedem Fall eine vorhersagbare Roundtrip-Time für die Echtzeit-Daten eingehalten wird. Wie in Kapitel 4, Abschnitt 4.6 gezeigt, erfüllt SERCOS III diese Echtzeit-Anforderungen nur aufgrund folgender Einschränkungen: 1) alle Devices im Netzwerk müssen SERCOS III unterstützen und 2) es muss eine einfache physikalische Ring- oder Linien-Topologie ohne Switches vorliegen. Jedes Device ist selber für das Einhalten seiner Zeitfenster für den Echtzeit- und UC-Kanal zuständig. Es stellt mit Hilfe des SERCOS III Protokolls sicher, dass es seine Telegramme zur richtigen Zeit versendet. Die Quality of Service bezüglich der Roundtrip-Time wird also in den Devices realisiert. Hat man Devices im Netzwerk, die SERCOS III nicht unterstützen, sind diese nicht in der Lage die Zeitfenster einzuhalten und die geforderte durchgehend kollisionsfreie Echtzeit-Kommunikation ist nicht mehr möglich. Die Roundtrip-Time und der Jitter sind dann abhängig vom Cross-Traffic dieser Devices, was die Echtzeit-Fähigkeit beeinträchtigt. Der Einsatz von SERCOS III in Netzwerken mit Switches, also komplexeren Netzwerktopologien als ein Ring oder eine Linie, ist aus den gleichen Gründen nicht möglich. Cross-Traffic über die Switches hätte, da Standard Ethernet vorliegt, denselben Effekt auf Roundtrip-Time und Jitter.

Eine Möglichkeit SERCOS III in Netzwerken mit Switches und Devices, die das Protokoll nicht unterstützen, einzusetzen, wäre die Abbildung von SERCOS III auf TSN. Durch die Verwendung von TSN auf OSI-Layer 2 wird das Einhalten der Zeitfenster durch alle Devices möglich. Die Devices müssen nicht mehr selber sicherstellen, dass ihre Frames (Echtzeit-Telegramme und nicht-Echtzeit Frames) zur richtigen Zeit versendet werden, sondern das zeitgerechte Versenden der Frames geschieht über die Verwendung eines geeigneten OSI-Layer 2 Protokolls, wie z.B. TT. D.h. Quality of Service bezüglich der Roundtrip-Time wird über das auf OSI-Layer 2 verwendete Protokoll realisiert und nicht mehr in den Devices. Somit wird auch der Einsatz von Switches und Devices, die SERCOS III nicht unterstützen, möglich, ohne Beeinträchtigung der Echtzeit-Fähigkeit. Für eine solche Abbildung von SERCOS III auf TSN wurde ein Simulationsmodell erstellt.

Anforderungen

Aus der Analyse ergeben sich folgende Anforderungen an das Simulationsmodell:

1) Es muss ein Netzwerk, bestehend aus Devices (Hosts) und Switches, zur Verfügung gestellt werden.

2) Es muss eine SERCOS III Anwendung zur Verfügung gestellt werden, welche die Funktionalität, wie in Kapitel 4 vorgestellt, bietet.

3) Auf OSI-Layer 2 müssen unterschiedliche Ethernet-Protokolle zur Verfügung gestellt werden: Best-Effort (Standard Ethernet) und TSN (TT, AVB und RC).

4) Das SERCOS III Protokoll muss auf unterschiedliche Ethernet-Protokolle (Best-Effort, TT, AVB und RC) abgebildet werden. Dabei muss die in Kapitel 4 beschriebene Funktionalität erhalten bleiben.

5) Die Generierung von unterschiedlich großer Belastung des Netzwerks durch Cross-Traffic muss möglich sein, um zu untersuchen, wie sich Roundtrip-Time und Jitter bei unterschiedlicher Menge an Cross-Traffic verhalten.

6) Das Messen von Roundtrip-Time und Jitter muss möglich sein, um zu bewerten, ob die SERCOS III Quality of Service bezüglich Roundtrip-Time und Jitter eingehalten wird.

5.2 Konzept

Ausgehend von den in Abschnitt 5.1 hergeleiteten Anforderungen wird in diesem Abschnitt das Konzept des Simulationsmodells vorgestellt.

Abdeckung von SERCOS III durch das Simulationsmodell

Für die Zwecke dieser Arbeit muss das SERCOS III Protokoll nicht bis ins letzte Detail implementiert werden. Deshalb werden nur die folgenden Teile von SERCOS III im Simulationsmodell abgedeckt:

1) Die grundlegende Echtzeit-Kommunikation über das Zeitschlitzverfahren: aufteilen der Bandbreite in RTC und UCC und einhalten des Telegramm-Timings.

2) Das Master-Slave-Prinzip: Es gibt nur einen Master, der Rest sind Slaves.

3) Aufbau der Telegramme (MDTs und ATs) nach Kapitel 4.

4) Netzwerktopologie: Die zulässigen Netzwerktopologien sind der Ring und die Linie, wobei es sich bei einer Abbildung von SERCOS III auf TSN nicht mehr um einen physikalischen Ring/Linie handeln muss, sondern um einen logischen Ring/Linie.

Folgende Teile des SERCOS III Protokolls wurden hier weggelassen:

1) Für die Abbildung von SERCOS III auf TSN ist das Protokoll im laufenden Betrieb interessant und weniger seine Initialisierung und Konfiguration. Deshalb wurden die 4 verschiedenen Kommunikationsphasen für die Initialisierung und Konfiguration zu 2 Phasen zusammengefasst (CP0 und CP2) und z.T. durch eine statische Konfiguration des Simulationsmodells ersetzt.

2) Auch die Implementierung der Uhren-Synchronisation nach SERCOS III ist für eine Abbildung auf TSN nicht notwendig, da eine zeitliche Synchronisation der Netzwerkteilnehmer bei SERCOS III nur auf Anwendungsebene benötigt wird, weshalb diese hier entfällt.

3) Das "Hot Plugging" ist ein zusätzliches Feature, welches die Grundfunktionalität erweitert, sie aber nicht wesentlich verändert, weshalb es weggelassen wurde.

Diese Abdeckung erfüllt die Anforderung 2.

Schichten des Simulationsmodells

Um die Abbildung des SERCOS III Protokolls auf unterschiedliche Ethernet-Protokolle zu ermöglichen, wird das Simulationsmodell in 4 Schichten aufgeteilt:
1) den Physical Layer (OSI-Layer 1),
2) den Data-Link Layer (OSI-Layer 2),
3) einen TSN-Interface Layer (kein "Layer" im Netzwerk-Sinn!)
4) eine Anwendungsschicht (Application Layer).
(Siehe: Abbildung 5.1, Seite 28)

Auf dem Application Layer laufen die Anwendungen, welche den Echtzeit-Traffic (SERCOS III) und den nicht-Echtzeit Cross-Traffic generieren. Der TSN-Interface Layer erlaubt die Abbildung von SERCOS III auf TSN (und Best-Effort) (Details Siehe: Abschnitt "Komponenten des Simulationsmodells": 3) TSN-Interface-Layer Komponente). Dabei handelt es sich nicht um einen "Layer" im Netzwerk-Sinn (wie etwa OSI-Layer 2), sondern der TSN-Interface Layer bietet dem Application Layer ein Interface für den

Zugriff auf OSI-Layer 2 (auf dem unterschiedliche Protokolle verwendet werden können, z.B. Best-Effort oder TT). Damit kann Anforderung 4 erfüllt werden.

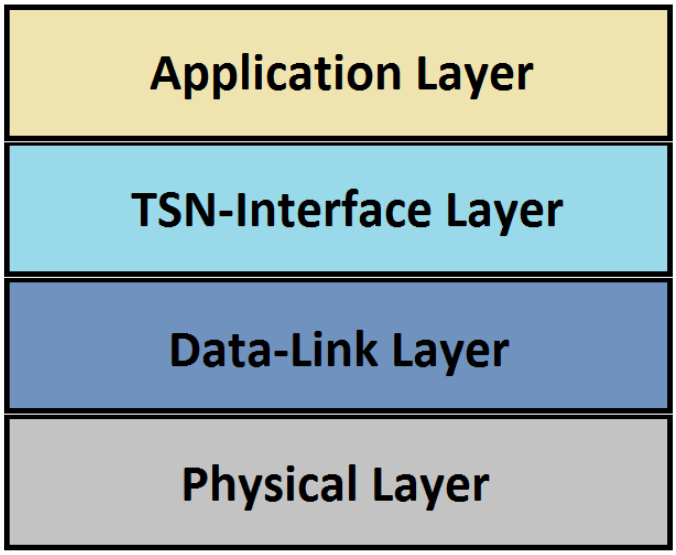

Abbildung 5.1: Schichten des Simulationsmodells (Quelle: eigene Grafik)

Komponenten des Simulationsmodells

Das Simulationsmodell stellt folgende zwei Arten von Komponenten für Netzwerkteilnehmer zur Verfügung: Devices auf denen Anwendungen laufen können (Hosts) und Switches. Damit können Netzwerke für unterschiedliche Szenarien zusammengestellt werden. Damit wird Anforderung 1 erfüllt.

Ein SERCOS III Device im Simulationsmodell besteht aus den folgenden Komponenten (Siehe: Abbildung 5.2, S. 30):

1) Einer SERCOS III Anwendung, welche die Echtzeit-Telegramme (SERCOS III Frames) erstellt und verarbeitet wie in Kapitel 4 beschrieben. Damit wird die Anforderung 2 erfüllt.

2) Data-Link Layer (OSI-Layer 2) Komponenten (Buffer, Ports) mit denen die Ethernet-Protokolle Best-Effort, TT, AVB und RC realisiert werden können. Damit wird Anforderung 3 erfüllt.

3) Einer TSN-Interface-Layer Komponente zur Abbildung von SERCOS III auf die Ethernet-Protokolle Best-Effort, TT, AVB und RC. Bei der TSN-Interface-Layer Komponente handelt es sich um ein austauschbares Modul, d.h. je nachdem ob SERCOS III auf Best-Effort, TT, AVB oder RC abgebildet werden soll, wird das Best-Effort-, TT-, AVB-, oder RC-Modul verwendet. Damit wird Anforderung 4 erfüllt. Für die Abbildung ist es notwendig, dass die SERCOS III Frames in jeweils TT-, AVB-, oder RC-Frames gekapselt werden. Auf dem Data-Link Layer (OSI-Layer 2) wird dann durch Verwendung von TT, AVB oder RC dafür gesorgt, dass das Timing der SERCOS III Frames eingehalten wird.

