Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Jonas Engler

Network-Calculus-basierte Timing-Analyse mit Generierung von angenäherten Worst-Case-Simulationsszenarien für IEEE 802.1 TSN-Netzwerke

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Jonas Engler

Network-Calculus-basierte Timing-Analyse mit Generierung von angenäherten Worst-Case-Simulationsszenarien für IEEE 802.1 TSN-Netzwerke

Masterarbeit eingereicht im Rahmen der Masterprüfung

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

Betreuender Prüfer: Prof. Dr. Franz Korf
Zweitgutachter: Prof. Dr. Wolfgang Fohl

Eingereicht am: 30. September 2016

Jonas Engler

Thema der Arbeit
Network-Calculus-basierte Timing-Analyse mit Generierung von angenäherten Worst-Case-Simulationsszenarien für IEEE 802.1 TSN-Netzwerke

Stichworte
IEEE, 802.1, TSN, AVB, Worst-Case, Netzwerk, Latenzen, Verzögerung, Network Calculus, Simulation

Kurzzusammenfassung
Diese Abschlussarbeit ist auf Worst-Case-Latenzen in IEEE 802.1 TSN- und AVB-Netzwerken ausgerichtet. Basierend auf einer vorgenommenen Modellierung von Worst-Case-Szenarien wird mithilfe eines Network-Calculus-Modells für AVB- und TSN-Netzwerke eine Abschätzung für maximale Latenzen vorgenommen. Ein Tool, das ebenfalls auf Basis der Worst-Case-Modelle entwickelt wurde, zeigt, wie realistisch die Abschätzungen sind und zeigt Engpässe in der Planung eines Netzwerkes anhand einer von ihm generierten angepassten Simulation.

Jonas Engler

Title of the paper
Network Calculus based timing analysis with generation of approximated worst case simulation scenarios for IEEE 802.1 TSN networks

Keywords
IEEE, 802.1, TSN, AVB, worst case, network, latencies, delay, network calculus, simulation

Abstract
This thesis focusses on worst case latencies in IEEE 802.1 TSN and AVB networks. A modeling of worst case scenarios is used for an estimation of maximum latencies with a Network Calculus model for AVB and TSN networks. Based on the worst case scenarios, a tool was created to generate simulations capable of reaching these latencies to show the usefulness of the modeling and the Network Calculus estimation.

Inhaltsverzeichnis

1 Einleitung **1**
- 1.1 Einführung . . . 1
- 1.2 Motivation . . . 3
- 1.3 Ziele und Struktur der Arbeit . . . 3

2 Grundlagen **4**
- 2.1 Ethernet . . . 4
- 2.2 TDMA . . . 5
- 2.3 802.1Q . . . 6
- 2.4 AVB . . . 8
- 2.5 TTEthernet . . . 10
- 2.6 TSN . . . 11
  - 2.6.1 Funktionsweise . . . 11
  - 2.6.2 Shaper . . . 11

3 Related Work **15**
- 3.1 Basis dieser Arbeit . . . 15
- 3.2 Stand der Wissenschaft . . . 16
  - 3.2.1 Modular Performance Analysis . . . 16
  - 3.2.2 Formale Analyse per busy period . . . 17
  - 3.2.3 Shaper-Analyse . . . 17
  - 3.2.4 Compositional Performance Analysis . . . 17
  - 3.2.5 Warteschlangentheorie . . . 18

4 Worst-Case-Modellierung **20**
- 4.1 Vorgehen . . . 20
- 4.2 AVB . . . 22
  - 4.2.1 Einflüsse durch Best Effort . . . 22
  - 4.2.2 Einflüsse durch AVB-Klasse B . . . 23
  - 4.2.3 Einflüsse durch AVB-Klasse A . . . 24
  - 4.2.4 Gesamt-Szenario . . . 37
  - 4.2.5 Cycle Piling . . . 37
- 4.3 TSN . . . 42
  - 4.3.1 Einflüsse durch TSN-Pakete . . . 43

4.3.2 Gesamt-Szenario . . . 45
4.3.3 Cycle Piling und Bandbreitenbeschränkung . . . 50

5 Analyse **52**
5.1 Analyse-Methoden . . . 52
5.2 Network Calculus . . . 54
5.2.1 Mathematische Grundlagen . . . 54
5.2.2 Einführung . . . 55
5.2.3 Eigenschaften eines Flows . . . 56
5.2.4 Min-Plus-Algebra . . . 56
5.2.5 Arrival- und Service-Curve . . . 58

6 Network-Calculus-Modelle **61**
6.1 AVB . . . 61
6.1.1 NC-System unter AVB . . . 61
6.1.2 Parameter . . . 63
6.1.3 Arrival- und Service-Curve . . . 64
6.1.4 Worst-Case-Verzögerung . . . 65
6.2 TSN . . . 66
6.2.1 Begrenzungen und Einschränkungen des Modells . . . 67
6.2.2 Parameter . . . 70
6.2.3 Arrival- und Service Curve . . . 71
6.2.4 Worst-Case-Verzögerung . . . 72

7 Simulationsgenerierung **73**
7.1 Funktionen . . . 73
7.1.1 Benutzer-Funktionen . . . 73
7.1.2 Interne Funktionen . . . 75
7.2 Konzept . . . 76
7.2.1 Programmiersprache und Bibliotheken . . . 76
7.2.2 GUI . . . 76
7.2.3 Netzwerkkonstruktion (simpel) . . . 77
7.2.4 Netzwerkkonstruktion (vollständig) . . . 78
7.2.5 Routing der Pakete . . . 80
7.2.6 Erkennung Crosstraffic . . . 81
7.2.7 Neuplanung des Netzwerkes . . . 81
7.2.8 Erstellung neues Netzwerk . . . 83
7.3 Umsetzung . . . 83
7.3.1 Oberfläche . . . 83
7.3.2 Netzwerkkonstruktion . . . 84
7.3.3 Neuplanung des Traffics . . . 86
7.3.4 Prototyp-Funktionalität . . . 88
7.3.5 Erweiterungsmöglichkeiten . . . 90

7.4 Qualitätssicherung 91

8 Evaluierung **95**
8.1 Kleines Netzwerk 95
8.1.1 Aufbau 96
8.1.2 Ausgangslatenz 96
8.1.3 Generierte Simulation 97
8.1.4 Network-Calculus-Abschätzung 98
8.1.5 Einordnung 98
8.2 Worst-Case-Netzwerk 99
8.2.1 Aufbau 99
8.2.2 Ausgangslatenz 100
8.2.3 Generierte Simulation 101
8.2.4 Network-Calculus-Abschätzung 101
8.2.5 Einordnung 102
8.3 Komplexes Netzwerk 102
8.3.1 Aufbau 102
8.3.2 Ausgangslatenz 103
8.3.3 Generierte Simulation 104
8.3.4 Network-Calculus-Abschätzung 105
8.3.5 Einordnung 106
8.4 Überblick 106

9 Zusammenfassung und Ausblick **108**

Anhang **X**
1 Netzwerkerstellung X
2 Neuplanung Netzwerk XI

1 Einleitung

1.1 Einführung

Von der Pferdekutsche über Fahrzeuge mit Verbrennungsmotor bis zu der Renaissance der Elektromotoren ist die Automobilindustrie einem ständigen Wandel unterworfen. Neben diesen offensichtlichen Änderungen, die auch Einfluss auf die audiovisuelle Gestalt des Fahrzeuges nehmen, bleiben viele Entwicklungen dem Beobachter verborgen. Wie wird eigentlich die Scheibenwischer-Geschwindigkeit geregelt? Warum öffnet sich das Fenster, wenn ich einen Knopf drücke? Wieso piept das Auto, wenn ich einparke und einem anderen Auto zu nahe komme? Es gibt natürlich noch unzählige weitere Funktionen in Autos, die ähnliche Fragestellungen aufwerfen. Wie genau die einzelnen Einstellungen und Rückmeldungen ablaufen ist unterschiedlich, aber eines haben sie gemeinsam: Sie brauchen eine Kommunikations-Infrastruktur innerhalb des Fahrzeuges, um von Start- zu Endpunkt zu kommunizieren.

Um das zu bewerkstelligen, kann jedes Gerät mit seinem Steuer- oder Sensor-Mechanismus direkt verbunden werden. Vom Technologie-Level ist das der Ansatz, der noch in den späten 1970er und frühen 1980er Jahren gefahren wurde. Der sich daraus ergebende Nachteil bei einer steigenden Anzahl von Endgeräten ist offensichtlich: Die Kabel, die dafür gelegt werden müssen, werden zahlreicher und machen einen nicht unwesentlichen Teil des Gesamtgewichtes des Fahrzeug aus. Das verringert die Leistung des Fahrzeugs und ist teuer.

Die Lösung dafür sind Feldbusse. Diese bieten Kommunikation über eine einzige Leitung mit vielen angeschlossenen Teilnehmern. Der CAN-Bus [1] wird noch heute in nahezu allen modernen Fahrzeugen verbaut und dient als Infrastruktur der Kommunikation. Alle Daten, die vorher über eigene Leitungen transferiert wurden, können so über eine gemeinsame Leitung laufen. Das verringert Gewicht und Kosten.

Seit diesem Technologiewechsel sind mittlerweile mehr als 30 Jahre vergangen. Andere Feldbus-Technologien haben ebenfalls Einzug gehalten und erweitern die Möglichkeiten des Datenver-

sands. Ein Problem dieser Feldbusse ist für heutige Anforderungen die zu geringe Bandbreite. Bereits seit einigen Jahren sind Infotainment-Systeme im Auto nicht mehr die Ausnahme und benötigen teilweise eigene Infrastrukturen, um ihre Daten korrekt senden zu können. Infotainment umfasst neben Info-Systemen, beispielsweise Rückfahrkameras oder Sensoren-Rückmeldungen, die Entertainment-Sparte. Die Anforderungen der Endanwender steigen stetig und Streaming von Filmen in 1080p- oder 4k-Auflösungen erfordern hohe gestreamte Datenmengen. Diese übersteigen die Kapazität eines CAN-Busses um einige Größenordnungen.

Eine Lösung für dieses Problem bietet die weit verbreitete Ethernet-Technologie. Ein CAN-Bus kann nur eine Bandbreite von 1Mbit/s aufweisen. Über ein Ethernet-Netzwerk können je nach Spezifikation Daten mit einer Geschwindigkeit von 10Mbit/s, 100Mbit/s, 1Gbit/s, 10Gbit/s, 40Gbit/s oder 100Gbit/s gesendet werden. Die Ethernet-Technologie ist keine Neuheit, erste Ideen und Aufzeichnungen datieren bis in die 1970er Jahre zurück. Allerdings waren die Feldbusse bis vor einigen Jahren eine Komfort-Zone, aus der sich die großen Automobil-Hersteller nur ungern entfernen, weil bisher noch alles mit dieser inzwischen veralteten Technologie funktioniert. Vorreiter für einen Technologie-Wandel ist in diesem Fall die Branche, die neue Technologien mit deutlich geringerer Latenz in neue Modelle einbringen kann: Die Luftfahrt. Mit dem AFDX-Protokoll [2], das auf Ethernet aufsetzt, ist Ethernet als Backbone der Kommunikation in Flugzeugen bereits zahlreich vertreten. Eine entscheidende Änderung zum Standard-Ethernet ist die „Echtzeit“-Ausrichtung von AFDX, die für kritische Daten schnellere Sendungen ermöglicht.

Die fehlende Echtzeitfähigkeit von Standard-Ethernet ist auch in Fahrzeugen ein Problem. Einige sicherheitsrelevante Daten müssen schnell ihr Ziel erreichen und können nicht darauf warten, dass andere Pakete mit ihrer Sendung fertig werden. Um die Möglichkeit einer schnellen Sendung zu erhalten, muss eine Erweiterung des Ethernet-Standards verwendet werden. „Audio/Video Bridging“ (AVB) geht einen Schritt in diese Richtung und priorisiert seine Paketsendungen. Auf AVB aufbauend ist aktuell der Nachfolger „Time Sensitive Networking“ (TSN) in Entwicklung, der zusätzlich zu Prioritäten einen Mechanismus einbaut, der eine noch schnellere Sendung von ausgewählten Paketen garantiert. Für die beiden Protokolle AVB und TSN werden im Rahmen dieser Arbeit Worst-Case-Latenz-Szenarien ihrer Pakete entwickelt.

1.2 Motivation

Innerhalb der CoRE-Projektgruppe [3] an der HAW Hamburg wird die Kommunikation in Automobil-Netzwerken untersucht. Die Untersuchungen beziehen sich vorallem auf die Echtzeitkommunikation. Neben einem reinen Echtzeit-Ansatz – TTEthernet, vermarktet von der TTTech Computertechnik AG [4] – wird in dieser Gruppe das AVB-Protokoll und sein Nachfolge-Protokoll TSN untersucht. Die Gruppe arbeitet mit Hardware-Komponenten, die diesen Protokollen nachempfunden sind und kleine Netzwerke aufbauen können.

Vor allem beschäftigt sich die Projektgruppe mit der Simulationsumgebung OMNeT++ [5] und dem dafür existierenden Simulationsmodell INET [6] für Ethernet-Netzwerke. Darauf aufbauend wird das Simulationsmodell CoRE4INET [7] entwickelt, das unter anderem die Möglichkeit bietet, TTEthernet und AVB zu simulieren. Diese beiden Protokolle können in Kombination verwendet werden, um ein TSN-ähnliches Protokoll zu erhalten.

Der dritte Baustein, die theoretische Analyse der Protokolle, ist der Fokus dieser Arbeit. Die in Hardware und Simulation erstellten Netzwerke sind nicht in der Lage, für jede Netzwerkkonfiguration eine Aussage über den Worst-Case bezüglich der Latenzen der versendeten Pakete zu geben. Ein formaler Analyse-Rahmen, der in dieser Arbeit für AVB- und TSN-Netzwerke aufgezeigt werden wird, soll diese „Unzulänglichkeit“ von Hardware und Simulation ausgleichen.

1.3 Ziele und Struktur der Arbeit

Am Ende dieser Arbeit sollen formale Analyse-Modelle für AVB und TSN stehen. Die formalen Modelle werden eine nach oben begrenzte Abschätzung für Latenzen von Paketen an einzelnen Stationen im Netzwerk geben. Zusätzlich soll ein Tool erstellt werden, das die Möglichkeit hat, bestehende Simulationsnetzwerke so anzupassen, dass sich diese dem Worst-Case aus den formalen Analyse-Modellen annähern. Das Tool und die formalen Analyse-Modell werden anhand einer Worst-Case-Modellierung entworfen und in einer Evaluierung in Relation zueinander gesetzt.

In Kapitel 2 werden Grundlagen geboten. Kapitel 3 zeigt verwandte Themen und Arbeiten. Kapitel 4 erläutert Worst-Case-Szenarien für AVB und TSN. Kapitel 5 stellt Analyse-Methoden vor, die in Kapitel 6 für das AVB- und TSN-Protokoll verwendet werden. In Kapitel 7 werden Konzept und Realisierung des Tools vorgestellt. Die Evaluierung von den Analyse-Modellen und des Tools findet in Kapitel 8 statt. Zum Abschluß werden in Kapitel 9 noch Fazit und Ausblick geboten.

2 Grundlagen

Zum Verständnis der betrachteten Technologien innerhalb dieser Arbeit bietet dieses Kapitel einen Überblick über die Funktionsweisen und Anwendungsgebiete der unterschiedlichen Netzwerk-Standards.

2.1 Ethernet

Als Ethernet wird die weit verbreitete Netzwerk-Technologie bezeichnet, die eine Reihe von Standards für Software und Hardware (Kabel, Switche, etc.) zur Verfügung stellt, die für LANs (Local Area Networks) verwendet werden. Sie hat sich gegen andere Netzwerk-Technologien wie Token Ring oder Apple Talk durchgesetzt (vgl. [8]) und wird heute umfangreich eingesetzt.

Ein Ethernet-Netzwerk ist aus Knoten aufgebaut, die über Zwischenstationen – Switches genannt – miteinander verbunden werden. Die Struktur der Kommunikation basiert auf Paketen. Diese Pakete folgen einem Standard-Aufbau, der sich neben den eigentlich transportierten Daten – dem Payload – zusätzlich aus dem sogenannten Overhead – den Informationen, die für den korrekten Transport des Pakets über alle Stationen des Ethernet-Netzwerkes benötigt werden – zusammensetzt. In Abb. 2.1 wird die Struktur eines Ethernet-Pakets gezeigt.

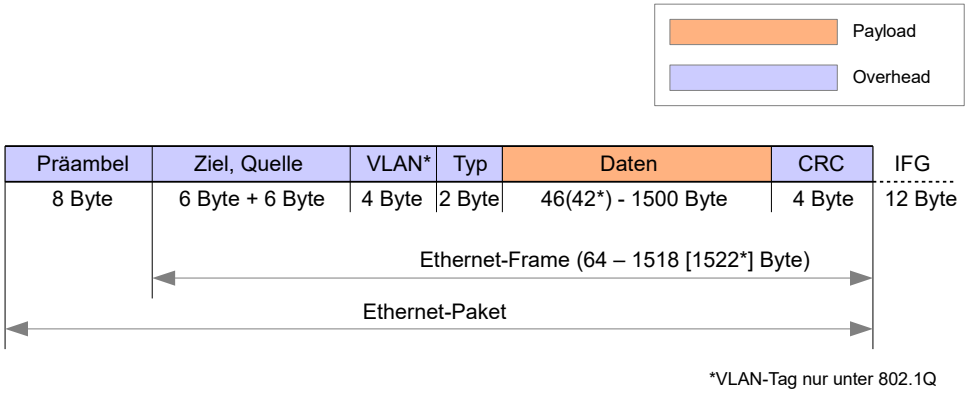

Abbildung 2.1: Standard-Ethernet-Paket-Aufbau (mit VLAN-Tag für 802.1Q-Ethernet)

Eine Interframe-Gap (IFG) wird zwischen zwei Paketen eingesetzt. Eine Übertragung von Daten über Ethernet setzt sich aus mehreren Paketen zusammen, die von einem Knoten (Quelle) über das Netzwerk zu einem anderen Knoten (Ziel) übertragen werden. Treten mehrere Pakete für die Versendung über eine Leitung in Konkurrenz, so entscheidet das Warteschlangen-Prinzip („FIFO“ – First In, First Out) über die Reihenfolge der Versendung. Dieses Verhalten ist explizit umgesetzt, indem Pakete vor der Versendung in eine Warteschlange (Queue) verschoben werden.

Oberhalb der Ethernet-Technologie kommen Protokolle zum Einsatz. Beispiele dafür sind Netzwerkprotokolle wie IPX/SPX oder TCP/IP. Letzteres kommt in heutigen Netzwerken in großer Anzahl zur Anwendung.

Aufbauend auf dem Standard-Ethernet gibt es weitere Protokolle, die auf bestimmte Anwendungsfälle zugeschnitten sind. Nach einem Blick auf das Übertragungsverfahren TDMA wird dieses Kapitel abschließend auf diese weiteren Ethernet-Protokolle eingehen.

2.2 TDMA

Echtzeit-Anwendungen setzen voraus, dass beispielsweise Kamerabilder in „Echtzeit“, das heißt ohne merkliche Verzögerung, dem Benutzer angezeigt werden. Eine Schwäche des Ethernet ist die mangelhafte Echtzeit-Fähigkeit. Die Dienstgüte, engl. *Quality of Service (QoS)*, für die Übertragung sicherheitsrelevanter Pakete kann je nach Anwendungsfall eine Ende-zu-Ende-Latenz (Zeitspanne von Bereitstellung des Pakets im Quelle-Knoten bis zum Ankommen am Ziel-Knoten) von $< 1ms$ erfordern. Werden im Kommunikationsnetz eines Fahrzeuges beispielsweise Daten für eine automatische Notfallbremsung verschickt, darf keine hohe Verzögerung auftreten.

Über eine gewisse Anzahl von Switches ist es nicht möglich, eine Garantie für so niedrige Latenzen geben zu können, da andere Pakete die Leitung belegen könnten, auf der das sicherheitsrelevante Paket gesendet werden soll. Unter Ethernet ist das Prinzip der Präemption, die Unterbrechung einer Paketsendung zugunsten einer anderen Paketsendung, nicht möglich. Begonnene Paketübertragungen werden immer komplett ausgeführt. Da eine Paketübertragung nicht unterbrochen werden kann, muss das sicherheitsrelevante Paket warten, bis die Leitung wieder frei ist.

Für die Übertragung kann eine Steuerung der Paket-Sendezeiten vorgenommen werden, um dieser potentiellen Blockade entgegenzuwirken. Dazu wird die Zeit, in der Pakete auf einer Leitung übertragen können, in Zeitabschnitte eingeteilt. Einige dieser Zeitabschnitte werden dann

für die Übertragung spezieller Pakete reserviert. Eine solche Aufteilung wird beispielhaft in Abb. 2.2 veranschaulicht. Das Verfahren, das eine solche Unterteilung von Sende-Zeitabschnitten vornimmt, wird **T**ime **D**ivision **M**ultiple **A**ccess (TDMA) genannt.

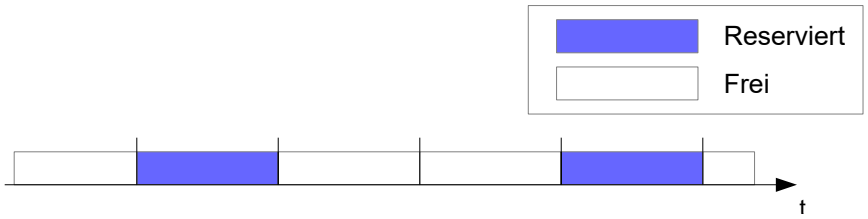

Abbildung 2.2: Reservierte Zeitblöcke in einer Übertragung

Zusätzlich zu der Aufteilung der Sendeabschnitte müssen für diese Art der Paket-Versendung unterschiedliche Pakete verwendet werden. Bei einer Aufteilung in Paketklassen dürfen zu einem Zeitabschnitt, der reserviert ist, nur Pakete einer TDMA-Klasse senden. Alle anderen Paketklassen müssen auf einen freien Abschnitt warten, um versendet zu werden.

Ein TDMA-Paket kann ohne Blockade durch andere Pakete versendet werden und weist gegenüber Paketen in einer Standard-Ethernet-Übertragung eine verringerte Latenz auf. Pakete einer Nicht-TDMA-Paketklasse können auf diese Weise eine verschlechterte Latenz gegenüber Standard-Ethernet aufweisen, da sie in reservierten Blöcken immer warten müssen, auch wenn keine andere Übertragung aktiv ist.

TDMA-Pakete können sicherheitsrelevante Nachrichten mit der angefordert niedrigen Latenz transportieren. Damit dieser Vorteil nicht in eine signifikante Verschlechterung der Latenz aller übrigen Paketklassen resultiert, sollten reservierte Zeitslots so klein wie möglich gehalten werden. Die beiden Ethernet-Protokolle TTEthernet und TSN (Abschnitt 2.5 und 2.6) operieren in einem TDMA-Verfahren und benutzen sich ähnelnde Verfahren zum Erreichen von Echtzeitfähigkeit in einem modifizierten Ethernet-Netzwerk.

2.3 802.1Q

Aufsetzend auf Standard-Ethernet wird mit dem 802.1Q-Protokoll [9] die Paket-Struktur des Standard-Ethernet-Pakets erweitert. Diese Änderung zeigt sich in dem in Abb. 2.1 gezeigten VLAN-Tag. Dieses sorgt dafür, dass ein Paket in verschiedene (Prioritäts-)Klassen und VLANs eingeordnet werden kann. Ein VLAN (Virtual LAN) macht ein Teilnetzwerk innerhalb des

gesamten Netzwerks aus. Zur Anwendung kommen die Paketklassen, wenn ein Paket von einem Knoten oder Switch auf eine Leitung gelegt werden soll. Die übliche Queue, die zur Versendung von Paketen mit diesen befüllt wird, wird für 802.1Q-Ethernet in mehrere Queues aufgeteilt. Jede Paketklasse erhält eine eigene Queue. Die Aufteilung der Queues wird in Abb. 2.3 beispielhaft veranschaulicht.

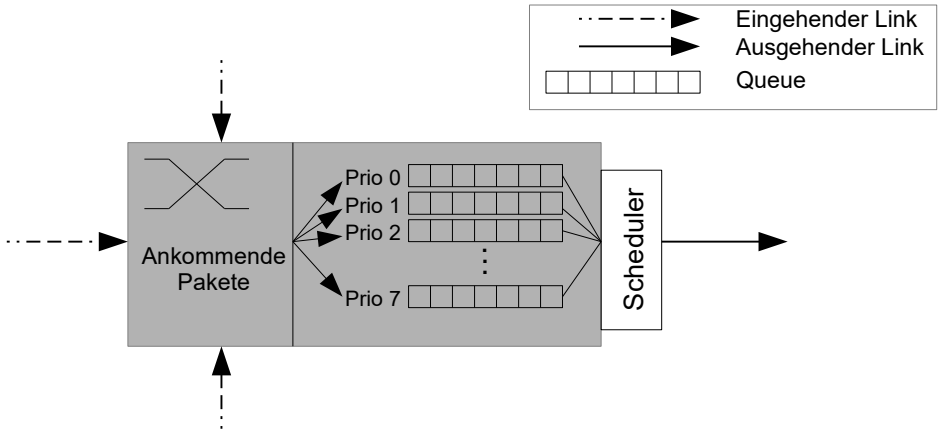

Abbildung 2.3: Aufteilung der Warteschlangen in 802.1Q-Ethernet

Das VLAN-Tag ist in TPID und TCI (zu je 16 Bit) aufgeteilt. Das TCI-Feld ermöglicht acht verschiedene Paketklassen bzw. Prioritäten für Pakete. Sind mehrere Pakete aus verschiedenen Queues sendebereit, entscheidet der Scheduler anhand der Paketklasse, welches Paket als nächstes versendet wird. Konkurrierende Pakete der gleichen Paketklasse werden weiterhin mit dem FIFO-Prinzip behandelt.

Durch die unterschiedlichen Prioritäten können niedrigpriore Pakete verzögert werden. Tritt ein kontinuierlicher Strom von Paketen einer hohen Prioritätsstufe auf, kann eine Queue niedrigerer Priorität vollständig blockiert werden. Pakete dieser niedrigen Prioritätsstufe würden somit nie ihr Ziel erreichen. Dieser Nachteil steht dem Vorteil gegenüber, dass Pakete nur auf Pakete gleicher oder höherer Prioritätsstufe warten müssen. Zusätzlich können diese Pakete von einem Paket einer niedrigeren Priorität blockiert werden, wenn dieses bereits mit dem Senden begonnen hat und daher nicht unterbrochen werden kann.

Dieser Nachteil kann durch eine TDMA-Struktur der Paket-Versendung gemildert werden, was in Abschnitt 2.6 für TSN veranschaulicht wird. TSN ist ein Protokoll, das aus AVB weiterentwickelt wird. AVB setzt auf 802.1Q-Ethernet auf und wird nachfolgend erklärt.

2.4 AVB

Audio/**V**ideo **B**ridging ist ein Protokoll, das auf 802.1Q-Ethernet basiert. Es setzt sich aus vier Standards zusammen [10–13], die alle Teilbereiche des Protokolls definieren.

„Ursprünglich wurde AVB entwickelt, um die Übertragung von Audio- und Video-Signalen dynamisch über Ethernet laufen zu lassen. Da moderne Fahrzeuge audiovisuelle Daten für ihre Infotainment-Segmente übertragen müssen, fand AVB auch im Automotive-Bereich Beachtung.“ [14]

Bereiche außerhalb des Infotainment-Segments sind ebenfalls für eine medienorientierte Kommunikations-Infrastruktur prädestiniert. Eine Rückfahrkamera beispielsweise muss in Echtzeit ihre Daten an den Nutzer weitergeben, um Sicherheitsrisiken zu minimieren.

Das Protokoll ordnet die Prioritätsklassen eigens definierten Paketklassen zu. Die beiden höchsten Prioritäten werden den Klassen „AVB-Class A“ (nachfolgend als Klasse A bezeichnet) und „AVB-Class B“ (nachfolgend als Klasse B bezeichnet) zugeordnet. Eine dritte Paketklasse kann die verbleibenden Prioritäten verwenden und wird als „Best Effort“ (BE) bezeichnet.

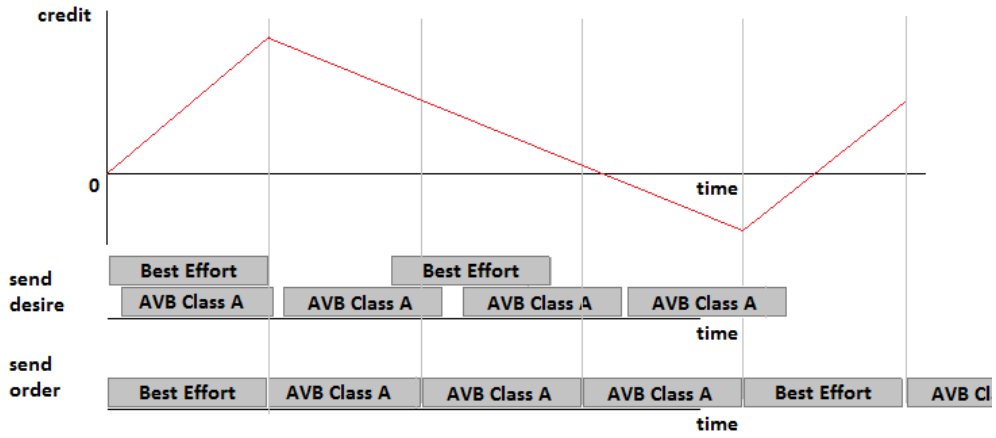

Abbildung 2.4: Kredit und Paketversendung unter AVB [15]

Eine Änderung des Protokolls gegenüber dem 802.1Q-Standard ist die Erweiterung des Schedulers um ein Kredit-System, den sogenannten Credit Based Shaper (CBS). Die beiden Klassen A und B haben zwar die höchsten Prioritäten an den Switches, dürfen aber nur senden, wenn ihr klassenspezifischer Kredit mindestens den Wert 0 aufweist. Der Kredit einer

Klasse ist grundsätzlich 0, bis ein Paket dieser Klasse ein Paket versenden möchte. Mit diesem Mechanismus soll sichergestellt werden, dass Pakete niedrigerer Priorität senden können, was ein Vorteil gegenüber dem 802.1Q-Standard ist.

Ein beispielhafter Sendeablauf aus Klasse-A-Paketen und BE-Paketen wird in Abb. 2.4 veranschaulicht. Da das BE-Paket mit seiner Sendung beginnt, bevor das AVB-Paket sendebereit ist, kann das BE-Paket nicht unterbrochen werden. Da nun ein AVB-Paket der Klasse A senden möchte, steigt der Kredit dieser Klasse, während andere Paketklassen senden. Hat das BE-Paket seine Sendung abgeschlossen, ist der Kredit der Klasse A mindestens 0, daher darf das Klasse-A-Paket senden. Mit der Sendung eines Pakets dieser Klasse fällt der Kredit wieder.

Das nächste BE-Paket darf nicht sofort senden und muss auf ein AVB-Paket warten, das nach ihm angekommen ist, da der Kredit weiterhin mindestens 0 ist. Fällt der Kredit unter 0, darf kein Paket der Klasse A mehr senden.

Die Raten des steigenden und fallenden Kredits hängen direkt voneinander ab und werden durch die genutzte Bandbreite der Paket-Klassen festgelegt. Die genutzte Bandbreite wiederum ist abhängig von der reservierten Bandbreite der Paketklassen. Mithilfe des Stream Reservation Protocols (SRP) wird ein Mechanismus zur Reservierung von Bandbreite integriert. Soll eine gewisse Datenmenge mit einem AVB-Stream (eine Sequenz aus mehreren AVB-Paketen) übertragen werden, kann für die Netzwerkroute des Streams die Bandbreite reserviert werden, die dafür benötigt wird. Ist die Bandbreite an jeder Station des Streams verfügbar, ist die Reservierung erfolgreich. Die Standards geben die Empfehlung, eine Reservierung von maximal 75% der gesamten Bandbreite des verwendeten Netzwerk-Links für Klasse A und B zu ermöglichen [10]. Das sorgt dafür, dass mindestens 25% der Bandbreite für Pakete niedrigerer Prioritäten verwendet werden kann. Das konkrete Reservierungsverhältnis wird in den Krediten der Klassen umgesetzt.

Die Perioden, die zugleich den sogenannten „Measurement Interval“ ausmachen, der beiden Klassen A und B liegen bei $125\mu s$ bzw. $250\mu s$. Innerhalb dieser Perioden muss eine Versendung aller Pakete einer Klasse möglich sein, ohne den reservierten Prozentsatz zu überschreiten, wenn außer diesen Paketen keine anderen Paketklassen übertragen werden. Übertragen auch Pakete anderer Paketklassen, kann die Übertragung aller AVB-Pakete einen längeren Zeitraum als den Measurement Interval in Anspruch nehmen. Wartet ein AVB-Paket auf die Versendung, steigt der Kredit der zugehörigen Klasse mit der Rate

$$idleSlope_{tcl} = R,$$

mit R für die gesamte reservierte Bandbreite der Paketklasse tcl. Ist eine Übertragung eines Pakets der Paketklasse tcl aktiv, so verringert sich der Kredit der Klasse mit der Rate

$$sendSlope_{tcl} = idleSlope_{tcl} - C,$$

mit C für die gesamte Bandbreite des Netzwerk-Links. Für einen Netzwerk-Link der Bandbreite 100Mbps ergeben sich bei einer Reservierung von 60% der Bandbreite durch Klasse-A-Pakete also die Werte $idleSlope_A = 60\text{Mbps}$ und $sendSlope_A = -40\text{Mbps}$.

Auf das AVB-Protokoll wurde in den Untersuchungen der vorigen Arbeiten [14, 15] der Haupt-Fokus gelegt. Auch diese Arbeit wird in den Worst-Case-Analysen weiterhin die Latenzen der AVB-(ähnlichen-)Paketklassen untersuchen, dabei aber die weiterentwickelte Version des Protokolls (TSN) in die Modellierung des Worst-Case miteinbeziehen. Die Standards von AVB garantieren für die Übertragung eines Pakets über 7 Hops (7 Switches) eine Latenz von maximal 2ms. Die Gültigkeit dieser Garantie kann aufgrund von Ergebnissen der Arbeit [15] angezweifelt werden. Zusätzliche Pakete des TSN-Protokolls werden diese Latenz weiter verschlechtern.

2.5 TTEthernet

TTEthernet steht für **T**ime **T**riggered **E**thernet und ist ein Ethernet-Protokoll, das von der Firma TTTech Computertechnik AG [4] vermarktet und vertrieben wird. Das Protokoll basiert auf dem TDMA-Ansatz und ermöglicht die Übertragung von Paketen mit vergleichsweise niedrigen Latenzen. Eine Einhaltung von Ende-zu-Ende-Latenzen von unter 1ms ist mit dem Protokoll auch über eine Vielzahl von Switches möglich.

TTE ist ein Protokoll, das in der Projektgruppe CoRE [3] neben der Realisierung in Hardware für die Erstellung eines TDMA-basierten Ethernet-Protokolls in einem Simulations-Framework für die Simulationsumgebung OMNeT++ [5] verwendet wird. Zusammen mit einem in der Projektgruppe entwickelten Simulations-Modell für das AVB-Protokoll ergibt sich die Möglichkeit, beide Protokolle zu verbinden [7], um die Grundlage für ein Simulationsmodell zu schaffen, das starke Ähnlichkeiten zu TSN hat. Um Analyse-Ergebnisse, die das TSN-Protokoll betreffen, zu evaluieren, bietet sich dieses Simulationsmodell an. Das TSN-Protokoll wird nachfolgend erklärt.

2.6 TSN

Time **S**ensitive **N**etworking ist die Weiterentwicklung des AVB-Protokolls. Die IEEE 802.1 TSN Task Group [16] arbeitet an der (Weiter-)Entwicklung der Standards 802.1AS, 802.1Qbu, 802.1Qbv, 802.1Qcc, 802.1CB, 802.1Qch und 802.1Qci, die neben Erweiterungen des 802.1Q-Standards auch verschiedene neue Shaper-Varianten für das Protokoll vorsehen. Das Ziel der Weiterentwicklung ist vorallem, eine schnellere Übertragung einzelner Pakete zu ermöglichen.

2.6.1 Funktionsweise

TSN setzt auf AVB auf. Das bedeutet, dass CBS und SRP weiterhin verfügbar sind. Die Erweiterungen beziehen sich auf die Verwendung neuer Shaper. Drei verschiedene Varianten werden momentan diskutiert (CQF, Burst Limiting und Time Aware) und werden nachfolgend erklärt.

Zusätzlich zu der Ergänzung um neue Shaper wird innerhalb der TSN Task Group in Zusammenarbeit mit der IEE 802.3 Working Group [17] an der Erweiterung des Standard-Ethernets um Möglichkeiten der Frame Preemption gearbeitet. Diese Funktionsweise wird im Standard 802.1Qbu [18] festgehalten, der sich momentan — wie die anderen TSN-Standards — noch in der Draft-Phase befindet.

2.6.2 Shaper

Dieser Abschnitt wird die drei Alternativen für die neuen Shaper in TSN beleuchten. Für die Beschreibung der Shaper wird teilweise die Beschreibung aus der Arbeit [19] verwendet.

Burst Limiting Shaper

Der Burst Limiting Shaper (BLS) ist in seiner Funktionsweise an den CBS angelehnt. Wie der Name erahnen lässt, soll der BLS das Auftreten von Bursts in definierten Grenzen halten. Bursts sind dabei mehrere Pakete einer bestimmten Klasse (Control-Data Traffic, CDT), die durch ihr geballtes Auftreten (=Burst) dafür sorgen, dass ein anderes Paket dieser Klasse nicht mehr senden kann. Die Sendevoraussetzungen für ein CDT-Paket sind:

- Der Kredit dieser Queue, also der CDT-Paketklasse, ist unter max_level
- Die Priorität dieser Queue ist höher als die aller anderen Queues

Wird ein CDT-Frame übertragen, steigt der Kredit mit der Rate $sendSlope$. Wartet ein Frame der CDT-Paketklasse, fällt sein Kredit mit der Rate $idleSlope$. Der Unterschied zum ursprünglichen

CBS von AVB liegt in der Umkehrung der Kreditfunktionen. Unter AVB war der $idleSlope$ eine steigende Funktion und der $sendSlope$ eine fallende.

Die CDT-Paketklasse fungiert grundsätzlich wie unter AVB die Paketklasse A. CDT-Frames werden mit der höchsten Priorität versendet. Das Konzept der „dynamic queue priority“ sieht jedoch eine Änderung der Prioritäten der Queues vor. Erreicht der Kredit der CDT-Klasse den Wert max_level, wird die Priorität der CDT-Klasse auf die niedrigste Priorität gesetzt, die möglich ist. Haben die CDT-Pakete lange genug pausiert, ist ihr Kredit so weit gefallen, dass er den vordefinierten Wert $resume\_level$ erreicht. Zu diesem Zeitpunkt wird die Priorität der CDT-Klasse auf den ursprünglichen (höchsten) Wert gesetzt. Ähnlich zu AVB-Klasse A sorgt dieses Vorgehen dafür, dass Pakete der CDT-Klasse nicht die Möglichkeit haben, niedrigere Prioritäten unendlich zu verzögern.

Cyclic Queuing and Forwarding

Ehemals als Peristaltic Shaper benannt, ist der Cyclic Queuing and Forwarding (CQF) Shaper nach wie vor in einer frühen Entwicklungsphase begriffen. Der CQF Shaper ist der Ansatz, der laut Aussage von Thangamuthu et al. [19] am einfachsten zu konfigurieren ist. Peristaltic, zu Deutsch „Peristaltisch“ stand dabei für eine Muskelbewegung einiger Hohlorgane, wie beispielsweise der Speiseröhe. Die Muskelbewegung sorgt dafür, dass Objekte durch die Organe geleitet werden.

„*Cyclic queuing and forwarding (CQF) is a method of traffic shaping that can deliver deterministic, and easily calculated, distributed delays for time sensitive traffic streams. In essence, the technique involves the use of two transmission queues and a cycle timer.*“ [20]

Die Aufteilung in zwei Queues geschieht mit einer Aufteilung der Sendezeitabschnitte in sich abwechselnde Abschnitte mit den Namen *odd* und *even*. Während einer Phase nimmt eine der beiden Queues ankommende Frames an, sendet sie aber nicht weiter. Gleichzeitig sendet die andere Queue, ohne neue Frames anzunehmen. Der Übergang in die jeweils andere Phase sorgt für einen Tausch der Rollen.

Die Motivation des Shapers ist eine Begrenzung der Latenzen unter bestimmten Voraussetzungen. „*The attraction of CQF as a technique for handling time-sensitive traffic is that, if the cycle time is set appropriately, the latency introduced by any one hop in the network is completely bounded by the cycle time*[...]“ [20]

Ein Cycle hat die Länge $2T$, wobei T jeweils für die Länge der beiden Phasen steht. Diese Eigenschaft des Shapers sorgt dafür, dass die Worst-Case-Ende-zu-Ende-Latenz eines Pakets aus der Summe der einzelnen Hop-Delays errechnet werden kann. Das soll auch gegen den Einfluss störenden Cross-Traffics gültig sein, so lange alle Frames in den vorgesehenen Cycles an den Switches eintreffen. Eine Synchronisierung und Ausrichtung der Cycles ist nötig, damit kein Paket seinen Cycle verpasst und so dafür sorgen würde, dass die oben genannte Aussage über Ende-zu-Ende-Latenzen nicht mehr zuträfe.

