Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Soeren Rumpf

Ein ressourcenschonender Stack zur Unterstützung von AVB und Time Triggered Ethernet Verkehr

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Soeren Rumpf

Ein ressourcenschonender Stack zur Unterstützung von AVB und Time Triggered Ethernet Verkehr

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

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

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

Eingereicht am: 14. November 2014

Soeren Rumpf

Thema der Arbeit
Ein ressourcenschonender Stack zur Unterstützung von AVB und Time Triggered Ethernet Verkehr

Stichworte
Audio/Video Bridging, Time-Triggered Ethernet, Netzwerkstack, Fahrzeugnetz, TTEthernet

Kurzzusammenfassung
In den heutigen Automobilfahrzeugen werden mit zunehmender Sicherheit, Leistung und Komfort die Kommunikationsnetze immer komplexer. Um die wachsenden Datenmengen der Vielzahl von Kameras, Sensoren und Steuergeräten in einem Netzwerk übertragen zu können, reicht die Bandbreite von Übertragungsnetzen wie CAN, LIN, MOST und Flex-Ray nicht mehr aus. Aufgrund der Notwendigkeit den Bandbreitenanforderungen gerecht zu werden und auch zeitkritische Daten übertragen zu können, bietet das TTEthernet Protokoll eine Echtzeit-Lösung. In dieser Arbeit wird eine vorhandene hardware-basierte TTEthernet-Implementierung um das Audio/ Video Bridging (AVB) Protokoll erweitert, um den hohen Konfigurationsaufwand von TTEthernet zu verringern. Die Zusammenführung von AVB und TTEthernet sowie das Verhalten dieser Implementierung wird anhand von Tests verifiziert und analysiert.

Title of the paper
A resource-efficient stack to support AVB and time triggered ethernet traffic

Keywords
audio-video bridging, time-triggered ethernet, network stack, in-car network, TTEthernet

Abstract
In today's automotive-vehicles, the communication networks become more complex by increasing security, performance and comfort. To transfer the growing amount of data from the variety of cameras, sensors and controller, the bandwidth of transmission networks like CAN, LIN, MOST and Flex-Ray is insufficient. To cope with the bandwidth requirements and to transmit time-critical data, the TTEthernet protocol offers a real-time solution. In this thesis an existing hardware-based TTEthernet implementation is being extended by the audio/ video bridging (AVB) protocol to reduce the high configuration effort of TTEthernet. The combination of AVB and TTEthernet, as well as the behaviour of this implementation will be verified and analyzed based on tests.

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

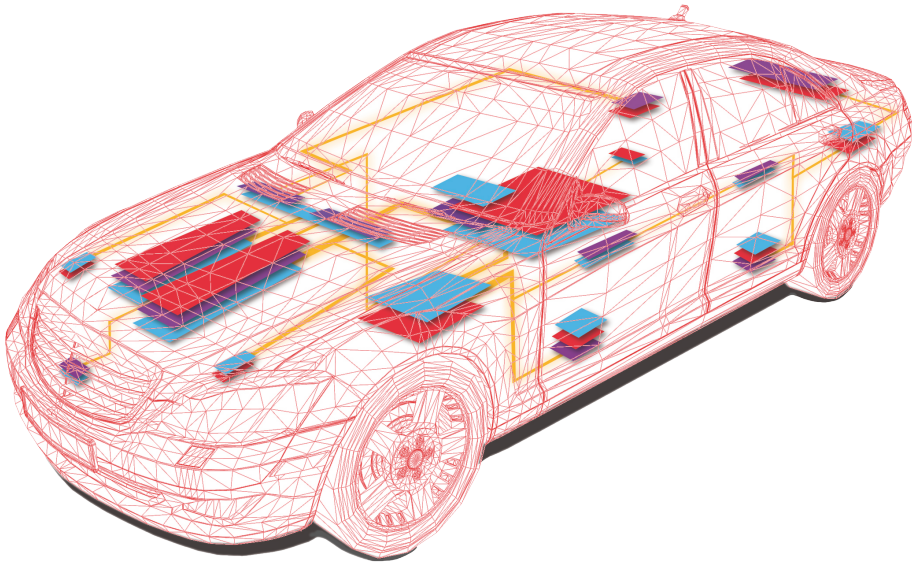

Abbildung 1.1.: Vernetzung elektronischer Komponenten in einem modernen Oberklassefahrzeug (Quelle: CoRE Arbeitsgruppe).