4) Einer UCC-Anwendung mit der unterschiedliche Mengen nicht-Echtzeit Cross-Traffic (Standard Ethernet Frames) generiert werden können. Damit wird Anforderung 5 erfüllt.

Erfassung von Roundtrip-Time und Jitter

Die Erfassung der Roundtrip-Time und des Jitter erfolgt durch die SERCOS III Anwendung des SERCOS III Master Devices über Time-Stamps die in den durch das Netzwerk gesendeten Nachrichten, welche die Echtzeit-Telegramme enthalten, integriert sind. Damit wird Anforderung 6 erfüllt.

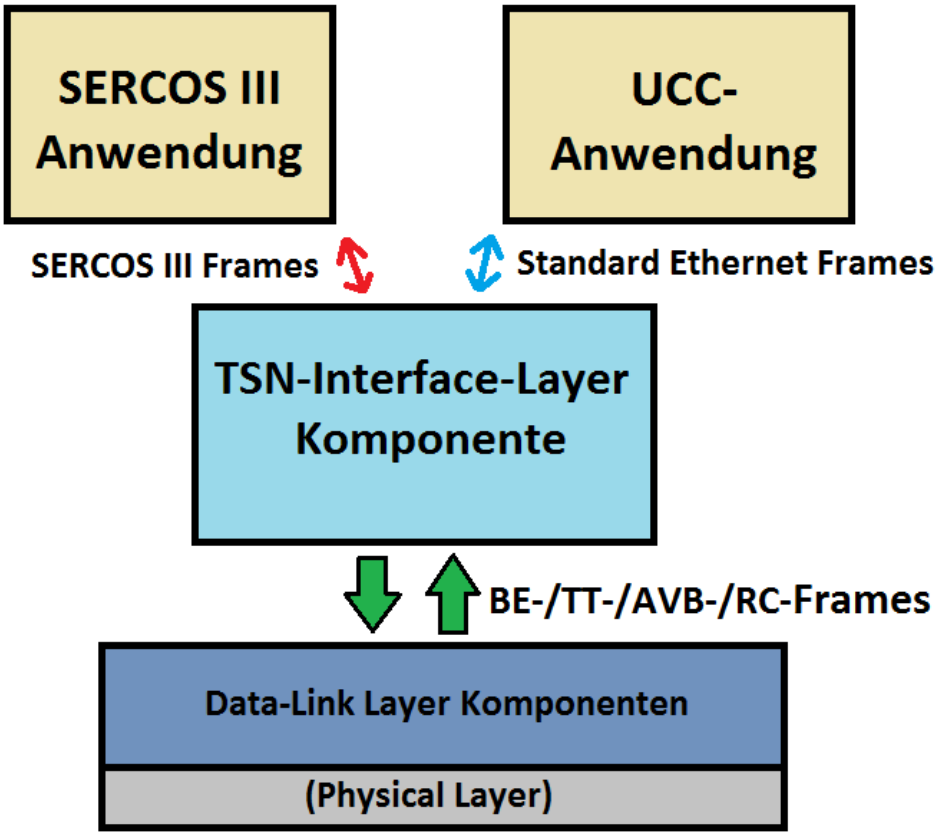

Abbildung 5.2: Komponenten des Simulationsmodells (Quelle: eigene Grafik)

5.3 Umsetzung

Dieser Abschnitt beschreibt die Umsetzung des Simulationsmodells. Im Rahmen dieser Arbeit wurde auf dem TSN-Interface Layer neben Best-Effort nur TT implementiert. Der modulare Aufbau des Simulationsmodells erlaubt allerdings das einhängen von entwickelten AVB- oder RC-Modulen ohne Modifikation der bereits erstellten Komponenten.

5.3.1 Netzwerk und Nachrichten

Netzwerk

Um SERCOS III in einem Netzwerk zu simulieren, werden folgende Netzwerkteilnehmer benötigt: für eine klassische Simulation von SERCOS III nur SERCOS III Devices, für eine Abbildung von SERCOS III auf bspw. TT zusätzlich noch Switches und nicht-SERCOS III Devices zur Erzeugung von Cross-Traffic.
SERCOS III Devices werden im Rahmen des Simulationsmodells selbst zur Verfügung gestellt (Siehe Abschnitt 5.3.4). Für die Switches können die Module "TTE-EtherSwitch" oder "TTEAVBEtherSwitch" von CoRE4INET verwendet werden. Für die nicht-SERCOS III Devices können die Module "TTEEtherHost" oder "TTEAVBEtherHost" von CoRE4INET verwendet werden. Für die 100 Mbit/s Ethernet Links des Netzwerks wird "Eth100M" von INET verwendet. Die Übertragungsgeschwindigkeit von "Eth100M" ist auf $2 \cdot 10^8$ m/s festgesetzt, was realistisch die für SERCOS III Netzwerke vorgeschriebene Ethernet-Hardware wiedergibt.
Aus diesen Modulen werden auch die Netzwerke für die Tests des Simulationsmodells (Siehe: Kapitel 6) zusammengesetzt.

Nachrichten

In dem Netzwerk werden zwei Arten von Nachrichten verschickt: SERCOS III Echtzeit-Nachrichten (Telegramme) und UC-Nachrichten (Standard-Ethernet Frames). Für die Telegramme wurde die Nachrichtenklasse "SercosIIITelegram" erstellt (Siehe: Abbildung 5.3, S. 32). Diese enthält neben den SERCOS III Echtzeitdatenfeldern (DLPDU Header, Service-Channel Feld, Device-Control Feld und Connections) Felder für die Konfiguration der Slaves für die Echtzeitkommunikation ("sequenceCounter", "durationUC", "durationRCBlockBeforeUC"), ein Feld für die Erfassung der Roundtrip-Time des Telegramms ("sendTime") und ein Feld welches festlegt, ob das Telegramm an den

Nachfolger oder Vorgänger im Ring oder in der Linie geschickt werden soll ("sendToNextNotPrevious"). Das "SercosIIITelegram" wird zum Versenden über das Netzwerk in einen Ethernet-Frame der Nachrichtenklasse "EthernetIIFrame" von INET gekapselt. Der Ether-Typ des "EthernetIIFrame" wird auf den SERCOS III Ether-Typ (0x88CD) gesetzt. Bei der Abbildung von SERCOS III auf TT wird der "EthernetIIFrame" zusätzlich in einen von CoRE4INET zur Verfügung gestellten "TTFrame" gekapselt. Für die UC-Nachrichten werden einfach Standard "EthernetIIFrames" verwendet.

```
packet SercosIIITelegram {
    //SERCOS III data fields:
    SercosIIITelegramDLPDUHeader dlpduHeader;
    SercosIIITelegramSVCField serviceChannelField;
    SercosIIITelegramDeviceControlField deviceControlField;
    SercosIIITelegramConnectionsField connectionsField;

    //Send telegram to either the next or previous node in ring/line.
    bool sendToNextNotToPrevious;
    //Sequence counter for slave indexing (initialization).
    uint16_t sequenceCounter;
    //Duration of the UC-chanel.
    double durationUC;
    //Duration of the RT-block in cycle before UC-channel.
    double durationRTBlockBeforeUC;
    //Time when the telegram was sent (for Roundtrip-time and jitter calculation).
    simtime_t sendTime;
}
```

Abbildung 5.3: Aufbau des SERCOS III Telegramms im Simulationsmodell (Quelle: eigene Grafik, Screenshot)

5.3.2 SERCOS III Anwendungen

Es gibt zwei verschiedene SERCOS III Anwendungen: eine SERCOS III Master Anwendung "SercosIIIApplicationMaster" und eine SERCOS III Slave Anwendung "SercosIIIApplicationSlave". Die Aufgabe von "SercosIIIApplicationMaster" ist einerseits die Steuerung der Slaves im laufenden Betrieb (Kommunikationsphase: CP4) und andererseits das Einlesen aller SERCOS III Protokoll Parameter wie z.B. "Anzahl der Slaves", "Topologie" (Ring oder Linie), "Anzahl der Telegramme" (jeweils 2 oder 4 MDTs/ATs), "Dauer des UC-Kanals" oder die Konfiguration der "Connections". Alle Parameter, welche auch die Slaves benötigen, wie "Dauer des UC-Kanals" oder die Konfiguration der "Connections", verteilt "SercosIIIApplicationMaster" per Telegramm in den Initialisierungsphasen. Im Simulationsmodell sind das die beiden Kommunikationsphasen CP0 und CP2: in CP0 verteilt "SercosIIIApplicationMaster" die "Dauer des UC-Kanals" und die Slaves bestimmen mit Hilfe des "sequenceCounters" ihren Topologie-Index, der in CP4 für den Zugriff auf die Connections und den Service-Channel und das Device-Control Feld nötig ist. Slave1 erhält den "sequenceCounter", der mit "1" initialisiert wird, zuerst, liest ihn aus und erhält so den Topologie-Index "1". Slave1 inkrementiert den "sequenceCounter" anschließend und sendet ihn weiter an Slave2 usw. In CP2 verteilt "SercosIIIApplicationMaster" die Konfiguration der "Connections" an alle Slaves. Die Konfiguration der "Connections" liest der Master bei Initialisierung des Simulationsmodells aus einer XML-Datei ein.
In CP4 erstellt "SercosIIIApplicationMaster" "SercosIIITelegrams" mit Dummy-Daten. Diese werden an die Slaves verschickt. "SercosIIIApplicationMaster" trägt dabei bei jedem "SercosIIITelegram" nur ein, ob es an den Nachfolger oder Vorgänger gesendet werden soll. Jede "SercosIIIApplicationSlave" erhält diese "SercosIIITelegrams" und verarbeitet sie. Die Dummy-Daten vom Master werden ausgelesen und eigene Dummy-Daten in die "SercosIIITelegrams" geschrieben.

Module

Für die Module (.ned) der SERCOS III Anwendungen wurde ein Basismodul "SercosIIIApplicationBase" erstellt (Siehe: NED Klassendiagramm (Abbildung 5.4, S. 34)) von dem "SercosIIIApplicationMaster" und "SercosIIIApplicationSlave" abgeleitet sind. "SercosIIIApplicationBase" hat ein "input" Gate für eintreffende "SercosIIITelegrams" und ein "output" Gate über das verarbeitete "SercosIIITelegrams" weiter geschickt werden. Es implementiert das Interface "ISercosIIIApplication" welches diese beiden Gates

spezifiziert und ist abgeleitet von dem CoRE4INET Basismodul für Anwendungen "ApplicationBase", welches wiederum von dem OMNeT++ Basismodul "cSimpleModule" abgeleitet ist. "SercosIIIApplicationMaster" implementiert das CoRE4INET Interface "IScheduled", weil der "Scheduler" von CoRE4INET zum Versenden der "SercosIIITelegrams" verwendet wird. "SercosIIIApplicationMaster" benötigt keine weiteren Gates und hat als Parameter nur die SERCOS III Protokoll Parameter und Scheduling Parameter. "IScheduled" ist abgeleitet von "ITimed" welches wiederum von "cSimpleModule" abgeleitet ist. "SercosIIIApplicationSlave" benötigt keine weiteren Gates oder Parameter.