Unter CQF ist, wie unter AVB, für jede Klasse ein Measurement Interval vorgesehen. Der Trick, dass die schlechteste Ende-zu-Ende-Latenz ohne Berücksichtigung von Cross-Traffic festgelegt werden kann, liegt in der Wahl der Größe von T. Der Zeitraum T sollte groß genug gewählt werden, um alle ankommenden Pakete eines Measurement Intervals unterzubringen und zusätzlich noch Platz für mindestens ein maximales BE-Paket bieten. Es gilt, dass T ein Vielfaches des Measurement Intervals, oder, im Falle von verschiedenen Paketklassen, ein Vielfaches des kleinsten gemeinsamen Vielfachens der Measurement Intervals beider Klassen sein sollte.

Eine Verbesserung des Zeitverhaltens gegenüber des AVB-Protokolls ist für diesen Ansatz des neuen Shapers nicht zu erkennen, daher ist die Tauglichkeit des Shapers für zeitkritische Versendungen anzuzweifeln.

Time Aware Shaper

Der letzte Shaper, der für TSN vorgesehen ist, ist der Time Aware Shaper (TAS). Dieser kommt dem oben vorgestellten TDMA-Verfahren nahe. Der TAS sieht eine Paketklasse vor, die in der Priorität über den alten AVB-Klassen A und B steht. Neben der hohen Priorität wird sichergestellt, dass dieser Klasse zugehörige Pakete nicht durch andere Pakete blockiert werden können. Die TDMA-typische Einteilung der Sendezeiten wird unter TAS mithilfe von Gates vorgenommen. In Abb. 2.5 wird gezeigt, wie dieses Konzept auf den bisherigen CBS von AVB aufgesetzt wird.

Ähnlich wie bei TTEthernet muss im Voraus festgelegt werden, wann die TAS-Pakete versendet werden sollen. Um die Versendung aller anderen Pakete durch TAS-Pakete nicht unnötig zu verzögern, ist ein möglichst kleines Zeitfenster für deren Versendung vorgesehen. Abb. 2.5 zeigt die Gates für Scheduled Traffic (TAS) und Other Traffic (alle anderen Paketklassen). Diese können offen (grün) oder geschlossen (rot) sein. Sind die Gates offen, kann eine Sendung

begonnen werden. Es gibt nun die Möglichkeit, die Gates für TAS-Pakete so zu konfigurieren, wie Abb. 2.5 es darstellt. Die Gates der anderen Klasse werden bereits geschlossen, bevor das TAS-Gate geöffnet wird. Somit wird eine Überlappung der Sendungen verhindert. Ein gewisser Zeitraum vor der Versendung von TAS-Paketen wird auf diese Weise immer geblockt.

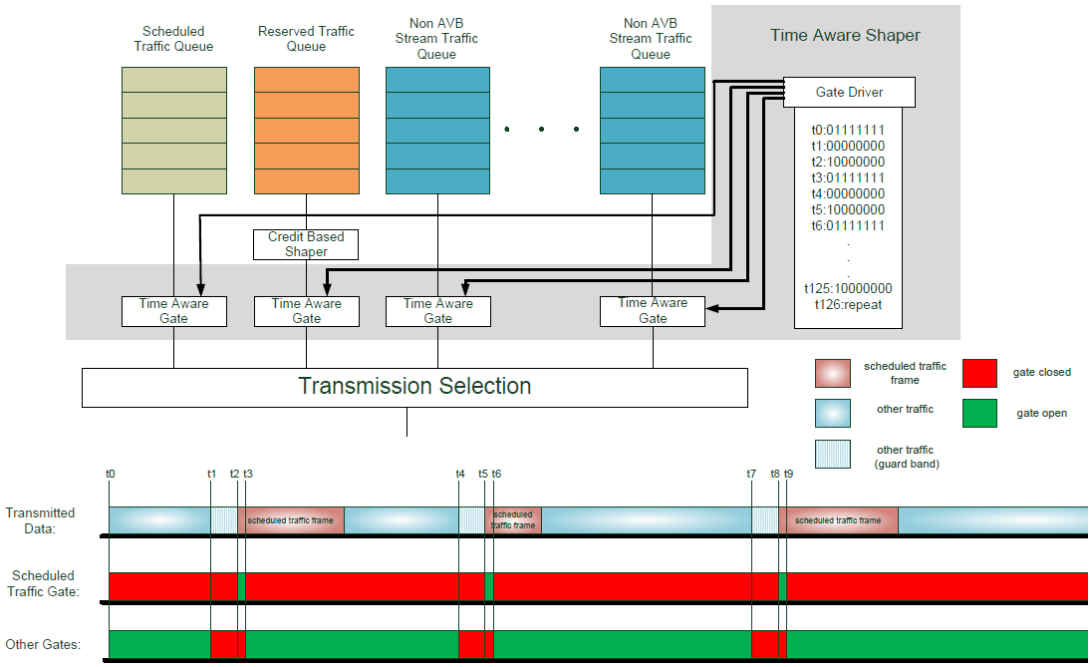

Abbildung 2.5: Time Aware Shaper Konzept [21]

Der TAS ist der einzige Shaper, der relativ niedrige Ende-zu-Ende-Latenzen ($< 1ms$) über mehrere Hops erreichen kann und wird in dieser Masterarbeit für alle Untersuchungen bezüglich der Latenz in TSN-Netzwerken verwendet. Der Hauptgegenstand der Untersuchung wird die Latenzen der unteren Paketklassen betreffen, da für die TAS-Pakete unter der Voraussetzung eines synchronisierten Netzwerkes eine konstante Latenz angenommen werden kann.

3 Related Work

Dieser Abschnitt fasst zusammen, worauf diese Arbeit basiert und welche verwandten Themen im aktuellen wissenschaftlichen Fokus stehen. Unterteilt wird nachfolgend in Arbeiten, die für die Vorarbeit dieser Masterarbeit entscheidend waren, und Arbeiten, die den momentanen Stand der Wissenschaft auf relevanten Gebieten der Worst-Case-Timing-Analysen von AVB und/oder TSN darlegen und als Wissensbasis für diese Masterarbeit verwendet werden können.

3.1 Basis dieser Arbeit

Hinführend zum Thema dieser Masterarbeit wurde das Verfahren Network Calculus untersucht. Dieser kann Aussagen über die nötige Größe von Queues und maximalen Verzögerungen innerhalb von definierten Systemen (beispielsweise eines Switches) treffen. So können die Verzögerungen von Paketen an Ausgangs-Queues von Switches abgeschätzt werden. Ein Grundlagenkapitel zur Erklärung des Network Calculus wird in Abschnitt 5.2 geboten. Die grundlegende Funktionsweise des Network Calculus stammt aus dem Buch „Network calculus: a theory of deterministic queuing systems for the internet“ [22] von Jean-Yves Le Boudec und Patrick Thiran. Aufbauend auf dessen Algebra zur Abschätzung der Worst-Case-Eigenschaften eines Systems wird von Georges et al. in „Strict Priority versus Weighted Fair Queueing in Switched Ethernet Networks for Time Critical Applications“ [23] eine Erweiterung des allgemeinen Network Calculus auf Ethernet-Netzwerke entworfen. Zur Anwendung dieses Modells auf AVB-Netzwerke wird von Rene Queck in „Analysis of Ethernet AVB for automotive networks using Network Calculus“ [24] eine Parametrisierung für AVB-Netzwerke entworfen. Aufbauend auf diesem Modell wird in der vorangegangenen Arbeit [14] eine Untersuchung und Evaluierung des AVB-Network-Calculus-Modells vorgenommen, die zu einer Korrektur und erneuten Evaluierung des Modells in der darauf folgenden Arbeit [15] führte. Die Erkenntnisse dieser Vorarbeiten wird als Grundlage für die Forschung innerhalb dieser Masterarbeit dienen.

3.2 Stand der Wissenschaft

Die Analyse eines Netzwerks ist neben praktischen Verfahren, beispielsweise Simulationen, auf formale Methoden angewiesen, um die Verzögerungen und daraus resultierenden Latenzen einzelner Pakete und Streams abzuschätzen. Gegenüber praktischen Methoden sind sie in der Lage, die Latenzen auf eine obere Grenze abzuschätzen. Zusätzlich zu dem in Abschnitt 5.2 gezeigten Network-Calculus-Verfahren existieren weitere Verfahren, die in diesem Abschnitt nachfolgend kurz beschrieben werden.

3.2.1 Modular Performance Analysis

In „Timing analysis of Ethernet AVB-based automotive E/E architectures“ [25] von Reimann et al. wird das Verfahren Modular Performance Analysis (MPA), das auf dem Real Time Calculus (RTC) basiert, angewendet, um AVB-Netzwerke auf Worst- und Best-Case Szenarien zu untersuchen. Der RTC definiert die Ausgangs-Queue eines Switches oder Knotens als Ausgangspunkt eines definierten RTC-Systems. Der RTC weist im Grundsatz starke Ähnlichkeiten zu dem hier vorgestellten Network Calculus auf. Eine Modifizierung des Verfahrens erlaubt, Nachrichten auf ihre Latenz zu untersuchen, die länger als ein maximales Ethernet-Paket sind und folglich aus mehreren Ethernet-Paketen bestehen müssen. Das Verfahren berücksichtigt nicht, dass ein AVB-Netzwerk erhöhte Verzögerungen für AVB-Streams verursachen kann, indem blockierende Pakete doppelt auf den Ausgangs-Queues auftreten. Dieses doppelte Auftreten von AVB-Paketen wird nachfolgend in dieser Masterarbeit erklärt. In der Betrachtung von Nachrichten, die größer als ein maximal großes AVB-Paket sind, wird die Gesamtlatenz vom Start der Übertragung des ersten Pakets bis zum Abschluss der Sendung des letzten Pakets interessant, daher ist das doppelte Auftreten von blockierenden Paketen ein Problem, dass nur bei einem der Pakete der Nachricht auftreten kann. Da nachfolgende Pakete der Nachricht diese erste Verzögerung sogar „aufholen“ können, ist für diese Art der Latenz-Betrachtung ein Ignorieren der doppelten blockierenden AVB-Pakete legitim. Die Ergebnisse der Worst-Case-Abschätzungen in Arbeit [25] werden mit Berechnungen zur Ermittlung von Worst-Case-Latenzen aus dem AVB-Standard 802.1Qav [10] verglichen und zeigen, dass die Abschätzungen der Autoren optimistischer, aber auch realistischer, sind. Der Fokus in dieser Masterarbeit soll anschließend an vorherige Arbeiten weiterhin auf der Analyse einzelner Pakete eines AVB-Streams liegen. Das MPA-Verfahren bietet in der aktuellen Version keine Basis für diese Art der Analyse. Bei einer Erweiterung der, in dieser Masterarbeit vorgestellten, Analyse-Verfahren auf Nachrichten aus mehreren Ethernet-Paketen könnte dieses Verfahren aber als Vergleichsverfahren verwendet werden.

3.2.2 Formale Analyse per busy period

Bordoloi et al. untersuchen in „Schedulability Analysis of Ethernet AVB Switches“ [26], ob Pakete in AVB-Netzwerken versendet werden können, ohne eine vorgegebene Deadline zu verletzen. Dazu wird eine Worst-Case-Response-Time(WCRT)-Analyse für diese Pakete durchgeführt. Die Annahmen des Verfahrens sind ähnlich zur Network-Calculus-Abschätzung. Pakete höherer, niedrigerer und gleicher Priorität werden für die WCRT-Analyse berücksichtigt. Die „busy period“ beschreibt den Zeitraum, der für die Analyse des Ethernet-Pakets betrachtet wird. Pakete außerhalb dieser busy period werden nicht miteinbezogen. In [26] fehlt ebenfalls die Erweiterung der Annahmen auf doppelte AVB-Pakete. Diese spielt bei Analysen von AVB-Netzwerken eine wichtige Rolle und wird in Kapitel 4 genauer erklärt. Die resultierende WCRT-Analyse weist damit einen Fehler auf, der für eine Verwendung innerhalb dieser Masterarbeit korrigiert werden müsste. In angepasster Variante könnte dieses Verfahren als Vergleichsverfahren für verwendete Analyse-Methoden innerhalb dieser Masterarbeit verwendet werden.

3.2.3 Shaper-Analyse

Einen Überblick über die Entwicklung des TSN-Protokolls wird von Thangamuthu et al. in „Analysis of Ethernet-switch traffic shapers for in-vehicle networking applications“ [19] gegeben. Dieser beinhaltet Beschreibungen zu den möglichen Traffic-Shapern des Protokolls und evaluiert die Einsetzbarkeit der verschiedenen Ansätze anhand von Worst-Case-Analysen. Abgesehen von dem erhöhten Konfigurationsaufwand mit TDMA-Paketen ziehen die Autoren der Arbeit den Schluss, dass sich der TDMA-Ansatz (Time Aware Shaper — TAS) von TSN für die Latenz-Reduzierung am besten eignet. Dieser Schluss hilft, den Fokus dieser Masterarbeit auf den TAS-Ansatz von TSN zu legen.

3.2.4 Compositional Performance Analysis

Einen weiteren Ansatz zur theoretischen Analyse wird von Thiele et al. in „Formal worst-case timing analysis of Ethernet TSN’s time-aware and peristaltic shapers“ [27] geboten. Mit der Methode „Compositional Performance Analysis“ (CPA) wird ein Verfahren vorgestellt und aus vorherigen Arbeiten weiterentwickelt, das ebenfalls eine Abschätzung der Worst-Case-Verzögerung von Paketen an Ausgangs-Queues von Switches ermöglicht. Das CPA-Verfahren transformiert die Eigenschaften eines Netzwerkes und seine Konfiguration in eine Form, die dem Prozess-Scheduling innerhalb von CPUs ähnelt. Für die Abschätzung des Worst-Case der Latenzen werden die transformierten Systeme in Netzwerk-Darstellungen zurücktransformiert.

Die Analyse verfolgt einen iterativen Ansatz. In einem Netzwerk, das als gerichteter Graph modelliert wird, werden Pakete, Übertragungszeiten und Störungen durch andere Pakete konfiguriert und daraus Verzögerungen für ein untersuchtes Paket abgeleitet. Die Ergebnisse dieser Modellierung werden in ein neues Modell überführt, dass die störenden Einflüsse so verändert, dass eine größere Verzögerung für das untersuchte Paket eintritt. Dieser „globale Analyse-Loop“ (Iteration) wird so lange wiederholt, bis keine Änderung der Modelle mehr auftritt.

Für diese Masterarbeit ist die Erkenntnis der Autoren bezüglich der Shaper-Wahl in TSN hilfreich. Für eine Reduzierung der Latenzen gegenüber dem AVB-Protokoll ist die Verwendung des TAS sinnvoll. Diese Aussage bekräftigt die Motivation in dieser Masterarbeit, das TSN-Protokoll mit Fokus auf den TAS zu betrachten, um die Echtzeittauglichkeit des Protokolls zu ergründen.

3.2.5 Warteschlangentheorie

In der Abbildung von verschiedenen Prozessen eignen sich Zustandsräume, die aus Markov-Ketten aufgebaut werden. Auf diesen Markov-Ketten aufbauend existiert die Warteschlangentheorie, die ursprünglich zur Erforschung von Telekommunikationsdiensten entwickelt worden ist. Detaillierte Informationen zu der Vorgeschichte und Anwendung dieser Analyse finden sich im Werk „Grundlagen der Warteschlangentheorie“ von Dieter Baum [28]. Einen speziellen Zugang zu der Anwendung der Warteschlangentheorie unter anderem auf Computernetzwerke wird von Klaus-Dieter Thies in „Elementare Einführung in die Wahrscheinlichkeitsrechnung, Informationstheorie und stochastische Prozesse mit Warteschlangentheorie für Computernetzwerke - Mit einer wahrscheinlichkeitstheoretischen Leistungsanalyse des Ethernet“ [29] gegeben. Die Warteschlangentheorie eignet sich besonders für moderne Computernetzwerke, beispielsweise Ethernet, da diese bereits in Warteschlangen modelliert sind. Daher lässt sich mit überschaubarem Aufwand eine Aussage über die Latenzen eines Pakets, basierend auf seiner Aufenthaltsdauer in einer Warteschlange, treffen. Für Standard-Ethernet ist die Anwendung der Warteschlangentheorie ein probates Mittel, um einheitliche Paketklassen zu analysieren.

Für 802.1Q-Ethernet und seine Ableger — dazu gehören AVB und TSN — ist die Warteschlangentheorie in modifizierter Weise ein Kandidat für die Analyse der Paket-Latenzen. Aufgrund der Aufteilung der Warteschlangen auf verschiedene Prioritätsklassen ist eine solche Modifizierung mit einer Neugestaltung der zugrundeliegenden Markov-Prozesse verbunden und damit ein erheblicher Aufwand, der den Rahmen dieser Masterarbeit leicht überschreiten kann. Das

wissenschaftliche Gebiet der Warteschlangentheorie, vorallem in Bezug auf 802.1Q-Ethernet, ist ein aktuell unangetastetes Forschungsgebiet, was anhand des potentiell großen Aufwands einer Neuentwicklung eines passenden Markov-Prozesses nachvollziehbar ist. Im weiteren Verlauf dieser Masterarbeit wird die Warteschlangentheorie nicht zum Einsatz kommen.

4 Worst-Case-Modellierung

Eine Berücksichtigung aller möglichen Einflüsse auf ein Paket ist notwendig, um den Worst-Case der Verzögerung des Pakets zu erhalten. In Kapitel 5 werden formale Analyse-Methoden vorgestellt, die in Kapitel 6 für die Anwendung auf AVB- und TSN-Netzwerke modelliert und parametrisiert werden. Die für Kapitel 6 zugrundeliegenden Annahmen bezüglich möglicher Paket-Sendeverläufe in Worst-Case-Szenarien werden in diesem Kapitel vorgestellt.

4.1 Vorgehen

In den folgenden Abschnitten wird auf mögliche Sendeablauf-Szenarien der zu versendenden Pakete in AVB- und TSN-Netzwerken eingegangen. Der konkrete Ablauf der Paketsendungen wird wie in Abb. 4.1 skizziert.

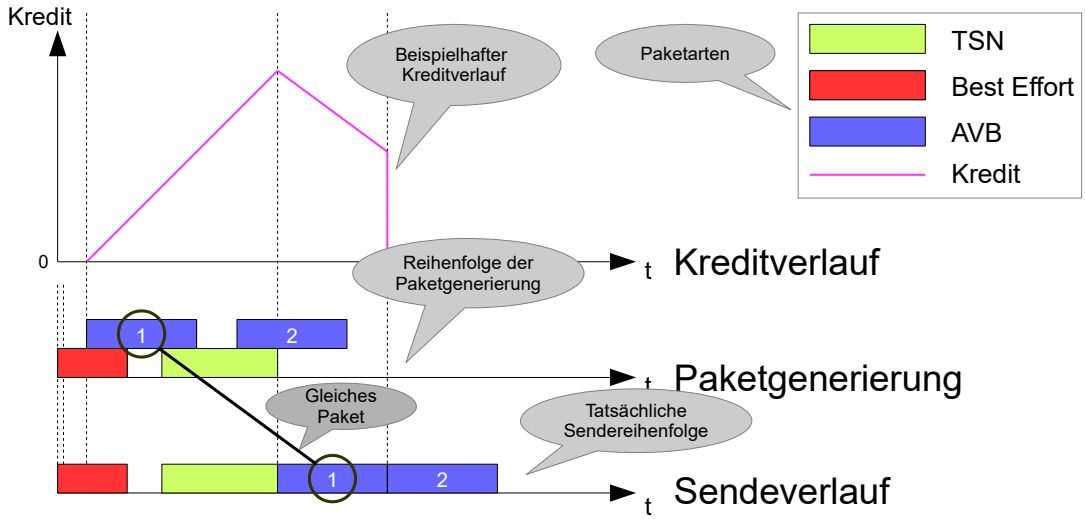

Abbildung 4.1: Beispiel eines Sendeablauf-Szenarios

Entscheidend für die Verzögerung des untersuchten Pakets ist der zeitliche Abstand von der Generierung des Pakets bis zu seiner tatsächlichen Versendung. Die Generierung eines

Pakets kann auf verschiedene Quellen zurückgeführt werden. Entweder wird das Paket zu dem betreffenden Zeitpunkt von dem Knoten, für den dieses Paket-Szenario entworfen wurde, generiert und ist für die Versendung bereit, oder die Generierung betrifft Switches. Im zweiten Fall umfasst sie, dass das Paket an dem Switch ankommt und weitergeleitet werden soll. Der Zeitpunkt der „Paketgenerierung“ ist als der Zeitpunkt zu verstehen, an dem das betreffende Paket vollständig im Switch oder Netzwerkknoten vorliegt und weitergeleitet werden kann.

Für AVB-Streams sind insbesondere die Ende-zu-Ende-Latenzen interessant. Diese ergeben sich aus der zeitlichen Differenz zwischen der Bereitstellung des Paketes durch die Anwendung beim Sender und dem vollständigen Empfang des Pakets durch die Anwendung beim Empfänger.

Für die Modellierungen in diesem Kapitel werden einzelne Stationen betrachtet. Ein Ablauf wie in Abb. 4.1 betrifft nur die Verzögerung eines Pakets an einer Station. Auf eine Ende-zu-Ende-Latenz des betreffenden Pakets kann aus diesen Szenarien nicht zwingenderweise geschlossen werden. In Einzelfällen ist dies möglich, für die Worst-Case-Modellierung ist aber entscheidend, dass die Verzögerung an genau der betreffenden Station – und nicht im gesamten Netzwerk – den schlechtesten Fall abdeckt.

Die Begrenzung auf einzelne Stationen bedeutet nicht, dass Sendeabläufe in anderen Knoten und Switchen keine Rolle für das Szenario spielen. Im Gegenteil: Für die Worst-Case-Betrachtung von AVB-Paketen ist ein mögliches Szenario in den vorherigen Stationen der Burst-Pakete entscheidend, um eine größtmögliche Verzögerung zu generieren. Burst-Pakete sind in diesem Fall aufeinanderfolgende Pakete der Paketklasse des untersuchten Paketes, die eine Sendung des untersuchten Pakets verzögern können.

Für die Modellierung der Einflüsse durch andere Pakete wird überprüft, wann und in welcher Anzahl diese Burst-Pakete eintreffen können. Dafür werden auch die Stationen betrachtet, die diese Pakete an die untersuchte Station weiterleiten. Aus den sich ergebenden Annahmen für das Eintreffen der Pakete ergibt sich der Paketgenerierungs-Verlauf an der untersuchten Station. Anhand dieses Verlaufs und dem jeweiligen Kreditwert der Paketklassen ergibt sich der Sendeablauf. Für die in diesem Kapitel vorgestellten Worst-Case-Modelle für AVB- und TSN-Netzwerke werden alle Paketklassen, die auf das untersuchte Paket einen verzögernden Einfluss haben können, betrachtet und in Kombination berücksichtigt.

4.2 AVB

Dieser Teil des Kapitels beschäftigt sich mit der Worst-Case-Latenz für ein Paket eines AVB-Klasse-A-Streams. Worst-Case-Latenzen von AVB-Klasse-B-Paketen und Best-Effort-Paketen werden nicht gesondert betrachtet. Die folgenden Abschnitte befassen sich mit den Einflüssen auf Pakete von AVB-Klasse-A-Streams. Zusätzlich wird ein Blick auf mögliches Cycle Piling geworfen. Cycle Piling kann zu einer unendlichen Aufstauung von Paketen führen.

4.2.1 Einflüsse durch Best Effort

Best-Effort(BE)-Pakete werden in AVB-Netzwerken mit einer Priorität versendet, die niedriger als die von AVB-Klasse-A- oder AVB-Klasse-B-Paketen ist. Damit ein BE-Paket die Sendeerlaubnis erhält, müssen folgende Voraussetzungen erfüllt sein:

- Das betreffende Paket steht in der BE-Ausgangs-Queue an vorderster Stelle
- Kein Paket einer anderen Paketklasse hält grade das Senderecht an diesem Link

Aktive Sendungen anderer Klassen müssen abgeschlossen sein. Die, bis zum Abschluss der anderen Sendung, verstrichene Zeit kann Kreditwerte der AVB-Klassen so beeinflussen, dass die Sendeerlaubnis für BE-Pakete neu bewertet werden muss.
Verzögerungen eines AVB-Pakets durch BE-Pakete können durch folgende Szenarien auftreten:

1. Zum Zeitpunkt der Sendereitschaft des AVB-Pakets ist eine BE-Übertragung aktiv.
2. Zum Zeitpunkt der Sendereitschaft des AVB-Pakets ist auch ein BE-Paket sendebereit. Der Wert des Kredits für die Klasse des AVB-Pakets liegt im negativen Bereich.

Fall 1 ist unabhängig von dem Kreditwert der Klasse des AVB-Pakets. Durch fehlende Präemptions-Mechanismen unter Ethernet kann das BE-Paket in seiner Sendung nicht unterbrochen werden. Fall 2 tritt auf, wenn vor der Sendebereitschaft des AVB-Pakets andere AVB-Pakete übertragen wurden und den Kreditwert der Klasse herabgesenkt haben.

BE-Pakete sind Standard-Ethernet-Pakete (vgl. Abb. 2.1). Ein maximal großes BE-Paket ist 1526 Byte (inklusive Overhead) bzw. 1538 Byte groß, wenn die erforderliche Lücke (Interframe-Gap = IFG) zwischen dem BE-Paket und dem nachfolgenden Paket mitgerechnet wird. Allein die Blockierung durch die Versendung eines maximal großen BE-Pakets (Maximum Transmission Unit = MTU) kann bei 100 Mbit/s für ein AVB-Paket eine Verzögerung von ~125μs bedeuten.

Diese Verzögerung tritt dann auf, wenn die Sendung des BE-Pakets (MTU) kurz vor der Sendebereitschaft des AVB-Pakets beginnt (Fall 1 – unabhängig vom Kreditwert). Die größtmögliche Verzögerung durch ein BE-Paket tritt auf, wenn zum Zeitpunkt der Sendebereitschaft des AVB-Pakets ein negativer Kreditwert vorliegt und ein BE-Paket (MTU) vor dem Anstieg des Kredits auf 0 seine Sendung beginnt (Fall 2). Diese beiden Möglichkeiten sind in Abb. 4.2 veranschaulicht.

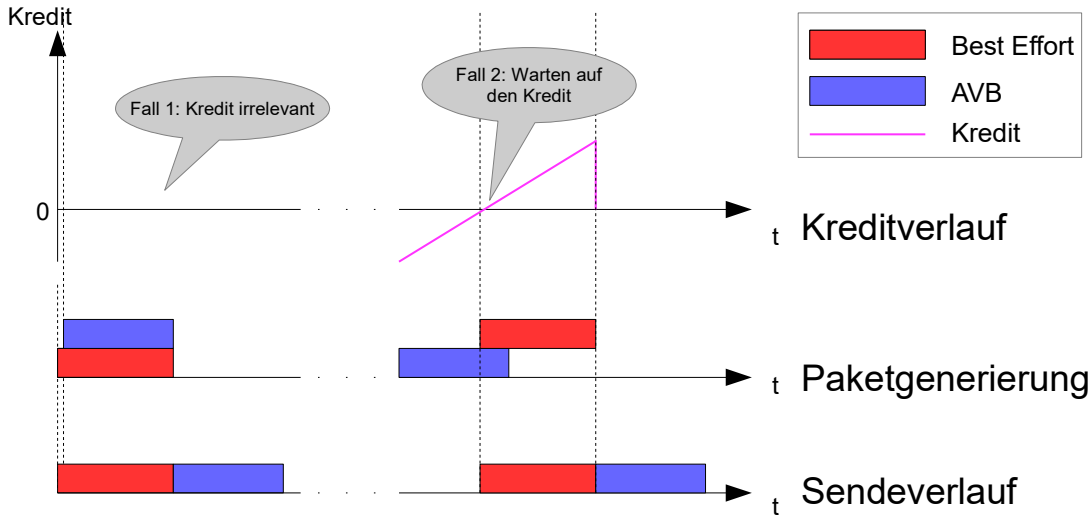

Abbildung 4.2: Verzögerung durch BE-Pakete

Blockiert ein BE-MTU-Paket die Versendung eines AVB-Pakets, steigt der Kredit der AVB-Klasse an. Gesetzt den Fall, dass die Übertragung des BE-Pakets erst beginnt, wenn der Kredit der AVB-Klasse fast wieder den Wert 0 erreicht hat, wird der Kreditwert nach dem Ende der BE-Übertragung positiv sein. Eine Verzögerung durch ein weiteres BE-Paket ist dann nicht mehr möglich. Der schlimmste Fall der Verzögerung durch ein BE-Paket tritt durch Fall 2 (vgl. Abb. 4.2) mit einem BE-MTU-Paket auf. Da der negative Kredit durch andere Pakete der untersuchten Paketklasse aufgetreten ist, wird diese Art der Verzögerung auch in Abschnitt 4.2.3 behandelt.

4.2.2 Einflüsse durch AVB-Klasse B

AVB-Klasse-B-Pakete werden prioritär unter Klasse-A-Paketen eingeordnet. Ähnlich wie BE-Pakete können Klasse-B-Pakete nur senden, wenn die übergeordneten Prioritätsklassen – in diesem Fall AVB-Klasse A – gerade kein Senderecht halten.

Die Voraussetzungen für eine Sendeerlaubnis sind:

- Das betreffende Paket steht in der Klasse-B-Ausgangs-Queue an vorderster Stelle
- Kein Klasse-A-Paket hat die Sendeerlaubnis

Analog zu BE-Paketen gilt, dass aktive Paketübertragungen abgewartet werden müssen und zu einer Neubewertung der Sendeerlaubnis führen können. Klasse-B-Pakete stehen zwar prioritär über BE-Paketen, weisen aber in ihren Möglichkeiten, die Sendung von Klasse-A-Paketen zu verzögern, keine besseren Eigenschaften als BE-Pakete auf. Für die Entwicklung eines Worst-Case-Modells für AVB-Klasse-A-Pakete wird daher nicht auf Klasse-B-Pakete zurückgegriffen. Die Modellierung wird ausschließlich aus BE-Paketen und (anderen) Klasse-A-Paketen bestehen.

4.2.3 Einflüsse durch AVB-Klasse A

Pakete dieser Klasse haben alle die gleiche Priorität wie das untersuchte Paket, das hier nachfolgend als Paket x referenziert wird.

Ein AVB-Stream reserviert für die Sendung seiner Pakete an jeder Station, die er im Netzwerk passiert, mindestens die Bandbreite, die nötig ist, um die Pakte des Streams weiterzuleiten. Die Höhe der reservierten Bandbreite leitet sich aus den *Measurement Intervals* der Klassen ab. Für Klasse A beträgt dieser 125μs, für Klasse B 250μs. Die Reservierung für AVB-Streams ermöglicht die Reservierung eines oder mehrerer Pakete innerhalb dieses Intervalls. Zur vereinfachten Veranschaulichung und für die hier vorgestellten Modelle wird angenommen, dass ein Stream für einen Measurement Interval immer nur ein Paket reserviert. Die Länge des Measurement Intervals ist äquivalent zu dem durchschnittlichen Abstand der einzelnen Pakete der Streams, daher lässt sich von der Paketgröße direkt auf die benötigte Bandbreite schließen. Für Klasse A können bei der Annahme einzelner Paketreservierungen pro Measurement Interval insgesamt 8000 Pakete in der Sekunde gesendet werden. Die benötigte Bandbreite für einen Stream der Klasse A ist somit $\frac{FrameSize \cdot 8000}{s}$, für Klasse B $\frac{FrameSize \cdot 4000}{s}$, mit $FrameSize$ für die Größe des Pakets in Bit/Byte inklusive Overhead.

Der Measurement Interval sorgt gleichzeitig dafür, dass die mögliche Anzahl an AVB-Streams pro Station begrenzt wird. Der Empfehlung aus Standard IEEE 802.1Qav [10] folgend, wird die Reservierung durch die höchste Prioritätsklasse (in diesem Fall AVB-Klasse-A) auf 75% begrenzt. Der Traffic durch AVB-Klasse-A-Pakete darf daher nicht mehr als 75% der Bandbreite des entsprechenden Links beanspruchen.

Die maximale Anzahl von AVB-Streams richtet sich nach der Größe der einzelnen Pakete der Streams. Möchte ein Stream s beispielsweise ein Paket der Größe 1171 Byte (inklusive Overhead) pro Measurement Interval verschicken, so beansprucht er, wenn die Reservierung erfolgreich verläuft, annähernd 75% der gesamten Bandbreite eines 100-Mbit/s-Links. Andere Streams dieser Klasse können dann auf den Ausgangs-Queues, die durch Pakete des Streams s passiert werden, keine erfolgreiche Reservierung mehr durchführen, bis dieser seine Reservierung beendet. 1171 Byte ist die maximale Größe eines AVB-Paket, das innerhalb eines 100Mbit-Netzwerkes verschickt wird.

Die minimale Größe eines AVB-Pakets beträgt 85 Byte (für die Reservierung, in der Übertragung hat das Paket tatsächlich nur 84 Byte). Um nicht mehr als 75% der Bandbreite an einer Station zu belegen, können maximal 13 dieser Minimal-AVB-A-Pakete durch Streams an den Stationen reserviert werden. Für diese Arbeit wird genommen, dass ein AVB-Paket mindestens 64 Byte Payload aufweist, was die maximale Anzahl der AVB-Pakete pro Station auf 10 begrenzt.

Alle Pakete der AVB-Klasse-A, ausgenommen Paket x, sind der Burst für Paket x. Die Anzahl der, dem Burst zugehörigen, Pakete wird durch die oben beschriebene Regulierung in den Grenzen der möglichen Bandbreitenreservierung gehalten. Je nach der Größe von Paket x kann die Verzögerung aus keinem Paket, oder mehreren Paketen, die in Summe die von Paket x nicht beanspruchte Bandbreite benötigen, bestehen.

Burst-Eigenschaften

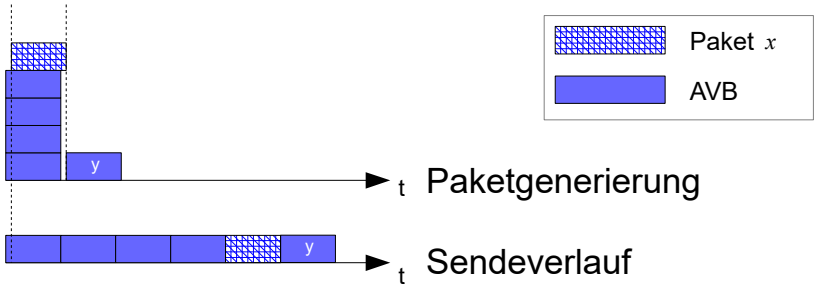

Abbildung 4.3: AVB-Burst mit nicht vollständig differierenden Burst-Quellen. Paket y kommt aus der gleichen Quelle wie eines der anderen AVB-Pakete. Sendeverlauf ist ohne Pausen durch Kreditschwankungen skizziert.

Das in Abschnitt 5.2 vorgestellte Analyse-Verfahren Network Calculus wird den Burst als die Anzahl Bytes aus der Summe der Klasse-A-Pakete, die eine erfolgreiche Reservierung an

der gleichen Ausgangs-Queue wie Paket x an der untersuchten Station haben, definieren. Da der Burst grundsätzlich vor Paket x abgearbeitet werden muss, bedeutet das für Pakete des Bursts, dass sie aus verschiedenen Quellen bezogen werden müssen. Ist das nicht der Fall, können die Pakete des Bursts nicht alle gleichzeitig eintreffen. Das bedeutet, dass nicht jedes Paket des Bursts für eine Verzögerung sorgen würde. In Abb. 4.3 wird ein dementsprechender Sendeverlauf skizziert. Das Paket y kommt aus der gleichen Quelle wie eines der anderen Pakete und kann deshalb nicht verzögernd in die Versendung von Paket x eingreifen.

Bei der Betrachtung von Sendeabläufen, die aus AVB-Paketen der gleichen Klasse bestehen, tritt wieder das FIFO-Prinzip bezüglich der Abarbeitungsreihenfolge in den Vordergrund. Ein Paket dieser Klasse, dass nicht vor der Generierung von Paket x seine Generierung abgeschlossen hat, zieht keine Verzögerung für Paket x nach sich.

Alle Burst-Pakete beeinflussen den Kredit der Paketklasse. Bei der Abarbeitung mehrerer Pakete bedeutet das, dass der Kredit während der Sendung der Pakete unter 0 fallen kann. Wenn in dieser Situation weitere Pakete in der Ausgangs-Queue stehen, entstehen Pausen zwischen den einzelnen Paketen. Unter Berücksichtigung der Kreditschwankungen ergibt sich eine Konstellation wie im Sendeablauf in Abb. 4.4.

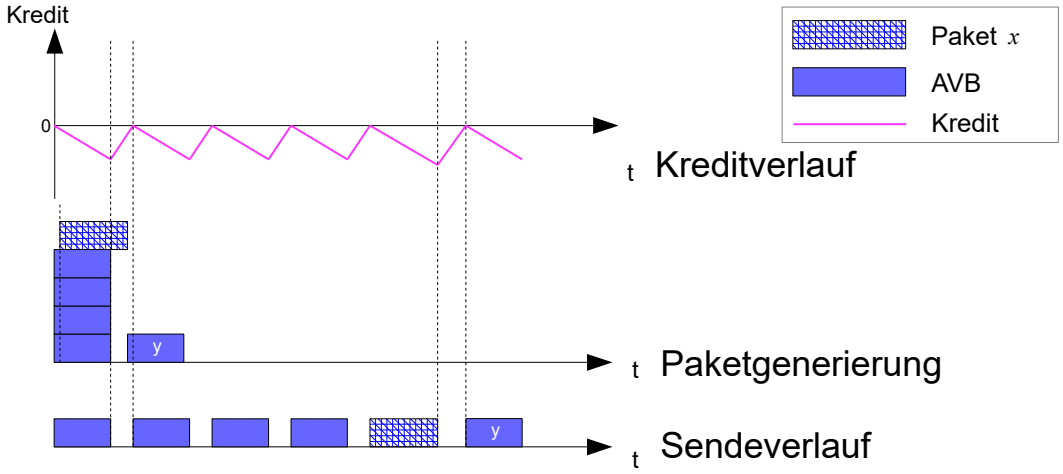

Abbildung 4.4: Pausen zwischen AVB-Paketen durch Kreditschwankungen.

Das in Abb. 4.4 veranschaulichte Szenario hat die gleiche Verzögerung zur Folge wie ein Szenario mit einem einzelnen AVB-Paket, das die gleiche Größe wie die Summe der Paketgrößen aller AVB-Pakete (ausgenommen Paket x und y) hat. Für die Modellierung des Worst-Case für

AVB-Netzwerke ist eine Unterteilung der Burst-Pakete in möglichst kleine Pakete entscheidend. Der Grund dafür wird nachfolgend erklärt.

Burst-Herkunft und doppelte Pakete

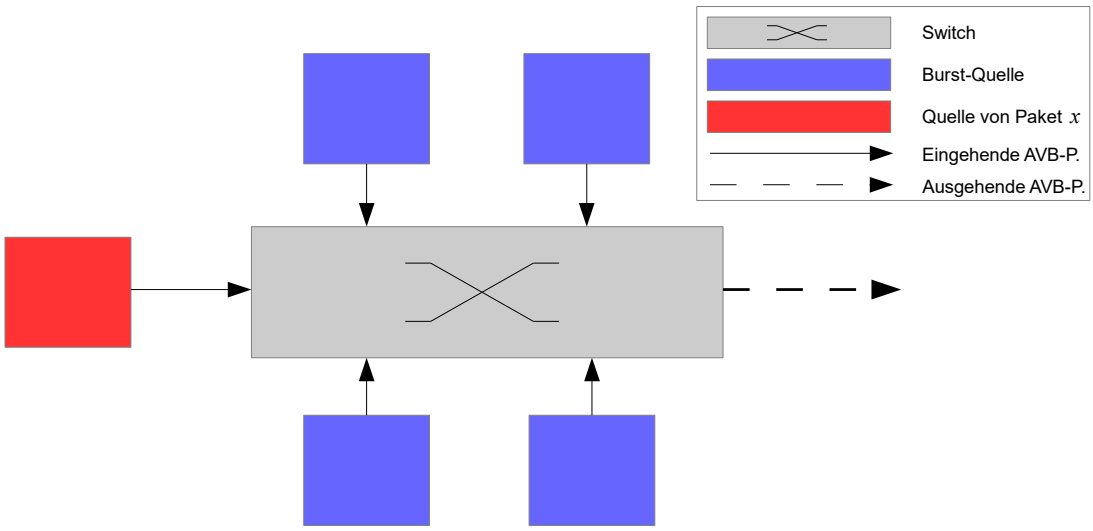

Abbildung 4.5: Herkunft von Burst-Pakten an einem Switch.