In heutigen Automobilfahrzeugen steigt mit zunehmender Sicherheit, Leistung und Komfort auch die Anzahl der darin verbauten technischen Geräte. Um neue Technologien ermöglichen zu können, verfügt ein Fahrzeug über eine Vielzahl von Sensoren, Kameras und Steuergeräten (siehe Abbildung 1.1). Dadurch werden die heutigen Kommunikations- und Bordnetze immer komplexer und die zu übertragenden Datenmengen steigen. Damit die verschiedenen sicherheitsrelevanten Daten übertragen werden können, kommen mehrere Übertragungsnetze wie CAN, LIN, MOST und Flex-Ray zum Einsatz. Da deren Bandbreite für die Übertragung und Verarbeitung von zum Beispiel mehreren Kamerabildern und den zukünftigen Infotainment-Systemen nicht ausreichen, wird ein Netzwerk mit einer höheren Bandbreite benötigt. Ein

solches standardisiertes Netzwerk ist das heutzutage weit verbreitete Standard Ethernet IEEE 802.3. Um ein solches Netzwerk im Automobil nutzen zu können, muss gewährleistet sein, dass sicherheitsrelevante Informationen, wie beispielsweise das Auslösen des Airbags, in Echtzeit übertragen werden und zuverlässig ankommen. Ein echtzeitfähiges Protokoll mit hoher Bandbreite ist das von der Firma TTTech entwickelte Time-Triggered-Ethernet (TTEthernet) Protokoll (vgl. Steiner (2008)). Dieses Protokoll geht mit einem hohen Konfigurationsaufwand einher, der bei zunehmender Kommunikationsnetzkomplexität steigt. Um den Aufwand zu verringern, gibt es Protokolle wie das Audio/ Video Bridging (im weiteren Verlauf AVB genannt).

Mit dem Erweitern des AVB-Standards um zeitbasierte Kommunikation beschäftigen sich zur Zeit Automobilhersteller und die IEEE TSN Task Group (vgl. IEEE 802.1 TSN Task Group (a) und IEEE 802.1 TSN Task Group (b)). TSN steht für Time Sensitive Networking, das seit November 2012 den Nachfolger von AVB bildet. Die TSN Task Group und die Automobilhersteller arbeiten an einem Konzept, noch geringere Latenzen unter 2 ms zu erreichen und eine Preemption von Frames zu ermöglichen, damit sichergestellt werden kann, dass auch die zeitkritische Kommunikation im Auto nicht zu spät übermittelt wird.

1.1. Zielsetzung der Arbeit

Das Hauptziel dieser Arbeit ist die Entwicklung eines ressourcenschonenenden Stacks zur Unterstützung von AVB und Time Triggered Ethernet Verkehr. Diese Arbeit ist in der „Communication over Realtime Ethernet“ Arbeitsgruppe (CoRE RG) an der HAW-Hamburg entstanden. Als Basis für die Umsetzung dieser Arbeit dient eine bestehende Implementierung eines echtzeitfähigen Time Triggered Ethernet Stacks aus der genannten Arbeitsgruppe. Dieser Stack soll durch das Implementieren von AVB-Verkehr erweitert werden, so dass Time Triggered Ethernet und AVB-Verkehr zusammengeführt werden. Das Hauptziel setzt sich aus zwei Teilen zusammen. Im ersten Teil der Arbeit wird die bestehende Time Triggered Ethernet-Implementierung (vgl. Society of Automotive Engineers - AS-2D Time Triggered Systems and Architecture Committee (2011)) um AVB Komponenten erweitert. Im zweiten Teil wird die Implementierung und Zusammenführung der beiden Protokolle geprüft und analysiert.

Da die TSN Task Group an einem Ansatz arbeitet, Time Triggered Verkehr als zusätzliche Nachrichtenklasse für AVB zu spezifizieren, wird in dieser Arbeit untersucht welche Veränderungen dafür an dem AVB Standard nötig sind. Außerdem soll eine Probeimplementierung des Ansatzes umgesetzt werden. Diese soll anschließend auf bestimmte Testfälle überprüft und somit sichergestellt werden, dass der Ansatz sich wie geplant verhält und funktioniert.

1.2. Aufbau der Arbeit

Die Arbeit ist wie folgt gegliedert: In Kapitel 2 wird das grundlegende Wissen über die verwendeten Protokolle erklärt, das für den weiteren Verlauf der Arbeit notwendig ist. Kapitel 3 gibt einen Überblick über die für diese Arbeit verwendete Hardware und deren Funktionen. Das vierte Kapitel beleuchtet verwandte Arbeiten, die mit dem Thema dieser Arbeit in Zusammenhang stehen und interessante Ansätze zeigen. In Kapitel 5 werden die Anforderungen an diese Arbeit analysiert und das daraus resultierende Konzept erklärt. Weiterhin wird in Kapitel 6 die Implementierung und Evaluierung des Konzepts beschrieben und getestet. Abschließend wird im letzten Kapitel ein Fazit der Arbeit, sowie ein Ausblick über zukünftige Arbeiten gegeben.

2. Grundlagen

In diesem Kapitel werden die Grundlagen für das weitere Verständnis der Arbeit vermittelt. Um die Zielsetzung zu erreichen, wird die grundlegende Funktionsweise von Audio/Video Bridging und Time-Triggered Etherenet erläutert.