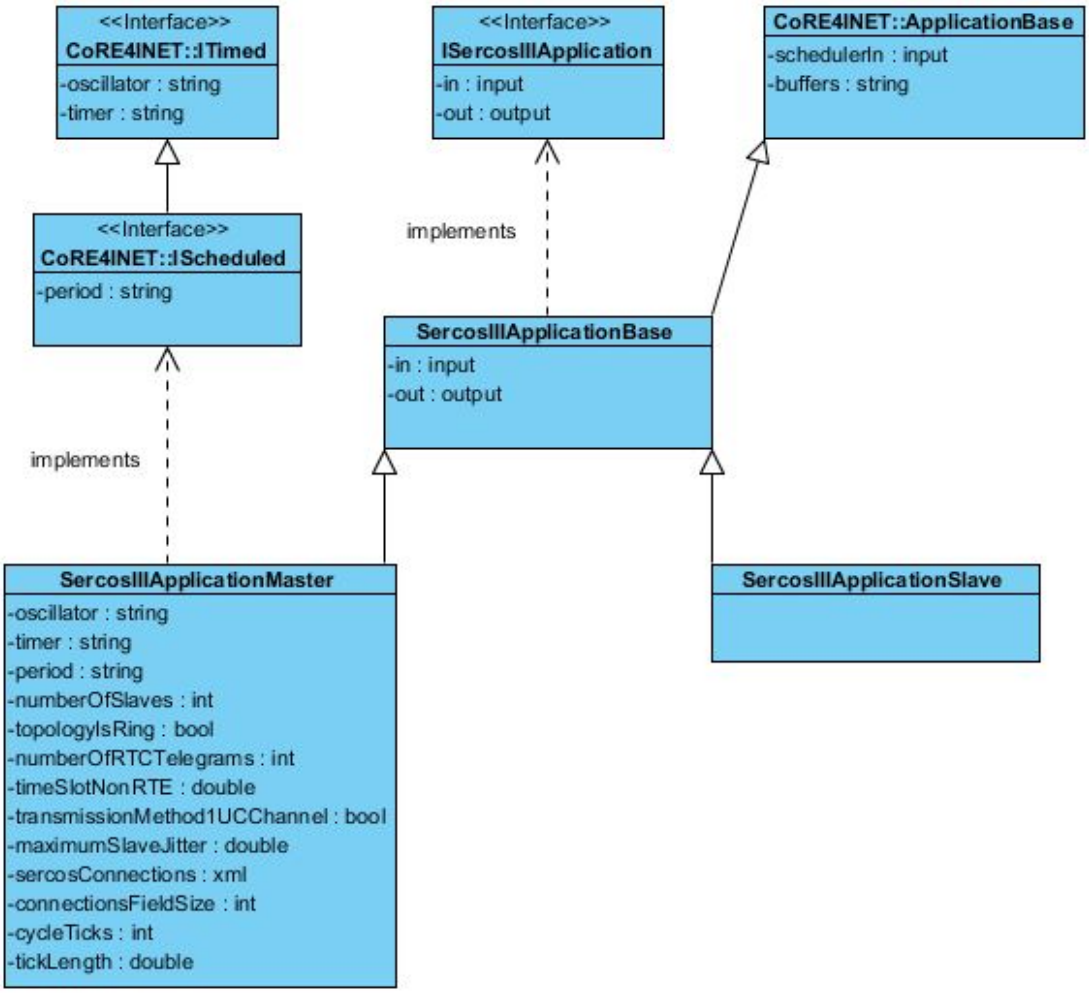

Abbildung 5.4: Die SERCOS III NED-Module in der Hierarchie (Quelle: eigene Grafik, erstellt mit Visual Paradigm)

Verhalten

Die Vererbungshierarchie der C++ Klassen, welche das Verhalten der Module beschreiben, ist analog zu derjenigen der NED Module (Siehe: Klassendiagramm (Abbildung 5.5, S. 36) Für eine bessere Übersicht wurden Felder und Methoden die in diesem Kontext weniger relevant sind weggelassen (z.B. die ”initialize”-Methoden)). Auf ein Interface für Klassen für SERCOS III Anwendungen wurde verzichtet, da die Basisklasse vollkommen ausreichend ist. ”SercosIIIApplicationMaster” ist neben dieser Basisklasse auch von ”Scheduled” abgeleitet. Aus den Parametern des Simulationsmodells ermittelt ”SercosIIIApplicationMaster” die Sendezeiten aller Telegramme (MDTs/ATs) im Kommunikationszyklus und registriert für jedes zu sendende Telegramm beim ”Scheduler” ein ”SchedulerActionTimeEvent” mit der entsprechenden Sendezeit. Der ”Scheduler” benachrichtigt ”SercosIIIApplicationMaster” dann, wenn die jeweilige Zeit erreicht ist, indem er eine Nachricht an ”SercosIIIApplicationMasters” ”shedulerIn” Gate schickt. Daraufhin versendet ”SercosIIIApplicationMaster” das entsprechende ”SercosIIITelegram” über sein ”output” Gate. Für das Registrieren der ”SchedulerActionTimeEvents” hat ”SercosIIIApplicationMaster” die Methode ”registerSchedulerEvent”, welche als Parameter die Sendezeit bekommt.

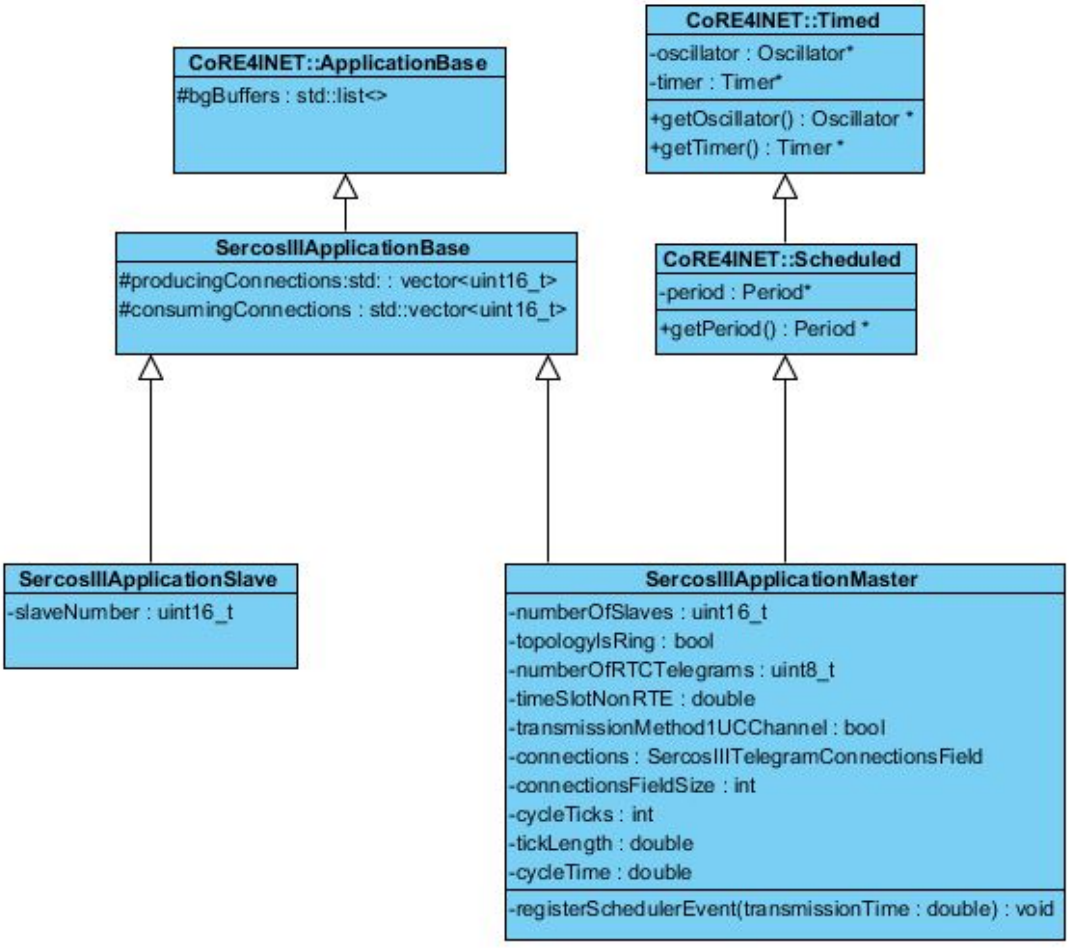

Abbildung 5.5: Klassendiagramm der SERCOS III Applications (Quelle: eigene Grafik, erstellt mit Visual Paradigm)

5.3.3 TSN-Interface Layer

Das TSN-Interface Layer Modul ist die zentrale Komponente, welche der SERCOS III Anwendung ein Interface für den Zugriff auf den OSI-Layer 2 bietet. Je nachdem welches Protokoll auf OSI-Layer 2 verwendet werden soll, kann ein anderes TSN-Interface Layer Modul eingesetzt werden. Im Rahmen dieser Arbeit wurden ein Best-Effort Modul für Standard Ethernet und ein TT Modul für die Abbildung von SERCOS III auf TT implementiert.