Burst-Pakete müssen aus verschiedenen Quellen kommen (vgl. Abb. 4.5), um eine Verzögerung durch jedes Burst-Paket zu gewährleisten. Die Bandbreitenreservierung durch AVB-Streams sieht vor, dass alle Pakete, die eine erfolgreiche Reservierung an der Ausgangs-Queue einer Station haben, innerhalb des Measurement-Intervals versendet werden können, wenn nur diese Paketklasse auf dem genutzten Link sendet. Neben der Möglichkeit, dass BE- oder Klasse-B-Pakete verhindern, dass alle erfolgreich reservierten Pakete innerhalb des Measurement Intervals versendet werden, gilt die Annahme, dass die reservierten Pakete jeweils ein Mal innerhalb des Measurement Intervals auftreten. Ist die Station ein Knoten, in dem die Pakete der Streams generiert werden, dann ist diese Annahme korrekt. Für den Fall, dass die Station ein Switch ist und die Burst-Pakete von anderen Stationen gesendet werden, kann eine andere Situation auftreten.

Die Station, die ein Burst-Paket an den untersuchten Switch weiterleitet, kann die Versendung des AVB-Pakets unter bestimmten Umständen verzögern. Der Measurement Interval sorgt auch an dieser Station dafür, dass AVB-Pakete im Abstand von 125μs verschickt werden. Das bereits erklärte Verhalten von AVB-Paketen, wenn ein BE-Paket ihre Sendung verzögert, führt

dazu, dass der Abstand zwischen zwei Paketen des selben Streams geringer als 125μs sein kann. Abb. 4.6 veranschaulicht den Fall, dass ein BE-MTU-Paket die Sendung eines Pakets verzögert.

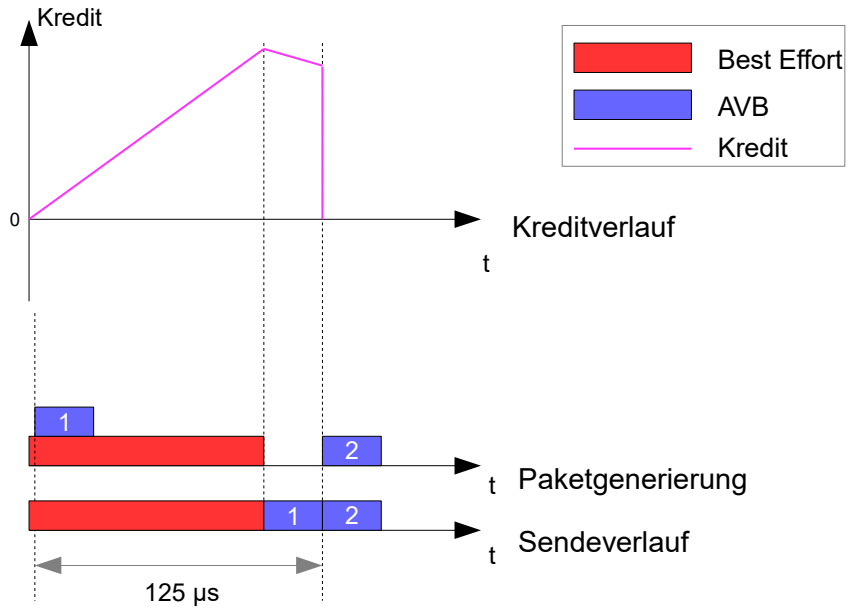

Abbildung 4.6: Aufeinanderfolgende Sendung von zwei AVB-Paketen eines Streams [15]

Das BE-Paket sorgt in diesem Fall dafür, dass zwei Pakete des gleichen Streams unmittelbar nacheinander gesendet werden. Das hat zur Folge, dass an dem untersuchten Switch zwei Pakete des gleichen Streams direkt hintereinander eintreffen.

An dieser Stelle stellt sich die Frage, ob das doppelte Eintreffen eines AVB-Pakets bereits dem Worst-Case für AVB-Pakete entspricht. In der Station, in dem in Abb. 4.6 die doppelte Paketsendung eines Streams erfolgt, wird das nächste Paket des Streams 125μs nach der doppelten Paketversendung geschickt. Ein Szenario, in dem drei Pakete direkt hintereinander senden, kann nicht auftreten. BE-Pakete, die dieses Verhalten provozieren könnten, können nicht groß genug sein, um einen Zeitraum von zwei Measurement Intervals abzudecken. Für diese Station kann es in Bezug auf hintereinander gesendete AVB-Pakete des selben Streams keinen ungünstigeren Fall als den in Abb. 4.6 veranschaulichten geben. Mehr als zwei Pakete hintereinander sind nicht möglich.

Da die Station vor dem untersuchten Switch auch ein Switch sein kann, können für ihn ähnliche Eigenschaften des eingehenden AVB-Verkehrs auftreten, wie sie in Abb. 4.6 durch

doppelte ausgehende Pakete verursacht werden. Die aufeinanderfolgenden zwei Pakete können dann wiederum durch ein BE-MTU-Paket blockiert werden. Das dritte Paket des Streams, das 125μs später am Switch eintrifft, wird dann ebenfalls gleich im Anschluss an die beiden ersten Pakete gesendet. Dieses Sendeablauf-Szenario wird in Abb. 4.7 gezeigt.

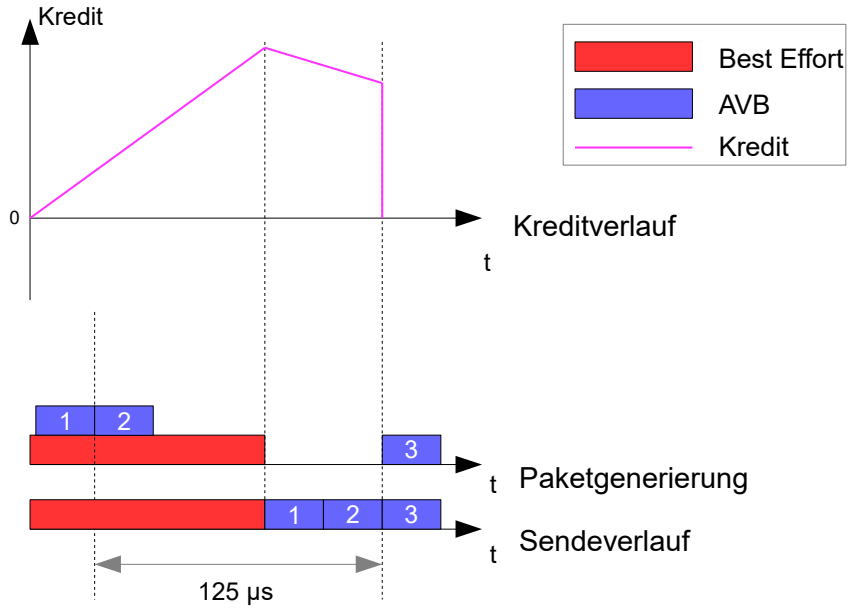

Abbildung 4.7: Doppelte Pakete der Vorstation führen zu drei Paketen in Folge

Eine Fortführung dieses Gedankengangs hätte eine stetig wachsende Anzahl hintereinander verschickter Pakete zur Folge. Das bedeutet allerdings nicht, dass die Anzahl hintereinander eintreffender Pakete je nach Anzahl der Vorgänger-Stationen des Streams bis in die Unendlichkeit wachsen kann. Der Kredit der AVB-Klasse sorgt dafür, dass nicht unendlich viele Pakete hintereinander geschickt werden können. Je nach den Raten $idleSlope$ und $sendSlope$, die die Kreditzu- und -abnahme steuern, kann die aufeinanderfolgende Sendung von AVB-Paketen begrenzt sein. Das BE-MTU-Paket hebt den Kredit auf seinen höchstmöglichen Wert. Aber auch dieser wird ab einer gewissen Anzahl von AVB-Paketen nicht mehr ausreichen, um alle Pakete zu versenden. Ein solches Szenario wird in Abb. 4.8 dargelegt.

In Bezug auf den $idleSlope$ an der Station vor dem untersuchten Switch ist ein weiteres Detail der Worst-Case-Modellierung wichtig: Um zu gewährleisten, dass die Pakete des AVB-Streams unmittelbar hintereinander verschickt werden, kann kein anderer AVB-Stream auf der selben Ausgangs-Queue Pakete reserviert haben. Das führt dazu, dass die gesamte reservierte Bandbreite ($idleSlope$) nur durch die reservierte Bandbreite dieses einen Streams bestimmt

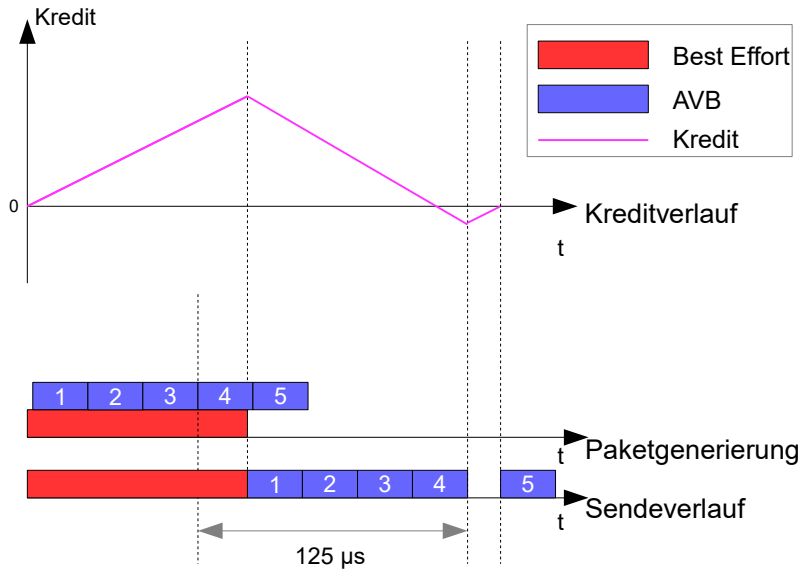

Abbildung 4.8: Hintereinander geschickte AVB-Pakete werden durch Kredit verzögert.

wird. Praktisch bedeutet das, dass nach der Sendung eines Pakets der Kredit negativ ist und erst nach Verstreichen des restlichen Zeitraums des Measurement Intervals wieder den Wert 0 erreicht. Wenn keine störenden Pakete anderer Klassen an der Station senden, ist der Sende- und Kreditverlauf ähnlich dem Ablauf in Abb. 4.9.

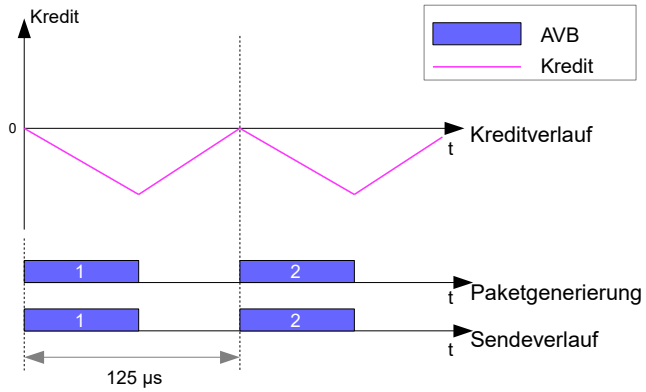

Abbildung 4.9: Kreditschwankungen bei einzelnen reservierten AVB-Paketen.

Dieses Verhalten sorgt dafür, dass – auch wenn mehrfach hintereinander AVB-Pakete des selben Streams eintreffen – ausschließlich ein Störfaktor wie das in den Abbildungen 4.6, 4.7 und 4.8 herangezogene BE-MTU-Paket dafür sorgen kann, dass mehr als ein AVB-Paket (des

selben Streams) hintereinander geschickt wird. Die Tatsache, dass für das Worst-Case-Modell einzelne reservierte Pakete mit begrenzter Größe an den Vorgänger-Stationen des Bursts für Paket x angenommen werden, führt dazu, dass ein Sende-Szenario wie in Abb. 4.7 oder 4.8 nicht auftreten kann. Ein Rechenbeispiel zeigt die Begrenzung der aufeinanderfolgenden Pakete auch bei großen AVB-Paketen:

Die Gesamtbandbreite für den Link der Ausgangs-Queue wird für dieses Beispiel mit 1 Bit/s angenommen. Der Measurement Interval wird auf 1s festgelegt. Die Pakete können zudem eine rationale Größe in Bits annehmen. Für 75% der Bandbreite reserviert ein AVB-Stream ein Paket der Größe 0,75 Bits an der Ausgangs-Queue. Das BE-MTU-Paket hat eine Größe von 0,99 Bits (auch in 100 Mbit-Netzwerken beansprucht ein BE-MTU-Paket nicht die volle Zeitspanne des Measurement Intervals). Die Werte der Kreditkurven ergeben folgende Werte:

$$idleSlope_{AVB\_max} = 0,75Bit/s, sendSlope_{AVB\_max} = -0,25Bit/s \tag{4.1}$$

Der höchstmögliche Kreditwert $HiCredit$ wird erreicht, wenn die Übertragung des BE-Pakets zu einem Zeitpunkt startet, an dem der Kredit 0 beträgt. Die Übertragung des BE-MTU-Pakets dauert exakt 0,99 s.

$$HiCredit_{AVB\_max} = 0,75Bit/s \cdot 0,99s = 0,7425Bit \tag{4.2}$$

Die Senkung des Kredits wird durch die Versendung der AVB-Pakete bewerkstelligt. Die Versendung eines AVB-Pakets beeinflusst den Kredit um den folgenden Betrag:

$$-0,25Bit/s \cdot 0,75s = -0,1875Bit \tag{4.3}$$

Der Kredit wird durch jedes versendete AVB-Paket um 0,1875 Bit verringert. Die Anzahl der zu versendenden Pakete, bis der Kredit wieder 0 beträgt, ist:

$$HiCredit \rightarrow 0_{AVB\_max} = \frac{HiCredit_{AVB\_max}}{0,1875} = 3,96 \tag{4.4}$$

Folglich ist der Kredit nach der Versendung von vier Paketen im negativen Bereich. Da keine Pakete reserviert werden können, die eine Größe von 0,75 Bits überschreiten, können nicht

mehr als vier AVB-Pakete unmittelbar aufeinander folgen.

Ein solches maximales AVB-Paket kommt als Burst-Paket nicht in Frage, da die Reservierung von Paket x erfolgreich abgeschlossen werden muss. Mit der Annahme, dass Paket x ein Minimal-AVB-Paket (Payload = 64 Byte) sein kann, ist die Größe – angewendet auf das oben beschriebene Rechenbeispiel – nach unten auf 0,06848 Bit begrenzt. Wenn der Burst aus einem Paket besteht, kann dieses theoretisch die Größe $0,75Bit - 0,06848Bit = 0,68152Bit$ annehmen. Das hat an der Station, die dieses Burst-Paket an den untersuchten Switch sendet, die folgenden Werte für $idleSlope$ und $sendSlope$ zur Folge:

$$idleSlope_{burst_1} = 0,68152Bit/s, sendSlope_{burst_1} = -0,31848Bit/s \tag{4.5}$$

Es ergibt sich für die Versendung dieser maximal großen Burst-Pakete:

$$HiCredit \rightarrow 0_{burst_1} \approx 3,11 \tag{4.6}$$

Das bedeutet, dass von Paketen dieser Größe maximal vier unmittelbar hintereinander an dem untersuchten Switch eintreffen können.

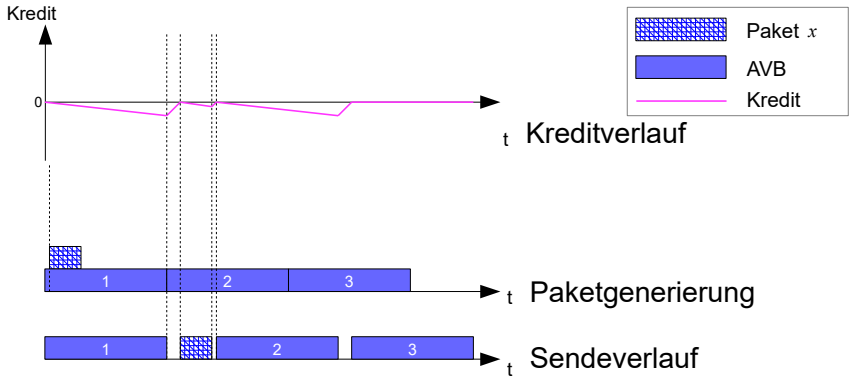

Abbildung 4.10: Mehrere Burst-Pakete der gleichen Quelle.

Wenn mehrere Pakete des selben Streams hintereinander am untersuchten Switch eintreffen, ergibt sich das in Abb. 4.10 skizzierte Bild des Sendeablaufs. Für die Worst-Case-Modellierung ist in diesem Fall festzuhalten, dass lediglich das erste Paket des Burst-Streams eine Verzögerung von Paket x zur Folge hat. Besteht der Burst aus einem Paket, wirken sich hintereinander

eintreffende Pakete des Bursts nicht negativer auf die Verzögerung von Paket x aus, als wenn nur ein einzelnes Paket des Bursts einträfe.

Eine Änderung der Auswirkung kann erzielt werden, wenn der Burst aus mehr als einem Paket besteht. Mit der Annahme, dass das zweite Paket des Bursts ein Minimal-AVB-Paket ist, ergeben sich für die Station, die dieses Burst-Paket an den untersuchten Switch versendet, folgende Werte für die entsprechende Ausgangs-Queue:

$$idleSlope_{burst_{min}} = 0,06848Bit/s, sendSlope_{burst_{min}} = -0,93152Bit/s \tag{4.7}$$

Daraus ergibt sich die folgende Anzahl an Paketen dieser Größe, die zur Senkung des Kredits auf den Wert 0 führen:

$$HiCredit \rightarrow 0_{burst_{min}} \approx 1,06 \tag{4.8}$$

Ist ein Burst-Paket ein AVB-Minimal-Paket, dann können von ihm also maximal zwei Pakete unmittelbar hintereinander an dem untersuchten Switch eintreffen.

Die verbleibende Bandbreite wird auf das zweite Burst-Paket verteilt. Dieses hat folgende Raten an seinen Stationen vor dem untersuchten Switch:

$$idleSlope_{burst_2} = 0,61304Bit/s, sendSlope_{burst_2} = -0,38696Bit/s \tag{4.9}$$

Zur Senkung des Kredits auf den Wert 0 wird die folgende Anzahl an Paketen benötigt:

$$HiCredit \rightarrow 0_{burst_2} \approx 2,56 \tag{4.10}$$

Von einem Paket dieser Größe können also maximal drei Pakete unmittelbar nacheinander im untersuchten Switch eintreffen. Wenn diese beiden Burst-Pakete am Switch eintreffen, kann sich der in Abb. 4.11 gezeigte Sendeablauf ereignen.

Im Vergleich zu dem Ablauf aus Abb. 4.10 ist zu erkennen, dass Paket x nach der Generierung des zweiten großen Burst-Paketes generiert wird. Diese Reihenfolge sorgt dafür, dass Paket x durch mehr als jeweils ein Paket aller Burst-Streams blockiert wird. Die beiden kleineren Burst-Pakete müssen abgewartet werden, bis die Sendung von Paket x möglich ist.

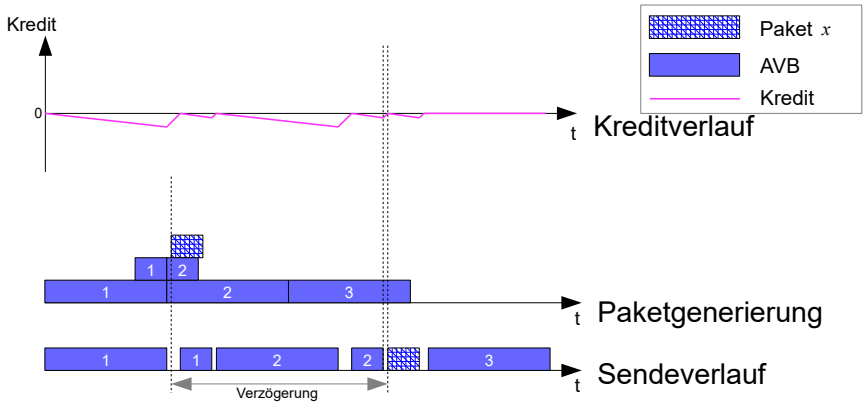

Abbildung 4.11: Burst-Pakete aus zwei Quellen mit unterschiedlichen Größen

Bei der Erweiterung dieses Szenarios auf drei Burst-Pakete wird ein Schema erkennbar, das die Erweiterung um möglichst viele unterschiedliche Burst-Quellen nahelegt. Um drei Burst-Pakete zu ermöglichen, wird ein weiteres Minimal-AVB-Paket von einer dritten Quelle hinzugefügt. Das große Burst-Paket hat dann folgende Kreditraten:

$$idleSlope_{burst_3} = 0,54456Bit/s, sendSlope_{burst_3} = -0,45544Bit/s \tag{4.11}$$

Die Kreditsenkung auf einen Wert von 0 nach Sendung eines BE-MTU-Pakets erfordert die folgende Anzahl an AVB-Paketen:

$$HiCredit \rightarrow 0_{burst_3} \approx 2,17 \tag{4.12}$$

Es ist möglich, drei Pakete dieses Bursts hintereinander auf dem untersuchten Switch zu empfangen. Das sich ergebende Worst-Case-Szenario wird in Abb. 4.12 veranschaulicht.

Das große Burst-Paket wird im Vergleich zu dem Zwei-Pakete-Szenario kleiner. Die dadurch entfallende Verzögerung wird durch das neu hinzugekommene Burst-Paket ausgeglichen. Diese Umstellung sorgt dafür, dass die beiden Streams der kleineren Burst-Pakete mit jeweils zwei Paketen eine Versendung von Paket x blockieren und die Verzögerung im Vergleich zum vorherigen Szenario vergrößern. Mit jeder weiteren Burst-Quelle erhöht sich die Verzögerung von Paket x. Auffallend ist, dass der Burst-Stream mit den großen Paketen nur mit einem seiner Pakete eine Verzögerung von Paket x herbeiführen kann. Die kleineren Burst-Pakete hingegen blockieren alle die Versendung von Paket x.

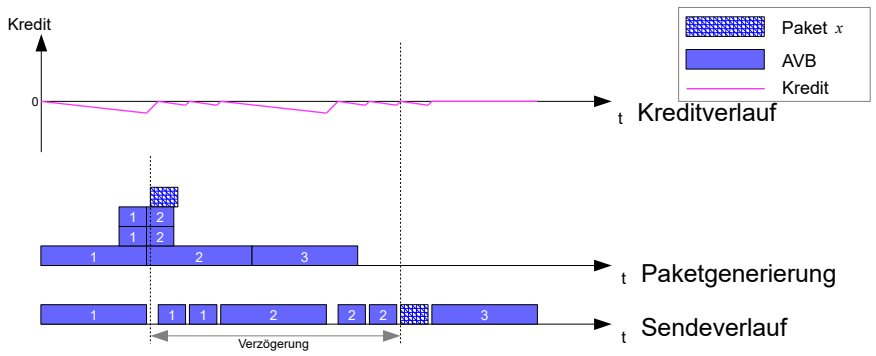

Abbildung 4.12: Burst-Pakete aus drei Quellen mit unterschiedlichen Größen

Die größtmögliche Verzögerung in einem Szenario aus unterschiedlich großen Burst-Paketen wird herbeigeführt, wenn folgende Eigenschaften der Sendeabfolge gelten:

1. Von allen Burst-Quellen werden zwei Pakete unmittelbar aufeinanderfolgend empfangen
2. Das größte Burst-Paket erhält von allen AVB-Paketen zuerst die Sendeerlaubnis
3. Die jeweils zweiten Pakete aus den Burst-Quellen erhalten vor Paket x die Sendeerlaubnis

Sind diese Eigenschaften gegeben, wird Paket x von den folgenden Paketen blockiert:

- Ein Paket des AVB-Streams, der die größten Burst-Pakete generiert
- Zwei Pakete aller anderen AVB-Streams, die Burst-Pakete generieren

Die AVB-Streams der kleineren Pakete können maximal zwei Pakete unmittelbar hintereinander an den Vorgänger-Stationen senden. Könnten sie drei oder mehr Pakete in Folge senden, würden diese eine weitere Verzögerung von Paket x zur Folge haben, wenn sie vor Paket x in der Ausgangs-Queue ankämen.

Ein AVB-Stream, der als einziger an den Vorgängerstationen reserviert und mindestens drei Pakete in Folge schicken kann, muss mehr als 50% der Bandbreite reservieren. Das bedeutet, dass so ein Stream das größte AVB-Paket am untersuchten Switch sein müsste. Ein zweiter AVB-Stream im Burst hätte dementsprechend nicht die Möglichkeit, drei Pakete in Folge an den untersuchten Switch zu schicken. Von dem größten Burst-Stream verzögert in jedem Fall nur ein Paket die Versendung von Paket x und da der kleinere Burst-Stream maximal zwei Pakete in Folge schicken kann, ist die Folge daraus, dass eine Verzögerung durch drei AVB-Pakete des gleichen AVB-Streams nicht möglich ist.

Der Burst-Stream mit den größten Paketen verzögert das Paket x mit einem seiner Pakete. Das bedeutet, dass von allen Burst-Streams jeweils zwei Pakete in das Worst-Case-Szenario der Verzögerung mit einfließen, mit Ausnahme des Burst-Streams mit den größten Paketen.

Abb. 4.13 veranschaulicht den Fall, dass alle Burst-Pakete die gleiche Größe haben.

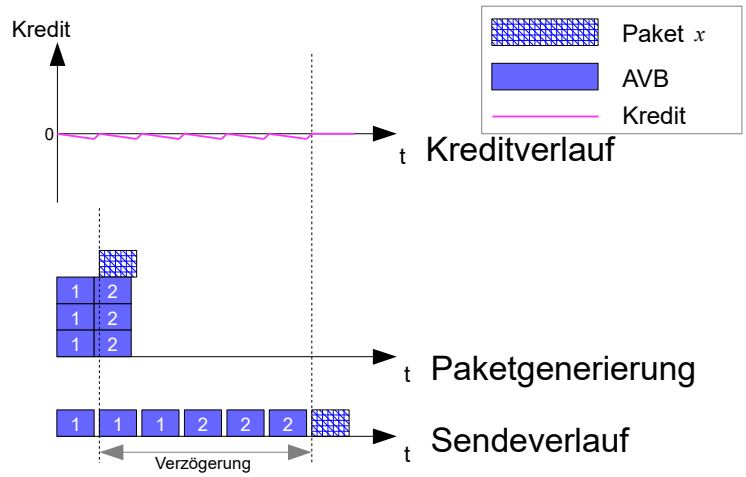

Abbildung 4.13: Burst-Pakete aus drei Quellen mit gleicher Größe

Trotz der gleichen Größe wird nur eines der Burst-Pakete in der Verzögerung nicht berücksichtigt. Dieses Szenario kann auf bis zu neun Burst-Pakete minimaler Größe erweitert werden, um die verfügbare Bandbreite vollständig auszunutzen. Ein Szenario mit Burst-Paketen aus fünf verschiedenen Quellen wird in Abb. 4.14 gezeigt.

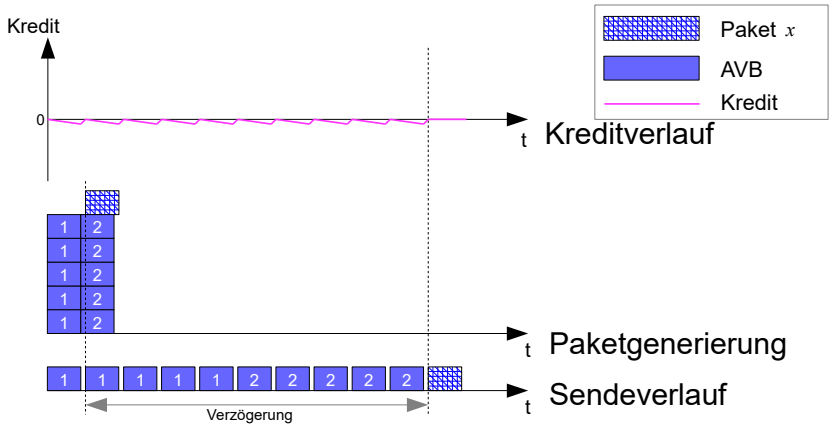

Abbildung 4.14: Burst-Pakete aus fünf Quellen mit gleicher Größe

4.2.4 Gesamt-Szenario

Das gesamte Worst-Case-Szenario für AVB-Netzwerke setzt sich aus einem Sendeablauf mit Paket x, AVB-Burst-Paketen und einem BE-MTU-Paket zusammen. Bevor Paket x seine Sendung beginnen kann, müssen die Worst-Case-Szenarien aus den Abschnitten 4.2.1 (BE-Pakete) und 4.2.3 (AVB-Klasse-A) berücksichtigt werden. Die Reihenfolge wird durch die Eigenschaften des Kreditsystems vorgegeben. Damit Pausen zwischen AVB-Paketen für eine größere Verzögerung sorgen, müssen AVB-Pakete zu einem Zeitpunkt gesendet werden, an dem der Kredit möglichst niedrig ist. Sendet zu Anfang des Szenarios ein BE-MTU-Paket, steigt der Kredit stark an und verhindert die meisten Pausen zwischen den AVB-Paketen. Das BE-MTU-Paket muss nach dem gesamten Burst (=allen AVB-Paketen außer Paket x) gesendet werden. Treffen an dem untersuchten Switch Burst-Pakete aus vier verschiedenen Quellen ein, ergibt sich das in Abb. 4.15 skizzierte Worst-Case-Szenario. Das mit „1*“ markierte Paket ist in diesem Fall das Burst-Paket, das nicht in die Verzögerung von Paket x mit einfließt.

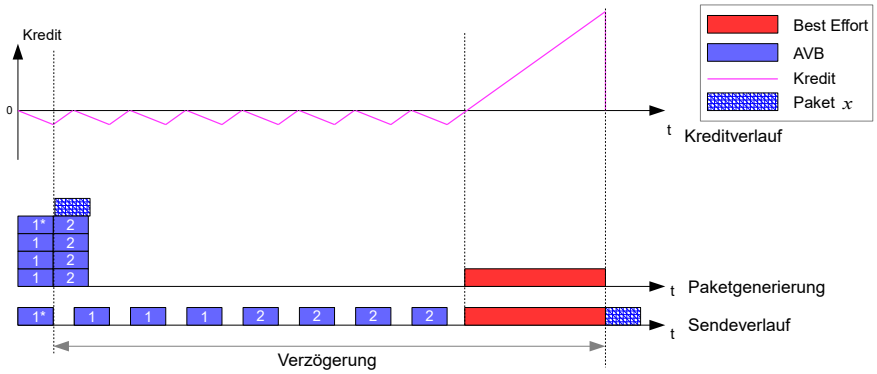

Abbildung 4.15: Worst-Case-Szenario an einem AVB-Switch

Diese Konstellation der Paket-Generierung beschreibt das Worst-Case-Szenario für ein AVB-Paket an einer Station seiner Sendung. In Kapitel 6 wird bei der Entwicklung und Beschreibung eines formalen Analyse-Modells für AVB-Netzwerke auf die Annahmen aus diesem Szenario zurückgegriffen.

4.2.5 Cycle Piling

Cycle Piling bezieht sich auf Netzwerke, die ihre Pakete in regelmäßigen Abständen schicken. Eine Zeitspanne, die alle Pakete enthält, ist ein Zyklus (Cycle). Cycle Piling drückt aus, dass nicht alle Paketsendungen innerhalb der Zyklen platzieren werden können und damit für eine Aufstauung und potentiell unendliche Latenz der Pakete sorgen.

Ein AVB-Netzwerk arbeitet bei der Versendung von AVB-Paketen mit Zyklen. Die Zyklen eines AVB-Netzwerks werden Measurement Interval genannt und beschreiben die benötigte Zeit für die Sendung aller reservierten AVB-Pakete, wenn kein Paket einer anderen Klasse sendet. Für Klasse-A-Pakete sind das 125μs. Werden beispielweise 3 AVB-Pakete reserviert, wird versucht in einem Zeitraum von 125μs jedes dieser Pakete einmal zu senden.

Da die Möglichkeit besteht, dass Pakete eines Streams zweimal hintereinander auftreten, verlängert sich die Zeit, die zum Senden aller Pakete aller AVB-Streams, ohne zwischenzeitliche Sendungen anderer Paketklassen, benötigt wird, von 125μs auf 250μs. Berücksichtigend, dass für den Worst-Case dieser Fall auftritt, wird der Begriff *Zyklus* in diesem Abschnitt zeitlich äquivalent zu 250μs aufgefasst.

Diese Zeitvorgabe kann eventuell nicht eingehalten werden, wenn Pakete anderer Paketklassen auf dem gleichen Link senden. Sendet innerhalb des 250-μs-Zyklus ein BE-MTU-Paket, werden die verbleibenden Pakete des Zyklus verzögert. Je nach Anzahl und Größe der AVB-Pakete kann dies dazu führen, dass einige AVB-Pakete erst gesendet werden, wenn die 250μs des Zyklus bereits abgelaufen sind. Die Pakete des nachfolgenden Zyklus werden dadurch um die Zeitspanne verzögert, die die Sendung der verbleibenden Pakete des ersten Zyklus die 250μs übersteigt.

Wäre der zeitliche Verlauf des nachfolgenden Zyklus, abgesehen von dem verspäteten Start, identisch mit dem des ersten Zyklus, würden sich die Pakete in nachfolgenden Zyklen um die gleiche Zeitspanne verschieben und eine kontinuierlich schlechter werdende Latenz der Pakete nach sich ziehen. Das würde bedeuten, dass in AVB-Netzwerken Cycle Piling auftreten kann. Dieser Abschnitt analysiert, wie sich Pakete innerhalb der Zyklen eines AVB-Netzwerkes verhalten und ob Cycle Piling wirklich auftreten kann.

Ob Cycle Piling auftritt, wird an einem theoretischen AVB-Netzwerk-Modell veranschaulicht. Die relevanten Kenngrößen des Netzwerks und der Stream-Reservierung werden nachfolgend in Tabelle 4.1 bei einer Link-Bandbreite C aufgeführt.

Alle Zeitspannen der Sendeabläufe lassen sich aus den in Tabelle 4.1 aufgeführten Werten ermitteln. x erfüllt die im Standard empfohlene Richtlinie, dass maximal 75% der Bandbreite für AVB-Verkehr verwendet werden soll. t_{BE} ist in seiner Größe eingeschränkt, weil ein reales BE-Paket maximal 123,04μs lang sendet und ebenfalls kleiner als die Hälfte eines 250-μs-Zyklus ist. Diese Eigenschaft ist entscheidend dafür, wie lange nach der Versendung eines BE-Pakets

Variable	Bedeutung	Einheit	Bedingung/Wert
x	$idleSlope$	bit/s	$0 < x \leq 75\%$ von C
T	Zeitspanne des Zyklus	Zeit	$250\mu s$
t_{BE}	Sendezeit des BE-Pakets	Zeit	$t_{BE} < 0,5 \cdot T$
C	Bandbreite der Links des Netzwerks	$\frac{Byte}{Zeit}$	100Mbit/s

Tabelle 4.1: Paketeigenschaften im Worst-Case-Szenario

ausschließlich AVB-Pakete gesendet werden können.

Nachfolgend werden die relevanten Zeitspannen, die innerhalb eines Sendeablaufs eines Zyklus auftreten, gegeben:

- Sendezeit aller AVB-Pakete (ohne Warten auf Kredit): $T \cdot x$
- Wartezeit durch Senden aller AVB-Pakete, wenn Kredit am Anfang 0 ist: $T \cdot (1 - x)$
- Zeit, bis der Kredit nach Sendung des BE-Pakets wieder 0 erreicht: $t_{BE} \cdot \frac{x}{1-x}$

Der erste Zyklus besteht in der Sendungsfolge aus den folgenden Abschnitten:

$$Cycle_1 = T \cdot x + T \cdot (1 - x) + t_{BE} = T + t_{BE} \tag{4.13}$$

Ist $t_{BE} > 0$, dann dauert der erste Zyklus länger als T. Der zweite Zyklus startet mit einer Verzögerung von t_{BE}. Der zweite Zyklus unterscheidet sich gegenüber dem ersten Zyklus darin, dass der Kredit zu Beginn der Übertragung der Pakete des zweiten Zyklus nicht 0 beträgt, sondern in dem Zeitraum t_{BE} um $t_{BE} \cdot x \cdot C$ steigt. Durch den positiven Kredit fällt die Wartezeit nach der Versendung von AVB-Paketen mindestens teilweise weg.

Der Verlauf des zweiten Zyklus hängt davon ab, ob die Versendung der enthaltenen AVB-Pakete dafür sorgt, dass zu einem Zeitpunkt während des Zyklus der Kredit den Wert 0 erreicht und damit ermöglicht, dass anteilig Wartezeiten (auf das Steigen des Kredits) auftreten und im Anschluss an die AVB-Pakete des Zyklus erneut ein BE-Paket senden kann. Um das zu ermitteln, wird die Zeit, die nach der BE-Paketsendung aus Zyklus 1 zum Erreichen eines Kreditwerts von 0 benötigt wird, von der Sendezeit der AVB-Pakete abgezogen. Ist dieser Wert größer als 0, dann ist in dem zweiten Zyklus wieder eine Sendung eines BE-Pakets möglich. Mit der Notation „$[k]^+$“, deren Wert entweder k (wenn $k \geq 0$), oder 0 (wenn $k < 0$) entspricht, hat der Zweite Zyklus folgenden Sendeablauf:

$$Cycle_2 = Cycle_1 - T + T \cdot x + \left[T \cdot x - t_{BE} \cdot \frac{x}{1-x}\right]^+ \cdot \frac{1-x}{x} + t_z,$$
$$t_z = \begin{cases} t_{BE}, & \text{wenn } T \cdot x - t_{BE} \cdot \frac{x}{1-x} > 0 \\ 0, & \text{sonst} \end{cases} \tag{4.14}$$

Wie sich der Sendeablauf verhält, hängt davon ab, ob der Kredit innerhalb des Zyklus den Wert 0 erreicht (ausgedrückt durch t_z). Es gibt zwei Fälle für den Sendeablauf des zweiten Zyklus. Der erste beschreibt, dass der Kredit während des zweiten Zyklus den Wert 0 erreicht:

$$\begin{aligned} Cycle_{2_1} &= Cycle_1 - T + T \cdot x + \left[T \cdot x - t_{BE} \cdot \frac{x}{1-x}\right]^+ \cdot \frac{1-x}{x} + t_z \\ &= t_{BE} + T \cdot x + T \cdot x \cdot \frac{1-x}{x} - t_{BE} \cdot \frac{x}{1-x} \cdot \frac{1-x}{x} + t_{BE} \\ &= t_{BE} + T - t_{BE} + t_{BE} \\ &= t_{BE} + T \end{aligned} \tag{4.15}$$

Dieser erste Fall des zweiten Zyklus hat exakt die gleiche Länge wie der erste Zyklus, der nachfolgende Zyklus würde das gleiche Verhalten aufweisen. Die Verzögerung, mit der die Zyklen starten, überschreitet in diesem Fall niemals den Wert t_{BE}. Für Fall 1 tritt somit kein Cycle Piling auf.

Fall 2, kein Erreichen des Kreditwerts von 0 innerhalb des Zyklus, hat den folgenden Sendeablauf zur Folge:

$$Cycle_{2_2} = Cycle_1 - T + T \cdot x + 0 = t_{BE} + T \cdot x \tag{4.16}$$

Die Verzögerung des darauf folgenden Zyklus ist dann $Cycle_{2_2} - T$. Im dritten Zyklus muss der Kredit spätestens wieder den Wert 0 erreichen, da in einem AVB-Netzwerk bei einer Maximalbelegung von 75% nur 3 von 4 Measure-Intervals ($125\mu s$) voll mit AVB-Paketen belegt sein können und die hier definierte Zyklus-Zeit eine Länge von 2 Measure-Intervals hat. Die Sendung eines BE-Pakets und das anteilige Warten auf Kredit werden somit in Zyklus 3 wieder möglich. Davon unabhängig sieht der Sendeablauf in Zyklus 3 folgendermaßen aus:

$$Cycle_3 = Cycle_{2_2} - T + T \cdot x + \left[T \cdot x - (t_{BE} \cdot \frac{x}{1-x} - T \cdot x)\right]^+ \cdot \frac{1-x}{x} + t_z,$$
$$t_z = \begin{cases} t_{BE}, & \text{wenn } T \cdot x - t_{BE} \cdot \frac{x}{1-x} - T \cdot x > 0 \\ 0, & \text{sonst} \end{cases} \tag{4.17}$$

Der Ausdruck in den eckigen Klammern ändert sich, weil die Zeit, bis der Kredit den Wert 0 erreicht, um die AVB-Sendezeit aus Zyklus 2 – also $T \cdot x$ – verringert werden muss. Wie bereits oben beschrieben, kann in Zyklus 3 nicht mehr der Fall auftreten, dass während des Zyklus nicht der Kreditwert 0 erreicht wird. Die Fallunterscheidung ist hier möglich, wirklich eintreten wird aber nur der nachfolgend beschriebene Fall, dass der Kredit den Wert 0 erreicht:

$$\begin{aligned} Cycle_{3_1} &= Cycle_{2_2} - T + T \cdot x + T \cdot x \cdot \frac{1-x}{x} - (t_{BE} \cdot \frac{x}{1-x} - T \cdot x) \cdot \frac{1-x}{x} + t_{BE} \\ &= t_{BE} + T \cdot x - T + T \cdot x + T \cdot x \cdot \frac{1-x}{x} - t_{BE} \cdot \frac{x}{1-x} \cdot \frac{1-x}{x} + T \cdot x \cdot \frac{1-x}{x} \\ &+ t_{BE} \\ &= t_{BE} + T \cdot x - T + T - t_{BE} + T \cdot x \cdot \frac{1-x}{x} + t_{BE} \\ &= t_{BE} + T \cdot x + T \cdot x \cdot \frac{1-x}{x} \\ &= t_{BE} + T \end{aligned} \tag{4.18}$$

Dieser dritte Zyklus verhält sich in der Länge des Sendeablaufs wieder wie Zyklus 1. Das bedeutet, dass der nachfolgende Zyklus das gleiche Verhalten wie $Cycle_{2_2}$ und der darauffolgende Zyklus das Verhalten von $Cycle_{3_1}$ aufweist. Auch für den zweiten Fall des zweiten Zyklus gilt somit, dass kein Cycle Piling auftreten kann. Für die Worst-Case-Betrachtung von AVB-Netzwerken bedeutet das, dass der schlimmste Fall der Verzögerung bereits im ersten Zyklus auftritt und eine Betrachtung des ersten (250μs-)Zyklus für die Entwicklung eines Worst-Case-Modells ausreicht. Das mögliche Auftreten von Cycle Piling wird auch für die Entwicklung des TSN-Modells gesondert betrachtet.

4.3 TSN

In diesem Abschnitt wird analog zu Störfaktoren für AVB-Pakete in Abschnitt 4.2 ein Blick auf mögliche Störfaktoren für AVB-Pakete in TSN-Netzwerken gegeben. Diese Störfaktoren bieten einen Rahmen für die Aufstellung eines formalen Analyse-Modells für TSN.

Für alle folgenden Annahmen gilt, dass von den möglichen drei Shapern des TSN-Protokolls (s. Abschnitt 2.6) der TAS verwendet wird. Dieser bietet den neuen TSN-Paketen, die mit der höchsten Priorität senden, gegenüber den AVB-Klasse-A-Paketen des AVB-Protokolls eine deutlich verringerte (Ende-zu-Ende-)Latenz. Um sicherheitsrelevante, zeitkritische Nachrichten zu schicken, sollte bei der Planung des Netzwerk-Verkehrs daher auf die Verwendung von TSN-Paketen zurückgegriffen werden.

Davon abgesehen existieren Paketklassen niedrigerer Priorität. Die bisherigen AVB-Pakete reihen sich prioritär unter den TSN-Paketen ein. Auch weitere AVB-Klassen sind in dem nicht abgeschlossenen TSN-Protokoll denkbar. BE-Pakete sind weiterhin prioritär unter den AVB-Klassen einzuordnen. Für die Paketklassen ergeben sich die in Tabelle 4.2 aufgeführten Einflüsse durch andere Paketklassen.

Paketklasse	**Beeinflusst durch**
TSN-Paket	TSN-Pakete
AVB-Paket	TSN-Pakete, AVB-Pakete, BE-Pakete
BE-Paket	TSN-Pakete, AVB-Pakete, BE-Pakete

Tabelle 4.2: TSN-Paketklassen und Einflüsse auf ihre Latenz

In dieser Worst-Case-Modellierung wird angenommen, dass TSN-Pakete, deren Versendung vor der Inbetriebnahme des Netzwerkes bereits fertig geplant sein muss, keine Verzögerung erfahren. Die Latenz der TSN-Pakete wird ausschließlich durch die Geschwindigkeit der passierten Links bestimmt. Das ändert sich, wenn mehrere TSN-Pakete auf den gleichen Sendeslot an einem Link zugeteilt werden. Dieser Fall könnte eine Verzögerung der TSN-Pakete durch andere TSN-Pakete verursachen. Da diese Verzögerung statisch berechenbar ist, wird sich die Worst-Case-Modellierung in diesem Abschnitt nicht auf die Verzögerung eines TSN-Paketes fokussieren, sondern kritische Szenarien für die AVB-Pakete entwerfen. Diese Szenarien bauen auf dem Worst-Case-Szenario aus Abschnitt 4.2 auf.

4.3.1 Einflüsse durch TSN-Pakete

Der einzige zusätzliche Störfaktor für AVB-Pakete gegenüber dem in Abschnitt 4.2 entwickelten Worst-Case-Modell sind die TSN-Pakete. Die Störcharakteristik der TSN-Pakete unterscheidet sich grundsätzlich von den niedrigprioren BE-Paketen und gleichprioren AVB-Burst-Paketen, weil TSN-Pakete prioritär über den AVB-Paketen stehen.

Der Ethernet 802.1Q-Standard, der Prioritäten an Stationen im Netzwerk möglich macht und auf dem AVB und TSN aufbauen, hat das potentielle Problem, dass viele hochpriore Pakete dafür sorgen können, dass sehr wenige oder gar keine Pakete niedrigerer Priorität versendet werden. Die TSN-Pakete, die nicht die Bandbreiten-Beschränkungen der AVB-Pakete berücksichtigen, können in TSN-Netzwerken diese „Unfairness“ verursachen. Das bedeutet, dass TSN-Pakete potentiell dafür sorgen können, dass kein einziges Paket anderer Klassen versendet wird.

In einem ersten Schritt wird der Einfluss auf ein Paket (Paket x) betrachtet, wenn Paket x neben den TSN-Paketen das einzige Paket im Netzwerk ist. Der (Worst-Case-)Einfluss lässt sich in drei Fälle unterscheiden:

1. Alle Abstände zwischen TSN-Paketen bieten genügend Platz für die Versendung von Paket x
2. Mindestens eine Lücke zwischen TSN-Paketen ist groß genug für Paket x
3. Keine Lücke zwischen TSN-Paketen ist groß genug für Paket x

Der erste Fall erlaubt die Versendung von Paket x. Jede Lücke ist groß genug für Paket x, daher richtet sich der Worst-Case nach dem größten TSN-Paket, das auf dem gleichen Link wie Paket x sendet. Beschreibt TSN_{max} die durch das größte TSN-Paket beanspruchte Sendezeit und C die Link-Bandbreite, wird die Verzögerung von Paket x durch $delay_{TSN_1}$ in Formel 4.19 beschrieben. $\frac{x}{C}$ beschreibt die Zeit, die Paket x zur Sendung über den Link benötigt.

$$delay_{TSN_1} = TSN_{max} + \frac{x}{C} \tag{4.19}$$

Fall 2 hat ein anderes Verhalten als Fall 1. Wird Paket x zu einem Zeitpunkt generiert, an dem keine ausreichend großen Lücken im TSN-Sendeplan vorhanden sind, muss Paket x warten, bis eine ausreichend große Lücke auftritt, um seine Sendung starten zu können. Die Verzögerung wird wie in Abb. 4.16 dargestellt auftreten.

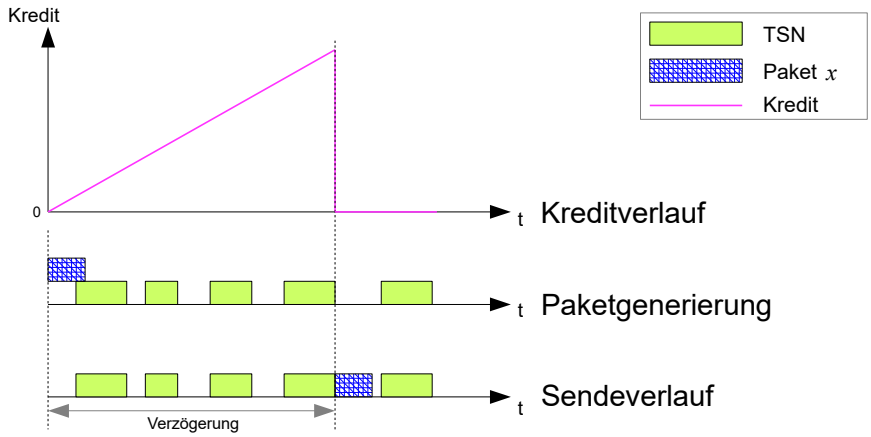

Abbildung 4.16: Lücken zwischen TSN-Paketen sind nicht alle ausreichend groß

TSN-Pakete werden in einem sich wiederholenden Kreislauf terminiert. Dieser Kreislauf (Cycle – unterscheidet sich von den in Abschnitt 4.2.5 genannten Zyklen) hat eine Gesamtlaufzeit. Ist in einem Kreislauf nur eine einzige Lücke zwischen TSN-Paketen vorhanden, die eine Sendung von Paket x ermöglicht, kann Paket x im schlechtesten Fall zu einem Zeitpunkt generiert werden, der unmittelbar nach dieser Lücke liegt. Das bedeutet, dass Paket x den gesamten Cycle abwarten muss, bis eine Sendung möglich ist. Um diesen Sachverhalt vereinfacht mathematisch auszudrücken, lassen sich für die Worst-Case-Betrachtung alle TSN-Pakete, inklusive der (nicht ausreichend großen) Lücken zwischen ihnen, zu einem einzelnen TSN-Paket zusammenfassen. Damit ergibt sich das in Abb. 4.17 skizzierte Bild des Sendeverlaufs.

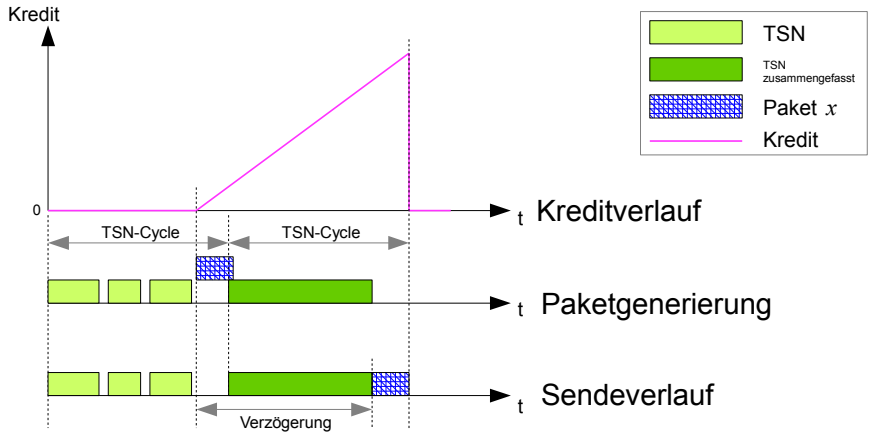

Abbildung 4.17: Zusammenfassung von TSN-Paketen eines Cycles

Die Sendezeit des zusammengefassten TSN-Block beschreibt für den Fall einer einzigen ausreichend großen Lücke den Worst-Case der Verzögerung für Paket x durch TSN-Pakete. Der entstandene TSN-Block sorgt dafür, dass dieses Szenario wieder Fall 1 entspricht. Da alle Lücken (hier nur eine Lücke) groß genug sind, wird das größte TSN-Paket (hier der zusammengefasste TSN-Block) zum Einsetzen in Formel 4.19 verwendet.

Gibt es mehr als eine Lücke, die für die Versendung von Paket x ausreichend groß ist, werden alle Sequenzen aus TSN-Paketen, deren Abstände voneinander nicht groß genug für die Versendung von Paket x sind, zu TSN-Blöcken zusammengefasst. Auf diese Weise kann ebenfalls mit Formel 4.19 gearbeitet werden. Die Sendezeit des größten entstehenden Blocks wird für TSN_{max} eingesetzt.

Der dritte Fall, keine ausreichend großen Lücken zwischen TSN-Paketen, hat eine unendliche Verzögerung für Paket x zur Folge. Das oben genannte Zusammenfassen der TSN-Pakete würde einen unendlich großen TSN-Block zur Folge haben. Für nachfolgende Modellierungen und Analyse-Modelle wird angenommen, dass dieser Fall nicht eintritt. Es muss im Sendeplan also mindestens eine Lücke zwischen TSN-Paketen existieren, die ausreichend groß für die Versendung von Paket x ist.

4.3.2 Gesamt-Szenario

Das hier entwickelte Worst-Case-Modell setzt auf das in Abschnitt 4.2 entwickelte Worst-Case-Modell für AVB-Netzwerke auf. Werden die hinzugekommenden TSN-Pakete in einer Netzwerkkonfiguration nicht verwendet, können sie keinen Einfluss auf die anderen Paketklassen ausüben. In diesem Fall entspricht das Worst-Case-Szenario eines AVB-Paketes dem in Abb. 4.15 gezeigten Sendeverlauf.

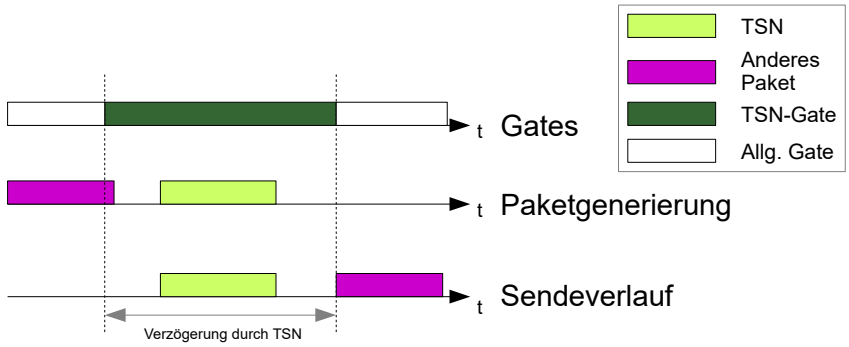

Abbildung 4.18: Blockierung eines Pakets durch TSN-Gate

Sind TSN-Pakete zur Versendung auf einem Link geplant, den Paket x passiert, ändert sich das Szenario je nach Größe und Abstand der einzelnen TSN-Blöcke. Ein TSN-Block beschreibt die Zeitspanne, die die Versendung von Nicht-TSN-Paketklassen blockiert werden. Diese Zeitspanne ist nicht von der Größe der TSN-Pakete abhängig, sondern kann von der Zeitspanne der Öffnung des TSN-Gates abgeleitet werden. Das ist so, da die TSN-Pakete Sendeslots benutzen, die nicht zwingenderweise die exakte Größe des Pakets haben. Abb. 4.18 zeigt ein Beispiel für die Verzögerung eines Nicht-TSN-Paketes durch das TSN-Gate. Ein Nicht-TSN-Paket kann nur senden, wenn das allgemeine Gate geöffnet ist.

Umgekehrt gilt, dass TSN-Pakete nur dann senden dürfen, wenn ihr Gate geöffnet ist. Ist ein TSN-Paket nicht korrekt mit dem Gate auf der Station synchronisiert, kann eine Wartezeit für TSN-Pakete entstehen. Nachfolgend gilt die Annahme, dass TSN-Pakete grundsätzlich mit den Gates synchronisiert sind. Zur Vereinfachung des Modells steht im folgenden Szenario die Größe eines TSN-Pakets synonym für die Größe eines TSN-Gates. Die Verzögerung durch ein TSN-Paket ist somit immer abhängig von dem Gate, in dem es sendet und Lücken zwischen den Paketen stehen synonym für Lücken zwischen geöffneten TSN-Gates.

In einem Szenario, das Pakete aus allen Paketklassen enthält, kann ähnlich zu Abschnitt 4.3.1 in drei Fälle bezüglich der Abstände zwischen den TSN-Paketen eingeteilt werden:

1. Mindestens eine Lücke zwischen TSN-Paketen bietet genügend Platz für den Worst-Case-Delay, den Paket x durch Nicht-TSN-Pakete erfahren kann
2. Die Lücken zwischen TSN-Paketen reichen nicht für den in Fall 1 genannten Worst-Case-Delay aus, bieten aber jeweils genügend Platz für mindestens je ein AVB-Paket
3. Keine Lücke zwischen TSN-Paketen ist groß genug für andere Pakete

Fall 3 wird für diese Modellierung ausgeschlossen, da die Annahme gilt, dass Lücken zwischen den TSN-Paketen eine Versendung von Paket x zulassen. Zusätzlich gilt die Annahme, dass jedes AVB-Paket gesendet werden kann.

Fall 1 hat eine Ergänzung zu dem in Abb. 4.15 skizzierten Worst-Case-Szenario für AVB-Netzwerke zur Folge. Die Verzögerung wird um das größte TSN-Paket erweitert und ergibt ein Szenario wie in Abb. 4.19. Die Reihenfolge der Pakete ist wieder an dem Kreditsystem von AVB orientiert. Das TSN-Paket kommt nach allen anderen Paketen des Delays, damit die Wartezeiten durch den AVB-Kredit weiterhin auftreten. Das BE-Paket muss ebenfalls vor dem

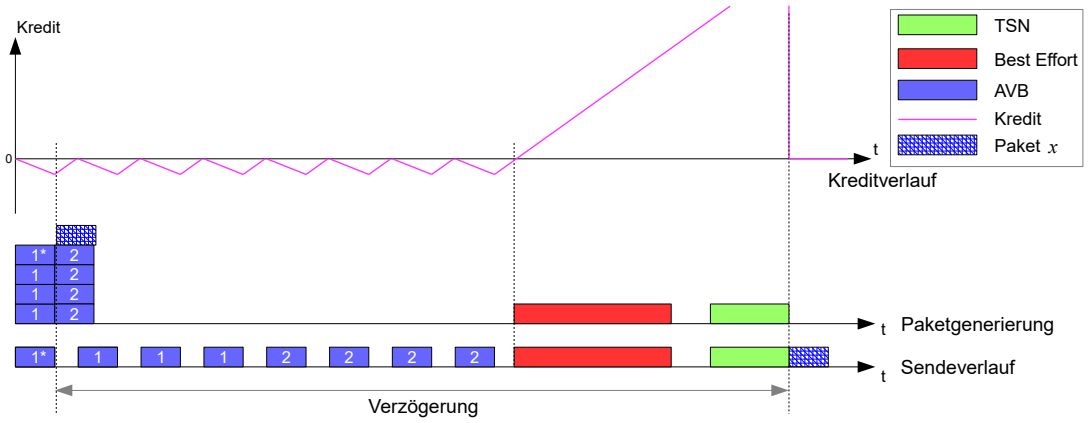

Abbildung 4.19: TSN-Cycle groß genug: TSN-Paket am Ende des Worst-Case-Sendeverlaufs

TSN-Paket versendet werden, da gestiegener Kredit anderenfalls für einen Vorzug von Paket x vor dem BE-Paket sorgen würde.

Tritt Fall 2 auf, richtet sich der Sendeablauf nach der Größe der Lücken zwischen den TSN-Paketen. Die größte Lücke zwischen den TSN-Paketen wird betrachtet, um das, den Lücken zwischen den TSN-Paketen entsprechende, Worst-Case-Szenario zu finden.

In der größten Lücke startet die Übertragung aller anderen Pakete. Die Unterbrechung durch ein TSN-Paket sorgt dafür, dass die verbleibenden AVB-Pakete in den folgenden Lücken übertragen werden. In Abb. 4.20 wird beispielhaft an zwei verschiedenen Sendeabläufen präsentiert, wie sich eine (zu) kleine Lücke zwischen TSN-Paketen auf den Sendeverlauf auswirkt.

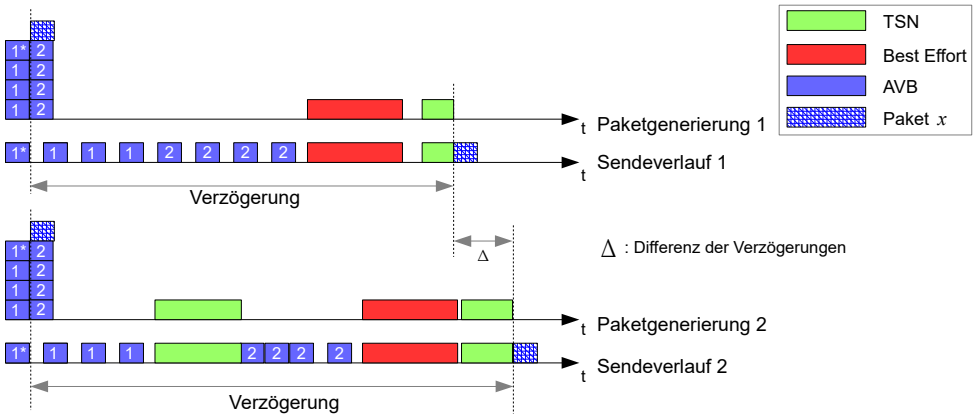

Abbildung 4.20: TSN-Lücken zu klein: Aufteilung der Sendungen

Sendeverlauf 1 zeigt den Fall, in dem alle Pakete (bis auf Paket x) vor dem TSN-Paket gesendet werden können. Sendeverlauf 2 ändert sich dem gegenüber durch das hinzugekommene zweite TSN-Paket. Dieses sorgt für eine Unterbrechung der Sendung der Nicht-TSN-Pakete. Ein Punkt, der in der formalen Analyse eine Rolle spielen wird, ist das Kreditverhalten im zweiten Sendeverlauf. Der Kredit ist in Abb. 4.20 nicht mit aufgeführt, das erste TSN-Pakete sorgt aber für einen „außerplanmäßigen“ Anstieg des Kredits. Der so gestiegene Kredit hat zur Folge, dass die folgenden AVB-Pakete (vornehmlich die ersten drei mit „2“ gekennzeichneten) keine Wartezeit aufgrund niedrigen Kredits haben. Die Lücke zwischen diesen Paketen schrumpft im Vergleich zu Sendeverlauf 1. Die insgesamt gestiegene Verzögerung ist nicht um das gesamte zusätzliche TSN-Paket gewachsen und es gilt die Aussage, dass $\Delta < TSN_1$ mit TSN_1 für das hinzugekommene TSN-Paket.

Dieses Szenario lässt sich auf eine Vielzahl von Lücken erweitern. Je nach der Größe der Lücken kann der Fall auftreten, dass eine Sendung des BE-Pakets nicht mehr möglich ist, dadurch fällt dieses im Szenario weg. Dieser Umstand kann dafür sorgen, dass zusätzliche TSN-Pakete eine Verringerung der Verzögerung bewirken. Ein Beispiel für ein solches Szenario wird in Abb. 4.21 veranschaulicht.

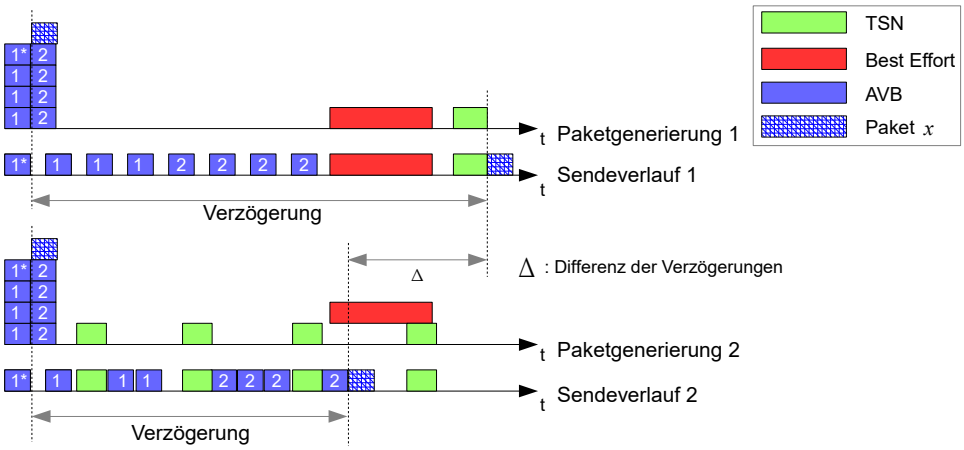

Abbildung 4.21: Mehrere kleine TSN-Lücken: Kein Platz für BE

Neben den vorgegebenen Größen der AVB- und TSN-Pakete geht diese Modellierung von einem variierend großen BE-Paket aus, um die Sendung eines BE-Pakets sicherzustellen. Das bedeutet, dass nicht zwingend ein BE-MTU-Paket in die Verzögerung mit einfließt. Tritt der oben beschriebene Fall auf, dass die Lücke zwischen TSN-Blöcken nicht zur Versendung des BE-MTU-Pakets ausreicht, wird die Größe des BE-Pakets so weit reduziert, bis eine Versendung

innerhalb der Lücke möglich ist. Wenn die Lücken zwischen den TSN-Paketen groß genug für das BE-Paket sind, davon abgesehen aber keine anderen (AVB-)Pakete in der gleichen Lücke senden können, muss das BE-Paket im Sendeverlauf an erster Stelle stehen. Anderenfalls sorgt der gestiegene Kredit durch die TSN- und AVB-Pakete dafür, dass das BE-Paket nicht mehr senden kann. Der durch das BE-Paket ansteigende Kredit kann in Kombination mit dem, durch die TSN-Pakete, gestiegenen Kredit dafür sorgen, dass die Kredit-Wartezeiten zwischen den AVB-Paketen weiter schrumpfen oder komplett wegfallen. Abgesehen von diesem Fall ist für den Worst-Case eines bestimmten TSN-Schedulings die späteste mögliche Sendung des BE-Pakets zu verwenden, damit Kredit-Wartezeiten zwischen den AVB-Paketen auftreten.

Eine ausreichend große Lücke ist als Startpunkt des Sendeablaufs kein Garant für das, diesem Netzwerk-Scheduling entsprechende, Worst-Case-Szenario. Wenn alle Pakete innerhalb einer Lücke zwischen TSN-Paketen untergebracht werden können, tritt möglicherweise eine kleinere Verzögerung auf, als wenn die Sendung in mehrere Lücken aufgeteilt wird. Das ist der Fall, wenn die zwischendurch gesendeten TSN-Pakete eine größere Verzögerung als die Kreditwartezeit der Burst-Pakete zur Folge haben. Als Startpunkt für den Sendeablauf ist die größte Lücke zwischen TSN-Paketen daher nicht immer optimal.
Für die formale Analyse sind folgende Punkte zu beachten:

- Das Worst-Case-Szenario für Klasse-A-Pakete in AVB-Netzwerken bleibt für das Worst-Case-Szenario von AVB-Paketen in TSN-Netzwerken, wenn möglich, vollständig erhalten und wird um das größte TSN-Paket erweitert
- Kann das Szenario nicht vollständig in einer Lücke zwischen TSN-Paketen stattfinden, wird es in mehrere Lücken verteilt
  - Das BE-Paket wird möglichst spät gesendet
  - Kreditschwankungen sind für nachfolgende Lücken zu beachten
  - Je nach Lückengröße kann die Größe des störenden BE-Pakets variieren
- Für eine universelle formale Betrachtung der Szenarien kann nicht vorausgesetzt werden, dass die größte Lücke zwischen TSN-Paketen der beste Startpunkt ist
- Abstände zwischen TSN-Paketen, die für Paketsendungen zu klein sind, führen zu einer Zusammenfassung dieser TSN-Pakete

Die Lückencharakteristik wird bei der Entwicklung des formalen Modells erneut diskutiert.

4.3.3 Cycle Piling und Bandbreitenbeschränkung

Unter TSN muss die Cycle-Piling-Problematik neu betrachtet werden. In AVB-Netzwerken kann Cycle Piling nicht auftreten, weil der Kredit-Mechanismus dafür sorgt, dass alle AVB-Pakete versendet werden. Der entscheidende Unterschied zu TSN ist bei AVB, dass alle anderen Pakete im Netzwerk mit niedrigerer Priorität senden und der Kredit-Mechanik unterworfen sind. Die Anzahl und Größe der Nicht-AVB-Pakete ist zweitrangig, da sie, wie in Abschnitt 4.2.5 gezeigt, keine Aufstauung der AVB-Pakete bewirken können, da nur der nicht-AVB-reservierte Bandbreitenbereich von ihnen genutzt wird.

Diese Kreditmechanik ist in TSN-Netzwerken weiterhin vorhanden, findet aber für TSN-Pakete keine Anwendung, da diese prioritär über den AVB-Paketen stehen. Hier ändert sich das Bild dahingehend, dass TSN-Pakete immer die Möglichkeit zum Senden erhalten. Sie können mehr als den nicht-AVB-reservierten Bandbreitenbereich verwenden und relativ einfach dafür sorgen, dass Cycle Piling entsteht. Ein simples Beispiel zeigt einen solchen Fall:

Paketklasse	Beanspruchte Bandbreite
TSN-Pakete	30%
AVB-Pakete	75%

Tabelle 4.3: Cycle-Piling-Paket-Szenario unter TSN

In der Datenmenge, die von den TSN- und AVB-Paketen zur Versendung bereitsteht und der Datenmenge, die von ihnen versendet wird, zeigt sich die Charakteristik des Cycle Pilings. Für eine Bandbreite von 100 Mbit/s wird diese in Tabelle 4.4 dargestellt.

Paketklasse	Datenmenge generiert	gesendet	nicht gesendet
TSN-Pakete	3,75 MB	3,75 MB	-
AVB-Pakete	9,375 MB	8,75 MB	0,625 MB

Tabelle 4.4: Datenmengen bei 100 Mbit/s Bandbreite für Pakete aus Tab. 4.3

Die Daten in Tabelle 4.4 gelten unter der Annahme, dass keine Pakete anderer Paketklassen generiert werden. Diese könnten die gesendete Datenmenge der AVB-Pakete weiter senken. Die nicht versendeten Daten der AVB-Pakete resultieren, wenn die Annahme dieser Arbeit gilt, dass die Ausgangs-Queues unendlich groß sind, in immer weiter steigenden Latenzen der AVB-Pakete. In realen Netzwerken, die mit begrenzten Queue-Größen arbeiten, bedeutet die Aufstauung, dass Pakete verloren gehen und eine unendliche Latenz aufweisen.

Ohne weitere Beschränkungen ist ein Cycle-Piling-Szenario unter TSN also recht einfach zu konstruieren. Beschränkungen für die Bandbreite von TSN- und insbesondere von AVB-Paketen

müssen formuliert werden, um die formale Analyse für AVB-Pakete in TSN-Netzwerken nicht in einen Rahmen zu fassen, der mit dem Vorzeichen arbeitet, dass eine ungünstige Konfiguration zu einer unendlichen Worst-Case-Latenz führt.

Der einfachste Ansatz, um Cycle Piling zu verhindern, ist eine Ausrichtung nach der verwendeten Bandbreite des TSN-Traffics. AVB-Streams können nur die Bandbreite reservieren, die nicht durch TSN-Pakete verwendet wird. Für die, im Beispiel aus Tabelle 4.3 aufgeführte, TSN-Bandbreite von 30% gilt dann, dass maximal 70% der restlichen Bandbreite von AVB-Paketen reserviert werden können. Diese Aufteilung kann zu Problemen mit der Versendung von Paketen mit niedrigerer Priorität als AVB-Pakete führen. Sind 100% der Bandbreite auf TSN- und AVB-Verkehr aufgeteilt, bleibt kein Platz für Pakete anderer Klassen.

Dem kann entgegengewirkt werden, indem die, nicht durch TSN-Pakete verwendete, Bandbreite erneut aufgeteilt wird. Das *Fairness*-Prinzip, das AVB auszeichnet und dafür sorgt, dass alle Pakete, die senden wollen, letztendlich auch senden können, kann in einem ähnlichen Ausmaß erreicht werden, wenn das gleiche Verhältnis zwischen AVB- und anderem Verkehr wiederhergestellt wird. Die Aufteilung der Bandbreite erfolgt dann wiefolgt:

Paketklasse	**Maximale Bandbreite**
TSN-Pakete	variabel, x
AVB-Pakete	$0,75 \cdot (1 - x)$
Andere Klassen	$0,25 \cdot (1 - x)$

Tabelle 4.5: Faires Scheduling unter TSN

In einem Netzwerk, in dem TSN-Pakete insgesamt 10% der Bandbreite beanspruchen, bleiben für AVB-Pakete 75% der verbleibenden 90% der Bandbreite. Insgesamt sollten AVB-Pakete also nicht mehr als 67,5% der Bandbreite reservieren. Den anderen Paketklassen bleiben 25% der 90% Bandbreite, was 22,5% der Gesamtbandbreite entspricht. Auf diese Weise kann gewährleistet werden, dass weiterhin alle Pakete irgendwann versendet werden.

Diese Anpassung sorgt dafür, dass Cycle Piling nicht auftreten kann. Für die Erstellung des formalen Analyse-Modells wird in der Parametrisierung des Netzwerks vorausgesetzt, dass die Bandbreiten nach diesem Schema aufgeteilt worden sind. Eine Konfiguration, die diese Vorgaben nicht erfüllt, wird deshalb von dem formalen Analyse-Modell möglicherweise nicht abgedeckt werden. Zur Einführung der formalen Analyse-Modelle werden nachfolgend Analyse-Methoden beschrieben.

5 Analyse

Für die Betrachtung von Netzwerkeigenschaften bieten sich mehrere Analyse-Methoden an. Statische Merkmale, wie Topologie und Bandbreiten der Links, bieten den Rahmen für die Betrachtung der Eigenschaften der Streams und anderer Paketsendungen. Gegenstand der Analyse sind Übertragungszeiten einzelner Pakete und benötigte Queue-Größen an den versendenden Queues der Switche und Knoten. Der Fokus in dieser Arbeit liegt auf den Übertragungszeiten, Queue-Größen werden daher als unendlich angenommen.

5.1 Analyse-Methoden

Es existieren unterschiedliche Ansätze der Analyse. Nächstliegend ist die Betrachtung eines **realen Netzwerkes**. Für die Betrachtung von Ende-zu-Ende-Latenzen können Zeitstempel der Übertragung an Start- und Ziel-Knoten ausgelesen werden. Für komplexe Netzwerkarchitekturen ist dieses Vorgehen relativ aufwendig, da die Topologie des Netzwerks mit allen Hardware-Ressourcen gebaut und eine Konfiguration der Paketsendungen vorgenommen werden muss.

Zur Begrenzung des Aufwands kann die Analyse an einer **Simulation** eines Netzwerkes ausgeführt werden. Die Simulation bietet Möglichkeiten zur einfacheren Konfiguration ganzer Netzwerke inklusive ihrer Paketsendungen. Mit der Simulationsumgebung OMNeT++ [5], die speziell für die Simulation von Netzwerken ausgelegt ist, ist neben der Erstellung von Simulationsmodellen für bestimmte Netzwerkprotokolle auch die Erfassung von Analyse-Daten einfach zu bewerkstelligen. Das Simulationsmodell *INET* [6] bietet den Zugriff auf eine Vielzahl von Ethernet-Protokollen. Darauf aufbauend kann auf das Simulationsmodell *CoRE4INET* für TTEthernet und AVB zurückgegriffen werden, das innerhalb der Projektgruppe CoRE [3] an der HAW Hamburg erstellt wurde.

Die Betrachtung eines realen Netzwerkes und dessen Abbildung in einer Simulation können Aussagen über Latenzen innerhalb des Netzwerkes treffen. In Bezug auf den Worst-Case der Latenzen können diese Aussagen aber unzureichend sein. Der Worst-Case ist in diesem

Fall dadurch definiert, dass ein einzelnes Paket (das zu analysierende Paket x) betrachtet wird und dessen Ende-zu-Ende-Latenz durch andere Pakete so stark wie möglich verzögert wird.

Ein Fall, in dem sich Pakete so stauen, dass Paket x den Worst-Case erlebt, kann in einem realen Netzwerk und somit auch in der Simulation auftreten. Ein solches Szenario muss aber nicht vorkommen. Dabei ist egal, ob das Netzwerk bzw. die Simulation über einen langen Zeitraum läuft. Theoretisch kann bei unendlicher Laufzeit nie der schlechteste Fall eintreten. Für Aussagen über Worst-Case-Latenzen sind daher reale Netzwerke und Simulationen nur bedingt geeignet. Im späteren Verlauf dieser Arbeit wird noch erläutert, warum eine Simulation Schwierigkeiten hat, den Worst-Case abzubilden und mit welcher Änderung des Simulationsmodells eine Annäherung eines solchen Szenarios dennoch möglich ist. Die Probleme der Simulation bezüglich Worst-Case-Aussagen sind teilweise darin begründet, dass eine Simulation den Worst-Case nicht gezielt herbeiführen soll. Der Fokus liegt auf der korrekten Abbildung aller Netzwerkeigenschaften und der Streams.

Im Gegensatz dazu bieten **formale Analyse-Methoden** eine vollständige Ausrichtung auf die Worst-Case-Szenarien. Formal bedeutet, dass keine Aussagen anhand von Beobachtungen getroffen werden, sondern mit Rechenmodellen gearbeitet wird, die einen Worst-Case konstruieren. Ein typisches Vorgehen zur Erstellung von formalen Analyse-Methoden für Übertragungseigenschaften von Paketen in Netzwerken besteht in der Modellierung aller Einflüsse auf die Pakete und der Konstruktion eines Ablaufs, der die größtmögliche Verzögerung für Paket x nach sich zieht. Diese Konstruktion wird in einem formalen Rahmen gefasst, die eine beliebige Manipulation der Einflüsse erlaubt. Das Worst-Case-Szenario wird durch das Modell beschrieben, das die schlechtesten Einflüsse vereint. Eine Worst-Case-Modellierung ohne formalen Rahmen findet sich in Kapitel 4. Der fehlende formale Rahmen bedeutet für die Modelle aus Kapitel 4, dass die Modelle der AVB- und TSN-Worst-Case-Modellierung keinen parametrierbaren Rahmen aufweisen, der zu einer eindeutigen Aussage führt. Die Modelle berücksichtigen dennoch alle denkbaren Einflüsse durch andere Paketklassen, diskutieren diese und formulieren sie mitunter zu Annahmen, die für die Entwicklung der formalen Analyse-Modelle verwendet werden. Darauf aufbauend werden in Kapitel 6 Modelle für AVB und TSN vorgestellt bzw. entworfen, die die schlechtesten Fälle dieser Protokolle abhängig von Parametern berechenbar machen sollen.

Im folgenden Abschnitt wird die formale Analyse-Methode Network Calculus vorgestellt. Neben anderen Analyse-Modellen, die teilweise kurz in Kapitel 3 vorgestellt wurden, ist der Network Calculus auf die formale Analyse von Netzwerken ausgerichtet und war bereits in

den Arbeiten [14, 15] die Grundlage zur Analyse von AVB-Netzwerken. In Kapitel 6 werden formale Worst-Case-Modelle für AVB und TSN auf Basis des Network Calculus erklärt und (weiter)entwickelt.

5.2 Network Calculus

Die formale Analyse-Methode, die in dieser Arbeit im weiteren Verlauf verwendet wird, ist der Network Calculus [22]. Er wird in den folgenden Kapiteln bei der Analyse von AVB- und TSN-Netzwerken zum Einsatz kommen. In den vorangegangenen Arbeiten [14, 15] wurde ein Network-Calculus-Modell für AVB weiterentwickelt. Dieses Modell und seine erneute Weiterentwicklung zur Berücksichtigung von TSN-Paketen wird in Kapitel 6 erläutert. In den folgenden Unterabschnitten wird einen Überblick über den grundlegenden Aufbau des Network Calculus und seiner Anwendung gegeben.

5.2.1 Mathematische Grundlagen

Rechnungen im Network Calculus sind auf unäre und binäre Operatoren angewiesen, die an dieser Stelle eingeführt werden.

Infimum

Das Infimum ist die größte untere Schranke einer Menge. Oft entspricht dieses dem Minimum der Menge, bei einem offenem Intervall ist das aber nicht der Fall.

$$\text{Beispiel: } \mathit{inf}\{n > 0 : \tfrac{1}{n}\} = 0$$

Supremum

Das Supremum ist die kleinste obere Schranke einer Menge. Oft entspricht dieses dem Maximum der Menge, bei einem offenem Intervall ist das aber nicht der Fall.

$$\text{Beispiel: } \mathit{sup}\{n > 0 : 1 - \tfrac{1}{n}\} = 1$$

$[\,]^+$-Notation

$$\text{Beispiel: } R[t]^+ = \begin{cases} R(t), & \text{wenn } R(t) \geq 0 \\ 0, & \text{sonst} \end{cases}$$

Funktionsfaltung

Die in der Funktionsanalyse verwendete Faltung (auch Konvolution genannt) zweier Funktionen (f und g) liefert ein Produkt ($f * g$) und ist definiert durch:

$$(f * g)(t) := \int_0^t f(t-s)g(s)ds \quad \text{für } f(t<0) = g(t<0) = 0$$

Die resultierende Funktion beschreibt den gewichteten Mittelwert der Funktion g mit der Gewichtung f abhängig von s.

5.2.2 Einführung

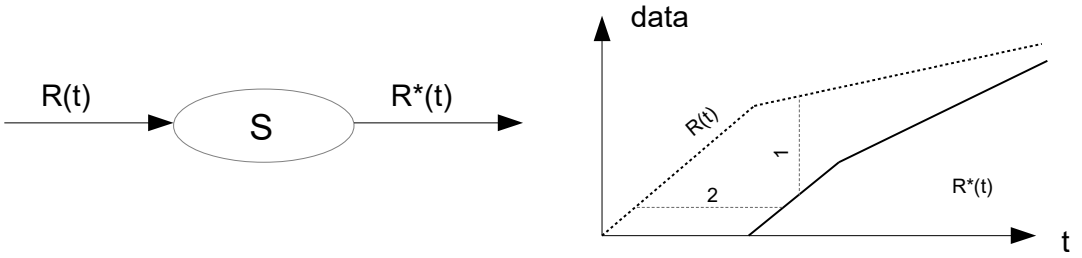

Abbildung 5.1: Network Calculus: Ein System S mit Input-Funktion $R(t)$, Output-Funktion $R^*(t)$ und dem zeitlichen Verlauf der gesendeten Daten dieser Flows [14]

Der Network Calculus untersucht die Eigenschaften von Flows (Ströme). In Bezug auf Netzwerke sind das die Eigenschaften von Daten-Flows, beispielsweise in einem Ethernet-Netzwerk. Die Flows werden in Input- und Output-Flows aufgeteilt und beziehen sich auf ein definiertes System. Ein beispielhaftes System S und ein mögliches Verhalten seiner Eingangs- und Ausgangsströme $R(t)$ bzw. $R^*(t)$ wird in Abb. 5.1 veranschaulicht.