Module

Es wurde ein TSN-Interface Layer Basismodul "TSNInterfaceBase" entwickelt von dem alle anderen Module für die unterschiedlichen OSI-Layer 2 Protokolle ("TSNInterfaceBestEffort" und "TSNInterfaceTT") abgeleitet sind (Siehe: NED Klassendiagramm (Abbildung 5.6, S. 38)). "TSNInterfaceBase" hat jeweils ein "input" Gate für "SercosIIITelegrams" ("SERCOSIIIin") und UC-Nachrichten ("UCin") von den über dem TSN-Interface Layer liegenden SERCOS III- und UC-Anwendungen, jeweils ein "input" Gate für Ethernet Frames (sowohl SERCOS III, als auch UC) von den beiden OSI-Layer 2 Ports ("BEinPort1", "BEinPort2") und ein "output" Gate ("SERCOSIIIout") um vom OSI-Layer 2 erhaltene SERCOS III Telegramme an die SERCOS III Anwendung weiterzugeben (UC-Nachrichten werden nicht über ein separates Gate, sondern über einen Buffer an die UC-Anwendung weitergegeben (Siehe Abschnitt 5.3.4)). Die Ethernet-Frames an den OSI-Layer 2 werden nicht über "output" Gates, sondern direkt an Buffer versendet (Siehe Abschnitt 5.3.4). Es implementiert das Interface "ITSNInterface", welches diese Gates spezifiziert und ist abgeleitet von CoRE4INETs "ApplicationBase". Außerdem hat "TSNInterfaceBase" Parameter zum Versenden von Ethernet Frames auf OSI-Layer 2: die MAC Adressen des Nachfolgers und Vorgängers im Ring oder in der Linie ("nextNodeAddress" und "previousNodeAddress"), die eigene MAC Adresse ("deviceMACAddress"), den Ether-Typ für die SERCOS III Frames (Default: 0x88CD) und ob ein Port deaktiviert ist ("deactivatePort") und über welchen der beiden Ports empfangene Frames gesendet werden sollen ("operatingModeFastForwarding"). "TSNInterfaceBestEffort" benötigt keine weiteren Gates oder Parameter. Über die beiden Gates "BEinPort1" und "BEinPort2" erhält es alle Frames vom darunter liegenden OSI-Layer 2. "TSNInterfaceTT" hat zusätzlich noch zwei Gates für TT-Frames die über einen der beiden Ports gekommen sind ("TTinPort1" und "TTinPort2"). Standard Ethernet Frames vom OSI-Layer 2 erhält es weiterhin über "BEinPort1" und "BEinPort2".

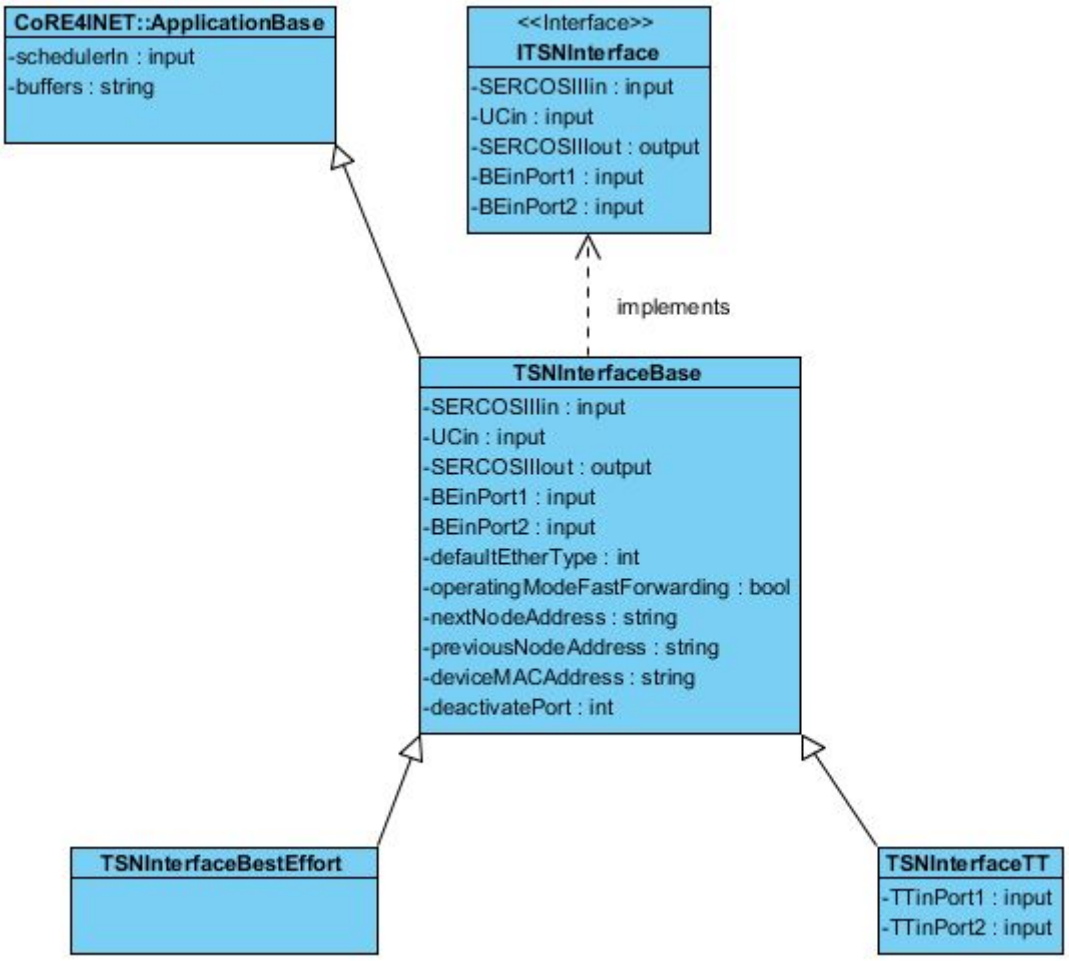

Abbildung 5.6: Die TSN-Interface NED-Module in der Hierarchie (Quelle: eigene Grafik, erstellt mit Visual Paradigm)

Verhalten

Die Vererbungshierarchie der C++ Klassen, welche das Verhalten der Module beschreiben, ist analog zu derjenigen der NED Module (Siehe: Klassendiagramm (Abbildung 5.7, S. 40) Für eine bessere Übersicht wurden Felder und Methoden die in diesem Kontext weniger relevant sind weggelassen (z.B. die ”initialize”-Methoden)). Auf ein Interface für Klassen für TSN-Interface Layer Anwendungen wurde verzichtet, da die Basisklasse vollkommen ausreichend ist. ”TSNInterfaceBase” ist von CoRE4INETs ”CTApplicationBase” abgeleitet, welche wiederum von ”ApplicationBase” abgeleitet ist. Das liegt daran, dass alle TSN-Interface Layer Klassen, bis auf ”TSNBestEffort”, nicht mit Best-Effort Buffern auskommen, sondern z.B. TT-Buffer, RC-Buffer oder AVB-Buffer benötigen, da sie SERCOS III auf TT, RC oder AVB abbilden. Alle diese Buffer sind von CoRE4INETs ”CTBuffer” abgeleitet und ”CTApplicationBase” ermöglicht über die Variable ”ctBuffers” Zugriff auf ”CTBuffer”.
Die von ”TSNInterfaceBase” abgeleiteten Klassen (”TSNBestEffort” und ”TSNTT”) regeln das Versenden von Ethernet-Frames auf OSI-Layer 2. Sie kennen die MAC Adressen des Nachfolgers und Vorgängers im Ring oder in der Linie (”nextNodeAddress” und ”previousNodeAddress”) und die eigene MAC Adresse (”deviceMACAddress”). Sie setzen den Ether-Typ für die SERCOS III Frames (Default: 0x88CD) und wissen ob ein Port deaktiviert ist (”deactivatePort”). Sie entscheiden, unter Berücksichtigung des Modus des Devices (”operatingModeFastForwarding”) und ob ein Port deaktiviert wurde, über welchen der beiden Ports zu sendende Frames geschickt werden sollen.

Best Effort

Mit ”TSNBestEffort” als TSN-Interface Layer Komponente wird auf OSI-Layer 2 Best-Effort verwendet, also Standard Ethernet, wie es für SERCOS III vorgesehen ist. Ethernet-Frames (sowohl SERCOS III als auch UC) vom OSI-Layer 2 werden von ”TSNBestEffort” an die darüber liegende Anwendung weiter gereicht. Die ”SercosIIITelegrams” werden dazu aus ihrem Ethernet-Frame (Ether-Type: 0x88CD) entpackt. Von der über ”TSN-BestEffort” liegenden SERCOS III Anwendung über ”SERCOSIIIin” erhaltene ”SercosII-ITelegrams” werden in Ethernet-Frames mit entsprechendem Ether-Type gepackt und sofort an BE-Buffer gesendet, von wo aus sie über einen der beiden Ports an das nächste

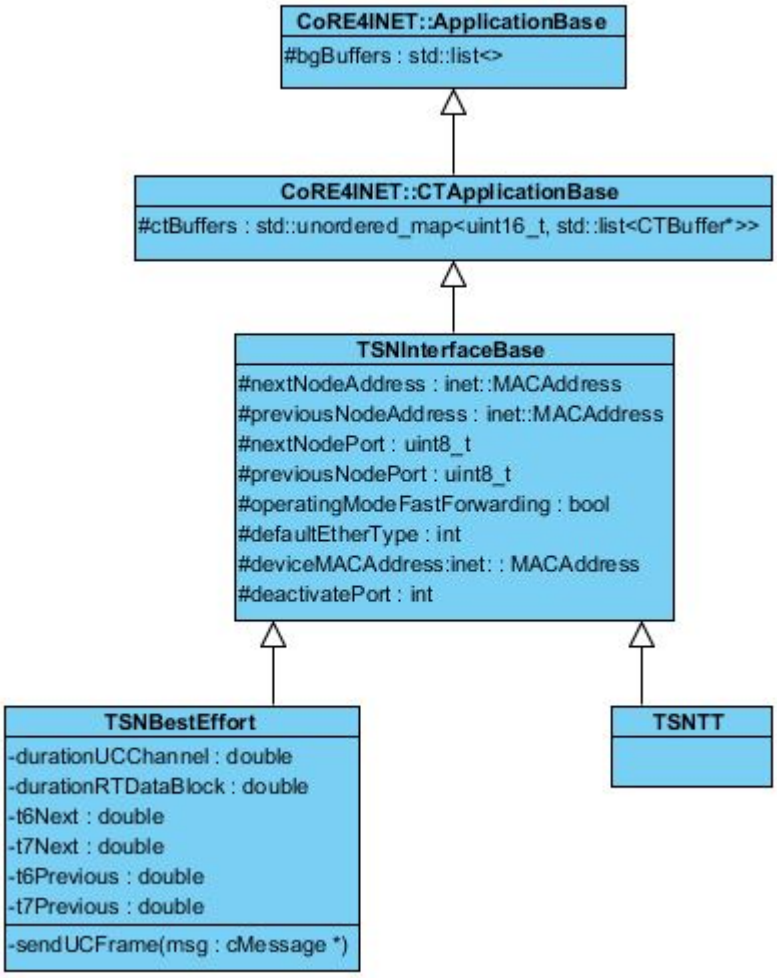

Abbildung 5.7: Klassendiagramm der TSN-Interface Module (Quelle: eigene Grafik, erstellt mit Visual Paradigm)

Device gesendet werden. Über ”UCin” oder einen der beiden Ports (”BEinPort1” oder ”BEinPort2”) erhaltene UC-Frames die an ein anderes Device adressiert sind, dürfen nur im UC-Kanal versendet werden. Dafür sorgt ”TSNBestEffort”. Der Anfang (t6) und das Ende (t7) des UC-Kanals sind immer relativ zum Anfang des MDT0. Ein Device bekommt im Fall der Ringtopologie 2 MDT0 und im Fall der Linientopologie dasselbe MDT0 2-mal innerhalb eines Zyklus. ”TSNBestEffort” berechnet bei jedem Eintreffen eines MDT0 die Zeiten t6 und t7 (jeweils ”t6Next” und ”t7Next” wenn das MDT0 vom Vorgänger im Ring oder in der Linie kam (also an den Nachfolger ”next” gesendet werden soll) und ”t6Previous” und ”t7Previous” wenn es vom Nachfolger im Ring oder in der Linie kam (also an den Vorgänger ”previous” gesendet werden soll)). Ein anstehender UC-Frame wird nur dann versendet, wenn er in das Zeitfenster passt. Zum Versenden von UC-Frames hat ”TSNBestEffort” die Methode ”sendUCFrame” die als Parameter einen Pointer auf den zu versendenden Frame bekommt.