Alle Ströme sind monoton steigend, fehlerfreie Übertragungen werden angenommen und sorgen dafür, dass keine Daten verloren gehen.

Ein System ist ein Netzwerk-Knoten, genauer der Teil des Knotens, der sich mit der Versendung des betrachteten Stroms beschäftigt. Unter Ethernet ist ein System ein Switch oder Netzwerk-Knoten, wird aber in der Analyse eines bestimmten Stroms auf die Komponenten reduziert, die einen Einfluss auf die Weiterleitungs-Geschwindigkeit des Stroms haben. Ein Datum, das vollständig im System angekommen ist, liegt in der Ausgangs-Queue und kann jederzeit weiter versendet werden.

5.2.3 Eigenschaften eines Flows

Sind ein Eingangsstrom $R(t)$ und Ausgangsstrom $R^*(t)$ gegeben, lassen sich zwei Werte direkt ablesen (vgl. Abb. 5.1):

Definition 1 (backlog) *Der $backlog$ zu Zeitpunkt t ist $R(t) - R^*(t)$*

Definition 2 (virtual delay) *Der $virtual\ delay$ zu Zeitpunkt t ist*
$d(t) = \underset{\tau \geq 0}{inf}\{R(t) \leq R^*(t + \tau)\}$

Der *backlog* lässt sich in Abb. 5.1 auf der rechten Seite an der mit 1 markierten Vertikale ablesen. Er drückt aus, wie viele Daten (in Bits) zu Zeitpunkt t im System angekommen, aber noch nicht weiter versendet worden sind. Für die Analyse ist dieser Wert interessant, wenn minimale Queue-Größen ausgewählt werden sollen.

Der *virtual delay* ist die Zeitspanne, um die ein Bit b verzögert wird, das zum Zeitpunkt t im System angekommen ist. Diese Betrachtung setzt ein FIFO-Verhalten der Ausgangs-Queue des Systems voraus. Es werden alle Bits, die vor dem Zeitpunkt t angekommen sind, vor b übertragen. In Abb. 5.1 lässt sich der *virtual delay* auf der rechten Seite an der mit 2 markierten Horizontale ablesen.

Die mithilfe des backlogs getroffenen Aussagen über Queue-Größen sind in der weiteren Analyse dieser Arbeit nicht von Belang, da für die Untersuchungen unendliche Queue-Größen angenommen werden. Der virtual delay hingegen wird benötigt, um eine Abschätzung der Worst-Case-Latenzen geben zu können. Er wird in den folgenden Abschnitten und Kapiteln den Kernpunkt der Analyse ausmachen.

5.2.4 Min-Plus-Algebra

Für die Abschätzung von Latenz-Zeiten werden Kurven gegenübergestellt, die den Worst-Case für Eingangs- und Ausgangsströme modellieren. Diese Kurven, die in Abschnitt 5.2.5 vorgestellt werden, sind teilweise voneinander abhängig. Die Abhängigkeit der beiden Kurven wird mithilfe der Min-Plus-Algebra (MPA) hergestellt. Die MPA unterscheidet sich von konventioneller Algebra in ihrer Definition einiger Operatoren.

Vergleich zur konventionellen Algebra

Die konventionelle Algebra umfasst bekannte Gesetzmäßigkeiten wie beispielsweise das Assoziativ-, Distributiv- oder Kommutativgesetz. Deren Definitionen unterscheiden sich teil-

weise für verschiedene binäre Operatoren. Das Kommutativgesetz bedeutet für die Operation „Addition“ $(+)$ sowie „Multiplikation“ $(\cdot)$, dass die Operanden ausgetauscht werden können, ohne das Ergebnis der Berechnung zu ändern. Für Berechnungen in der Min-Plus-Algebra bleibt das Kommutativgesetz erhalten, aber die Definition der Operatoren verändert sich.

Konventionelle Algebra	Min-Plus-Algebra
$min\{a, b\}$	$a \oplus b$
$a + b$	$a \otimes b$

Tabelle 5.1: Min-Plus-Algebra-Definitionen

Wie Tabelle 5.1 zeigt, hat eine Addition in der MPA als Ergebnis nicht die Summe der beiden Operanden, sondern deren Minimum. Eine Multiplikation funktioniert in der MPA wie eine Addition der konventionellen Algebra. Diese Umdeutung der Operatoren-Definitionen ist notwendig, um eine Funktionsfaltung vorzunehmen, die den Worst-Case des Ausgangs-Stromes modelliert. Es gibt weitere Änderungen an Operatoren in der MPA, die im Rahmen dieser Arbeit nicht zum Einsatz kommen und hier nicht gesondert erklärt werden.

Min-Plus-Faltung

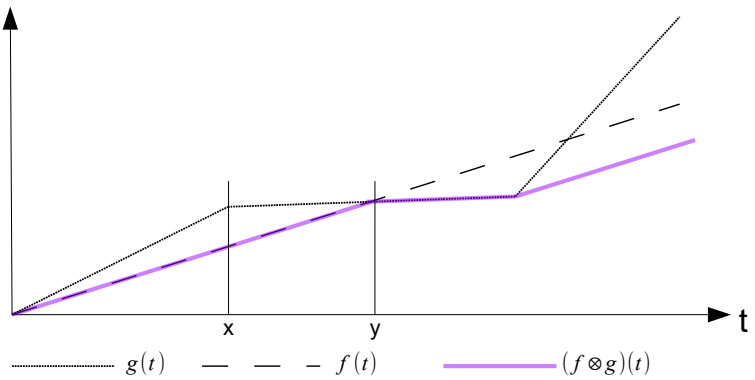

Abbildung 5.2: Min-Plus-Faltung von $f(t)$ und $g(t)$

Die in Abschnitt 5.2.1 zugrunde gelegte Definition einer Funktionsfaltung kann in der MPA mit der geänderten Operatoren-Definition vorgenommen werden. Sie hat die folgende Form:

$$(f \otimes g)(t) := \inf_{0 \leq s \leq t} \{f(s) + g(t - s)\} \tag{5.1}$$

Eine Min-Plus-Faltung ist bei der Modellierung des Worst-Case-Ausgangsstroms nötig. Diese Kurve wird Service-Curve genannt und detailliert in Abschnitt 5.2.5 erläutert. Abb. 5.2 zeigt ein Beispiel für eine Min-Plus-Faltung der Funktionen $f(t)$ und $g(t)$. Die resultierende Faltung ergibt sich nicht aus dem Minimum der beiden Funktionen $f(t)$ und $g(t)$, sondern aus dem Minimum der Werte, die mit den Raten der beiden Funktionen erreichbar sind.

5.2.5 Arrival- und Service-Curve

Die Arrival- und Service-Curve sind neben dem *virtual dealy* die beiden entscheidenden Komponenten des Network Calculus, um die Worst-Case-Latenz eines Datums abzuschätzen. Für ein definiertes System (s. Abb. 5.1) begrenzen sie den Eingangsstrom nach oben und den Ausgangsstrom nach unten. Der *virtual delay* dieser beiden Kurven drückt zu jedem Zeitpunkt t die Worst-Case-Verzögerung des zu diesem Zeitpunkt angekommenen Datums aus.

Arrival-Curve

Bevor die Service-Curve anhand der oben beschriebenen Min-Plus-Faltung genau definiert wird, folgt eine Beschreibung der Arrival-Curve. Diese modelliert den maximal möglichen Eingang an Daten in einem System. Alle eingehenden Daten, die vor Eintreffen des analysierten Datums eintreffen, müssen vor diesem versendet werden. Die Arrival-Curve wird mit einer Rate beschrieben, die in einem Ethernet-Netzwerk ausdrücken kann, mit welcher Bandbreite die ankommenden Pakete in den Knoten/Switch strömen. Eine Arrival-Curve $\alpha(t)$ muss folgender Definition entsprechen:

Definition 3 (Arrival-Curve) *Für eine monoton steigende Funktion $\alpha(t)$, definiert für $t \geq 0$, ist ein Flow R dann und nur dann eingeschränkt von $\alpha(t)$ wenn für alle $s \leq t : R(t) - R(s) \leq \alpha(t - s)$. In diesem Fall ist $\alpha(t)$ eine Arrival-Curve für R oder R ist α-smooth.*

Ein Beispiel für eine Arrival-Curve $\alpha(t)$, die für $R(t)$ gilt, wird in Abb. 5.3 veranschaulicht. Für die Erfüllung von Definition 3 genügt es nicht, wenn $R(t)$ stets unter $\alpha(t)$ liegt. Es muss gewährleistet sein, dass die Rate von $R(t)$ in keinem Intervall $[0, t]$ über der Rate von $\alpha(t)$ liegt. In Abb. 5.3 wird dies an der Anlegung des Ursprungs von $\alpha(t)$ an $R(t)$ zu mehreren Zeitpunkten überprüft. Liegt $R(t)$ nie oberhalb dieser „Kurvenschar“, ist die Definition einer Arrival-Curve für $\alpha(t)$ erfüllt.

Eine Arrival-Curve in einem Ethernet-Netzwerk kann durch die möglichen Bandbreiten eingehenden Verkehrs bestimmt werden. Hat ein Switch vier Links (Verbindungen zu Knoten

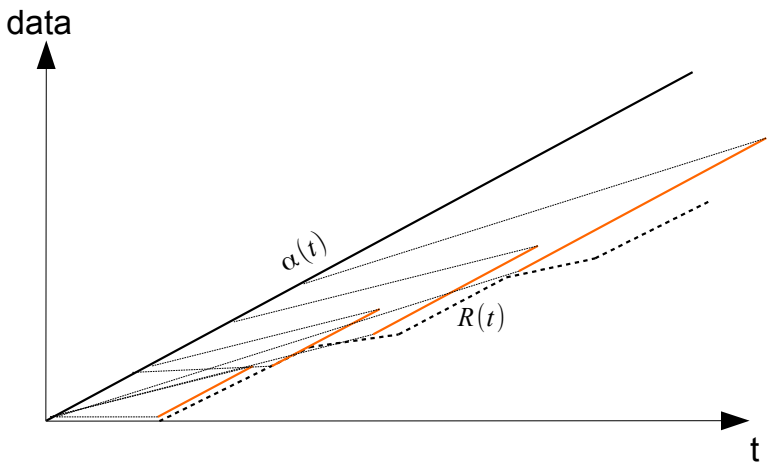

Abbildung 5.3: Network Calculus: Arrival-Curve $\alpha(t)$ und Input-Funktion $R(t)$

oder anderen Switchen) und wird die Analyse auf einem der Links (bzw. der Ausgangs-Queue des Switches, die auf diesen Link führt) ausgeführt, dann können die drei anderen Links alle eingehende Pakete liefern. Bei einer einheitlichen Link-Bandbreite von 100Mbps bedeutete dies eine Arrival-Curve mit einer Rate von 300Mbps. Das ist aber ein vereinfachtes Modell der Arrival-Curve. Für AVB und TSN wird ein Modell für eine Arrival-Curve in Kapitel 6 gegeben.

Service-Curve

Die Service-Curve soll ausdrücken, welcher Service (Dienst) von dem System (der Ausgangs-Queue) garantiert werden kann. Mit einem Rückblick auf Traffic-Shaper (s. dazu Abschnitt 2.6 im Grundlagen-Kapitel) ist die Vorstellung einer solchen „Minimal-Garantie“ möglich. Ein Standard-Ethernet-Paket wird, gemäß dem FIFO-Prinzip, sobald alle vorherigen Daten in der Ausgangs-Queue versendet worden sind, zum Zug kommen und den Link hinter der Ausgangs-Queue beanspruchen. Dies entspräche einem garantierten Service von 100%. Für ein AVB-Paket gelten andere Regeln. Auch wenn es an erster Stelle in seiner Ausgangs-Queue steht, ist eine Sendeerlaubnis nicht garantiert. Gemäß dem Fall, dass ein Standard-konformes AVB-Netzwerk mit einer maximal 75%-igen Reservierung durch AVB-Streams konfiguriert ist, liegt der garantierte Service für einen AVB-Stream bei höchstens 75%.

Wie in Abschnitt 5.2.4 angekündigt, ist der garantierte Service nicht nur von Shapern abhängig, sondern wird durch den Eingangsstrom $R(t)$ beeinflusst. Fällt beispielsweise der Eingangsstrom zu einem beliebigen Zeitpunkt unter den garantierten Service, kann die mini-

male Garantie nicht eingehalten werden. Aus diesem Grund muss mithilfe der Min-Plus-Faltung eine Faltung der Funktionen $R(t)$ und der minimalen Service-Garantie $\beta(t)$ vorgenommen werden. Eine Service-Curve entspricht der folgenden Definition:

Definition 4 (Service-Curve) *Man nehme ein System S mit einem Flow s durch S mit den Input- bzw. Output-Funktionen $R(t)$ und $R^*(t)$. $\beta(t)$ ist für den Flow s eine Service-Curve, dann und nur genau dann, wenn $\beta(t)$ monoton steigend, $\beta(0) = 0$ und $R^*(t) \geq (R \otimes \beta)(t)$ ist.*

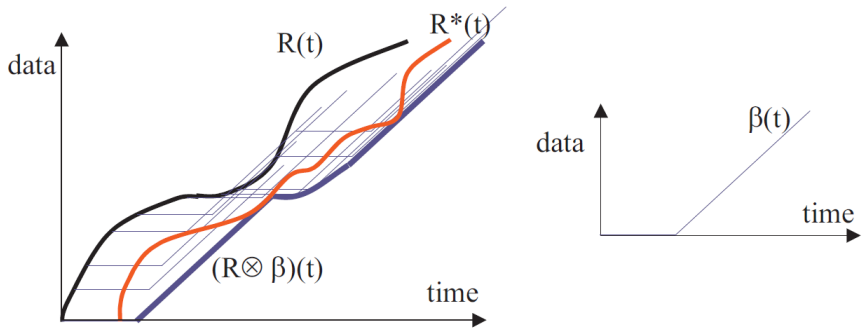

Abbildung 5.4: Network Calculus: Service-Curve [22]

Mit einer gegebenen Eingangsfunktion $R(t)$ und der Service-Curve $\beta(t)$ für den minimalen Service ergibt sich nach der Min-Plus-Faltung eine Kurve $(R \otimes \beta)(t)$, die $\beta(t)$ der Definition 4 entsprechen lässt. Ein Beispiel dieser Faltung wird in Abb. 5.4 gezeigt. Die Worst-Case-Verzögerung eines Datums ergibt sich durch den maximalen $virtual\ delay$ von Arrival- und Service-Curve. In den formalen Analyse-Modellen werden Arrival- und Service-Curve für AVB und TSN entworfen und anhand des $virtual\ delay$s Rückschlüsse auf die Worst-Case-Latenzen gezogen.

Das folgende Kapitel erklärt, wie die Arrival- und Service-Curve für AVB bzw. TSN modelliert werden müssen. Grundsätzlich gilt bei der Betrachtung von Ethernet 802.1Q-Netzwerken, dass die Arrival-Curve den Verkehr umfasst, der in der gleichen Paketklasse wie das untersuche Paket ist. Die Service-Curve hingegen drückt nicht nur die Beschränkung durch Traffic-Shaper aus, sondern modelliert zusätzlich statische Verzögerungen wie beispielsweise maximal große Best-Effort-Pakete.

6 Network-Calculus-Modelle

Dieses Kapitel wird den Network Calculus (s. Abschnitt 5.2) auf die Anwendung mit AVB und TSN konkretisieren. Das AVB-Modell ist im Vorfeld dieser Arbeit entwickelt worden [14, 15] und dient als Basis für das TSN-Modell. Untersuchte Paketklassen der Standards sind unter AVB die Pakete der Klasse A, unter TSN die höchste Prioritätsklasse unterhalb der neu eingeführten TSN-Pakete (die ebenfalls als AVB-Pakete bezeichnet werden). Da TSN-Pakete nach dem TAS-Ansatz modelliert werden, werden sie per TDMA versendet. Sind alle Knoten des Netzwerkes hinreichend synchronisiert, führt diese Vorgehensweise dazu, dass TSN-Pakete keiner paketklassenfremden Verzögerung unterliegen. Das Network-Calculus-Modell für TSN wird daher in Bezug auf TSN-Pakete mit der Annahme entworfen, dass diese mit einer statischen Latenz versehen sind. Im Gegensatz dazu stehen die Pakete der zweithöchsten Prioritätsklasse, also „AVB-Pakete“, die maßgeblich von den TSN-Paketen beeinflusst werden und deshalb der Fokus des Network-Calculus-Modells sind.

6.1 AVB

Georges et al. [23] legten mit ihrer Arbeit den Grundstein für die Betrachtung von „Switched Networks“ (Ethernet) in der Network-Calculus-Analyse. Daraus entwickelte Rene Queck [24] ein konkretes Modell für AVB. Dieses Modell wurde so erweitert bzw. korrigiert, dass allen Worst-Case-Szenarien (vgl. Kapitel 4) Rechnung getragen wird [14,15]. Die Erweiterung, die den größten Einfluss auf die Latenz hat, ist die Berücksichtigung doppelter AVB-Paketsendungen (s. Abschnitt 4.2.3).

6.1.1 NC-System unter AVB

Das in Abb. 6.1 veranschaulichte Network-Calculus(NC)-System eines AVB-Netzwerkes ist die AVB-Auslegung des in Abb. 5.1 vorgestellten allgemeinen NC-Systems. Es finden sich die Eingangsfunktion $R(t)$ und Ausgangsfunktion $R^*(t)$ wieder. Zusätzlich wird ein initialer Delay T_{tcl} definiert, der abhängig von der Paketklasse tcl eine Verzögerung aus anderen Paketklassen beschreibt, die im Worst-Case-Szenario auftritt.

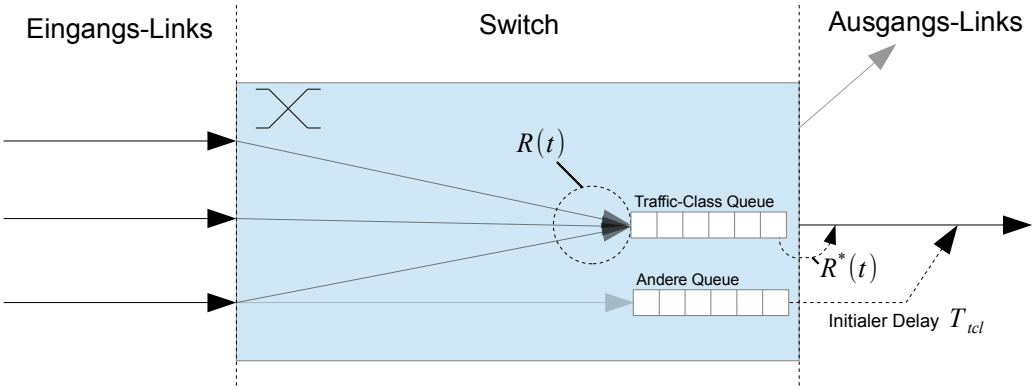

Abbildung 6.1: NC-System für ein AVB-Netzwerk

Um den Bezug zu dem untersuchten Paket zu betonen, wird nachfolgend mit Paket x referenziert. Paket x ist das Paket, das in der NC-Analyse in Bezug auf Worst-Case-Latenzen untersucht wird. Abschätzungen des Worst-Case beziehen sich immer auf Paket x.

Aus Abb. 6.1 ist ersichtlich, dass $R(t)$ und $R^*(t)$ aus den Paketen gespeist werden, die der gleichen Paketklasse wie Paket x angehören. In diesem NC-Modell ist Paket x ein Paket eines AVB-Klasse-A-Streams. Für die Zugehörigkeit zu $R(t)$ und $R^*(t)$ ist entscheidend, dass die Pakete über den gleichen Ausgangs-Link wie x verschickt werden. Sie befinden sich vor ihrer Versendung in der gleichen Ausgangs-Queue wie x. Vereinfacht ließe sich die Zugehörigkeit eines Paketes zu diesen Strömen an der Benutzung der gleichen Route wie x, mit Ausnahme des Eintreffens auf verschiedenen Eingangs-Links, innerhalb des Switches festmachen.

Obwohl die gleichen Pakete über die beiden Ströme laufen, ist eine Differenzierung von $R(t)$ und $R^*(t)$ notwendig. $R(t)$ beschreibt alle ankommenden Pakete. Diese können parallel eintreffen. $R^*(t)$ ist der „Flaschenhals“, der für eine schrittweise Abarbeitung der Ausgangs-Queue steht und zu jeder Zeit nur die Übertragung eines einzelnen Pakets erlaubt. Dieses Verhältnis der beiden Funktionen erlaubt eine Verzögerungs-Analyse für darin enthaltene Daten. Sollten über $R(t)$ so viele Daten in die Ausgangs-Queue übertragen werden, dass $R^*(t)$ diese nicht im gleichen Zeitintervall abarbeiten kann, entsteht eine Verzögerung.

Zusätzlich zu der beschränkenden Weiterleitung der AVB-Klasse-A-Pakete durch $R^*(t)$ werden Verzögerungen durch andere Paketklassen durch den initialen Delay T_{tcl}, in diesem Fall $tcl = A$ mit A für AVB-Klasse-A, modelliert. T_A kann aus den Paketen der prioritär darunterliegenden Paketklassen AVB-Klasse-B und Best-Effort (BE) bestehen. T_A tritt nicht in der

Ausgangs-Queue von x auf, beansprucht aber den gleichen Ausgangs-Link. Aus $R^*(t)$ und T_A wird sich die Service-Curve für AVB-Klasse-A-Pakete bilden.

Um das Worst-Case-Szenario im NC abzubilden, werden die ankommenden AVB-A-Pakete in $R(t)$ so geplant, dass sie gleichzeitig am Switch ankommen und alle vor Paket x in der Ausgangs-Queue stehen. Diese Anzahl an Paketen, die vor x an der Reihe sind, werden in der Summe ihrer Paketgrößen als „Burst“ zusammengefasst. Ein Burst σ_{tcl} wird spezifisch für eine Paketklasse angegeben und gibt den schlimmsten zu erwartenden Burst für diese Paketklasse an. Der Burst und die reservierte Bandbreite des Streams von Paket x sorgen für die Modellierung der Arrival-Curve für AVB-Klasse-A-Pakete.

Zu beachten ist in Bezug auf die Formulierung der Service-Curve, dass sich der Wert für σ_{tcl} aus allen anderen AVB-Paketen zusammensetzt, deren Streams eine erfolgreiche Reservierung für die Ausgangs-Queue von x haben. Das in Kapitel 4 erläuterte Auftreten doppelter AVB-A-Pakete an einem eingehenden Link wird von σ_{tcl} nicht abgedeckt, da σ_{tcl} die Datenmenge beschreibt, die sich aus der einzelnen Versendung der Burst-Pakete zusammensetzt. Folglich muss der doppelte Burst als $2\sigma_{tcl}$ modelliert werden (siehe Abschnitt 6.1.3).

6.1.2 Parameter

Mit diesen Parametern wird in den folgenden Abschnitten gearbeitet:

- T_{tcl}[s] — Initialer Delay für eine Traffic-Class tcl
- σ_{tcl}[bits] — Burst — Alle anderen Pakete erfolgreicher Stream-Reservierungen in tcl
- ρ_{tcl}[bit/s] — Die Bandbreite des betrachteten Streams (von Paket x)
- C[bit/s] — Die Bandbreite eines Links (i.d.R. 100 Mbit/s)
- R_{tcl}[bit/s] — Die reservierte AVB-Bandbreite ($idleSlope_{tcl}$)
- ω_{tcl}[bits] — Das größte Paket des Bursts inklusive Overhead und Sicherungsbyte

Der letztgenannte Parameter ω_{tcl} ist für die Berücksichtigung der in Kapitel 4 vorgestellten doppelten AVB-Pakete nötig. Aus diesen Parametern lassen sich Arrival- und Service-Curve für AVB-Netzwerke entwerfen.

6.1.3 Arrival- und Service-Curve

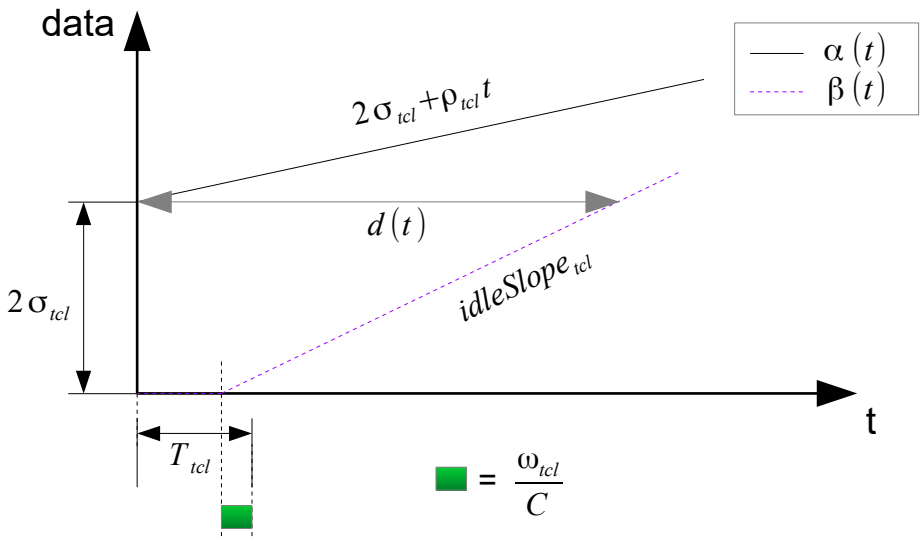

Abbildung 6.2: Arrival- und Service-Curve im AVB-NC-Modell

Mit den oben angegebenen Parametern lassen sich die Arrival- und Service-Curve wie in Abb. 6.2 modellieren. Die Arrival-Curve $\alpha(t)$ und Service-Curve $\beta(t)$ beschreiben die folgenden Funktionen:

$$\alpha(t) = 2\sigma_{tcl} + \rho_{tcl}t \tag{6.1}$$

$$\beta(t) = R_{tcl}\left[t - T_{tcl} + \frac{\omega_{tcl}}{C}\right]^{+} \tag{6.2}$$

Die Arrival-Curve beschreibt für alle Werte $t > 0$ die feste Rate ρ_{tcl}. Diese entspricht der reservierten Bandbreite des AVB-Streams von Paket x. Die Modellierung der Arrival-Curve setzt das Worst-Case-Szenario voraus, in dem alle anderen Pakete der gleichen Paketklasse von x zum Zeitpunkt $t = 0$ bereits in der Ausgangs-Queue vorhanden sind. Das erklärt die Verschiebung des Ursprungs von $\alpha(t)$ auf dem Ordinatenabschnitt um $2\sigma_{tcl}$, da der Burst σ_{tcl} diese anderen Pakete enthält. Aufgrund des, in Kapitel 4 erläuterten, möglichen doppelten Auftretens von Burst-Paketen wird der Burst mit dem Faktor 2 berücksichtigt.

Die Service-Curve wird mithilfe des $idleSlope_{tcl} = R_{tcl}$ mit $tcl = A$ entworfen, der die Rate beschreibt, mit der die AVB-Klasse-A-Ausgangs-Queue ihre Pakete versenden kann. Diese wird durch die gesamte reservierte Bandbreite aller AVB-A-Streams bestimmt. Die Verschiebung auf der x-Achse im Ursprung der Service-Curve entsteht aufgrund statischer Worst-Case-Szenario-Verzögerungen durch Pakete anderer Paketklassen, abgebildet durch den Parameter T_{tcl}. Die Zeitspanne $\frac{\omega_{tcl}}{C}$ reduziert diese Verschiebung, um zu berücksichtigen, dass nicht alle Pakete des Bursts doppelt in der Verzögerung auftauchen.

Der *virtual delay* zum Zeitpunkt $t = 0$ wird aufgrund der Eigenschaft $\rho_{tcl} \leq R_{tcl}$ in diesem System immer der größten Verzögerung des Pakets x entsprechen. Im folgenden Abschnitt wird der Maximalwert von $d(t)$ unabhängig vom Wert von t ermittelt.

6.1.4 Worst-Case-Verzögerung

Um mit der Arrival- und Service-Curve eine Aussage über die Worst-Case-Verzögerung eines Datums zu einem Zeitpunk $t \geq 0$ treffen zu können, muss der maximale *virtual delay* der beiden Kurven (vgl. $d(t)$ in Abb. 6.2) ermittelt werden. Das 3-Grenzen-Theorem [30] gibt über den *virtual delay* anhand von Arrival- und Service-Curve folgendes vor:

$$d(t) < \overline{D_{tcl}} = \sup_{t \geq 0} \Big\{ \inf_{\epsilon \geq 0} \{\alpha(t) \leq \beta(t+\epsilon)\} \Big\} \tag{6.3}$$

$\overline{D_{tcl}}$ beschreibt das Supremum für alle Werte, die $d(t)$ annehmen kann. Werden $\alpha(t)$ und $\beta(t)$ in Gleichung 6.3 eingesetzt, ergibt sich folgende Berechnung für $\overline{D_{tcl}}$, die in [15] bereits vorgestellt worden ist:

$$\begin{aligned} d(t) < \overline{D_{tcl}} &= \sup_{t \geq 0} \Big\{ \inf_{\epsilon \geq 0} \{\alpha(t) \leq \beta(t+\epsilon)\} \Big\} \\ &= \sup_{t \geq 0} \Big\{ \inf_{\epsilon \geq 0} \{2\sigma_{tcl} + \rho_{tcl} t \leq R_{tcl}[t + \epsilon - T_{tcl} + \frac{\omega_{tcl}}{C}]^{+}\} \Big\} \end{aligned} \tag{6.4}$$

Da $t \geq 0$, muss gelten $\{2\sigma_{tcl} + \rho_{tcl} t\} \geq 0$. Deshalb gilt $R[t + \epsilon - T_{tcl} + \frac{\omega_{tcl}}{C}]^{+} = R(t + \epsilon - T_{tcl} + \frac{\omega_{tcl}}{C})$. Mit dieser Festlegung und nach Übernahme der Ergebnisse aus obiger Rechnung:

$$\begin{aligned} \overline{D_{tcl}} &= \sup_{t \geq 0} \Big\{ \inf_{\epsilon \geq 0} \{ \ (\rho_{tcl} - R_{tcl})t + 2\sigma_{tcl} + R_{tcl}T_{tcl} - \frac{\omega_{tcl}}{C} R_{tcl} \leq R_{tcl}\epsilon\} \Big\} \\ &= \sup_{t \geq 0} \Big\{ \frac{(\rho_{tcl} - R_{tcl})t + 2\sigma_{tcl} + R_{tcl}T_{tcl} - \frac{\omega_{tcl}}{C} R_{tcl}}{R_{tcl}} \Big\} \end{aligned} \tag{6.5}$$

Aufgrund der Voraussetzung, dass ρ_{tcl} nur einen Teil der Bandbreite von R_{tcl} reservieren kann, gilt $\rho_{tcl} \leq R_{tcl}$. Nach Übernahme:

$$\begin{aligned} \overline{D_{tcl}} &= \sup_{t \geq 0} \left\{ \frac{(\rho_{tcl} - R_{tcl})t}{R_{tcl}} \right\} + \frac{2\sigma_{tcl} + R_{tcl}T_{tcl} - \frac{\omega_{tcl}}{C}R_{tcl}}{R_{tcl}} \\ &= \frac{2\sigma_{tcl} + R_{tcl}T_{tcl} - \frac{\omega_{tcl}}{C}R_{tcl}}{R_{tcl}} \end{aligned} \tag{6.6}$$

Anhand des maximalen *virtual delay* für $\alpha(t)$ und $\beta(t)$ ergibt sich für die Abschätzung der maximalen Verzögerung des Pakets x innerhalb eines Systems folgender Wert für $\overline{D_{tcl}}$:

$$\overline{D_{tcl}} = T_{tcl} + \frac{2\sigma_{tcl}}{R_{tcl}} - \frac{\omega_{tcl}}{C} \tag{6.7}$$

Gleichung 6.7 ermöglicht eine Abschätzung der Worst-Case-Verzögerung von Paket x innerhalb eines AVB-Netzwerk-Knotens. Sind sämtliche Annahmen des in Kapitel 4 entworfenen Worst-Case-Modells für AVB-Netzwerke korrekt, kann die Verzögerung von Paket x an einer so parametrisierten Station niemals über diesem Wert liegen.

6.2 TSN

Das Network-Calculus-Modell, das im vorigen Abschnitt für AVB-Netzwerke entwickelt wurde, wird als Basis für die Modellierung des TSN-Modells verwendet. Das zu untersuchende Paket x ist einer TSN-Paketklasse zuzuordnen, die einem AVB-Paket im AVB-Protokoll entspricht. Paket x ist also kein TDMA-Paket, sondern steht prioritär an zweiter Stelle.

Die Aufstellung des Modells beruht auf der in Abschnitt 4.3 vorgenommenen Worst-Case-Modellierung für Pakete in TSN-Netzwerken. Das Network-Calculus-System entspricht einem Netzwerk-Knoten. Am Beispiel eines Switches wurde dieses System bereits in Abb. 6.1 für AVB-Netzwerke beschrieben. Zusätzlich zu dem in Abschnitt 6.1 beschriebenen AVB-Modell ergeben sich, basierend auf den Annahmen aus Abschnitt 4.3, Vorbedingungen für den Entwurf des Modells:

- Einzelne TSN-Pakete senden innerhalb eines geöffneten TSN-Gates. Während ein TSN-Gate geöffnet ist, ist das allgemeine Gate geschlossen und es können keine Nicht-TSN-Pakete senden. Um den blockierenden Effekt von TSN-Paketen möglichst einfach zu

modellieren, steht in der nachfolgenden Betrachtung ein TSN-Paket/TDMA-Paket synonym für ein geöffnetes TSN-Gate.

- TSN-Pakete/TDMA-Pakete werden geplant versendet. Ihre Sendeslots sind fest. Der Sendeplan wiederholt sich in zeitlichen Intervallen, jede Periode ist ein TSN-Cycle. Für dieses Modell wird angenommen, dass alle Nicht-TDMA-Pakete, das schließt alle Pakete, die Paket x verzögern, und Paket x ein, innerhalb eines TSN-Cycles die Möglichkeit zur Versendung erhalten.
- Ist eine Lücke zwischen zwei TDMA-Paketen so klein, dass kein Paket einer anderen Klasse die Möglichkeit hat, seine Sendung durchzuführen, werden diese beiden TDMA-Pakete zu einem TDMA-Paket zusammengefasst. Das gewährleistet, dass alle Lücken groß genug sind, um mindestens ein Paket einer anderen Klasse zu versenden. Die nun vorhandenen Lücken müssen groß genug sein, um allen AVB-Paketen die Sendung zu ermöglichen. Auf diese Weise wird Cycle Piling unterbunden.
- Die Bandbreiten und Größen der AVB-Pakete, also Paket x und die Burst-Pakete, bleiben für die Worst-Case-Betrachtung unverändert. Das bedeutet, dass die Bandbreiten der Pakete nicht angepasst werden, um ein Sende-Szenario zu ermöglichen, das eine erhöhte Verzögerung für Paket x zur Folge hat.
- Die Größe des blockierenden BE-Pakets (im NC-Modell als T_{tcl} bzw. T_A modelliert) richtet sich nach dem Sendeablauf der TSN-Pakete. Nach Möglichkeit wird ein BE-MTU-Paket verwendet, das Paket wird aber nie größer sein als die größte Lücke zwischen TSN-Paketen. So wird gewährleistet, dass das BE-Paket senden kann.
- Die Bandbreitenaufteilung und mögliche AVB-Bandbbreitenreservierung richtet sich nach der Bandbreitenbeanspruchung des TDMA-Verkehrs. Das Verhältnis zwischen AVB- und prioritär niedrigeren Paketen (75% zu 25%) bleibt erhalten, deren summierte Bandbreite übersteigt aber nicht die verbleibende Bandbreite, die nicht durch TDMA-Pakete beansprucht wird. Machen beispielsweise TDMA-Pakete insgesamt 5% der Bandbreite aus, können AVB-Streams nicht mehr als 75% der verbleibenden 95% reservieren.

6.2.1 Begrenzungen und Einschränkungen des Modells

Je nach Konfiguration des TDMA-Verkehrs kann sich eine „komplizierte“ Berechnung/Abschätzung der Worst-Case-Latenz ergeben. Innerhalb eines TSN-Cycles gibt es die Möglichkeit, dass TDMA-Pakete nicht hintereinander gesendet werden, sondern in mehrere kleine Blöcke segmentiert werden. Ist das der Fall, muss auf die zuvor geführte Diskussion der Vorgehensweise

bei verschiedenen Lückengrößen Rücksicht genommen werden.

Die Lücken, die zwischen den TDMA-Paketen entstehen, können in verschiedene Größenordnungen eingeteilt werden. Eine Lücke muss, laut Vorbedingung, mindestens so viel Platz bieten, dass jedes der AVB-Pakete Platz zum Senden hat. Diese Lücken können auch so groß sein, dass alle (doppelten) Burst-Pakete und ein BE-Paket, dem Worst-Case-Szenario eines AVB-Netzwerkes entsprechend, in ihnen Platz finden.

Die Variante, die am schwersten zu berechnen ist, ist eine Mischung aus beiden Lückentypen. Wie bereits in Abschnitt 4.3.2 ausgeführt, kann ein solches Szenario dazu führen, dass alle Burst-Pakete und das BE-Paket in einer Lücke untergebracht werden. Dieser Fall führt allerdings nicht zwangsläufig zu der größten Verzögerung von Paket x. Um in so einem Szenario den Worst-Case zu finden, muss folgendermaßen vorgegangen werden:

1. Suche die nächste Lücke des TDMA-Schedules aus
2. Platziere den Sendeablauf aus Burst-Paketen und BE-Paket in dieser Lücke
3. Reicht die Lücke nicht für die vollständige Versendung aller Burst-Pakete:
   - Passt das BE-Paket in die nachfolgende Lücke und kann es dort gesendet werden (Kredit niedrig genug)?
     - Wenn ja: Verschiebe die Versendung von BE-Paket auf die nächste Lücke
     - Wenn nein: Sende BE-Paket noch vor Ende der aktuellen Lücke
   - Betrachte die nachfolgenden Lücken und verteile die Versendung der verbleibenden Burst-Pakete und (eventuell) des BE-Pakets auf diese. Reicht die nächste Lücke wieder nicht, wiederhole diesen Unterablauf
4. Notiere die insgesamt benötigte Zeit, bis Paket x senden kann
5. Ist diese Zeit höher als vorher notierte Werte, notiere sie als Worst-Case-Latenz
6. Beginne mit Schritt 1, wenn die nachfolgende Lücke noch nicht betrachtet worden ist

In einem solchen Szenario lässt sich die Abschätzung der Worst-Case-Latenz nicht einzig an den Parametern festmachen. Es müssen mehrere Startpunkte miteinander verglichen werden und das Maximum der Ergebnisse ermittelt werden, um den schlechtesten Fall zu finden. Im Vergleich zum NC-Modell für AVB-Netzwerke bedeutet das, dass Arrival- und Service-Curve

nicht eindeutig modelliert werden können, sondern ein Vergleich verschiedener Modellierungen angestellt werden muss. Folgende Parameter müssten für ein solches Szenario eingegeben werden:

- TT[bits] — Die gesamte Datenmenge an TDMA-Paketen, die während der Versendung aller Burst-Pakete und des BE-Pakets versendet wird. Zusätzlich umfasst TT alle Lücken vor den TDMA-Paketen, die nicht zur Versendung von Paketen verwendet werden kann.
- γ_i[bits] — Die Größe der Lücke i. Für alle Lücken gibt es einen γ_i-Wert, so dass die Lücken ihrer Reihenfolge im TSN-Cycle entsprechend durchnummeriert sind.
- ϕ_{tcl}[bits] — Teil des Bursts, der nicht in der Start-Lücke gesendet werden kann (abhängig von γ_i und σ_{tcl})
- $\psi_{tcl} = \left[\phi_{tcl} - \frac{(T_{tcl}+\frac{TT}{C})\cdot R_{tcl}\cdot C}{C-R_{tcl}}\right]^+$ — Hilfsgröße. Nicht negative Anzahl an Bits, die nicht in der ersten Lücke versendet werden kann, aber auf negativen Kredit warten muss.

Mithilfe dieser Parameter kann, zusätzlich zu den für das AVB-NC-Modell verwendeten Parametern, eine Abschätzung der Worst-Case-Latenzen in Bezug auf die Start-Lücke vorgenommen werden. ϕ_{tcl} und ψ_{tcl} werden verwendet, um Kreditschwankungen zu modellieren, die nach der Start-Lücke auftreten können. Ohne eine Arrival- und Service-Curve-Modellierung zu zeigen, sieht die Worst-Case-Latenz von Paket x wie in Formel 6.8 aus, wenn die Burst-Pakete und das BE-Paket mit ihrer Übertragung in der Lücke beginnen, die für die Größe von ϕ_{tcl} sorgt:

$$\overline{D_{tcl}} = T_{tcl} + \frac{2\sigma_{tcl}}{R_{tcl}} - \frac{\omega_{tcl}}{C} + \frac{TT}{C} + \frac{phi_{tcl} - \psi_{tcl}}{C} + \frac{\psi_{tcl}}{R_{tcl}} \tag{6.8}$$

Der Parameter T_{tcl} kann hier abhängig von Lückengrößen so variiert werden, dass er noch gesendet werden kann. Für verschiedene Größen von T_{tcl} wären Vergleiche denkbar. Ein Modell, das mit diesem Ansatz die Worst-Case-Abschätzung eines TSN-Netzwerkes ausführt, ist sehr auf die tatsächliche Konfiguration des Netzwerks ausgerichtet und bietet eine spezialisierte Betrachtung der Worst-Case-Szenarien.