Time-Triggered

Mit ”TSNTT” als TSN-Interface Layer Komponente wird auf OSI-Layer 2 TT verwendet. Damit kann die Abbildung von SERCOS III auf TT realisiert werden. Im Gegensatz zu ”TSNBestEffort” muss ”TSNTT” nicht dafür sorgen, dass alle Zeitfenster eingehalten werden, sondern TT auf OSI-Layer 2 sorgt dafür, dass die SERCOS III Frames (TT-Frames) zu den richtigen Zeiten versendet werden und dass die UC-Frames im UC-Kanal versendet werden. ”TSNTT” legt zu sendende UC-Frames einfach in Best-Effort Buffer, während ”SercosIIITelegrams” von der SERCOS III Anwendung in TT-Frames gekapselt werden (nachdem sie vorher nochmal in ”EthernetIIFrames” mit dem SERCOS III Ether-Type gekapselt wurden) und in TT-Buffern abgelegt werden. Zur Anwendungsschicht hin (”to upper layer”) funktioniert ”TSNTT” genauso wie ”TSNBestEffort”, nur dass die ”SercosIIITelegrams” vorher noch aus TT-Frames entpackt werden müssen.

5.3.4 SERCOS III Device

Das Zusammenspiel der vorgestellten Komponenten ergibt ein SERCOS III Device "SercosIIINodeTT" (Siehe: Abbildung 5.8), welches in jedem simulierten Netzwerk eingesetzt werden kann. Dieses enthält neben den vom Simulationsmodell zur Verfügung gestellten Modulen OSI-Layer 2 Module von CoRE4INET. Die Verwendung des "Schedulers" durch "SercosIIIApplicationMaster" wurde bereits beschrieben. Das TSN-Interface Layer Modul ("tsnInterfaceApplication") erhält "EthernetIIFrames" von 2 Input BE-Buffern (über die Gates: "BEinPort1" und "BEinPort2") und versendet "EthernetIIFrames" an 2 Output BE-Buffer. TT-Frames erhält "tsnInterfaceApplication" (über die Gates: "TTinPort1"und "TTinPort2") von 2 Input TT-Buffern (CoRE4INET "TTDoubleBuffer") vor die jeweils ein CoRE4INET "TTIncoming" geschaltet ist und versendet an 2 Output TT-Buffer vor die ebenfalls jeweils ein "TTIncoming" geschaltet ist. Die BE-Buffer, Output TT-Buffer und Input TT-Incomings sind mit zwei Ports verbunden. Für die Ports wurden "TTEPHYPorts" verwendet. Die Anzahl der benötigten TT-Buffer und Incomings, wenn auf OSI Layer 2 TT verwendet wird, hängt von der Anzahl der Telegramme ab, die versendet werden. Sind bspw. 4 MDTs/ATs konfiguriert, wird für jedes Senden und Empfangen eines Telegramms jeweils ein TT-Buffer und Incoming benötigt. Im Fall des Rings sind das für jedes Device jeweils 32 TT-Buffer und Incomings (es werden 2-mal jeweils 8 Telegramme gesendet und empfangen ($2 \cdot 8 \cdot 2 = 32$).

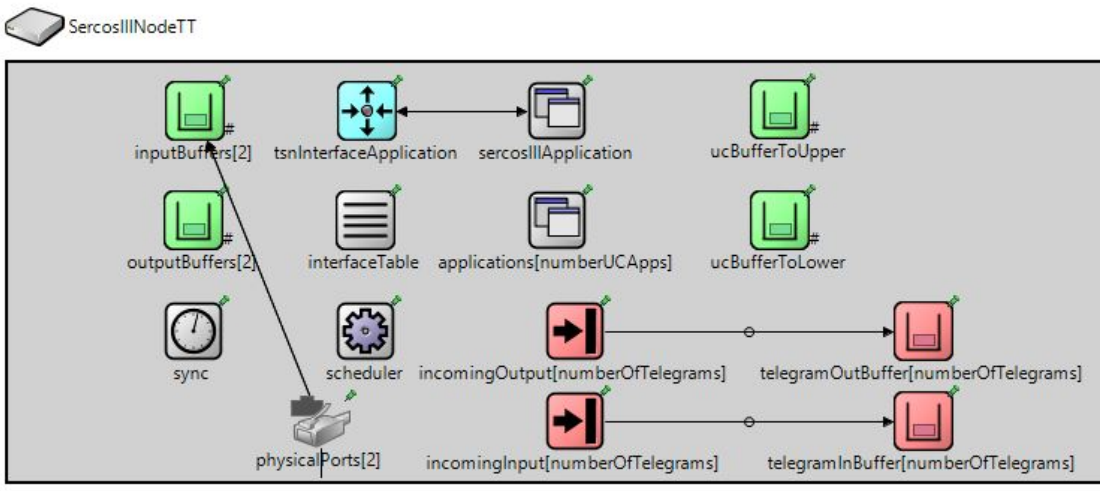

Abbildung 5.8: Aufbau eines SERCOS III Device mit TT (Quelle: eigene Grafik, Screenshot)

Für die Generierung von Cross-Traffic ist außerdem ein ”UCApplicationDummy” Modul enthalten, welches von CoRE4INETs ”ApplicationBase” abgeleitet ist und UC-Frames an einen BE-Buffer (”ucBufferToLower”) sendet, der sie an das TSN-Interface Layer Modul weiterleitet (Gate: ”UCin”). Das TSN-Interface Layer Modul schickt von einem anderen Device empfangene, an dieses Device adressierte UC-Frames über einen BE-Buffer (”ucBufferToUpper”) an ”UCApplicationDummy”. Dieses ”UCApplicationDummy” Modul kann auch in nicht-SERCOS III Devices (z.B. ”TTEEtherHost”) zum Erzeugen von Cross-Traffic verwendet werden. Die Größe der UC-Frames und wie häufig diese versendet werden sollen, kann eingestellt werden. Alle verwendeten BE-Buffer sind CoRE4INET ”BGQueueBuffer”.
Soll auf OSI-Layer 2 nur Standard Ethernet (”Best Effort”) verwendet werden, kann ”SercosIIINode” statt ”SercosIIINodeTT” verwendet werden. ”SercosIIINodeTT” ist von ”SercosIIINode” abgeleitet und erweitert diesen um die ”TTIncoming” und ”TTDoubleBuffer” Module.

Dadurch dass die SERCOS III Anwendungen komplett unabhängig von dem darunter liegenden TSN-Interface Layer sind und damit auch von dem auf OSI-Layer 2 verwendeten Protokoll (d.h. die SERCOS III Anwendung ”weiß” nicht einmal ob auf Layer 2 z.B. Best-Effort oder TT verwendet wird), kann die TSN-Interface Layer Komponente beliebig ausgetauscht werden und so SERCOS III auf die unterschiedlichen OSI-Layer 2 Protokolle abgebildet werden. Damit wurde das OMNeT++ Simulationsmodell für SERCOS III mit TSN Interface erfolgreich erstellt.

6 Qualitätssicherung, Tests und Evaluation

Dieses Kapitel befasst sich mit der Qualitätssicherung der Arbeit und den mit dem Simulationsmodell durchgeführten Tests und deren Evaluation. Da im Rahmen dieser Arbeit nur TSN-Interface Layer Module für die Verwendung von TT und Best-Effort auf OSI-Layer 2 implementiert wurden, beschränken sich die Tests entsprechend auf TT und Best-Effort auf OSI-Layer 2. AVB und RC können aber analog zu TT getestet werden.

6.1 Qualitätssicherung und Tests

Für die Qualitätssicherung und Tests wurden zwei Test-Netzwerke erstellt: 1) ein klassisches SERCOS III Netzwerk mit 4 SERCOS III Devices die einen physikalischen Ring bilden (Siehe: Abbildung 6.1) und 2) ein komplexeres, heterogenes Netzwerk mit 4 SERCOS III Devices, 2 Switches und 3 nicht-SERCOS III Devices (Siehe: Abbildung 6.2, S. 45).

Das erste Netzwerk wurde verwendet, um zu testen, ob das SERCOS III Simulationsmodell das SERCOS III Protokoll korrekt implementiert, d.h. ob die Telegramme richtig über das Netzwerk versendet werden und die garantierten Roundtrip-Times und Jitter eingehalten werden. Die Tests mit diesem klassischen SERCOS III Netzwerk liefern auch eine Grundlage für einen Vergleich mit der Abbildung von SERCOS III auf TSN. Das zweite Netzwerk wurde verwendet, um die Abbildung von SERCOS III auf TT zu untersuchen. Neben dem SERCOS III-Traffic und UC-Traffic der SERCOS III Devices generieren hierbei die nicht-SERCOS III Devices (”crossTrafficSource”) Cross-Traffic.

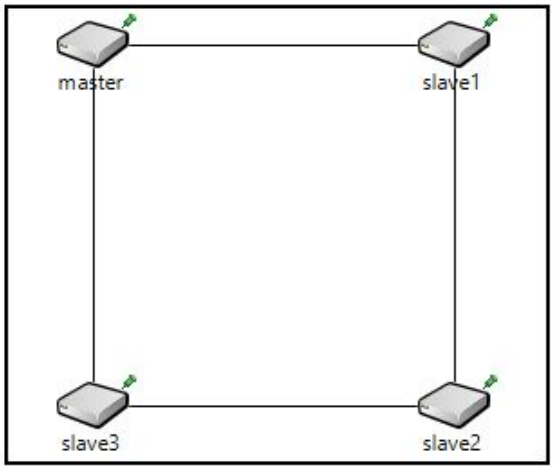

Abbildung 6.1: Ein klassisches SERCOS III Netzwerk. Die Länge der Links beträgt jeweils 10m. (Quelle: eigene Grafik, Screenshot)

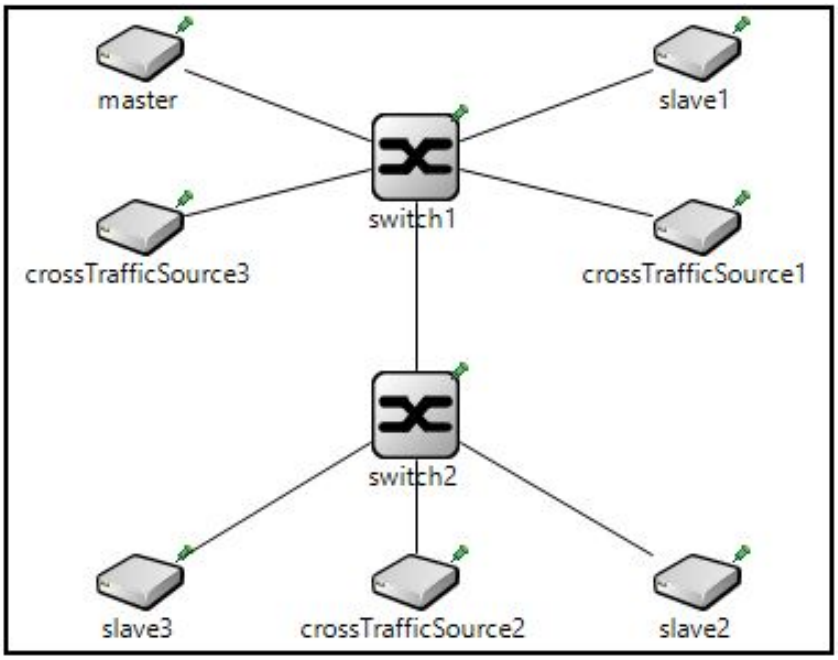

Abbildung 6.2: Ein komplexeres Netzwerk mit SERCOS III Devices, Switches und nicht-SERCOS III Devices. Die Länge der Links beträgt jeweils 4m. (Quelle: eigene Grafik, Screenshot)

Zum Generieren der Cross-Traffic Frames wird das vom Simulationsmodell zur Verfügung gestellte Modul "UCApplicationDummy" verwendet.

Testfälle

Zuerst wurden die Äquivalenzklassen für die Tests bestimmt. Das SERCOS III Protokoll ist unabhängig von der Größe der Echtzeit-Frames. D.h. das Verhalten ist allgemein gleich, für jede zulässige SERCOS III Frame-Größe (72 Byte bis 1526 Byte), einzig die absoluten Roundtrip-Times ändern sich. So ist die Roundtrip-Time größer, je größer die Echtzeit-Telegramme sind, da der Transmission Delay eines 1526 Byte Frames größer ist als der eines kleineren Frames. Das gleiche gilt für die Zeit, die von einem Device benötigt wird, um den Frame zu bearbeiten. Ein 1526 Byte Frame führt zu einem größeren Processing Delay als ein kleinerer Frame. Die Größe der UC-Frames hat keinen Einfluss auf das SERCOS III Protokoll und die Roundtrip-Times. Dieses Verhalten wird im Simulationsmodell dadurch realisiert, dass die Anwendungen unterschiedlich große Frames gleich behandeln. Deshalb wurden hier sowohl für die klassischen SERCOS III Tests, als auch für die Abbildung von SERCOS III auf TT die folgenden 4 Äquivalenzklassen identifiziert:
1) SERCOS III ohne UC-Traffic, Ring-Topologie,
2) SERCOS III mit UC-Traffic, Ring-Topologie,
3) SERCOS III ohne UC-Traffic, Linien-Topologie,
4) SERCOS III mit UC-Traffic, Linien-Topologie.

Für die Tests wurden folgende Repräsentanten der Äquivalenzklassen gewählt:

1) SERCOS III klassisch unter Verwendung des ersten Netzwerks (Siehe: Abbildung 6.1):

1.1) SERCOS III ohne UC-Traffic, Ring-Topologie, SERCOS III Frame-Größe: 72 Byte.

1.2) SERCOS III ohne UC-Traffic, Ring-Topologie, SERCOS III Frame-Größe: 1526 Byte.

1.3) SERCOS III mit UC-Traffic (UC-Frame Größe: 1526 Bytes, UC-Frame alle 150

us), Ring-Topologie, SERCOS III Frame-Größe: 72 Byte.

2) SERCOS III Abbildung auf TT unter Verwendung des zweiten Netzwerks (Siehe: Abbildung 6.2, S. 45):

2.1) SERCOS III ohne UC- und Cross-Traffic, logische Linien-Topologie, SERCOS III Frame-Größe: 74 Byte.

2.2) SERCOS III mit UC- und Cross-Traffic (UC-/Cross-Traffic Frame Größe: 1526 Bytes, UC-Frame alle 500 *us*), logische Linien-Topologie, SERCOS III Frame-Größe: 74 Byte.

2.3) SERCOS III mit UC- und Cross-Traffic (UC-/Cross-Traffic Frame Größe: 1526 Bytes, UC-Frame alle 300 *us*), logische Linien-Topologie, SERCOS III Frame-Größe: 74 Byte.

2.4) SERCOS III mit UC- und Cross-Traffic (UC-/Cross-Traffic Frame Größe: 1526 Bytes, UC-Frame alle 150 *us*), logische Linien-Topologie, SERCOS III Frame-Größe: 74 Byte.

3) Außerdem wurde der Einsatz von SERCOS III mit Abbildung auf BE in einem komplexeren Netzwerk mit Cross-Traffic untersucht. Dafür wurde ebenfalls das zweite Netzwerk verwendet (Siehe: Abbildung 6.2, S. 45) verwendet. Auf OSI-Layer 2 wurde hier also ausschließlich Standard Ethernet (Best-Effort):

3.1) SERCOS III ohne UC- und Cross-Traffic, logische Linien-Topologie, SERCOS III Frame-Größe: 72 Byte.

3.2) SERCOS III mit UC- und Cross-Traffic (UC-/Cross-Traffic Frame Größe: 1526 Bytes, UC-Frame alle 500 *us*), logische Linien-Topologie, SERCOS III Frame-Größe: 72 Byte.

3.3) SERCOS III mit UC- und Cross-Traffic (UC-/Cross-Traffic Frame Größe: 1526 Bytes, UC-Frame alle 300 us), logische Linien-Topologie, SERCOS III Frame-Größe: 72 Byte.

3.4) SERCOS III mit UC- und Cross-Traffic (UC-/Cross-Traffic Frame Größe: 1526 Bytes, UC-Frame alle 150 us), logische Linien-Topologie, SERCOS III Frame-Größe: 72 Byte.

Für alle Tests wurde ein Hardware Jitter von $0ns$ festgelegt. Es liegt allerdings ein Clock-Jitter von $< 0,3\mu s$ vor. Für jeden Test wird die Roundtrip-Time und deren Jitter gemessen, da das die beiden entscheidenden Kenngrößen des SERCOS III Protokolls sind. Für die Tests werden jeweils die folgenden Roundtrip-Times und Jitter erwartet:

1) Propagation- und Transmission Delays pro Link:

$$Propagation\ Delay = \frac{10\ m}{2 \cdot 10^8\ m/s} = 50ns = 0,05\mu s$$

$$Transmission\ Delay\ (72\ Byte) = \frac{576\ Bit}{100 \cdot 10^6\ Bit/s} = 5,76\mu s$$

$$Transmission\ Delay\ (1526\ Byte) = \frac{12208\ Bit}{100 \cdot 10^6\ Bit/s} = 122,08\mu s$$

Processing Delays: $0\mu s$ für alle Devices.

1.1) Erwartete Roundtrip-Time ($RTT_{exp.}$) und Jitter ($Jitter_{exp.}$):

$RTT_{exp.} = (5,76\mu s + 0,05\mu s) \cdot 4 = 23,24\mu s$

$Jitter_{exp.} < 0,3\mu s.$

1.2) Erwartete Roundtrip-Time ($RTT_{exp.}$) und Jitter ($Jitter_{exp.}$):

$RTT_{exp.} = (122,08\mu s + 0,05\mu s) \cdot 4 = 488,52\mu s$

$Jitter_{exp.} < 0,3\mu s.$

1.3) Es werden dieselben Roundtrip-Time und Jitter erwartet wie in 1.1). Erwartete Roundtrip-Time ($RTT_{exp.}$) und Jitter ($Jitter_{exp.}$):

$RTT_{exp.} = (5,76\mu s + 0,05\mu s) \cdot 4 = 23,24\mu s$

$Jitter_{exp.} < 0,3\mu s.$

2) Propagation- und Transmission Delays pro Link:

$$Propagation\ Delay = \frac{4\ m}{2 \cdot 10^8\ m/s} = 20ns = 0,02\mu s$$

$$Transmission\ Delay\ (74\ Byte) = \frac{592\ Bit}{100 \cdot 10^6\ Bit/s} = 5,92\mu s$$

Processing Delays: $1\mu s$ für alle Devices.

Bei der Abbildung von SERCOS III auf TT muss jedes Senden und Empfangen der Echtzeit-Telegramme (MDT0, MDT1, ...) gescheduled werden. Da dies nicht beliebig genau möglich ist, entsteht bei jedem Senden und Empfangen eine Verzögerung, abhängig von der Größe des konfigurierten Zeitfensters. Diese Verzögerung beträgt bei der hier verwendeten Konfiguration jeweils ca. $1\mu s$. Da die Zeitfenster für die unterschiedlichen Telegramm-Typen (MDT0, MDT1, ...) nicht exakt gleich sind, ist zwar für den Jitter eines Telegramm-Typs (z.B. MDT0) $< 0,3\mu s$ zu erwarten, aber der Jitter für alle Telegramme allgemein wird konfigurationsbedingt $> 0,3\mu s$ sein.