In TSN-Netzwerken ist eine Konfiguration, die ein solches Vorgehen motivieren würde, für Netzwerke aus dem Automotive-Bereich selten anzutreffen. Da die TDMA-Pakete den Zweck erfüllen sollen, sicherheitsrelevante Informationen zu transportieren, die meist mit wenigen Daten vermittelt werden können, beträgt die beanspruchte Bandbreite von TDMA-Paketen

in TSN-Netzwerken meistens einen niedrigen bis mittleren einstelligen Prozent-Bereich der gesamten Bandbreite. Ein Szenario, in dem die Versendung der Burst-Pakete und des BE-Pakets in zwei oder mehr Lücken aufgeteilt werden muss, tritt so nur selten auf.

Die spezialisierte Abschätzung mit Iteration über alle Start-Lücken kann in Anbetracht der Tatsache, dass dieser spezialisierte Fall nicht oft auftritt, durch eine Abschätzung ersetzt werden, die die Lückenproblematik für jedes Netzwerk angleicht und ein iteratives Vorgehen überflüssig macht. Die Annahme soll sein, dass innerhalb des TSN-Cycles nur ein einziger TSN-Block existiert, der alle TDMA-Pakete beinhaltet. Das bedeutet, dass nur eine Lücke zwischen den TDMA-Paketen existiert.

Die Beschränkungen und Erleichterungen, die dieses Vorgehen mit sich bringt sind folgende:

- Alle TDMA-Pakete folgen aufeinander. Sie können zu einem großen TDMA-Block zusammengefasst werden.
- Es muss keine Lückengröße betrachtet werden, da die Lücke ausreichend groß für die Versendung aller Burst-Pakete und des BE-Pakets sind
- Alle Fälle der spezialisierten Betrachtung sind abgedeckt
- Das BE-Paket variiert in seiner Größe nicht, sondern ist immer ein BE-MTU-Paket

Gegenüber dem spezialisierten Ansatz hat diese Vorgehensweise eine Abschätzung zur Folge, die im Ergebnis deutlich über den Werten der spezialisierten Abschätzung liegen kann. Diese Abschätzung erfüllt aber die Voraussetzung einer Maximalabschätzung, dass die Worst-Case-Latenz-Werte von Paket x niemals über ihrem Wert liegen können. Für den Entwurf des Modells und insbesondere der benötigten Parameter hat diese Einschränkung des Modells eine Auswirkung, die in den folgenden Abschnitten beleuchtet wird.

6.2.2 Parameter

Die Parameter des AVB-Modells bleiben für das TSN-Modell erhalten. Zusätzlich wird ein Parameter für die Modellierung der TDMA-Pakete eingeführt.

- T_{tcl}[s] — Initialer Delay für eine Traffic-Class tcl
- σ_{tcl}[bits] — Burst — Alle anderen Pakete erfolgreicher Stream-Reservierungen in tcl
- ρ_{tcl}[bit/s] — Die Bandbreite des betrachteten Streams (von Paket x)

- C[bit/s] — Die Bandbreite eines Links (i.d.R. 100 Mbit/s)
- R_{tcl}[bit/s] — Die reservierte AVB-Bandbreite ($idleSlope_{tcl}$)
- ω_{tcl}[bits] — Das größte Paket des Bursts inklusive IFG, Tags und Sicherungsbyte
- TT[bits] — Größe des TDMA-Blocks inklusive aller Sendekapazität, die vor dem Start der Sendung des TDMA-Blockes nicht durch andere Pakete belegt werden kann

Die Größe von TT richtet sich nach der Größe des TSN-Cycles. Zu TT zählen alle TDMA-Pakete, die innerhalb eines TSN-Cycles gesendet werden. Gegenüber dem spezialisierten Ansatz reicht hier die Erweiterung des NC-Modells für AVB um einen einzelnen Parameter.

6.2.3 Arrival- und Service Curve

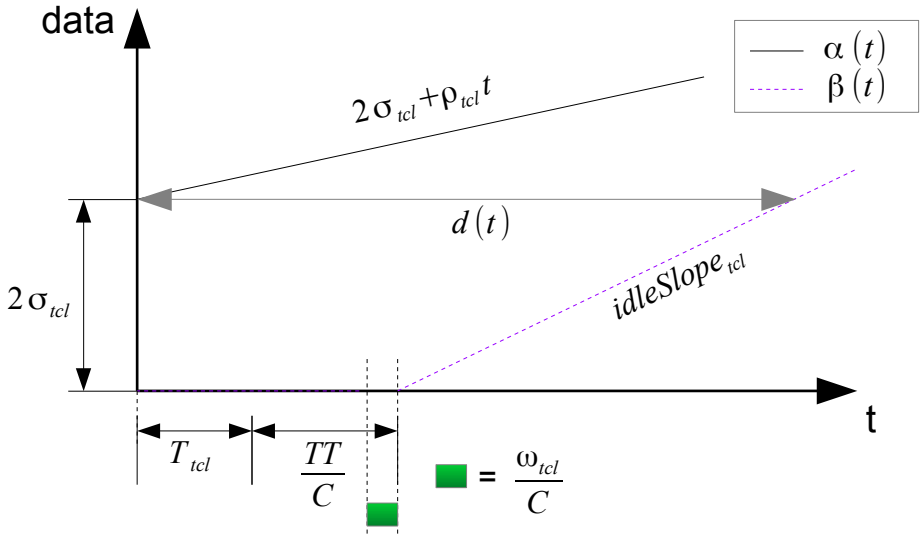

Abbildung 6.3: Arrival- und Service-Curve im TSN-NC-Modell

Das nicht spezialisierte Modell ist eine Erweiterung des vorhandenen AVB-NC-Modells. Die Eigenschaften der Arrival Curve bleiben erhalten, da diese weiterhin nur den eingehenden AVB- bzw. Burst-Traffic für die Ausgangs-Queue von Paket x beinhaltet. Die Arrival Curve $\alpha(t)$ für das TSN-NC-Modell wird folgendermaßen beschrieben:

$$\alpha(t) = 2\sigma_{tcl} + \rho_{tcl}t \tag{6.9}$$

Der hinzugekommene TDMA-Traffic lässt sich, da die Annahme eines einzelnen TDMA-Blocks gilt, an einer einzigen Stelle im Sendeablauf platzieren. Wie in Abschnitt 4.3 erläutert, muss der TDMA-Block für den Worst-Case am Ende des Sendeablaufs auftreten. Er verhält sich wie das BE-Paket, das als T_{tcl} bereits in der AVB-Service-Curve (s. Formel 6.2) modelliert wurde. Die Service Curve $\beta(t)$ für TSN-Netzwerke hat die folgende Form:

$$\beta(t) = R_{tcl}\Big[t - T_{tcl} + \frac{\omega_{tcl} - TT}{C}\Big]^{+} \tag{6.10}$$

Für Arrival- und Service-Curve ergibt sich die in Abb. 6.3 dargestellte Konstellation.

6.2.4 Worst-Case-Verzögerung

Analog zu der Anwendung des 3-Grenzen-Theorems (s. Formel 6.3 in Abschnitt 6.1.4) können Arrival- und Service-Curve des TSN-Modells in eine Worst-Case-Abschätzung des virtual delays überführt werden. Mit einer gekürzten Umrechnungen (Bedingungen dazu finden sich in der AVB-Berechnung für die Formeln 6.4 - 6.6) ergibt sich folgendes:

$$\begin{aligned}
d(t) < \overline{D_{tcl}} &= \underset{t\geq 0}{sup}\Big\{\underset{\epsilon\geq 0}{inf}\{\alpha(t) \leq \beta(t+\epsilon)\}\Big\} \\
&= \underset{t\geq 0}{sup}\Big\{\underset{\epsilon\geq 0}{inf}\{2\sigma_{tcl} + \rho_{tcl}t \leq R_{tcl}[t + \epsilon - T_{tcl} + \frac{\omega_{tcl} - TT}{C}]^{+}\}\Big\} \\
&= \underset{t\geq 0}{sup}\Big\{\frac{(\rho_{tcl} - R_{tcl})t + 2\sigma_{tcl} + R_{tcl}T_{tcl} - \frac{\omega_{tcl}-TT}{C}R_{tcl}}{R_{tcl}}\Big\} \\
&= \underset{t\geq 0}{sup}\Big\{\frac{(\rho_{tcl} - R_{tcl})t + 2\sigma_{tcl} + R_{tcl}T_{tcl} - \frac{\omega_{tcl}}{C}R_{tcl} + \frac{TT}{C}R_{tcl}}{R_{tcl}}\Big\} \\
&= \frac{2\sigma_{tcl} + R_{tcl}T_{tcl} - \frac{\omega_{tcl}}{C}R_{tcl} + \frac{TT}{C}R_{tcl}}{R_{tcl}}
\end{aligned} \tag{6.11}$$

Die Worst-Case-Latenz $\overline{D_{tcl}}$ von Paket x in einem TSN-Netzwerk lässt sich so, unter Einhaltung der Vorbedingungen mit Formel 6.12 abschätzen. Die Latenz von Paket x wird diesen Wert nicht überschreiten.

$$\overline{D_{tcl}} = T_{tcl} + \frac{2\sigma_{tcl}}{R_{tcl}} - \frac{\omega_{tcl}}{C} + \frac{TT}{C} \tag{6.12}$$

Das folgende Kapitel stellt ein Tool vor, das ebenfalls auf den Worst-Case-Modellen basiert.

7 Simulationsgenerierung

Neben den Worst-Case-Modellen und der formalen Analyse ist im Rahmen dieser Arbeit ein Tool entwickelt worden, das eine OMNeT++-Simulation generiert. Diese Simulation kann AVB-, BE- und TSN-Pakete enthalten und wird durch das Tool in eine Form gebracht, die ausgewählte Latenzen des Netzwerks an die Worst-Case-Szenarien aus den zuvor vorgestellten Worst-Case-Analysen und -Modellen annähert.

Das Tool hat eine simple Oberfläche, die die Auswahl einer Quell-Simulation erlaubt. Nach Auswahl eines AVB-Streams und dem Switch, an dem der Worst-Case generiert werden soll, wird eine Simulation erstellt, die die Pakete in der richtigen Reihenfolge für den Worst-Case platziert.

7.1 Funktionen

Dieser Abschnitt gibt einen Überblick über die Funktionen des Tools. Dabei wird zwischen Benutzer-Funktionen und internen Funktionen unterschieden. Benutzer-Funktionen werden im Tool durch die Auswahlmöglichkeiten der Bedienungsoberfläche realisiert. Interne Funktionen laufen im Hintergrund ab und sind nicht nach aussen sichtbar.

7.1.1 Benutzer-Funktionen

Der Benutzer erhält die Möglichkeit, Quell- und Ziel-Verzeichnisse zu setzen und kann auswählen, welcher AVB-Stream an welchem Switch untersucht werden soll.

Quell-Verzeichnis

Über einen Auswahl-Dialog kann das Quellverzeichnis ausgewählt werden. Zulässig sind nur Verzeichnisse, keine Datei-Pfade. Das Quell-Verzeichnis lässt sich auch in die zugehörige Box kopieren, diese ist ausschlaggebend für nachfolgende Gültigkeitsprüfungen. Das Eingabefeld kann durch andere Funktionen gesperrt werden.

Ziel-Verzeichnis

Analog zum Quell-Verzeichnis kann ein Ziel-Verzeichnis ausgewählt werden. Auch dieses Verzeichnis kann gesperrt werden.

Import des Quell-Verzeichnisses

Diese Schaltfläche bewirkt, dass das Quell-Verzeichnis überprüft wird. Enthält dieses Dateien einer Simulation, dann wird das Quell-Verzeichnis-Feld gesperrt. Diese Sperrung führt auch dazu, dass Menüs zur Auswahl des untersuchten Streams und Switches freigeschaltet werden.

Wird ein neues Quell-Verzeichnis gewählt (nur über den Auswahl-Dialog möglich), wird die Validierung aufgehoben, dass Quell-Verzeichnis-Feld entsperrt und die Auswahl von Stream und Switch deaktiviert.

Auswahl von Stream und Switch

Nach erfolgreichem Import des Quell-Verzeichnisses werden Daten aus dem Netzwerk importiert. Die Dropdown-Boxen bieten die Möglichkeit, einen AVB-Stream (nach Stream-ID) und einen Switch (nach Namen) auszuwählen. In der Worst-Case-Generierung wird die Simulation auf den Worst-Case für diesen Stream an dem ausgewählten Switch zugeschnitten.

Validierung der Auswahl

Sind alle Parameter ausgewählt, wird die Konfiguration auf Validität überprüft. Um die Generierung starten zu können, muss dieser Schritt erfolgreich verlaufen. Die Validität setzt, neben der Auswahl eines Streams und eines Switches, voraus, dass ein Ziel-Verzeichnis angegeben worden ist.

Ist die Validierung erfolgreich, werden alle Auswahl-Felder deaktiviert und die Generierung kann gestartet werden. Um die Deaktivierung der Felder zu widerrufen, kann ein neuer Quell-Pfad angegeben werden. Das sorgt dafür, dass Import, Auswahl von Stream und Switch und die Validitäts-Prüfung erneut ausgeführt werden müssen.

Worst-Case-Generierung

Ist die Konfiguration valide, lässt sich die Generierung starten. Diese erstellt aus dem Quell-Verzeichnis ein Netzwerk, überprüft alle Pakete und platziert die Pakete so, dass sie eine möglichst schlechte Latenz für den zuvor ausgewählten AVB-Stream zur Folge haben. Mehr zu diesem Schritt folgt in den internen Funktionen.

7.1.2 Interne Funktionen

Das Tool erstellt aus den importierten Simulations-Dateien eine eigene Repräsentation des Netzwerkes. Diese Struktur enthält Informationen über den Aufbau und die Verbindung aller Komponenten. Zusätzlich werden alle Paketinformationen und die Route der einzelnen Pakete im Netzwerk aufgezeichnet.

Ein- und Auslesen von Dateien

Das Tool hat die Möglichkeit Simulationsdateien einzulesen und zu interpretieren sowie Dateien zu erzeugen, die eine Simulation beschreiben.

Import der Netzwerkdaten

Bevor das Netzwerk in vollem Umfang in der internen Struktur des Tools nachgebaut wird, werden alle enthaltenen AVB-Streams und Switchnamen eingelesen. Dazu wird im Quell-Verzeichnis jede Datei eingelesen und auf Syntaxen geachtet, die AVB-Streams oder Switche auszeichnen.

Konstruktion des Netzwerkes

Ist das Netzwerk valide und wird die Generierung gestartet, wird das Netzwerk aus der Simulation in vollem Umfang nachgebaut. Das beinhaltet eine Liste aus Knoten (Start- bzw. End-Knoten und Switche). Dem Netzwerk entsprechend werden alle Verbindungen zwischen Knoten und Switches nachgebildet. Pro Verbindung wird zudem eine „Queue“ erstellt, die alle Pakete enthält, die über diese Verbindung versendet werden sollen. Diese Queue ist im programmatischen Sinne keine wirkliche Queue, sondern eine Liste der Pakete.

Die Pakete selber werden ebenfalls modelliert. Sie enthalten die Informationen darüber, wie groß sie sind, welchen Start- und Ziel-Knoten sie haben und welche „Queues“ sie passieren.

Erstellung des Worst-Case-Szenarios

Wenn die Konstruktion des Netzwerkes abgeschlossen ist, wird der ausgewählte Stream im Netzwerk lokalisiert. Der, in der Benutzeroberfläche, ausgewählte Switch ist dafür verantwortlich, welche „Queue“ untersucht wird. Alle Pakete, die neben dem ausgewählten Stream ebenfalls über diese Queue laufen (Crosstraffic), werden zusammengefasst. Die Informationen zu den Paketen werden interpretiert und die Dateien der Simulation werden so geändert, dass der Crosstraffic sich so platziert, dass die Pakete des untersuchten Streams eine möglichst hohe

Latenz auf dem untersuchten Switch aufweisen. Näheres zu diesem Schritt wird in Konzept und Umsetzung erläutert.

Generierung der Simulations-Dateien

Ist die Erstellung des Worst-Case-Szenarios abgeschlossen, wird das Worst-Case-Netzwerk in die Form einer Simulation gebracht. Die erzeugten Dateien werden im zuvor ausgewählten Ziel-Verzeichnis gespeichert.

7.2 Konzept

Dieser Abschnitt stellt das technische Konzept zur Entwicklung des Tools vor. Es werden verwendete Technologien und Mechanismen erklärt, die zur Realisierung des Tools verwendet werden.

7.2.1 Programmiersprache und Bibliotheken

Das Tool wird in Java entwickelt. Es wird keine externe Bibliothek verwendet, um die benötigte Infrastruktur so unkompliziert wie möglich zu gestalten. Alle Funktionen des Tools werden über Bibliotheken umgesetzt, die ein Teil der verwendeten Java-Version sind. Die verwendete Version ist Java 8.

7.2.2 GUI

Feld-Funktion	Feld-Typ
Quell-Verzeichnis	Text-Feld
Ziel-Verzeichnis	Text-Feld
Quell-Verzeichnis auswählen	Button
Ziel-Verzeichnis auswählen	Button
Netzwerk-Daten importieren	Button
Stream-Auswahl	Dropdown-Menü
Switch-Auswahl	Dropdown-Menü
Validierung	Button
Status-Anzeige	Label (nicht von Benutzer editierbar)
Generierung des Worst-Case	Button
Fortschritts-Anzeige	Progress-Bar

Tabelle 7.1: Eingabemöglichkeiten der Benutzeroberfläche

Die graphische Oberfläche (GUI) ist die Benutzer-Schnittstelle. Sie bietet alle Funktionen, die in Abschnitt 7.1.1 (Benutzer-Funktionen) beschrieben wurden. In der GUI enthaltene Felder werden in Tabelle 7.1 gezeigt.

7.2.3 Netzwerkkonstruktion (simpel)

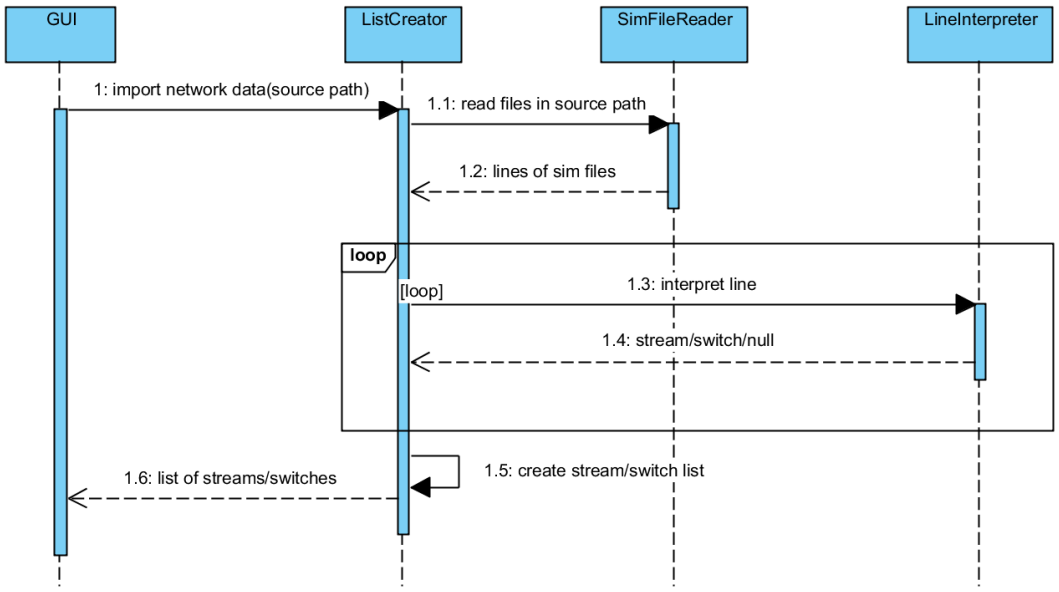

Abbildung 7.1: Einlesen von Streams und Switches

Für die Anzeige der Auswahlmöglichkeiten der Streams und Switche werden die Simulationsdateien eingelesen. Für eine OMNeT++-Simulationen werden Dateien mit den Endungen *.ini* und *.ned* verwendet. Erstgenannte Dateien enthalten Konfigurationen zu einzelnen Paketen, .ned-Dateien decken den Aufbau des Netzwerkes ab. Es werden alle .ini-Dateien des Quell-Verzeichnisses eingelesen, um ein Paket zu identifizieren. Da für die Auswahl auf der Benutzeroberfläche nur AVB-Streams und Switche von Interesse sind, wird nur nach diesen gesucht. Ein AVB-Stream wird identifiziert, wenn ein AVB-Talker, also die Quelle eines AVB-Streams, beschrieben wird. Eine Quelle ist in der Simulation eine Traffic-Source-App. Für AVBStreams ist das entweder eine *AVBTrafficSourceApp* oder eine *AVBTrafficSourceAppStatic*. Letztgenannte bezieht sich auf eine Traffic-Source-App, die den Generierungszeitpunkt eines AVB-Streams statisch festlegen kann.

Zum Ermitteln der Switches innerhalb des Netzwerks werden die .ned-Dateien der Simulation eingelesen. Alle Dateien, die einen Switch beinhalten, enthalten ein Modul, dass von der Klasse

TTEAVBEtherSwitch ableitet. Die beim Einlesen der Simulationsdateien identifizierten Streams und Switche werden in Listen gespeichert, die für die Auswahl auf der Benutzeroberfläche zur Verfügung gestellt werden. Abb. 7.1 stellt den Ablauf beim Import der Simulationsdaten dar.

7.2.4 Netzwerkkonstruktion (vollständig)

Hat der Benutzer den zu untersuchenden Stream und Switch gewählt, erfolgt eine vollständige Konstruktion des Netzwerkes. Die Infrastruktur des Netzwerkes (Knoten und Verbindungen) werden in einem *Network*-Objekt festgehalten. Die Abbildung des Netzwerkes erfolgt über Collections von verschiedenen Objekten, die Knoten, Switche und Pakete innerhalb des Netzwerkes darstellen.

Infrastruktur-Objekte

Das zentrale Objekt der Infrastruktur ist ein *Network*-Objekt. Ein Netzwerk hält eine Map über die „Hardware“ des Netzwerks. Diese Hardware ist vom abstrakten Typ *NetworkDevice*. Es gibt zwei Klassen, die diesen ableiten:

- NetworkNode
- NetworkSwitch

Ein *NetworkNode* entspricht einem Anfangs-/Endknoten und ein *NetworkSwitch* stellt einen Switch dar. Jeder Knoten hat eine Verbindung zu einem Switch. Jeder Switch hat Verbindungen zu Knoten und/oder anderen Switches. Diese Verbindungen werden in einem Switch in einer Liste der verbundenen *NetworkDevice*s vorgehalten. Jedes *NetworkDevice* hat eine Liste vom Typ *OutputQueue*. In einem *OutputQueue*-Objekt wird festgehalten, von welchem *NetworkDevice* die Pakete losgeschickt werden und an welchem *NetworkDevice* sie ankommen. Zusätzlich enthält es eine Liste der Pakete, die über diese Queue versendet werden.

Mit dieser Infrastruktur lässt sich die Peripherie eines Netzwerks darstellen. Es ist von jedem *NetworkDevice* aus möglich, die Verbindungen zu anderen Stationen des Netzwerkes nachzuvollziehen. Enthaltene Pakete in einem *OutputQueue*-Objekt lassen sich über verbundene *NetworkDevice*s in ihrer kompletten Route nachvollziehen.

Paketinformationen

Die Pakete des Netzwerkes werden über die abstrakte Klasse *EthernetPacket* modelliert. Alle Pakete in einer *OutputQueue* sind von diesem Typ. Ableitende Klassen sind:

- AVBStream
- BestEfforPacket
- TSNPacket

Ein *EthernetPacket* hält folgende Informationen:

- Payload-/Overheadgröße
- Gesamtgröße
- Start- und Zielknoten
- Liste aller passierten *OutputQueue*s

Die Liste ist als LinkedList konzipiert und wird so angelegt, dass die Stationen des Pakets im Netzwerk nacheinander eingepflegt werden. Die speziellen Paketarten haben erweiterte Eigenschaften. *AVBStream*-Objekte speichern die ID des zugehörigen Pakets, *BestEffortPacket*s haben einen Namen, damit sie identifiziert werden können und *TSNPacket*-Objekte halten neben einer ID Zusatzinformationen über das Paket, die in der Generierung der Simulationsdateien verwendet werden.

Erstellung des Netzwerks

Ähnlich wie für die simple Netzwerkkonstruktion werden die Simulationsdaten des Netzwerkes eingelesen und Zeile für Zeile überprüft. Im Vergleich zu der simplen Netzwerkkonstruktion ist die Analyse der Dateien tiefgehender, da auch die Eigenschaften der Pakete und Verbindungen zwischen einzelnen *NetworkDevice*s eingelesen werden.

Knoten und Switche werden anhand ihrer Modulnamen in .ned-Dateien identifiziert. Die Verbindungen der einzelnen Knoten und Switche untereinander werden in einer .ned-Datei, die das gesamte Netzwerk beschreibt, definiert. Alle Verbindungen werden aus dieser Datei extrahiert und in das Netzwerk eingepflegt.

Pakete werden an ihrer Syntax in .ini-Dateien erkannt. In der Datei, in der das Paket erkannt wurde, wird nach Parametern des Pakets gesucht. Dazu gehören ID, Ziel-Adresse, und/oder Payloadgröße je nach Paketart. In einer .ini-Datei, die solche Parameter eines Pakets setzt, werden (mit Ausnahme des Best-Effort-Pakets) keine Ziele der Pakete angegeben. Mit den IDs der AVB- und TSN-Pakete muss daher in einem zweiten Einlesen der Dateien der Endknoten der Pakete ermittelt werden.

Das Einlesen der Informationen und das Erstellen des Netzwerks läuft wie in Abb. 7.2 ab.

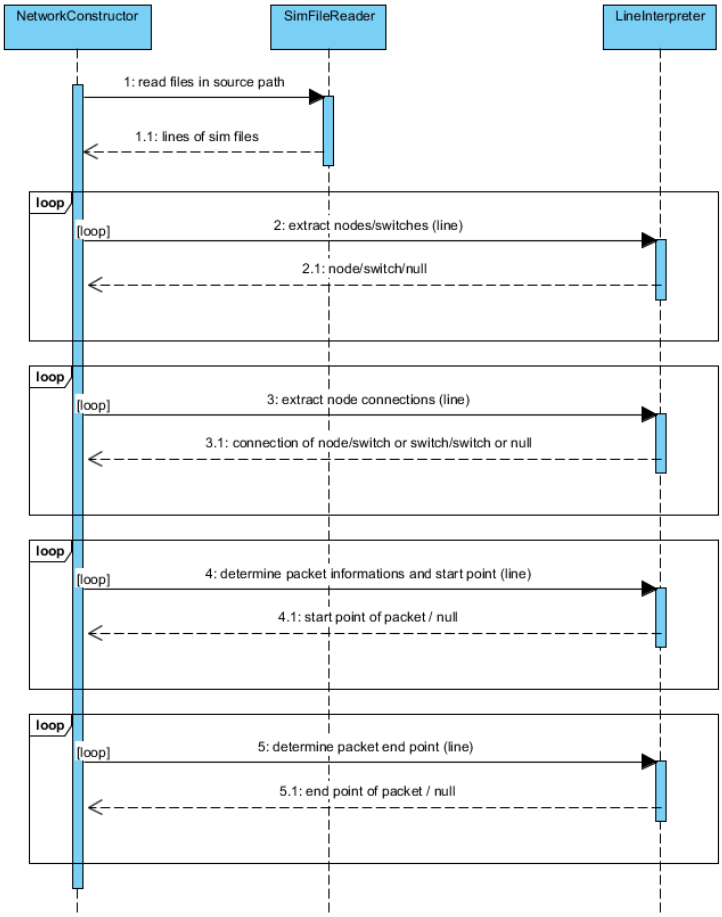

Abbildung 7.2: Einlesen von allen Netzwerkinformationen

7.2.5 Routing der Pakete

Der Aufbau des *Network*-Objekts erfordert, dass die die Routen der Pakete ermittelt werden. Mithilfe des Start- und Endknotens wird diese Route in einem rekursiven Durchsuchen aller angeschlossenen Switche ermittelt. Die Suche läuft nach dem folgenden Schema ab:

1. Starte im Startknoten des Pakets
2. Ermittle den angeschlossenen Switch
3. Ist Endknoten mit diesem Switch verbunden?
   - Nein: Fahre mit Schritt 4 fort
   - Ja: Gebe den Pfad vom Startknoten bis zu der OutputQueue zu dem Knoten zurück

4. Überprüfe alle angeschlossenen Switche
5. Ist dieser Switch bereits überprüft worden?
    - Nein: Fahre mit Schritt 3 fort
    - Ja: Gebe null zurück (falsche Route)

Dieser Vorgang wird ausgeführt, bis Schritt 2 eine Route zurückbekommt, die nicht den Wert *null* hat. Wird nur *null* zurückgegeben, konnte keine Route ermittelt werden. Die so ermittelten passierten Queues des Pakets werden in der LinkedList des Pakets festgehalten. Gleichzeitig wird an jeder *OutputQueue* festgehalten, welche Pakete über diese gesendet werden.

7.2.6 Erkennung Crosstraffic

Anhand der Auswahl in der GUI ist festgelegt, auf welchen Stream und auf welchen Switch der Worst-Case zugeschnitten werden soll. Die Struktur des Netzwerks ermöglicht die Identifizierung aller Pakete, die an der gleichen *OutputQueue* wie das untersuchte Paket gesendet werden. Die Erkennung des Crosstraffics ist trivial, weil eine Liste davon in dem *OutputQueue*-Objekt vorgehalten wird.

7.2.7 Neuplanung des Netzwerkes

Um den Worst-Case zu generieren, muss die Planung des Netzwerks gegenüber der Quell-Simulation geändert werden. Das bedeutet, dass alle Pakete zu festgelegten Zeitpunkten generiert werden und Pakete, die nicht zum Crosstraffic gehören, nicht geplant werden. Eine statische Generierung der Pakete kann nur durch eine Anpassung der TrafficSource-Apps der Simulation erreicht werden. Dieser Teil wird in dem CoRE4INET-Framework unter OMNeT++ umgesetzt und bewegt sich als einzige Implementierungsmaßnahme außerhalb der in Abschnitt 7.2.1 definierten Programmiersprache und Bibliotheken. Änderungen betreffen dabei die folgenden Punkte:

- Das Paket wird zu einem vorher festgelegten Zeitpunkt versendet
- Der Zyklus der Paketsendungen wird von 125μs auf 10 ms erhöht
- Pro Zyklus (10 ms) werden 2 Pakete pro Stream in einem Abstand von 125μs gesendet

Die Zyklusänderung hilft bei der Übersicht des Netzwerks, ist aber keine notwendige Änderung. In einem Zyklus werden zwei Pakete gesendet, um das Verhalten doppelter Burst-Pakete

weiterhin abbilden zu können. Für BE-Pakete wird eine ähnliche Änderung vorgenommen. Diese umfasst ebenfalls statische Generierungszeiten und einen Zyklus von 10 ms, aber keine doppelten Pakete pro Zyklus.

In dem Java-Tool umfasst die Neuplanung folgende Schritte:

1. Alle Crosstraffic-Pakete ermitteln.
2. Pro Link, der zu dem Switch führt, an dem der Worst-Case konstruiert wird, wird das größte Crosstraffic-Paket vermerkt, weitere Crosstraffic-Pakete werden aus der Konstruktion entfernt.
3. Pro Link, der zu dem Switch führt, an dem der Worst-Case konstruiert wird, werden entweder
   - existierende BE-Pakete so geplant, dass das Crosstraffic-Paket des Links doppelt im untersuchten Switch ankommt, oder
   - neue Knoten entworfen bzw. existierende Knoten so angepasst, dass sie BE-Pakete mit der gleichen Wirkung versenden.
4. Die Startknoten aller verbleibenden Crosstraffic-Pakete werden ermittelt
5. Die Crosstraffic-Pakete werden so geplant, dass sie an der Queue, die zu dem untersuchten Switch führt, durch das zuvor geplante BE-Paket blockiert werden.
6. Das Paket, auf das der Worst-Case zugeschnitten wird, wird dann so geplant, dass alle Crosstraffic-Pakete kurz davor an dem untersuchten Switch eintreffen.

Für die Planung der TSN-Pakete wird das Ausgangs-Netzwerk verwendet. Diese werden mit unverändertem Schedule verschickt. Alle anderen AVB- und BE-Pakete, die nicht zu der Konstruktion des Worst-Case beitragen, werden nicht versendet. Das verhindert eventuelle Verschiebungen des Sendeablaufs, die den schlechtesten Fall verhindern könnten.

Das neu geplante Netzwerk hat den Anspruch, den Vorbedingungen entsprechend den zuvor abgeschätzten Worst-Case-Wert der Network-Calculus so gut wie möglich anzunähern. Je nach Aufbau des Ausgangs-Netzwerkes können die Analyse und generierte Simulation sehr weit auseinander liegen. Dieses Thema wird im Kapitel Evaluierung (8) diskutiert.

7.2.8 Erstellung neues Netzwerk

Die Erstellung des Netzwerks erfolgt durch eine Erzeugung von Simulationsdateien. Analog zu der eingelesenen Simulation besteht diese aus mehreren Dateien mit der Endung .ini oder .ned, die jeweils für Paket- und Strukturinformationen verwendet werden. Gegenüber der Ausgangssimulation werden die TrafficSource-Apps auf die statische Variante umgestellt und Sendezeiten übergeben.

Falls neue Knoten hinzugefügt werden müssen, um zusätzlichen BE-Traffic zu erzeugen, hat das weitere Änderungen zur Folge, die auch die Infrastruktur des Netzwerks betreffen.

7.3 Umsetzung

Dieser Abschnitt wird einen Überblick über die Umsetzung des Tools geben. Die Oberfläche und ausgewählte Implementierungsdetails dienen als Übersicht der Umsetzung. Auf Code im laufenden Text wird verzichtet, um diesen Abschnitt übersichtlicher zu gestalten. Der Code findet sich zu Teilen im Anhang und vollständig auf der beigelegten CD.

7.3.1 Oberfläche

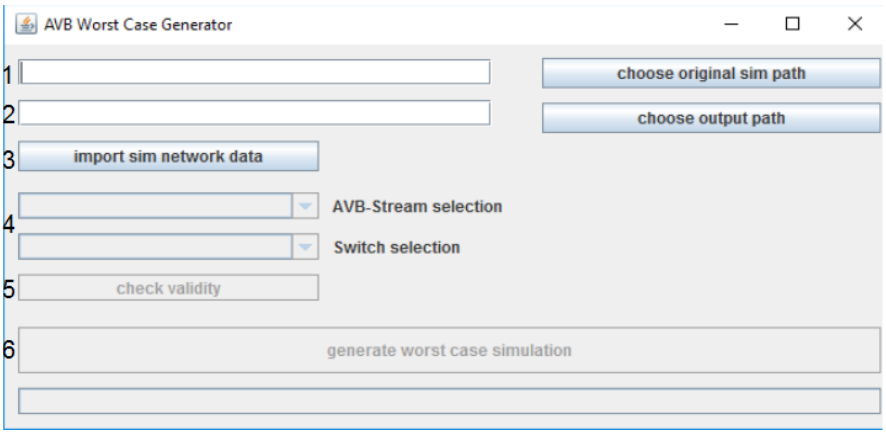

Abbildung 7.3: Benutzeroberfläche des Simulationsgenerierungstools

Die Benutzeroberfläche ist nach dem in Abschnitt 7.2.2 vorgestellten Konzept für die Oberfläche entwickelt worden. Alle Eingabefelder sind wie vorgesehen vorhanden. Die Bedienung des Tools erfolgt vollständig über diese Benutzeroberfläche. Die in Abb. 7.3 eingezeichnete Bedienungsreihenfolge wird nachfolgend erläutert:

1. **Auswählen des Quell-Simulationspfades**. Das Textfeld ist der Master und dessen Wert wird verwendet. Der Wert des Textfeldes ist auch über die Schaltfläche *choose original sim path* beeinflussbar.

2. **Auswählen des Ziel-Pfades**. Analog zu Quell-Simulationspfad. Über *choose output path* beeinflussbar.

3. **Importieren erster Daten**. Diese Schaltfläche bewirkt, dass die Auswahlfelder von Schritt 4 mit Werten befüllt werden.

4. **Auswählen von untersuchten AVB-Stream und Switch**. Auf den hier ausgewählten AVB-Stream wird ein Worst-Case-Szenario zugeschnitten, das am ausgewählten Switch eintritt.

5. **Validitäts-Check der Auswahl.** Sind alle Auswahlen korrekt, werden vorherige Eingabefelder gesperrt und die Generierung des Worst-Case-Szenarios ermöglicht (Schritt 6). Um die Validierung rückgängig zu machen, kann ein neuer Quell-Simulationspfad (ausschließlich per Schaltfläche) ausgewählt werden.

6. **Generierung des Worst-Case-Szenarios**. Diese Schaltfläche stößt die Hauptfunktionalität des Tools an. Es wird ein Netzwerk aus den importierten Daten konstruiert und eine Neuplatzierung von Paketen vorgenommen, um für den ausgewählten AVB-Stream das Worst-Case-Szenario zu generieren.

7.3.2 Netzwerkkonstruktion

Wird auf der Oberfläche die Generierung des Worst-Case-Szenarios gestartet (Schritt 6), konstruiert das Tool eine vollständige Repräsentation des Netzwerkes. Der Aufbau der Klassen, die den infrastrukturellen Aufbau des Netzwerkes ausmachen, lässt sich aus Abb. 7.4 entnehmen.

Der Knotenpunkt ist die Klasse *Network*. Dieses enthält alle Strukturen des Netzwerkes. Die Strukturen können Knoten oder Switche sein. Die abstrakte Klasse *NetworkDevice* fasst gemeinsame Eigenschaften von Knoten und Switches zusammen. Die Klasse *OutputQueue* ist hier zu nennen, da diese die Schnittstelle zu der Paketstruktur des Netzwerkes darstellt. Sich unterscheidende Eigenschaften von Knoten und Switches sind in den Klassen *NetworkNode* bzw. *NetworkSwitch* modelliert. Beim Aufbau des Netzwerks wird jede .ned-Datei der Quell-Simulation auf Netzwerkknoten oder -switche untersucht und diese anschließend in das *Network*-Objekt integriert.

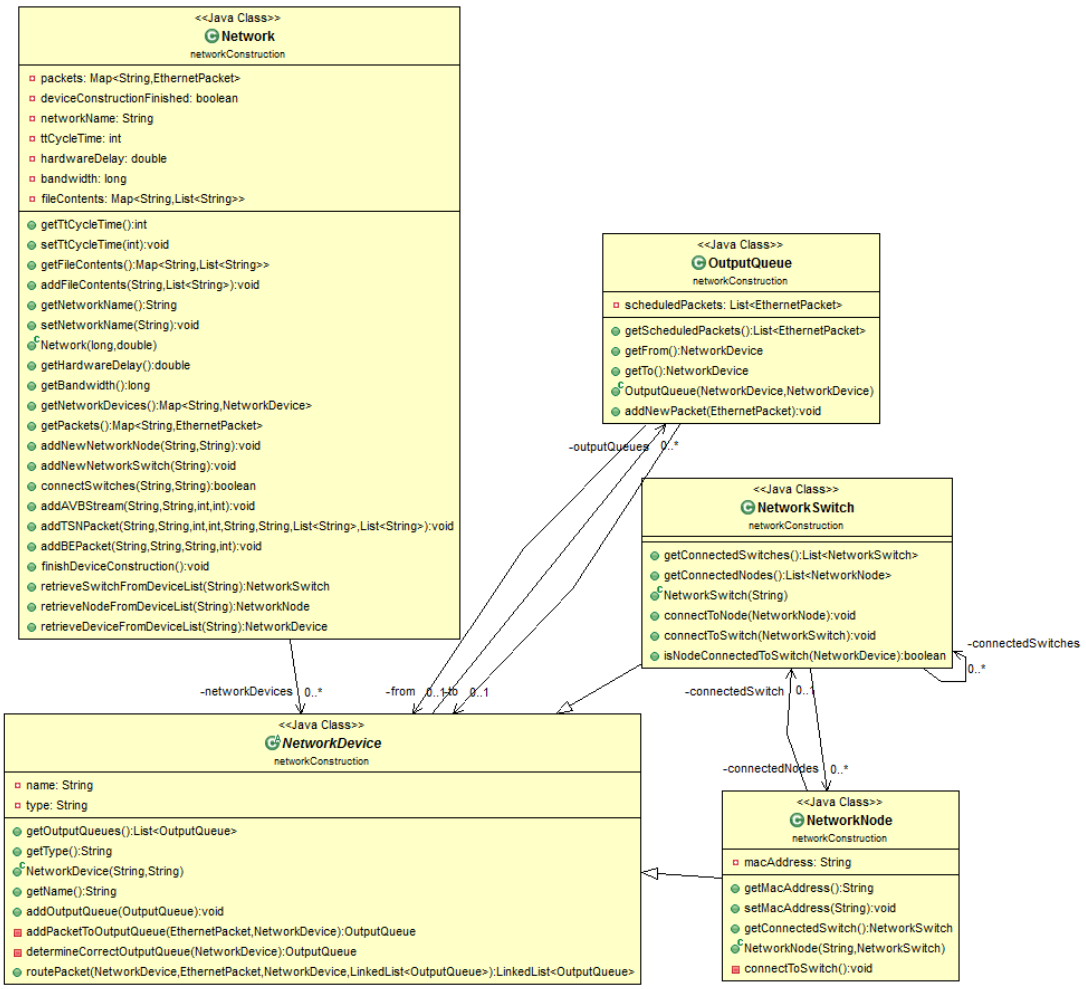

Abbildung 7.4: Klassendiagramm des Tools (Infrastruktur)

Die Pakete, die im Netzwerk versendet werden, sind über die Klasse *EthernetPacket* modelliert. Gemeinsame Eigenschaften wie Payload, Start- und Zielknoten und den Weg, den die Pakete im Netzwerk nehmen, sind Teil dieser Klasse. Der Aufbau der Pakete wird in Abb. 7.5 dargestellt.