2.1)-2.4) Es werden dieselben Roundtrip-Times und Jitter erwartet. Erwartete Roundtrip-Time ($RTT_{exp.}$) und Jitter ($Jitter_{exp.}$):

$RTT_{exp.} = (1\mu s + 5,92\mu s + 0,02\mu s + 1\mu s) \cdot 14 = 111,16\mu s$

$Jitter_{exp.} > 0,3\mu s.$
3) Propagation- und Transmission Delays pro Link:

$$Propagation\ Delay = \frac{4\ m}{2 \cdot 10^8\ m/s} = 20ns = 0,02\mu s$$

$$Transmission\ Delay\ (72\ Byte) = \frac{576\ Bit}{100 \cdot 10^6\ Bit/s} = 5,76\mu s$$

Processing Delays: $8\mu s$ für die Switches und $0\mu s$ für alle übrigen Devices.

Bei den Tests unter "3)" ist zu erwarten, dass die theoretischen Roundtrip-Times und Jitter, welche bei Kollisionsfreiheit von Echtzeit-Kanal und Cross-Traffic vorliegen würden, nicht eingehalten werden, da bei BE auf OSI-Layer 2 diese Kollisionsfreiheit nicht gegeben ist.

3.1)-3.4) Theoretische Roundtrip-Time ($RTT_{exp.}$) und Jitter ($Jitter_{exp.}$):

$RTT_{exp.} = (5,76\mu s + 0,02\mu s) \cdot 14 + 8\mu s \cdot 8 = 144,92\mu s$

$Jitter_{exp.} < 0,3\mu s.$

Qualitätssicherung

Für die Qualitätssicherung wurden die definierten Tests für das klassische SERCOS III Netzwerk durchgeführt. Alle Tests waren erfolgreich (Siehe: Abschnitt 6.2). Für die Überprüfung, ob die Telegramme korrekt über das Netzwerk versendet werden, wurden neben Zeitmessungen auch Ausgaben der SERCOS III Anwendungen analysiert. Anhand der Dummy-Daten und Ausgaben von Informationen der Telegramm Header konnte gezeigt werden, dass die Telegramme korrekt durch den Ring gelaufen sind und von den SERCOS III Anwendungen verarbeitet wurden.
Um eine Aussage über den Determinismus des Simulationsmodells machen zu können,

wurden für Test 1.1) ("SERCOS III klassisch ohne UC-Traffic, Ring-Topologie, SERCOS III Frame-Größe: 72 Byte") 5 Testdurchläufe durchgeführt (jeder Testdurchlauf dauerte 7 Kommunikationszyklen) und die Ausgaben der SERCOS III Anwendungen und gemessene Roundtrip-Times und Jitter verglichen. Alle waren wie erwartet identisch.
Außerdem wurden die Tests für die Abbildung von SERCOS III auf TT mit einer logischen Linien-Topologie durchgeführt. Da so jeweils einmal der Ring und einmal die Linie erfolgreich getestet wurden, wurde auf Tests für SERCOS III klassisch mit physikalischer Linien-Topologie und der Abbildung von SERCOS III auf TT mit logischem Ring verzichtet.

6.2 Ergebnisse und Evaluation

Für die Testfälle liegen folgende gemessene Zeiten vor:

1.1) Gemessene Roundtrip-Time (RTT) und Jitter ($Jitter$):
$RTT = 23,24\mu s$
$Jitter < 0,3\mu s$.

1.2) Gemessene Roundtrip-Time (RTT) und Jitter ($Jitter$):
$RTT = 488,52\mu s$
$Jitter < 0,3\mu s$.

1.3) Gemessene Roundtrip-Time (RTT) und Jitter ($Jitter$):
$RTT = 23,24\mu s$
$Jitter < 0,3\mu s$.

2.1-2.4) Gemessene Roundtrip-Time (RTT) und Jitter ($Jitter$):
$RTT = 111,16\mu s$
$Jitter < 3,96\mu s$.
$Jitter\ selber\ Telegramm - Typ < 0,3\mu s$.

3.1) Gemessene Roundtrip-Time (RTT) und Jitter ($Jitter$):
$RTT = 144,92\mu s$
$Jitter < 0,3\mu s$.

3.2) Gemessene Roundtrip-Time (RTT) und Jitter ($Jitter$):
$RTT = 144,92\mu s$
$Jitter < 0,3\mu s$.

3.3) Gemessene Roundtrip-Time (RTT) und Jitter ($Jitter$):
$\varnothing RTT = 344,67\mu s$
$Jitter < 200,35\mu s$.

3.4) Gemessene Roundtrip-Time (RTT) und Jitter ($Jitter$):
$RTT : divergiert$ (bedeutet Congestion des Netzwerks)
$Jitter : divergiert$ (bedeutet Congestion des Netzwerks)

Die gemessenen Zeiten (Roundtrip-Times und Jitter) entsprechen den erwarteten Zeiten. Somit wurde das Simulationsmodell einerseits validiert und andererseits gezeigt, dass eine Abbildung von SERCOS III auf TT eine konstante Roundtrip-Time und Jitter garantiert und damit das gewünschte Ziel - der Einsatz von SERCOS III in komplexen und heterogenen Netzwerken - erreicht wurde.
Man sieht in dem Diagramm (Abbildung 6.3, S. 53) dass sich SERCOS III auf TT abgebildet in einem komplexeren Netzwerk mit Cross-Traffic in Hinsicht auf die Roundtrip-Time genauso verhält wie das klassische SERCOS III in einem physikalischen Ring mit Standard Ethernet auf OSI-Layer 2. Die Roundtrip-Time ist konstant und unabhängig vom UC- bzw. Cross-Traffic. Der absolute Zeitunterschied liegt vor allem daran, dass sich beide Messungen auf unterschiedliche Netzwerke beziehen. Das gleiche gilt auch für den Jitter (Siehe: Abbildung 6.4, S. 53). Zum Vergleich nimmt bei SERCOS III auf BE abgebildet die Performance bezüglich der Roundtrip-Time und des Jitters in einem komplexeren Netzwerk wie erwartet mit zunehmendem Cross-Traffic ab. Der Einsatz von SERCOS III in einem anderen Netzwerk als einem physikalischen Ring oder einer physikalischen Linie ausschließlich mit SERCOS III Devices ist mit Standard Ethernet (BE) auf OSI-Layer 2 nicht möglich, sondern erfordert TSN auf OSI-Layer 2.

An dieser Stelle wird die Abbildung von SERCOS III auf TT abschließend bewertet:

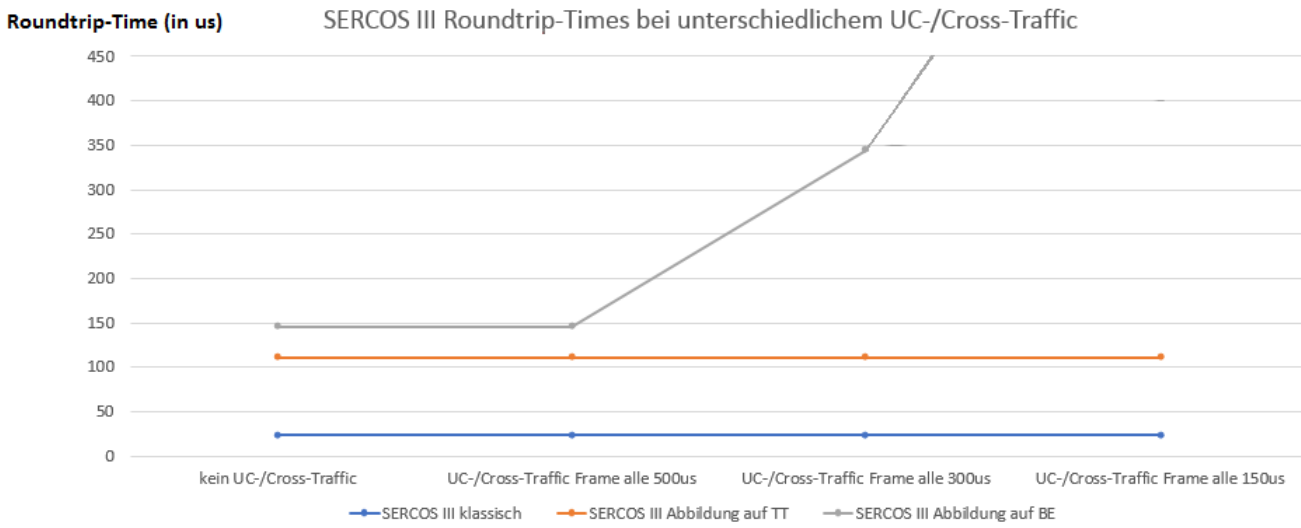

Abbildung 6.3: SERCOS III Roundtrip-Times bei unterschiedlichem UC-/Cross-Traffic (Quelle: eigene Grafik, Screenshot)

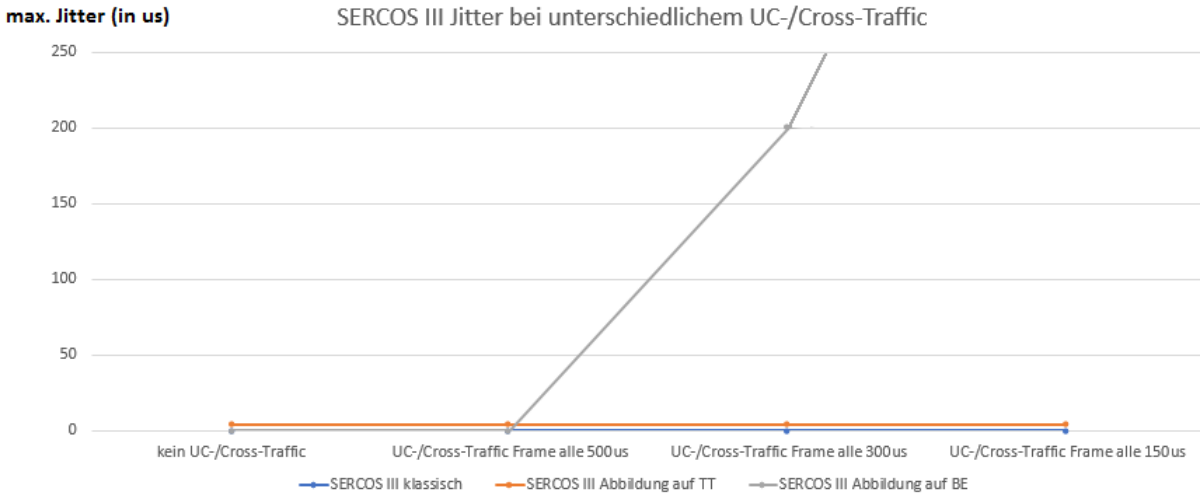

Abbildung 6.4: SERCOS III Jitter bei unterschiedlichem UC-/Cross-Traffic (Quelle: eigene Grafik, Screenshot)