Der Weg der Pakete wird in einer LinkedList festgehalten, die alle *OutputQueue*-Objekte vom Start bis zum Ziel des Pakets enthält. Da das *OutputQueue*-Objekt Informationen darüber enthält, von welchem *NetworkDevice* über diese Stelle an ein anderes *NetworkDevice* verschickt wird, kann über ein *OutputQueue*-Objekt ermittelt werden, welche anderen Pakete ebenfalls hier senden. Diese Mechanik ist ein wichtiger Bestandteil des Tools und ermöglicht die gewünschte Funktionalität der Worst-Case-Szenario-Erstellung.

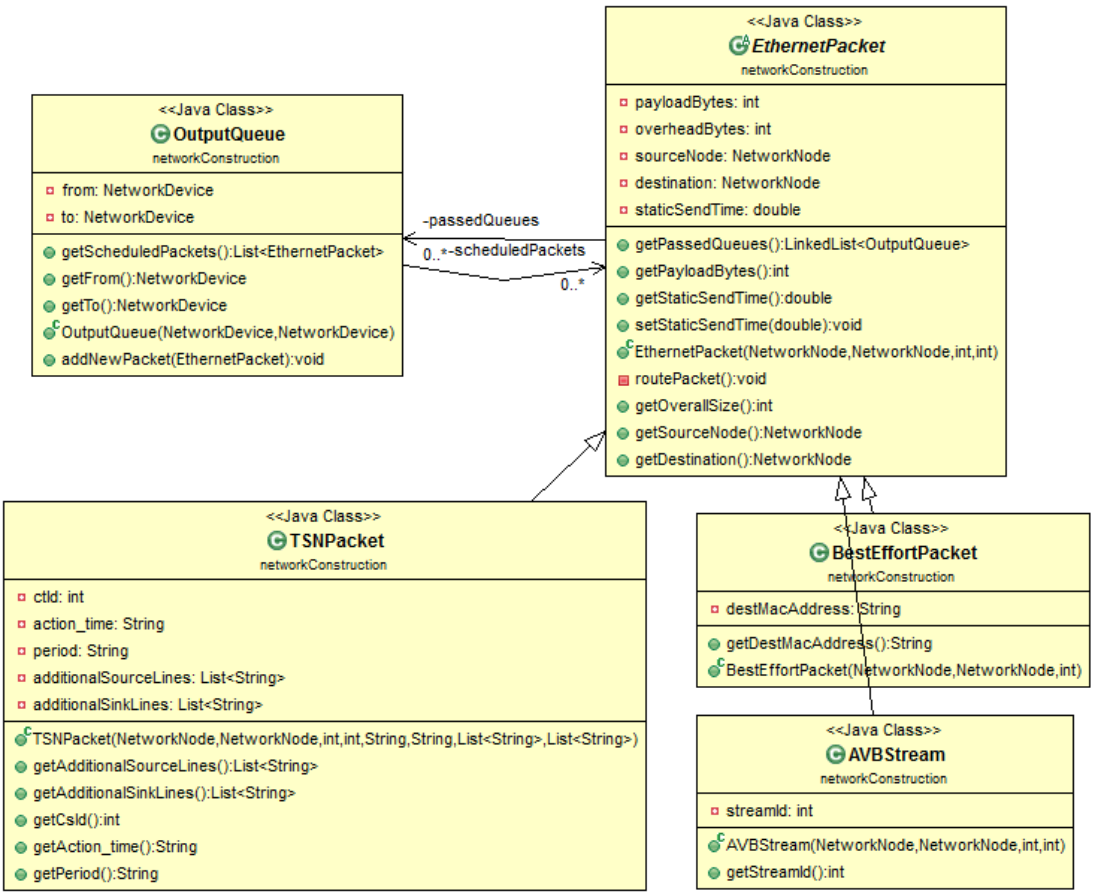

Abbildung 7.5: Klassendiagramm des Tools (Pakete)

Die Kernkomponente der Netzwerkkonstruktion ist das korrekte Routing der Pakete. Mithilfe der Infrastruktur aus Knoten und Switches kann ein Paket so geroutet werden, dass eine Liste der passierten Knoten und Switches erzeugt wird. Dieses Routing wird in *NetworkDevice* ausgeführt. Siehe dazu die Implementierung der Methode *routePacket(...)* in Listing 1 (Anhang).

7.3.3 Neuplanung des Traffics

Sind das Netzwerk und alle Pakete komplett aufgebaut, werden die Eigenschaften der Pakete untersucht. Je nach Auswahl auf der Benutzeroberfläche werden alle Pakete ermittelt, die auf der gleichen Queue wie der ausgewählte AVB-Stream an dem ausgewählten Switch senden.

Um die Klasse *WorstCaseScheduler*, die die Neuplanung aller Pakete koordiniert, ergibt sich die in Abb. 7.6 gezeigte Struktur der Klassen.

Abbildung 7.6: Klassendiagramm für Paketneuplanung

Die Klassen *AnalyzedStream* und *CrossTrafficPacket* sind aufgeteilt, da sie sich in den Eigenschaften leicht unterscheiden. In der Klasse *AnalyzedStream* wird ermittelt, welche Pakete zum Crosstraffic gehören. Diese werden in einer Liste gesammelt und in der Klasse *WorstCaseScheduler* weiter untersucht. Die Klasse ermittelt, ob mehrere Pakete des Crosstraffics über gleiche Links zum untersuchten Switch kommen, da immer nur ein Crosstraffic-Paket pro Lin den ausgewählten AVB-Stream blockieren kann. Der Ablauf wird in Listing 2 (Anhang) gezeigt.

Die Neuplanung des Verkehrs wird relativ zu einem 10ms-Zyklus der Static-Apps der Simulation vorgenommen. Die Apps bewirken, dass die AVB-Pakete zu einem vordefinierten Zeitpunkt generiert werden. Die Reservierung und Routenfindung sowie der Kreditmechanismus werden nicht außer Kraft gesetzt, nur der Startzeitpunkt der AVB- und Best-Effort-Pakete wird festgelegt. Die Apps sind auf der beigelegten CD als Quelle enthalten. Die Neuplanung des Crosstraffics wird in Listing 3 (Anhang) aufgeführt.

7.3.4 Prototyp-Funktionalität

Zum Abschluss dieser Arbeit ist das Tool in einem Prototypen-Status. Es liegen bereits einige Funktionen vor, Fehlerquellen sind aber vorhanden und müssen ausgiebiger getestet werden. Dieser Abschnitt gibt einen Überblick über enthaltene Funktionen und deren Status. Funktionen, die als „abgeschlossen“ gekennzeichnet sind, funktionieren im Rahmen dieser Arbeit vollständig, erheben aber keinen Anspruch auf ausnahmslos fehlerfreie Ausführung. Ein ausführlicher Test aller Funktionen wird vor der Weiterentwicklung des Tools empfohlen.

Funktionsübersicht

Funktion	Status
Pfade auswählen	abgeschlossen
Dateien einlesen	abgeschlossen
Validierung der Eingaben	Überprüfung nur rudimentär
Fortschritt anzeigen	abgeschlossen

Tabelle 7.2: Funktionen in der Simulationsauswahl

Funktion	Status
Dateien einlesen	abgeschlossen
Dateien nach Endung sortieren	abgeschlossen
Infrastruktur (Knoten, Switche) aufbauen	abgeschlossen
Pakete einlesen AVB	abgeschlossen
Pakete einlesen BE	abgeschlossen
Pakete einlesen TSN	potentiell fehleranfällig (bspw. „period“)
Pakete routen	AVB- und TT-Pakete haben nur ein Ziel

Tabelle 7.3: Funktionen der Netzwerkkonstruktion

Die hier vorgestellten Tabellen geben einen Überblick über den aktuellen Stand des Tools. Aufgeteilt werden diese nach Arbeitsschritten. Tabelle 7.2 zeigt die Funktionen während der Simulationswahl und damit verbundene Mechaniken, beispielsweise das Einlesen von Dateien. Ist eine Auswahl der Quell-Simulation getroffen, wird das Kernstück des Tools angesprochen. Tabelle 7.3 zeigt die Funktionen der Netzwerkkonstruktion. Ist die Netzwerkkonstruktion abgeschlossen, werden anhand des ausgewählten AVB-Streams und des ausgewählten Switches die Neuplanung des Netzwerkes vorgenommen. Die Status der zugehörigen Funktionen werden in Tabelle 7.4 gezeigt.

Funktion	**Status**
OutputQueue für CT identifizieren	abgeschlossen
Ein CT-Paket pro eingehendem Link	abgeschlossen
Neuplanung AVB	abgeschlossen
Neuplanung BE	Keine neuen Knoten (kann nötig sein)
Berücksichtigung TT	Nur auf vollen TT-Cycle

Tabelle 7.4: Funktionen der Neuplanung

Die Funktion „Berücksichtigung TT“ ist im Tool in einem geringen Umfang gegeben. Die Planung der AVB- und BE-Pakete wird so durchgeführt, dass ihre Sendung zu Abschluss eines 10ms-Zyklus endet. Ein Paket x des ausgewählten AVB-Streams wird so gesendet, dass ein TT-Paket, das zu Beginn eines solchen 10ms-Zyklus sendet, das Paket x blockieren würde. Auf diese Funktion wird im Unterabschnitt *Erweiterungsmöglichkeiten* eingegangen.

Nach der Neuplanung des Netzwerkes werden Dateien für eine neue Simulation erzeugt. Diese Erzeugung und die damit verbundenen Funktionen werden in Tabelle 7.5 gezeigt.

Funktion	**Status**
Erstellen von/Schreiben in Dateien	abgeschlossen
Zwischenspeichern und einfügen alter Inhalte	abgeschlossen
Paketquellen einfügen (AVB und BE)	abgeschlossen
Paketquellen einfügen (TT)	potentiell fehleranfällig
Paketsenken einfügen (AVB und BE)	abgeschlossen
Paketsenken einfügen (TT)	potentiell fehleranfällig
NED-Dateien kopieren	abgeschlossen
Neue Knoten erstellen	—

Tabelle 7.5: Funktionen der Ausgabe

Die Funktionen sind mit diesen Tabellen grob beschrieben. Für einige Funktionen, besonders aus Tabelle 7.4, wäre eine feingranularere Auflistung der Funktionen möglich. Aus Übersichtsgründen wird auf diesen Schritt verzichtet. Nicht abgeschlossene Funktionen werden nachfolgend diskutiert.

7.3.5 Erweiterungsmöglichkeiten

In den Tabellen 7.2-7.5 sind einige Funktionen enthalten, die nicht den Status „abgeschlossen“ aufweisen. Diese können folgendermaßen abgeschlossen und/oder erweitert werden:

- **Hardware-Delay:** Die Simulation hat einen Hardware-Delay pro Weiterleitung eines Pakets in einem Switch. Dieser liegt in der Regel bei $8\mu s$, kann aber umgestellt werden. Der Hardware-Delay könnte entweder auf der Oberfläche als Option hinzugefügt werden, oder aus den Simulationsdateien eingelesen werden.

- **Validierung der Eingaben:** Eine valide Auswahl besteht zur Zeit darin, dass ein Ausgabepfad ausgewählt wurde (s. *checkValidity()* in der Klasse *GuiBuilder*). Das kann um eine Überprüfung der Eingaben erweitert werden. Die Pfade können auf Gültigkeit überprüft werden und bei der gegenwärtigen Auswahl kann überprüft werden, ob der ausgewählte AVB-Stream überhaupt an diesem Switch sendet. Eine vorgezogene Importierung aller Daten könnte diese Überprüfung ermöglichen.

- **Pakete einlesen/Paketquellen und -senken für TT-Pakete:** Das Einlesen von Informationen von TT-Paketen funktioniert instabil. Die Eigenschaft „Periode“ wird eventuell nicht richtig angegeben. In der Klasse *IniCreator* werden in den Methoden *addTSNPacketource* und *addTSNPacketSink* TT-Pakete erstellt. Diese Methoden können angepasst werden, um diese Funktionalitäten zu verbessern.

- **Pakete routen:** Das Routing funktioniert nur für einen Zielknoten. AVB- und TT-Pakete, die mehr als ein Ziel haben, werden demnach nicht korrekt abgebildet. Eine Erweiterung, die diesen Umstand berücksichtigt, erfordert mehrere Änderungen. Die Klasse *EthernetPacket* hat derzeit nur einen Zielknoten. Stattdessen könnte eine Liste von Zielknoten angelegt werden. In der Klasse *NetworkConstructor* kann die Methode *findPacketDestinations()* angepasst werden, um mehrere Zielknoten zu berücksichtigen.

- **Berücksichtigung TT:** Bisher werden TT-Pakete für die Worst-Case-Planung nur zu einem geringen Teil berücksichtigt. Alle TT-Pakete werden so geplant wie in der Quell-Simulation. Eine Änderung kann die Neuplanung von TT-Paketen sein. Sie werden dann

so geplant, dass alle TT-Pakete, die den ausgewählten AVB-Stream blockieren können, hintereinander gesendet werden. Dadurch entstünde ein Block aus TT-Paketen, der eine größere Verzögerung verursachen würde. Alle TT-Pakete, die nicht den ausgewählten AVB-Stream blockieren können, sollten dann nicht mitgeplant werden.

Eine andere Änderung kann eine veränderte Platzierung der neugeplanten AVB- und TT-Pakete sein. Die Platzierung muss so gewählt werden, dass sie unmittelbar vor dem größten TT-Paket endet (exklusive des ausgewählten AVB-Streams). Auf diese Weise ist garantiert, dass mindestens das größte TT-Paket die Sendung des ausgewählten AVB-Streams verzögert. Für beide Änderungen ist ein möglicher Einstiegspunkt die Klasse *WorstCaseScheduler* in der Methode *rescheduleCT()*.

- **Neue Knoten für BE-Traffic:** Auf dem Link zum ausgewählten Switch muss ein BE-Paket platziert werden, um für alle Crosstraffic-Pakete ein doppeltes Eintreffen am ausgewählten Switch zu garantieren. Kommt der Crosstraffic von einem Knoten, der direkt an den ausgewählten Switch angeschlossen ist, kann direkt in diesem Knoten BE-Traffic generiert werden. Ist das nicht der Fall (mehr als ein Hop für den Crosstraffic), dann muss ein Knoten am vorigen Switch diesen BE-Traffic generieren. Ist dort kein Knoten vorhanden, müsste ein Knoten generiert werden. Der derzeitige Stand des Tools generiert den BE-Traffic auch bei Crosstraffic mit mehr als einem Hop im Startknoten dieses Crosstraffics. Dieses Vorgehen hat geringere Latenzen zur Folge. Einen Einstiegspunkt für diese Erweiterung bietet die Klasse *NedCreator*.

7.4 Qualitätssicherung

Die Qualitätssicherung testet die wichtigsten Funktionen des Tools. So wird garantiert, dass das Tool in dem getesteten Umfang korrekt arbeitet. Einige Funktionen wurden getestet und funktionieren korrekt. Dazu gehören:

- Kopieren von NED-Dateien
- Erstellen der Netzwerk-Infrastruktur
- Ausführung der erstellten Simulation unter OMNeT++

Einen Einblick in die korrekte Funktionsweise des Tools bietet die Importierung von Paketinformationen. Paketinformationen werden aus den INI-Dateien extrahiert. Für die Neuplanung ist wichtig, dass alle Informationen korrekt eingelesen werden. In Listing 7.1 wird die Ausgangs-INI-Datei gezeigt.

```
[General]
network = AVBSingleSwitchWC

**.s1_1.phy[*].mac.address = "0A-00-00-00-00-1A"

**.s1_1.numApps = 3

**.s1_1.app[0].typename = "TTTrafficSourceApp"
**.s1_1.app[0].action_time = 0us
**.s1_1.app[0].payload = 72Byte
**.s1_1.app[0].ct_id = 111
**.s1_1.app[0].period = "period[0]"
**.s1_1.app[0].buffers = "AVBSingleSwitchWC.s1_1.VL_TT_111"

**.s1_1.app[1].typename = "AVBTrafficSourceAppStatic"
**.s1_1.app[1].streamID = 11
**.s1_1.app[1].intervalFrames = 1
**.s1_1.app[1].payload = 74 Byte
**.s1_1.app[1].simTimeSend = 0.00008184

**.s1_1.app[2].typename = "BGTrafficSourceAppStatic"
**.s1_1.app[2].destAddress = "0A-00-00-00-00-23"
**.s1_1.app[2].payload = uniform(1500Byte, 1500Byte)
**.s1_1.app[2].sendInterval = uniform(125us, 125us)
**.s1_1.app[2].simTimeSend = 0.00008183

**.s1_1.phy[0].shaper.tt_buffers = "AVBSingleSwitchWC.s1_1.VL_TT_111
   "

**.s1_1.VL_TT_111_ctc.receive_window_start = 0 tick
**.s1_1.VL_TT_111_ctc.receive_window_end = 10000000 tick
**.s1_1.VL_TT_111_ctc.permanence_pit = 0 tick
**.s1_1.VL_TT_111.destination_gates = "AVBSingleSwitchWC.s1_1.phy
   [0].TTin"
**.s1_1.VL_TT_111.ct_id = 111
**.s1_1.VL_TT_111.sendWindowStart = 10 tick
**.s1_1.VL_TT_111_ctc.period = "period[0]"
**.s1_1.VL_TT_111.period = "period[0]"
```

Listing 7.1: Beispiel-INI-Datei „s1_1.ini“

Die Datei enthält drei Paketquellen, jeweils eine für die Paketklassen AVB, BE und TT. Nach dem Einlesen der Daten werden die Pakete in der Repräsentation des Netzwerks im zugehörigen *NetworkDevice* in der *OutputQueue* hinterlegt, die zum angeschlossenen Switch zeigt. Abb. 7.7 zeigt den Aufbau des Netzwerkes mit der Struktur des Knotens „s1_1“, der aus der in Listing 7.1 dargestellten Datei importiert wurde.

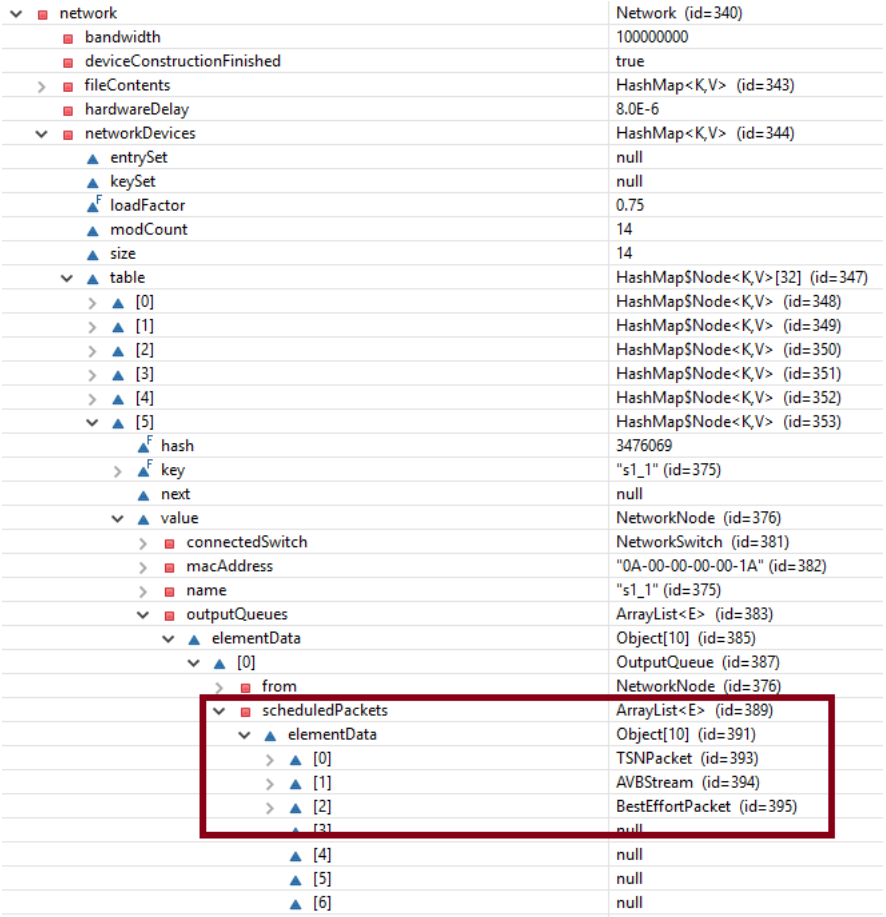

Abbildung 7.7: Geplante Pakete an NetworkDevice nach INI-Einlesen

Die Informationen der einzelnen Pakete werden ebenfalls importiert. Für TT-Pakete (im Tool als „TSNPacket“) werden zusätzliche Zeilen für die Konfiguration kopiert und ebenfalls im Objekt in der Liste „additionalSourceLines“ (Quelle des TT-Pakets) bzw. „additionalSinkLines“ (Senke des TSN-Pakets) vorgehalten. Die Informationen der Pakete werden in Abb. 7.8 gezeigt.

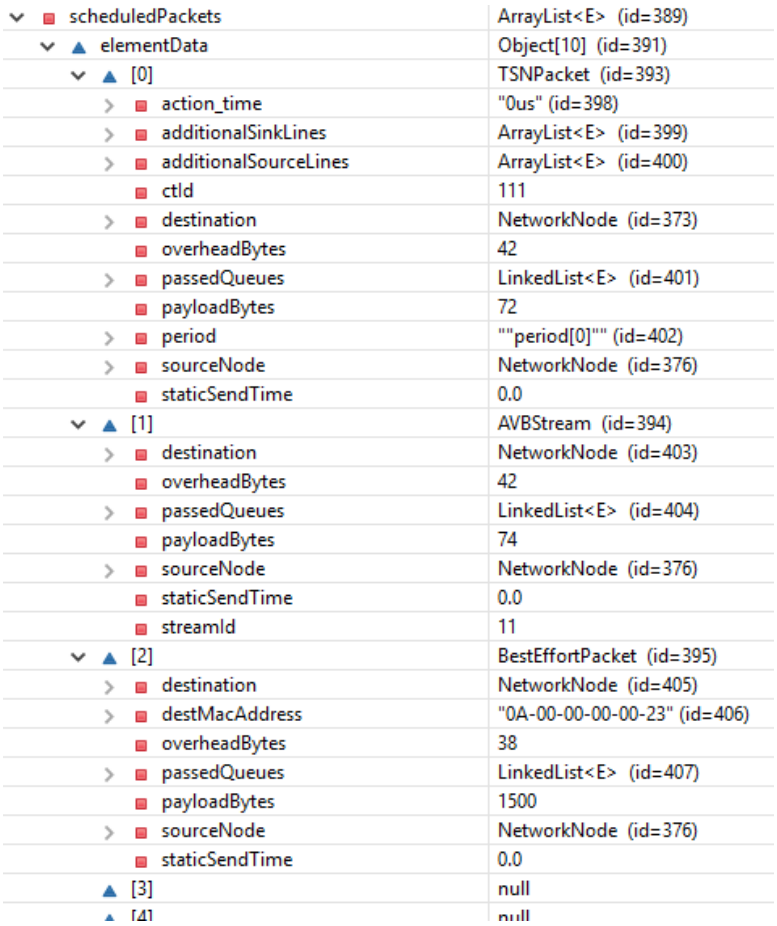

Abbildung 7.8: Detailinformationen der Pakete nach INI-Einlesen

Weitere Tests für die Sicherstellung korrekter Funktionsweisen sind speziell bei der Weiterentwicklung des Tools empfehlenswert. Dazu gehören:

- Vermeidung Null-Pointer in Netzwerkerstellung
- Fehlerfreie Generierung von TT-Paketen
- Fehlerfreie Generierung von zusätzlichen Knoten
- Korrekte Platzierung der Crosstraffic-Pakete unter Berücksichtigung von TT-Paketen

Die generierten Netzwerke werden im nachfolgenden Evaluierungs-Kapitel betrachtet.

8 Evaluierung

Die in Kapitel 4 vorgestellten Worst-Case-Modelle für AVB und TSN sind der erste Baustein, der für die, in Kapitel 6 erarbeiteten, Network-Calculus-Modelle als Basis dient. Das in Kapitel 7 vorgestellte Tool basiert in seiner Worst-Case-Planung ebenfalls auf den Worst-Case-Modellen. Dieses Kapitel untersucht, wie gut die Modellierungen aus Kapitel 4 umgesetzt worden sind, indem die Ergebnisse von Network-Calculus-Abschätzungen und generierten Simulationen gegenübergestellt werden.

Die Network-Calculus-Modelle haben den Anspruch, eine nach oben begrenzte Abschätzung der Latenz einzelner Pakete abzugeben. Alle Latenzen eines so abgeschätzten Pakets eines AVB-Streams in einem realen Netzwerk oder einer Simulationen können somit niemals über diesem abgeschätzten Wert liegen. Die aus dem Tool generierten Netzwerke werden bezüglich folgender Fragestellungen untersucht:

- Liegt die Latenz unter dem abgeschätzten Wert?
- Wie groß ist die Differenz zwischen Simulation und abgeschätztem Wert?
- Wo liegen die Schwächen des Tools bzw. der generierten Simulation?

Die Evaluierung wird anhand von drei Beispielnetzwerken vorgenommen. Das erste Netzwerk stellt ein Minimalbeispiel dar, das nur aus wenigen Komponenten und Paketen besteht. Das zweite und dritte Netzwerk werden umfangreicher gewählt, unterscheiden sich aber in ihren „Worst-Case-Fähigkeiten“. Eine genaue Erklärung der Netzwerke erfolgt in den jeweiligen Abschnitten.

8.1 Kleines Netzwerk

Das erste Netzwerk besteht lediglich aus einem Switch, an dem drei Knoten angeschlossen werden.

8.1.1 Aufbau

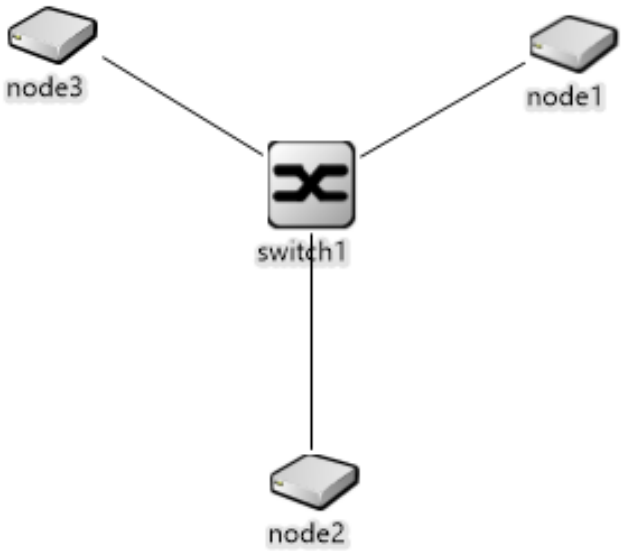

Abbildung 8.1: Evaluierung: Aufbau eines Minimal-Beispiel-Netzwerks

Alle Knoten des Netzwerks sind mit dem Switch verbunden (vgl. 8.1). Die im Netzwerk geplanten Pakete werden in Tabelle 8.1 aufgelistet.

Paketart	ID	Payload	Start	Ziel
AVB	1	350 Byte	node1	node3
AVB	2	350 Byte	node2	node3
TT	100	46 Byte	node1	node3

Tabelle 8.1: Pakete im Netzwerk aus Abb. 8.1

8.1.2 Ausgangslatenz

Das Paket mit der ID 1 wird auf die Worst-Case-Latenz untersucht. Entscheidend ist die Latenz von switch1 bis zu node3. In der Quell-Simulation ergibt eine Auswertung dieser Latenz nach einer simulierten Sekunde die in Abb. 8.2 veranschaulichten Ergebnisse.

Die Latenz liegt knapp unter dem Wert von 0,00016 Sekunden, was 0,16ms entspricht. Dieser Wert zeigt die Gesamtlatenz des Pakets von node1 bis zu node3 an. Anschließend wird das Netzwerk als Quell-Simulation für das Tool verwendet.

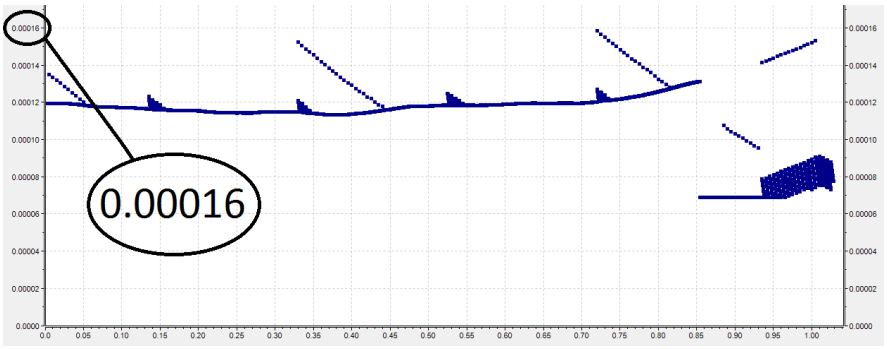

Abbildung 8.2: Evaluierung: Ausgangslatenz in Netzwerk aus Abb. 8.1

8.1.3 Generierte Simulation

Das Tool platziert die Pakete im Netzwerk so, dass alle AVB-Pakete, die das gleiche Ziel haben, vor dem ausgewählten Paket platziert werden. Die sich daraus ergebende Latenz wird in Abb. 8.3 gezeigt.

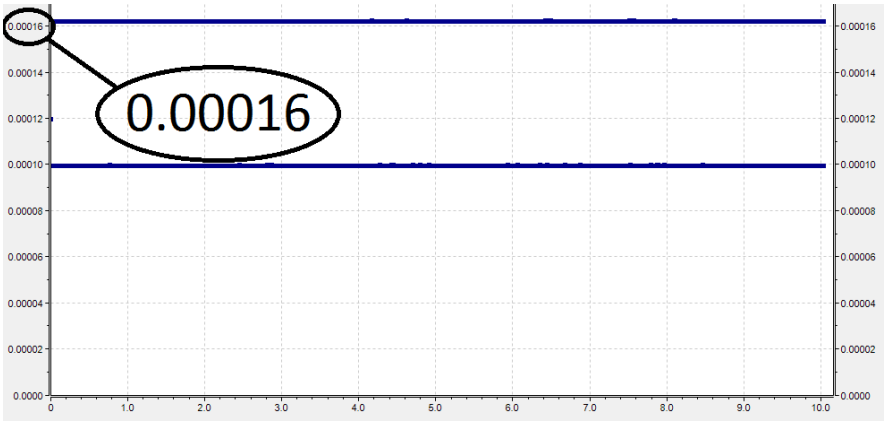

Abbildung 8.3: Evaluierung: Latenz nach Generierung in Netzwerk aus Abb. 8.1

Die Latenz liegt geringfügig über 0,00016s, also 160μs. Für dieses Netzwerk ist zu beachten: Einer der Drei Knoten dient als Ziel für die AVB-Streams. Die anderen beiden werden verwendet, um entweder den ausgewählten AVB-Stream oder den einen anderen AVB-Stream, der als Crosstraffic fungiert, zu generieren. Es gibt folglich keinen Knoten, der zur Versendung eines BE-MTU-Paketes verwendet werden könnte. Auch wenn in diesem Fall eine Ersetzung des Crosstraffic-Pakets durch ein BE-MTU-Paket eine höhere Latenz zur Folge hätte, wird die Priorität in der Generierung auf Crosstraffic-Pakete gelegt. Das einzige Paket, das in diesem generierten Netzwerk neu angelegt worden ist, ist ein Best-Effort-Paket an node2. Dieses sorgt

dafür, dass Stream 2 doppelt am Switch eintrifft (was in diesem Netzwerk keine Auswirkung auf die Latenz des untersuchten Streams hat).

8.1.4 Network-Calculus-Abschätzung

Für die Worst-Case-Abschätzung müssen alle Parameter dieser Konfiguration bestimmt werden. Diese werden in Tabelle 8.2 aufgeführt.

Parameter	Wert
T_{tcl}	0
σ_{tcl}	392 Byte
R_{tcl}	44,8 Mbit/s
ω_{tcl}	392 Byte
TT	84 Byte

Tabelle 8.2: Parameter für NC-Abschätzung in Netzwerk aus Abb. 8.1

Mit diesen Parametern lässt sich die Latenz nach oben abschätzen:

$$\begin{aligned}\overline{D_{tcl}} &= 0 + 2 \cdot \frac{392Byte}{44,8Mbit/s} - \frac{392Byte}{100Mbit/s} + \frac{84Byte}{100Mbit/s} \\ &= 0,00011536s = 115,36\mu s\end{aligned} \tag{8.1}$$

Diese Abschätzung zeigt die Verzögerung des Streams mit der ID 1 am Switch. Die Gesamtlatenz setzt sich aus diesem Wert und der Übertragungszeit des Pakets zusammen. Die Übertragungszeit des Pakets ergibt sich aus seiner Größe und der Bandbreite des Netzwerkes zuzüglich des Hardware-Delays ($8\mu s$). Mit L_{total} für die Latenz des Streams von seinem Startknoten bis zum Zielknoten ergibt sich die in Formel 8.2 angegebene Gesamtlatenz.

$$L_{total} = \overline{D_{tcl}} + 2 \cdot \frac{392Byte}{100Mbit/s} + 0,000008s = 0,00018608s = 186,08\mu s \tag{8.2}$$

8.1.5 Einordnung

Das erste Netzwerk bietet nicht viele Möglichkeiten, um die Latenz eines AVB-Streams negativ zu beeinflussen. Die beiden Störquellen sind ein anderes AVB-Paket und ein TT-Paket. Das TT-Paket wird in der generierten Simulation nicht in das Worst-Case-Szenario eingebaut, könnte aber einen weiteren Einfluss auf Stream 1 haben. Die Simulation hat momentan nur

eine geringfügig erhöhte Latenz gegenüber der Quell-Simulation. Das liegt daran, dass der schlechteste Fall — die Blockierung durch Stream 2 — in Teilen schon in der Quell-Simulation auftritt. Die generierte Simulation sorgt lediglich dafür, dass die größtmögliche Blockierung auftritt.

8.2 Worst-Case-Netzwerk

Das zweite Netzwerk wird mit einer Ausrichtung auf die Worst-Case-Moedellierung entworfen. Um den Worst-Case an einer Stelle bzw. Queue zu begünstigen, müssen viele Knoten an dem untersuchten Switch vorhanden sein. Das hier vorgestellte Netzwerk ist erneut nur über einen Switch verbunden. Die meisten verbundenen Knoten senden Crosstraffic-Pakete.

8.2.1 Aufbau

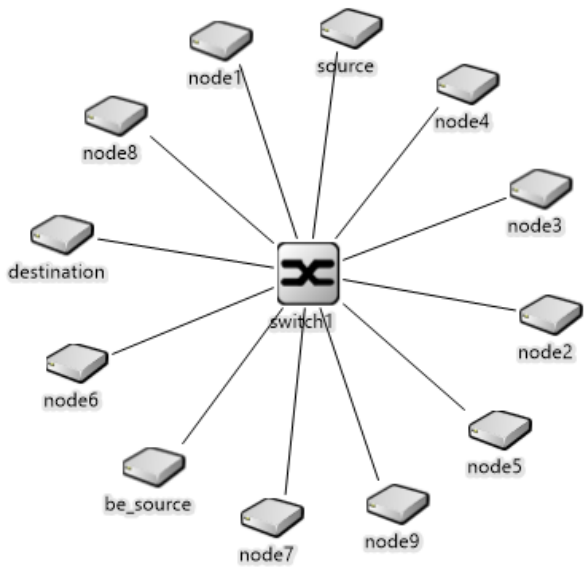

Abbildung 8.4: Evaluierung: Aufbau des Worst-Case-Netzwerks

Alle Knoten sind über einen zentralen Switch miteinander verbunden. Der Knoten „be_source“ sendet im Ausgangsnetzwerk keine Pakete und bietet eine Möglichkeit für die Generierung eines BE-MTU-Pakets. Der Aufbau lässt sich in Abb. 8.4 nachvollziehen. Die im Netzwerk geplanten Pakete werden in Tabelle 8.3 aufgelistet.

Paketart	ID	Payload	Start	Ziel
AVB	1	74 Byte	source	destination
AVB	11	74 Byte	node1	destination
AVB	12	74 Byte	node2	destination
AVB	13	74 Byte	node3	destination
AVB	14	74 Byte	node4	destination
AVB	15	74 Byte	node5	destination
AVB	16	74 Byte	node6	destination
AVB	17	74 Byte	node7	destination
AVB	18	74 Byte	node8	destination
AVB	19	74 Byte	node9	destination

Tabelle 8.3: Pakete im Worst-Case-Netzwerk

In diesem Netzwerk ist kein TT-Paket geplant. Dadurch werden in der Network-Calculus-Abschätzung keine Verzerrungen dadurch auftreten, dass in der generierten Simulation kein TT zur Konstruktion des Worst-Case berücksichtigt worden ist.

8.2.2 Ausgangslatenz

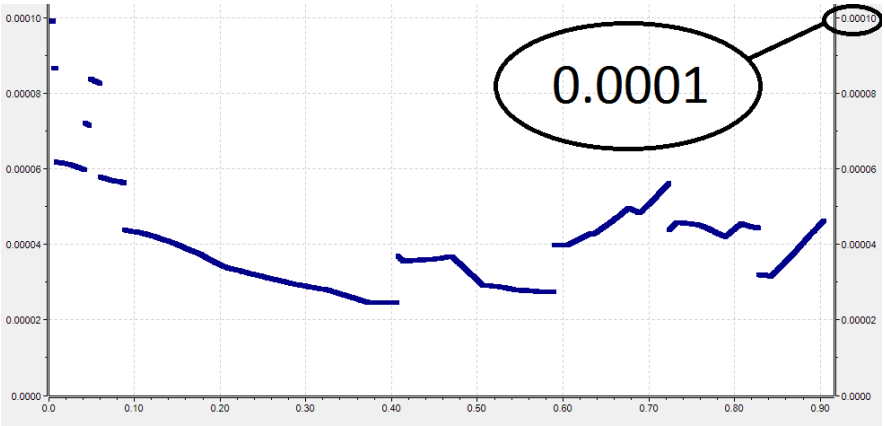

Abbildung 8.5: Evaluierung: Ausgangslatenz im Worst-Case-Netzwerk

Das Worst-Case-Szenario wird auf den AVB-Stream mit der ID 1 zugeschnitten. Dessen Latenzen vom Knoten source bis zum Knoten destination werden in Abb. 8.5 veranschaulicht. Ein Wert von 0,0001s, also 100μs, wird nicht überschritten. Im Vergleich zu dem kleinen Netzwerk liegt eine geringere maximale Ausgangslatenz darin begründet, dass alle Pakete in

dem Worst-Case-Netzwerk deutlich kleiner sind und daher eine geringere Übertragungszeit benötigen.

8.2.3 Generierte Simulation

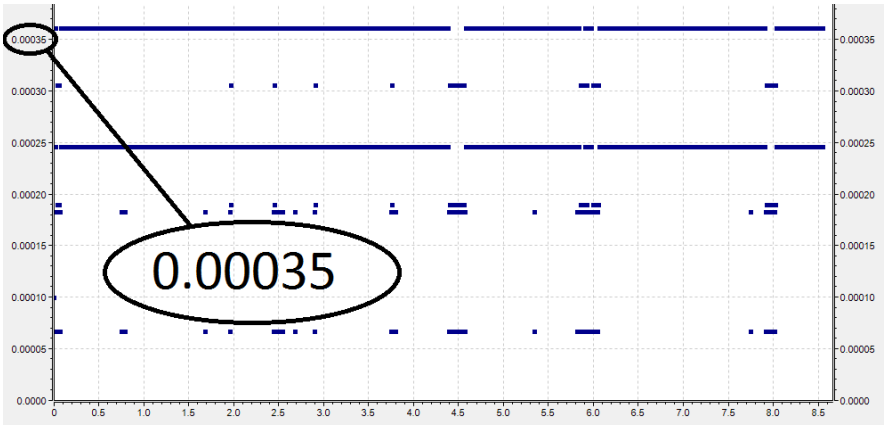

Abbildung 8.6: Evaluierung: Latenz nach Generierung im Worst-Case-Netzwerk

Die Latenz des AVB-Streams steigt in der generierten Simulation deutlich an. Alle AVB-Pakete, die von den anderen Knoten generiert werden, werden vor dem Paket mit der ID 1 versendet. Zusätzlich wird ein BE-MTU-Paket vom Knoten be_source versendet. Die maximale Latenz von Paketen des Streams mit der ID 1 liegt bei knapp 0,000361s, was 361μs entspricht.