Der große Vorteil einer Abbildung von SERCOS III auf TT gegenüber dem klassischen SERCOS III ist, dass SERCOS III auf diese Weise in komplexen Netzwerken mit Switches und anderen Devices, welche das SERCOS III Protokoll nicht unterstützen, verwendet werden kann.
Allerdings bringt die Abbildung von SERCOS III auf TT auch einige Nachteile mit sich: sie ist vor allem mit einem hohen offline Konfigurationsaufwand verbunden. Das Timing für das Senden und Empfangen jedes (SERCOS III) TT-Frames muss ermittelt und konfiguriert werden, was nicht beliebig exakt möglich ist und daher eine zusätzliche Verzögerung bedeutet. Auch ist es auf OSI-Layer 2 nicht mehr möglich, Standard Ethernet zu verwenden, sondern es muss TT unterstützt werden. Ein offensichtlicher Nachteil, der hier trotzdem erwähnt werden soll, ist die Kehrseite des Einsatzes von SERCOS III in komplexen Netzwerken: die Switches führen zu längeren Roundtrip-Times und sind zusätzliche ”Points of failure”, wobei die längeren Roundtrip-Times höchstens in extremen Anwendungsfällen relevant sein können. Das Problem der ”Points of failure” kann durch ausreichend Redundanz im Netzwerk gelöst werden.

7 Zusammenfassung und Ausblick

Zusammenfassung

Das Ziel dieser Arbeit war die Abbildung von SERCOS III auf TSN um den Einsatz von SERCOS III in komplexen, heterogenen Netzwerken zu ermöglichen. Dafür wurde ein OMNeT++ Simulationsmodell mit TSN-Interface entwickelt. Das Konzept dieses Simulationsmodells beruht auf der Bereitstellung eines Interfaces ("TSN-Interface Layer") über das die Anwendungsschicht, welche die SERCOS III Anwendungen realisiert, auf den OSI-Layer 2 zugreift. Dieser TSN-Interface Layer macht die Abbildung von SERCOS III auf TSN möglich. Im Rahmen dieser Arbeit wurden TSN-Interface Layer Komponenten implementiert, welche neben dem Zugriff auf Standard Ethernet (Best-Effort) den Zugriff auf TT erlauben, wodurch die Abbildung von SERCOS III auf TT möglich wird.
Mit diesem Simulationsmodell wurde zunächst das klassische SERCOS III (wie in Kapitel 4 definiert) simuliert. Das diente auch der Validierung des Simulationsmodells und lieferte ein Ausgangsszenario für den Vergleich mit der Abbildung von SERCOS III auf TT. Anschließend wurde die Abbildung von SERCOS III auf TT untersucht. Es zeigte sich, dass damit das Ziel, SERCOS III auch in komplexen, heterogenen Netzen einzusetzen, erreicht wird. Der Einsatz von SERCOS III muss also nicht auf einen einfachen physikalischen Ring oder eine Linie ohne nicht-SERCOS III Devices beschränkt werden, wenn auf OSI-Layer 2 TT verwendet wird. Die Abbildung von SERCOS III auf TT resultierte in einem Netzwerk mit Switches auch bei hohem Cross-Traffic in den erwarteten konstanten Roundtrip-Times und geringem Jitter, der ausschließlich von der Hardware und der Genauigkeit des TT-Schedulings abhängig ist. Allerdings ist der Einsatz von TT mit einem hohen offline Konfigurationsaufwand verbunden und durch Ungenauigkeit im TT-Scheduling entsteht eine zusätzliche Verzögerung, was minimal längere Roundtrip-Times bedeutet. Im Gegensatz dazu wurde gezeigt, dass bei Verwendung von Standard Ethernet auf OSI-Layer 2 (Best-Effort) der Einsatz von SERCOS III unter denselben Bedingungen zu längeren Roundtrip-Times und extremem Jitter führt, was der Grund für die Beschränkung von SERCOS III auf den Einsatz in einem physikalischen Ring

oder einer physikalischen Linie ohne Cross-Traffic bei Standard Ethernet auf OSI-Layer 2 ist.

Ausblick

Neben der Abbildung von SERCOS III auf TT, die im OMNeT++ Simulationsmodell implementiert wurde, wäre als nächstes eine Abbildung auf AVB und RC zu untersuchen. Dazu müsste das Simulationsmodell entsprechend erweitert werden. Da die Bibliothek CoRE4INET bereits die dafür nötigen Komponenten auf dem OSI-Layer 2 zur Verfügung stellt (z.B. AVB- und RC-Buffer), müssen diese nicht implementiert, sondern nur eingebaut werden. Die nächsten Schritte wären also die Erweiterung des Simulationsmodells um AVB und RC Komponenten auf dem TSN-Interface Layer (Siehe: Kapitel 5, Absatz 5.2), mit denen sich die Abbildungen realisieren lassen. Die Komponenten des Application Layer (Siehe: Kapitel 5, Absatz 5.2) - SERCOS III Anwendungen und UC-Anwendungen - können ohne Anpassung weiter verwendet werden, da die Komponenten auf dem TSN-Interface Layer austauschbar sind, weil die Architektur des Simulationsmodells den Einsatz von verschiedenen Modulen zum Zugriff auf unterschiedliche OSI Layer 2 Protokolle vorsieht.
Eine Abbildung von SERCOS III auf AVB hätte gegenüber TT den Vorteil, dass der hohe offline Konfigurationsaufwand entfällt. Allerdings sind bei AVB und RC größere Roundtrip-Times und Jitter zu erwarten.

Auch die Folgen der Abbildung von SERCOS III auf die unterschiedlichen Protokolle (TT, AVB und RC) für die Uhren-Synchronisation von SERCOS III könnten untersucht werden.

Das ”Guard band” vor einem geschedulten Frame bei TSN verhindert eine optimale Nutzung der Bandbreite, da es so groß sein muss, wie der größtmögliche Frame mit niedrigerer Priorität. Außerdem erhöht sich die Wartezeit von höher priorisierten Frames für den Zugang zum Medium wenn größere Best-Effort Frames versendet werden. Ein aktueller Ansatz zur Verbesserung von TSN hinsichtlich der Nutzung der Bandbreite und der Wartezeit von höher priorisierten Frames ist Frame-Preemption (Siehe: [Steinbach u.a. 2015]). Es ermöglicht eine Minimierung der Verschwendung von Bandbreite durch ”Guard bands” und der Wartezeit von höher priorisierten Frames für den

Zugang zum Medium. Bei der Abbildung von SERCOS III auf TSN bedeutet das für den Einsatz von TT eine effizientere Nutzung der Links und für den Einsatz von AVB und RC eine Verringerung der Roundtrip-Time und Jitter. Ein weiterer Aspekt wäre also die Untersuchung der Vorteile von Frame-Preemption bei der Abbildung von SERCOS III auf TSN. Dafür müsste insbesondere Frame-Preemption im Simulationsmodell implementiert werden.

Die Übertragung von SERCOS III über TSN aus dem OMNeT++ Simulationsmodell in eine reale Anwendung, z.B. in einer Industrie-Anlage oder im Automobil wäre die praktische Umsetzung der in dieser Arbeit ausgearbeiteten Grundlage.

Glossar

AVB Audio-Video Bridging, S.5

BE Best-Effort, S.4

DLPDU Data Link Protocol Data Unit, S.17

RC Rate-Constrained, S.4

SERCOS III Serial Real-time Communication System III, S.1

TDMA Time Division Multiple Access, S.4

TT Time-Triggered, S.4

TTE Time-Triggered Ethernet, S.4

TSN Time-Sensitive Networking, S.5

Literaturverzeichnis

[IEC 61158-4-19] DIN EN 61158-4-19, September 2015.

[IEC 61784-2] DIN EN 61784-2, Februar 2015, S.220-227.

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

[Steinbach u.a. 2011] STEINBACH, Till; DIEUMO KENFACK, Hermand; KORF, Franz; SCHMIDT, Thomas C.:An Extension of the OMNeT++ INET Framework for Simulating Real-time Ethernet with High Accuracy, 2011

[Steinbach u.a. 2015] STEINBACH, Till; LIM, Hyung-Taek; KORF, Franz; SCHMIDT, Thomas C.; HERRSCHER, Daniel; WOLISZ, Adam: Beware of the Hidden! How Cross-traffic Affects Quality Assurances of Competing Real-time Ethernet Standards for In-Car Communication, 2015 IEEE Conference on Local Computer Networks (LCN), Oktober 2015, S. 1-9, ISBN 978-1-4673-6770-7

[Meyer 2013] MEYER, Philipp: Simulationsbasierte Analyse der Integration von TDMA basierter Kommunikation in Ethernet AVB, 2013

[Sercos.DE] SERCOS - URL: http://www.sercos.de/

[OMNeT++ Community a] OMNeT++ Community: OMNet++ 5.0 - URL: https://omnetpp.org/

[OMNeT++ Community b] OMNeT++ Community: INET Framework for OMNet++ 5.0 - URL: https://inet.omnetpp.org/

[Simulcraft, Inc.] OMNEST - URL: https://omnest.com/

[TTTech a] Time-Triggered Ethernet - URL: https://www.tttech.com/technologies/deterministic-ethernet/time-triggered-ethernet/

[TTTech b] STEINER, Wilfried; BAUER Günther: TTEthernet: Time-Triggered Services for Ethernet Networks, 2009

[TTTech c] Time-Sensitive Networking - URL: https://www.tttech.com/technologies/deterministic-ethernet/time-sensitive-networking/

[Chongyuan u.a. 2010] CHONGYUAN, Hou; HANHONG, Jiang; YUAN, Yang; WANZHI, Rui; LIANWU, Hu: Research on Implementing Real Time Ethernet for Ship Power System, 2010

[Abel u.a. 2011] ABEL, Michael; CONTRERAS, Luis; KLEMM, Peter: Application of RT-Preempt Linux and Sercos III for Real-time Simulation, 2011

[Der-Wirtschaftsingenieur] Automatisierungspyramide - URL: https://www.der-wirtschaftsingenieur.de/