8.2.4 Network-Calculus-Abschätzung

Die Parameter für die Abschätzung werden in Tabelle 8.4 angegeben.

Parameter	Wert
T_{tcl}	123,04μs
σ_{tcl}	1044 Byte
R_{tcl}	74,88 Mbit/s
ω_{tcl}	116 Byte
TT	0

Tabelle 8.4: Parameter für NC-Abschätzung im Worst-Case-Netzwerk

Mit diesen Parametern lässt sich die Latenz nach oben abschätzen:

$$\begin{aligned}\overline{D_{tcl}} &= 0,00012304s + 2 \cdot \frac{1044Byte}{74,88Mbit/s} - \frac{116Byte}{100Mbit/s} \\ &= 0,000346117s = 346,117\mu s\end{aligned} \tag{8.3}$$

Die Gesamtlatenz des Streams beläuft sich auf den in Formel 8.4 ausgewiesenen Wert.

$$L_{total} = \overline{D_{tcl}} + 2 \cdot \frac{116Byte}{100Mbit/s} + 0,000008s = 0,000372677s = 372,677\mu s \tag{8.4}$$

8.2.5 Einordnung

Die Abweichung der maximalen Latenz in der Simulation von der Network-Calculus-Abschätzung beträgt ca. 11,5μs. Prozentual liegt die Abschätzung also ungefähr 3,2% über der Latenz, die in der Simulation erreicht wird. Diese Abweichung lässt sich durch geringfügige Ungenauigkeiten in der Planung der Pakete erklären. Wird das BE-Paket zu früh geplant, kann es zu Ausfällen von Wartezeiten für Stream 1 kommen. Um das BE-Paket sicher vor Stream 1 zu platzieren, wird das BE-Paket zur Zeit in einem Sicherheitsabstand davor geplant, was in der Ungenauigkeit der Plaung resultiert.

8.3 Komplexes Netzwerk

Das dritte vorgestellte Netzwerk besteht aus einer größeren Anzahl Switche. Die Analyse wird erneut auf den AVB-Stream gelegt, der vom Knoten „source“ zum Knoten „destination“ geschickt wird. Da der Fokus auf der Worst-Case-Generierung liegt, werden alle AVB-Streams innerhalb des Netzwerks mit dem Ziel „destination“ verschickt.

8.3.1 Aufbau

Gegenüber den ersten beiden Netzwerken besteht die Infrastruktur des Netzwerkes aus mehreren Switches. Alle Knoten des Netzwerks versenden AVB-Streams, mit Ausnahme der Knoten „tt1“ und „tt2“, die TT-Pakete verschicken. Die Eigenschaften der Pakete werden in Tabelle 8.5 aufgelistet.

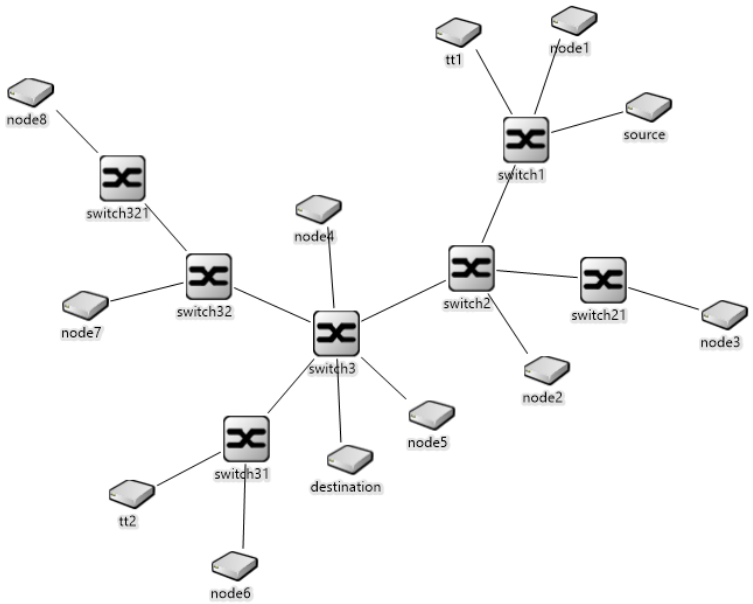

Abbildung 8.7: Evaluierung: Komplexes Netzwerk

8.3.2 Ausgangslatenz

Die Ausgangslatenz von Stream 1 wird in Abb. 8.8 gezeigt. Die Latenz des untersuchten AVB-Streams mit der ID 1 überschreitet einen Wert von 0,00014s über eine Strecke vom Knoten „source“ bis zum Knoten „destination“ nicht. Das entspricht 140μs.

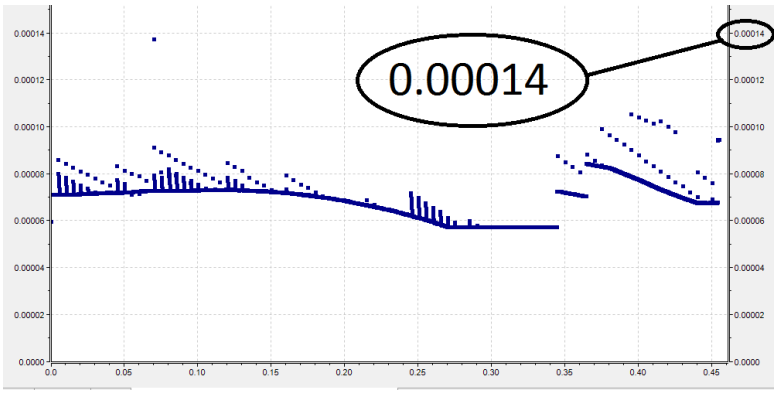

Abbildung 8.8: Evaluierung: Ausgangslatenz im komplexen Netzwerk

Paketart	ID	Payload	Start	Ziel
AVB	1	74 Byte	source	destination
AVB	11	64 Byte	node1	destination
AVB	12	64 Byte	node2	destination
AVB	13	64 Byte	node3	destination
AVB	14	74 Byte	node4	destination
AVB	15	74 Byte	node5	destination
AVB	16	74 Byte	node6	destination
AVB	17	79 Byte	node7	destination
AVB	18	84 Byte	node8	destination
TT	100	46 Byte	tt1	destination
TT	101	46 Byte	tt2	destination

Tabelle 8.5: Pakete im komplexen Netzwerk

8.3.3 Generierte Simulation

Die generierte Simulation richtet sich auf die Worst-Case-Latenz für den AVB-Stream 1 am Switch „switch3“ aus. An „switch3“ wird nur ein Crosstraffic-Paket auf jedem eingehenden Link zugelassen. Über den Link, der den Stream mit der ID 1 sendet, werden keine anderen AVB-Streams zugelassen. Das hat zur Folge, dass sich die geplanten Pakete im Netzwerk ändern, da nur Pakete berücksichtigt werden, die AVB-Stream 1 stören. Die neugeplanten Pakete werden in Tabelle 8.6 gezeigt.

Paketart	ID	Payload	Start	Ziel
AVB	1	74 Byte	source	destination
AVB	14	74 Byte	node4	destination
AVB	15	74 Byte	node5	destination
AVB	16	74 Byte	node6	destination
AVB	18	84 Byte	node8	destination
TT	100	46 Byte	tt1	destination
TT	101	46 Byte	tt2	destination

Tabelle 8.6: Pakete im komplexen Netzwerk nach Generierung

Die Streams 11, 12 und 13 fallen weg, da sie auf der gleichen Route wie Stream 1 liegen. Stream 17 fällt weg, da er auf der gleichen Route wie Stream 18 liegt. Stream 18 wird bevorzugt,

weil dessen Pakete eine höhere Payload haben. Zusätzlich werden an den Startknoten der AVB-Streams BE-Pakete geplant, damit die Pakete der AVB-Streams doppelt an switch3 eintreffen.

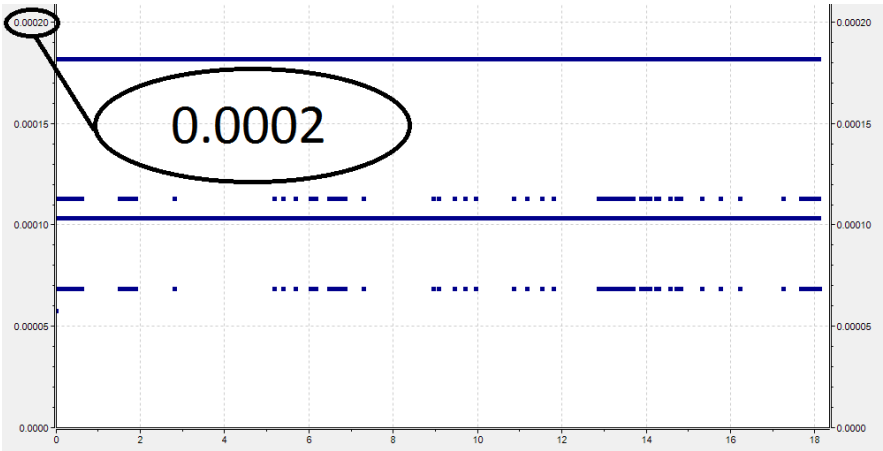

Abbildung 8.9: Evaluierung: Latenz nach Generierung im komplexen Netzwerk

Die Latenz von Stream 1 steigt im Gegensatz zu der Ausgangssimulation an. Dabei wird auf der Strecke zwischen „source“ und „destination“ eine Latenz von 0,00018s knapp überschritten. Diese 181μs bedeuten eine Differenz von etwa 45μs zu der Ausgangslatenz.

8.3.4 Network-Calculus-Abschätzung

Die Parameter für die Network-Calculus-Abschätzungen beruhen auf dem Quell-Netzwerk. Pakete, die in der generierten Simulation wegfallen, werden dessen ungeachtet für den Burst und den $idleSlope$ berücksichtigt.

Parameter	Wert
T_{tcl}	0
σ_{tcl}	913 Byte
R_{tcl}	66,432 Mbit/s
ω_{tcl}	126 Byte
TT	168 Byte

Tabelle 8.7: Parameter für NC-Abschätzung im komplexen Netzwerk

Im Ausgangsnetzwerk ist kein BE-Paket vorhanden. Das generierte Netzwerk hat kein BE-Paket mit dem Ziel „destination“, was darin begründet liegt, dass, wie bei dem ersten Netzwerk,

kein Knoten vorhanden ist, der diese Generierung übernehmen kann. Der Parameter T_{tcl} wird auf 0 gesetzt, um keine unnötige Verzerrung in der Network-Calculus-Abschätzung zu erhalten.

$$\begin{aligned}\overline{D_{tcl}} &= 0 + 2 \cdot \frac{913Byte}{66,432Mbit/s} - \frac{126Byte}{100Mbit/s} + \frac{168Byte}{100Mbit/s} \\ &= 0,000223254s = 223,254\mu s\end{aligned} \tag{8.5}$$

Die Worst-Case-Latenz von Stream 1 in Switch 3 beläuft sich auf den in Formel 8.6 ausgewiesenen Wert. Für die gesamte Übertragung überschreiten die Pakete von Stream 1 die in Formel 8.6 als L_{total} angegebene Latenz von source bis destination nicht.

$$L_{total} = \overline{D_{tcl}} + 2 \cdot \frac{116Byte}{100Mbit/s} + 0,000008s = 0,000240534s = 240,534\mu s \tag{8.6}$$

8.3.5 Einordnung

Die abgeschätzte Maximal-Latenz liegt etwa $60\mu s$ über dem erreichten Wert in der generierten Simulation. Der Grund dafür ist in erster Linie die Burst-Parametrisierung. Der Network Calculus geht davon aus, dass alle AVB-Pakete, die das gleiche Ziel wie Stream 1 haben, zu dem Burst zählen. Die generierte Simulation zeigt indessen, dass nur einige AVB-Streams einen störenden Einfluss haben. Ungenauigkeiten in der Planung von Paketstarts über mehrere Hops sorgen zusätzlich dafür, dass die Pakete nicht in der „optimalen“ Worst-Case-Szenario-Reihenfolge bzw. im „optimalen“ Worst-Case-Szenario-Abstand an switch3 ankommen. Die Latenz könnte durch manuelles nachjustieren der Generierungszeiten verbessert werden.

8.4 Überblick

Alle drei Netzwerke zeigen, dass eine Neuplanung der Pakete durch das Tool einen negativen Einfluss auf die Latenz des ausgewählten AVB-Streams haben. Je nach Ausgangs-Netzwerk und Route der Crosstraffic-Pakete sind die generierten Zeitpunkte näher an oder weiter entfernt von einem, Kapitel 4 entsprechenden, Worst-Case-Szenario dieses Netzwerkes.

Gleichzeitig wird das Network-Calculus-Modell für TSN-Netzwerke getestet. Alle simulierten Latenzen weisen einen geringeren Wert als die Network-Calculus-Abschätzung auf. Das stützt die Annahme, dass die Abschätzungen einen Wert ergeben, der die Latenzen nach oben be-

schränkt. Die Qualität der Abschätzungen weicht je nach Ausgangs-Netzwerk unterschiedlich stark von der Simulation ab.

Für die Abschätzung des ersten und dritten Netzwerks wäre die Verwendung der in Formel 6.8 aufgeführten spezialisierten Worst-Case-Betrachtung von TSN-Netzwerken zielführender, da die TT-Pakete nicht in einem Block auftreten und daher Lücken zwischen ihnen entstehen.

Die Evaluierung zeigt, dass die Abschätzungen eine korrekte Aussage treffen. Wie zu erwarten war, kann bei komplexeren, aus mehreren Switches bestehenden, Netzwerken die Genauigkeit der Abschätzung aufgrund der allgemeinen Parametrisierung schlechter werden. Dennoch kann die Abschätzung ein wichtiges Werkzeug sein, um den Traffic im Netzwerk optimal zu planen.

Das Tool hat im aktuellen Stand bei komplexen Netzwerken noch mit Ungenauigkeiten zu kämpfen, die bei weiterer Bearbeitung verringert werden können. Der Prototyp kann schon jetzt eine erhöhte Latenz für den ausgewählten AVB-Stream in der generierten Simulation gegenüber der Quell-Simulation vorweisen. Somit ist das Tool schon jetzt ein Werkzeug, das Latenz-Spitzen aufzeigt, die in der ursprünglichen Simulation nicht aufgefallen wären.

9 Zusammenfassung und Ausblick

Diese Arbeit untersucht das Timing in TSN-Netzwerken. Mit dem TAS existiert in TSN ein Shaper, der eine verringerte Latenz der versendeten Pakete gegenüber dem Vorgänger-Protokoll AVB aufweist. Die Übertragung zeitkritischer Daten ist auf diesem Weg umsetzbar, die Beeinträchtigung anderer Paketklassen durch die TSN-Pakete kann aber dazu führen, dass Daten, die hohe Anforderungen an die Netzwerk-Bandbreite stellen, verzögert werden. Die Analysen in dieser Arbeit beziehen sich auf die Latenzen dieser Paketklassen, die den AVB-Paketen des Vorgänger-Protokolls entsprechen.

Im Rahmen dieser Arbeit ist ein formales Analyse-Modell für die Analyse von AVB-Paketen in TSN entwickelt worden. Anhand des Network Calculus bietet dieses Modell eine Abschätzung der maximalen Verzögerungen für AVB-Pakete durch andere Paketklassen. Das Modell vereint alle Einflüsse, die an einer Station im Netzwerk auf ein AVB-Paket einwirken können. Die Worst-Case-Modellierung, die die Basis für das formale Analyse-Modell bietet, betrachtet Grenzfälle von auftretenden Paketkonstellationen und gibt einen sicheren Rückhalt, dass die Abschätzungen der formalen Analyse valide sind. Bestätigt wird diese Annahme durch eine Evaluierung des Modells an einigen Test-Netzwerken.

Mithilfe eines Tools, das im Zuge dieser Arbeit erstellt worden ist, können Netzwerk-Simulationen interpretiert und neugeplant werden. Die Neuplanung hat zum Ziel, die Worst-Case-Verzögerung des formalen Analyse-Modells mindestens anzunähern. Im gegenwärtigen Prototypen-Status des Tools ist ersichtlich, dass es fähig ist, Latenz-Spitzen in Netzwerken aufzuzeigen, die in der Ausgangs-Simulation höchstwahrscheinlich nie aufgetreten wären.

Sowohl die formalen Analyse-Modelle als auch die generierten Simulationen des Tools bieten eine neue Perspektive auf bestehende Netzwerke. Grenzfälle, die selten auftreten und in Simulationen möglicherweise nie zu beobachten sind, werden durch beide Analyse-Ansätze aufgedeckt. Anhand von Ergebnissen der formalen Analyse oder der Simulationsgenerierung kann eine Änderung in der Planung des Netzwerks vorgenommen werden. Für die Integration

von Ethernet-Kommunikation in Fahrzeuge können die formale Analyse und das Tool wichtig sein, um verlässlich Fehlerquellen aufzudecken und zu beheben. Mögliche Engpässe und Verzögerungen von Paketen können minimiert oder verhindert werden.

Die formale Analyse ist ein Werkzeug für die Analyse bestehender Netzwerke. Sie umfasst im Rahmen dieser Arbeit die Protokolle AVB und TSN. Unter TSN wird der Shaper TAS betrachtet. Die beiden anderen Shaper, BLS und CQF, können in zukünftigen Erweiterungen des Modells betrachtet werden. Eine vollständige Beschreibung des spezialisierten Worst-Case-Latenz-Ansatzes kann ebenfalls Gegenstand einer Erweiterung sein.

Das Tool ist in einem Prototypen-Status. Es ist möglich, die Funktionalitäten im bestehenden Tool zu verbessern oder neue Funktionen hinzuzufügen. Im Rahmen der CoRE-Projektgruppe wurde die DSL ANDL (Abstract Network Description Language) zur Erstellung von CoRE4INET-Netzwerken entwickelt. ANDL und das Tool können in einer Suite zusammengefasst werden, damit neben den ursprünglich generierten ANDL-Netzwerke auch Netzwerke erstellt werden, die auf Worst-Case-Latenzen eines AVB-Streams ausgerichtet sind.

Mit Ethernet in Fahrzeugen findet in absehbarer Zeit Technologiewechsel von Feldbussen zu modernen Kommunikationsstrukturen statt. Der Infotainment-Bereich wird immer umfassender und profitiert von diesem Wechsel. Mit den vorgestellten Analyse-Techniken können Fehlerquellen und Hindernisse, die dem Wechsel noch im Weg stehen, leichter identifiziert und verhindert werden.

Literaturverzeichnis

[1] Robert Bosch GmbH, *CAN Specification*, 1991. Version 2.0.

[2] Military & Aerospace Electronics, "*AFDX technology to improve communications on Boeing 787.*" http://www.militaryaerospace.com/articles/print/volume-16/issue-4/news/afdx-technology-to-improve-communications-on-boeing-787.html. zuletzt abgerufen am 26.09.2016.

[3] CoRE Research Group 2012-2013, "*CoRE Group.*" http://core.informatik.haw-hamburg.de/en/. zuletzt abgerufen am 06.05.2013.

[4] TTTech Computertechnik AG, "*TTEthernet: Deterministic Ethernet Network - TTTech.*" https://www.tttech.com/technologies/ttethernet/. zuletzt abgerufen am 23.12.2014.

[5] OpenSim Ltd, "*OMNeT++ Network Simulation.*" http://www.omnetpp.org/. zuletzt abgerufen am 24.01.2014.

[6] OpenSim Ltd, "*INET Framework for OMNeT++.*" http://inet.omnetpp.org/. zuletzt abgerufen am 24.01.2014.

[7] CoRE Research Group 2012-2016, "*CoRE Group.*" https://core4inet.core-rg.de/trac/. zuletzt abgerufen am 05.09.2016.

[8] W. Riggert, *Rechnernetze - Grundlagen - Ethernet - Internet.* M: Carl Hanser Verlag GmbH Co KG, 2014.

[9] Institute of Electrical and Electronics Engineers, Inc., "*IEEE 802.1: 802.1Q - Virtual LANs.*" http://www.ieee802.org/1/pages/802.1Q.html. zuletzt abgerufen am 23.12.2014.

[10] "IEEE standard for local and metropolitan area networks - virtual bridged local area networks amendment 12: Forwarding and queuing enhancements for time-sensitive streams," *IEEE Std 802.1Qav-2009 (Amendment to IEEE Std 802.1Q-2005)*, pp. C1–72, 2009.

[11] "IEEE standard for local and metropolitan area networks—virtual bridged local area networks amendment 14: Stream reservation protocol (srp)," *IEEE Std 802.1Qat-2010 (Revision of IEEE Std 802.1Q-2005)*, pp. 1–119, 2010.

[12] "IEEE standard for local and metropolitan area networks - timing and synchronization for time-sensitive applications in bridged local area networks," *IEEE Std 802.1AS-2011*, pp. 1–292, 2011.

[13] "IEEE standard for local and metropolitan area networks–audio video bridging (avb) systems," *IEEE Std 802.1BA-2011*, pp. 1–45, 2011.

[14] J. Engler, "Worst-Case-Timing unter AVB mit Network Calculus," in *Master Projekt 1*, Apr. 2015.

[15] J. Engler, "Evaluierung des AVB-Network-Calculus-Modells," in *Master Projekt 2*, Feb. 2016.

[16] Institute of Electrical and Electronics Engineers, Inc., "*IEEE 802.1 Time Sensitive Networking Task Group*." http://www.ieee802.org/1/pages/tsn.html. zuletzt abgerufen am 11.04.2016.

[17] Institute of Electrical and Electronics Engineers, Inc., "*IEEE 802.3 ETHERNET WORKING GROUP*." http://www.ieee802.org/3/. zuletzt abgerufen am 23.12.2014.

[18] "Standard for Local and metropolitan area networks–Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks Amendment: Frame Preemption," *802.1Qbu: Standard for Local and metropolitan area networks*, pp. 1–35, 2016.

[19] S. Thangamuthu, N. Concer, P. Cuijpers, and J. Lukkien, "Analysis of Ethernet-switch traffic shapers for in-vehicle networking applications," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2015*, pp. 55–60, March 2015.

[20] "Draft Standard for Local and Metropolitan Area Networks – Bridges and Bridged Networks – Amendment: Cyclic Queuing and Forwarding," *IEEE P802.1Qch/D0.1 Draft Standard for Local and metropolitan area networks – Bridges and Bridged Networks*, pp. 1–28, 2016.

[21] C. Boiger, "Time aware shaper in combination with qbu." Presentation on 802.1 TSN Task Group — March 2013, March 2013.

[22] J.-Y. Le Boudec and P. Thiran, *Network calculus: a theory of deterministic queuing systems for the internet*, vol. 2050. Springer, 2001.

[23] J.-P. Georges, T. Divoux, and E. Rondeau, "Strict Priority versus Weighted Fair Queueing in Switched Ethernet Networks for Time Critical Applications," in *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, pp. 141–141, April 2005.

[24] R. Queck, "Analysis of Ethernet AVB for automotive networks using Network Calculus," in *Vehicular Electronics and Safety (ICVES), 2012 IEEE International Conference on*, pp. 61–67, 2012.

[25] F. Reimann, S. Graf, F. Streit, M. Glas, and J. Teich, "Timing analysis of Ethernet AVB-based automotive E/E architectures," in *Emerging Technologies Factory Automation (ETFA), 2013 IEEE 18th Conference on*, pp. 1–8, Sept 2013.

[26] U. Bordoloi, A. Aminifar, P. Eles, and Z. Peng, "Schedulability analysis of Ethernet AVB switches," in *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2014 IEEE 20th International Conference on*, pp. 1–10, Aug 2014.

[27] D. Thiele, R. Ernst, and J. Diemer, "Formal worst-case timing analysis of Ethernet TSN's time-aware and peristaltic shapers," in *Vehicular Networking Conference (VNC), 2015 IEEE*, pp. 251–258, Dec 2015.

[28] D. Baum, *Grundlagen der Warteschlangentheorie.* Berlin Heidelberg New York: Springer-Verlag, 1. auflage ed., 2013.

[29] K.-D. Thies, *Elementare Einführung in die Wahrscheinlichkeitsrechnung, Informationstheorie und stochastische Prozesse mit Warteschlangentheorie fÃ¼r Computernetzwerke - Mit einer wahrscheinlichkeitstheoretischen Leistungsanalyse des Ethernet.* Aachen: Shaker Verlag, 1. aufl. ed., 2011.

[30] K.-S. J. Hielscher, "Dienstgüte-Garantien mit dem Network Calculus," 2015.

Tabellenverzeichnis

4.1 Paketeigenschaften im Worst-Case-Szenario 39
4.2 TSN-Paketklassen und Einflüsse auf ihre Latenz 42
4.3 Cycle-Piling-Paket-Szenario unter TSN 50
4.4 Datenmengen bei 100 Mbit/s Bandbreite für Pakete aus Tab. 4.3 50
4.5 Faires Scheduling unter TSN 51

5.1 Min-Plus-Algebra-Definitionen 57

7.1 Eingabemöglichkeiten der Benutzeroberfläche 76
7.2 Funktionen in der Simulationsauswahl 88
7.3 Funktionen der Netzwerkkonstruktion 88
7.4 Funktionen der Neuplanung 89
7.5 Funktionen der Ausgabe 89

8.1 Pakete im Netzwerk aus Abb. 8.1 96
8.2 Parameter für NC-Abschätzung in Netzwerk aus Abb. 8.1 98
8.3 Pakete im Worst-Case-Netzwerk 100
8.4 Parameter für NC-Abschätzung im Worst-Case-Netzwerk 101
8.5 Pakete im komplexen Netzwerk 104
8.6 Pakete im komplexen Netzwerk nach Generierung 104
8.7 Parameter für NC-Abschätzung im komplexen Netzwerk 105

Abbildungsverzeichnis

2.1 Standard-Ethernet-Paket-Aufbau (mit VLAN-Tag für 802.1Q-Ethernet) 4
2.2 Reservierte Zeitblöcke in einer Übertragung 6
2.3 Aufteilung der Warteschlangen in 802.1Q-Ethernet 7
2.4 Kredit und Paketversendung unter AVB [15] 8
2.5 Time Aware Shaper Konzept [21] 14

4.1 Beispiel eines Sendeablauf-Szenarios 20
4.2 Verzögerung durch BE-Pakete 23
4.3 AVB-Burst mit nicht vollständig differierenden Burst-Quellen. Paket y kommt aus der gleichen Quelle wie eines der anderen AVB-Pakete. Sendeverlauf ist ohne Pausen durch Kreditschwankungen skizziert. 25
4.4 Pausen zwischen AVB-Paketen durch Kreditschwankungen. 26
4.5 Herkunft von Burst-Pakten an einem Switch. 27
4.6 Aufeinanderfolgende Sendung von zwei AVB-Paketen eines Streams [15] . . . 28
4.7 Doppelte Pakete der Vorstation führen zu drei Paketen in Folge 29
4.8 Hintereinander geschickte AVB-Pakete werden durch Kredit verzögert. 30
4.9 Kreditschwankungen bei einzelnen reservierten AVB-Paketen. 30
4.10 Mehrere Burst-Pakete der gleichen Quelle. 32
4.11 Burst-Pakete aus zwei Quellen mit unterschiedlichen Größen 34
4.12 Burst-Pakete aus drei Quellen mit unterschiedlichen Größen 35
4.13 Burst-Pakete aus drei Quellen mit gleicher Größe 36
4.14 Burst-Pakete aus fünf Quellen mit gleicher Größe 36
4.15 Worst-Case-Szenario an einem AVB-Switch 37
4.16 Lücken zwischen TSN-Paketen sind nicht alle ausreichend groß 44
4.17 Zusammenfassung von TSN-Paketen eines Cycles 44
4.18 Blockierung eines Pakets durch TSN-Gate 45
4.19 TSN-Cycle groß genug: TSN-Paket am Ende des Worst-Case-Sendeverlaufs . . 47
4.20 TSN-Lücken zu klein: Aufteilung der Sendungen 47

4.21 Mehrere kleine TSN-Lücken: Kein Platz für BE 48

5.1 Network Calculus: Ein System S mit Input-Funktion $R(t)$, Output-Funktion $R^*(t)$ und dem zeitlichen Verlauf der gesendeten Daten dieser Flows [14] . . . 55
5.2 Min-Plus-Faltung von $f(t)$ und $g(t)$ 57
5.3 Network Calculus: Arrival-Curve $\alpha(t)$ und Input-Funktion $R(t)$ 59
5.4 Network Calculus: Service-Curve [22] 60

6.1 NC-System für ein AVB-Netzwerk 62
6.2 Arrival- und Service-Curve im AVB-NC-Modell 64
6.3 Arrival- und Service-Curve im TSN-NC-Modell 71

7.1 Einlesen von Streams und Switches 77
7.2 Einlesen von allen Netzwerkinformationen 80
7.3 Benutzeroberfläche des Simulationsgenerierungstools 83
7.4 Klassendiagramm des Tools (Infrastruktur) 85
7.5 Klassendiagramm des Tools (Pakete) 86
7.6 Klassendiagramm für Paketneuplanung 87
7.7 Geplante Pakete an NetworkDevice nach INI-Einlesen 93
7.8 Detailinformationen der Pakete nach INI-Einlesen 94

8.1 Evaluierung: Aufbau eines Minimal-Beispiel-Netzwerks 96
8.2 Evaluierung: Ausgangslatenz in Netzwerk aus Abb. 8.1 97
8.3 Evaluierung: Latenz nach Generierung in Netzwerk aus Abb. 8.1 97
8.4 Evaluierung: Aufbau des Worst-Case-Netzwerks 99
8.5 Evaluierung: Ausgangslatenz im Worst-Case-Netzwerk 100
8.6 Evaluierung: Latenz nach Generierung im Worst-Case-Netzwerk 101
8.7 Evaluierung: Komplexes Netzwerk 103
8.8 Evaluierung: Ausgangslatenz im komplexen Netzwerk 103
8.9 Evaluierung: Latenz nach Generierung im komplexen Netzwerk 105

Formelverzeichnis

4.1 Maximaler $idleSlope$ und $sendSlope$ . . . 31
4.2 Höchster Kreditwert . . . 31
4.3 Kreditsenkung durch AVB-Paket . . . 31
4.4 Maximale Anzahl AVB-Pakete, bis Kredit wieder 0 erreicht . . . 31
4.5 $idleSlope$ und $sendSlope$ an Vorgängerstation, mehr als ein Paket . . . 32
4.6 Anzahl Pakete, bis Kredit wieder 0 erreicht - Paket kleiner . . . 32
4.7 Minimaler $idleSlope$ und $sendSlope$ . . . 33
4.8 Minimale Anzahl AVB-Pakete, bis Kredit wieder 0 erreicht . . . 33
4.9 Raten für einen zweiten Burst-Stream in Vorgängerstation . . . 33
4.10 Anzahl Pakete des zweiten Burst-Stream, bis Kredit 0 erreicht . . . 33
4.11 Raten des größten Burst-Streams bei zwei weiteren Minimal-Streams . . . 34
4.12 Anzahl Pakete, bis Kredit 0 erreicht, bei großem Stream und zwei weiteren Streams 34
4.13 Cycle Piling: Zyklus 1 . . . 39
4.14 Cycle Piling: Zyklus 2 . . . 40
4.15 Cycle Piling: Zyklus 2 bei zwischenzeitigem Erreichen des Kreditwerts 0 . . . 40
4.16 Cycle Piling: Zyklus 2 ohne Erreichen des Kreditwerts 0 . . . 40
4.17 Cycle Piling: Zyklus 3 . . . 41
4.18 Cycle Piling: Zyklus 3 bei zwischenzeitigem Erreichen des Kreditwerts 0 . . . 41
4.19 TSN Worst Case: Abstände sämtlich groß genug . . . 43
5.1 NC: Min-Plus-Faltung . . . 58
6.1 AVB-NC: Arrival Curve . . . 64
6.2 AVB-NC: Service Curve . . . 64
6.3 AVB-NC: Worst-Case-Latenz nach 3-Grenzen-Theorem . . . 65
6.4 AVB-Worst-Case im Network Calculus, Teil 1 . . . 65
6.5 AVB-Worst-Case im Network Calculus, Teil 2 . . . 66
6.6 AVB-Worst-Case im Network Calculus, Teil 3 . . . 66
6.7 AVB-NC: Worst-Case-Delay für AVB-Stream . . . 66
6.8 TSN-NC: Spezialisierter Worst-Case-Delay für AVB-Stream . . . 69

6.9 TSN-NC: Arrival Curve 71
6.10 TSN-NC: Service Curve 72
6.11 TSN-Worst-Case im Network Calculus 72
6.12 TSN-NC: Allgemeiner Worst-Case-Delay für AVB-Stream 72
8.1 Evaluierung: NC-Abschätzung des Netzwerks aus Abb. 8.1 98
8.2 Evaluierung: Gesamtlatenz des Streams im kleinen Netzwerk 98
8.3 Evaluierung: NC-Abschätzung Worst-Case-Netzwerk 102
8.4 Evaluierung: Gesamtlatenz des Streams im Worst-Case-Netzwerk 102
8.5 Evaluierung: NC-Abschätzung Worst-Case-Netzwerk 106
8.6 Evaluierung: Gesamtlatenz des Streams im komplexen Netzwerk 106

Listings

7.1 Beispiel-INI-Datei „s1_1.ini“ 92

1 Routing von Paketen X
2 Aussortierung irrelevanter Crosstraffic-Streams XI
3 Neuplanung des Crosstraffics XIV

Anhang

1 Netzwerkerstellung

```
1 public LinkedList<OutputQueue> routePacket(NetworkDevice comingFrom,
      EthernetPacket packet, NetworkDevice destination, LinkedList<
     OutputQueue> currentRoute){
2                 if(this.getClass() == NetworkNode.class){       //
                      node
3                       NetworkNode tmpNode = (NetworkNode)this;
4                       currentRoute.addLast(
                            determineCorrectOutputQueue( tmpNode.
                            getConnectedSwitch()));
5                       return tmpNode.getConnectedSwitch().
                            routePacket(this, packet, destination,
                            currentRoute);
6                 }else{  //switch
7                       NetworkSwitch tmpSwitch = (NetworkSwitch)
                            this;
8                       if(tmpSwitch.isNodeConnectedToSwitch(
                            destination)){     //node is connected to
                             this switch
9                           currentRoute.addLast(
                                determineCorrectOutputQueue(
                                destination));
10                          return currentRoute;    //end of
                                recursion
11                      }else{  //node is not connected to this
                            switch -> search in connected switches
12                          LinkedList<OutputQueue>
                                tmpCurrentRoute = null;
13                          for (NetworkSwitch netwSwitch :
                                tmpSwitch.getConnectedSwitches())
                                 {
```

```
14                                          LinkedList<OutputQueue>
                                               tmpFurtherRouting = (
                                               LinkedList<OutputQueue>)
                                               currentRoute.clone();
15                                          tmpFurtherRouting.addLast(
                                               determineCorrectOutputQueue
                                               (netwSwitch));     //add
                                               routing to this switch
16                                          if(netwSwitch != comingFrom)
                                               {   //only look in
                                               switches not searched yet
17                                                  tmpCurrentRoute =
                                                       netwSwitch.
                                                       routePacket(this,
                                                        packet,
                                                       destination,
                                                       tmpFurtherRouting
                                                       );
18                                          }
19                                          //null = switch was not the
                                               correct one for further
                                               routing:
20                                          if(tmpCurrentRoute != null){
                                                   //if it's not null,
                                               the route has been
                                               completed
21                                                  break;
22                                          }
23                                  }
24                                  return tmpCurrentRoute; //null =
                                       false route
25                          }
26                  }
27          }
```

Listing 1: Routing von Paketen

2 Neuplanung Netzwerk

```
1 private void determineInterferingCrosstraffic(){
2                   AVBStream toBeAnalyzed = getAnalyzedAVBStream();
```

```
3                   OutputQueue investigatedQueue = getInvestigatedQueue
                        ();
4
5                   analyzedStream = new AnalyzedStream(toBeAnalyzed,
                        investigatedQueue, network.getHardwareDelay(),
                        network.getBandwidth());
6                   List<CrossTrafficPacket> crossTrafficPackets =
                        analyzedStream.getCrossTrafficPackets();
7                   List<OutputQueue> relevantQueues = new ArrayList<>()
                        ;
8
9                   //get all queues that are pointing to the
                        investigated switch
10                  Collection<NetworkDevice> networkDevices = network.
                        getNetworkDevices().values();
11                  for (NetworkDevice networkDevice : networkDevices) {
                            //iterate over devices
12                          for (OutputQueue outpQueue : networkDevice.
                                getOutputQueues()) { //iterate over
                                queues
13                                  if(outpQueue.getTo().getName().
                                        equals(switchName)){     //this
                                        queue is pointing to the
                                        investigated switch
14                                      relevantQueues.add(outpQueue
                                            );
15                                  }
16                          }
17                  }
18
19                  //get all streams in those queues and check if they
                        are contained in the crosstrafficPackets. Per
                        queue, only get the biggest crosstraffic packet
20                  for (OutputQueue relevantQueue : relevantQueues) {
21                          List<EthernetPacket> scheduledPackets =
                                relevantQueue.getScheduledPackets();
22                          CrossTrafficPacket chosenPacket = null; //
                                the packet with the biggest payload
23                          for (EthernetPacket packet :
                                scheduledPackets) {
```

```
24                                  //über ethernetpacket in den ct
                                       packets schauen ..
25                                  for (CrossTrafficPacket ctPacket :
                                       crossTrafficPackets) {
26                                       if(ctPacket.getPacket() ==
                                            packet){     //this
                                            packet is a crosstraffic
                                            packet
27                                            if(chosenPacket !=
                                                 null &&
                                                 chosenPacket.
                                                 getPacket().
                                                 getOverallSize()
                                                 < packet.
                                                 getOverallSize())
                                                 {
28                                                    chosenPacket
                                                         =
                                                       ctPacket;

                                                       //this
                                                       packet
                                                       has the
                                                       biggest
                                                       size
29                                            }else if (
                                                 chosenPacket ==
                                                 null){
30                                                    chosenPacket
                                                         =
                                                       ctPacket;

                                                       //this is
                                                        the
                                                       first
                                                       fitting
                                                       packet
31                                            }
32                                       }
33                                  }
```

```
34                          }
35                          if(chosenPacket != null)
                                interferingCrossTrafficPackets.add(
                                chosenPacket);
36                  }
37
38                  determineAndSetIdleSlope();     //set idleSlope in
                        CT packets
39          }
```

Listing 2: Aussortierung irrelevanter Crosstraffic-Streams

```
1 private void rescheduleCT() {
2                   //time taken by MTU BE:
3                   int overallsize = 1538; //in byte
4                   double bitsize = overallsize*8;
5                   double propagationTime = ((bitsize) / network.
                        getBandwidth());  //time in seconds for one hop
                        be
6
7                   //determine how much time is taken until every burst
                         packet and BE MTU was sent from the investigated
                         switch:
8                   //add up every ct package (burst packets twice):
9                   double timeForFullCT = 0.0;
10                  double timeForCurrentCT;
11                  for (CrossTrafficPacket crossTrafficPacket :
                        interferingCrossTrafficPackets) {
12                          if(crossTrafficPacket.getPacket().getClass()
                                .equals(AVBStream.class)){
13                                  timeForCurrentCT =
                                        crossTrafficPacket.
                                        getInvestigatedQueueWCTimeTaken()
                                        ;
14                          }else{
15                                  timeForCurrentCT = propagationTime;
                                            //full be mtu packet
16                          }
17                          timeForFullCT += timeForCurrentCT;
18                  }
```

```
19                  timeForFullCT += network.getHardwareDelay(); //+ hw
                        delay, because ct is forwarded inside
                        investigated switch
20
21
22                  //determine starting time, so ct is finished when
                        full 10 ms of the cycle have passed
23                  relativeStartingPoint = 0.01 - timeForFullCT;
24                  //add all crosstraffic packets to a new list of BE
                        and AVB packets
25                  //get propagation times of ct packages, subtract
                        prop time of BE MTU packet:
26                  for (CrossTrafficPacket crossTrafficPacket :
                        interferingCrossTrafficPackets) {
27                       if(crossTrafficPacket.getPacket().getClass()
                             .equals(AVBStream.class)){
28                            AVBStream op = (AVBStream)
                                 crossTrafficPacket.getPacket();
29                            double overallPropTimeForCT =
                                 propagationTime +
                                 crossTrafficPacket.
                                 getPropagationTime();
30                            //translate packet properties:
31                            AVBStream ctAVBStream = op;
32                            //starting time is relative to full
                                 the time the transmission of all
                                 burst packets on the investigated
                                  switch is taking
33                            double newStartTime =
                                 relativeStartingPoint -
                                 overallPropTimeForCT;
34                            ctAVBStream.setStaticSendTime(
                                 newStartTime);
35                            reScheduledAVBPackets.add(
                                 ctAVBStream);
36                            if(crossTrafficPacket.
                                 getNumberOfHops() == 1){
37                                 scheduleOneHopBE(ctAVBStream
                                      );
38                            }else{
```

```
39                                                  scheduleMultiHopBE(
                                                        ctAVBStream);
40                                          }
41                                  }
42                          }
43                  }
```

Listing 3: Neuplanung des Crosstraffics

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 30. September 2016 Jonas Engler