2.1. Audio/ Video Bridging

Audio/ Video Bridging (AVB) wurde von der AVB Task Group der IEEE 802.1 Working Group entwickelt und setzt sich aus einer Reihe von Standards zusammen, die in dem IEEE 802.1BA Standard zusammengefasst sind (vgl. Institute of Electrical and Electronics Engineers (2011b)). Durch die Standards wird ein synchrones und priorisiertes Streamen von Audio- und Videodaten mit sehr geringer und garantierter Latenzzeit in einem IEEE 802.3 Netzwerk mit Full-duplex Betrieb ermöglicht. Im Zuge von Erweiterungen und um eine noch niedrigere Latenzzeit zu erreichen, hat sich die AVB Task Group im November 2012 zur Time-Sensitive Networking (TSN) Task Group umbenannt, worauf in Kapitel 4.2 genauer eingegangen wird.

Grundlegende Begriffe anhand eines Beispielnetzwerks erklärt

Abbildung 2.1 zeigt ein AVB Netzwerk mit einer vereinfacht dargestellten Kommunikation. Das Netzwerk besteht aus einem Talker (Sender), zwei Bridges und zwei Listenern (Empfängern). Wenn der Talker einen Stream anbietet, leitet die Bridge das Angebot an die beiden Listener weiter. Das Angebot können die Listener annehmen, indem sie eine Bestätigung an die Bridge senden. Die Bestätigungen der Listener fasst die Bridge zu einer Nachricht zusammen und die benötigte Bandbreite wird reserviert. Diese Nachricht signalisiert dem Talker, dass ein oder mehrere Empfänger bereit sind und er mit der Übertragung beginnen kann. Um bei Übertragungen eine garantierte Latenzzeit zu ermöglichen, hat die AVB Task Group zwei Nachrichtenklassen spezifiziert. Diese sogenannten Stream Reservation (SR) Klassen werden als Klasse A und Klasse B unterschieden. Klasse A Nachrichten haben die höchste Priorität und garantieren eine maximale Latenz von 2 ms über sieben Knoten. Die Nachrichten der Klasse B

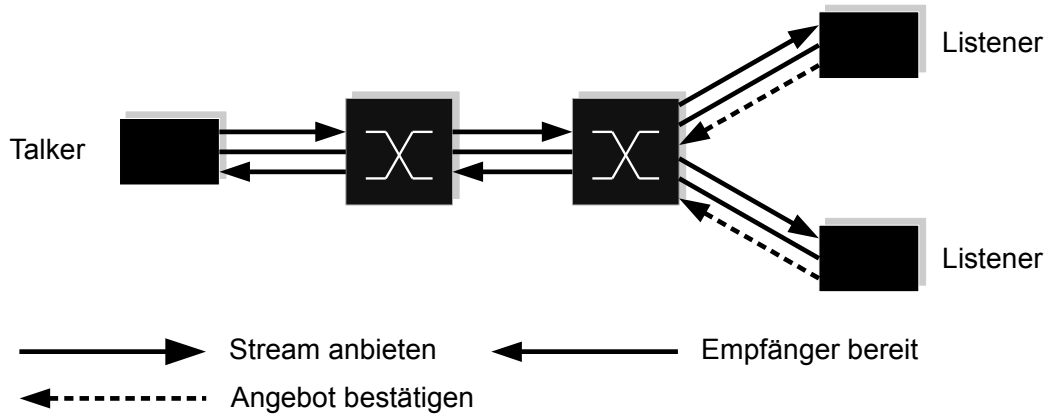

Abbildung 2.1.: Vorgang bei Bekanntgabe und Reservierung eines Streams in einem AVB Netzwerk.

haben die zweit höchste Priorität und garantieren eine maximale Latenz von 50 ms über sieben Knoten, weshalb Klasse A bevorzugt für zeitkritische Übertragungen verwendet wird.

2.1.1. AVB Standards und deren Funktionen

Um den zuvor beschrieben Betrieb ermöglichen zu können, werden die Standards kurz erläutert, bevor in den nachfolgenden Unterkapiteln darauf näher eingegangen wird.

- IEEE 802.1AB - Station and Media Access Control Connectivity Discovery

  Damit die von AVB garantierten Latenzen eingehalten werden können, beschränkt sich die Kompatibilität von AVB auf Teilnehmer, die die nötigen Ressourcen mit sich bringen. Um AVB-kompatible Teilnehmer von denen, die nicht kompatibel sind, trennen zu können, wird das Link Layer Discovery Protocol in dem IEEE 802.1AB Standard beschrieben.

- IEEE 802.1AS - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks

  Der IEEE 802.1AS Standard beschreibt unter anderem das Synchronisierungsmodell generalized Precision Time Protocol (gPTP), welches auf dem Precision Time Protocol (PTP) aus IEEE 1588 Standard aufbaut.

- IEEE 802.1BA - Audio Video Bridging (AVB) Systems

Die Architektur von AVB, die Bedeutung einer Domain und grundlegende Informationen, wie zum Beispiel die zulässige Verzögerungszeit eines Klasse A Paketes, aus den für AVB relevanten Standards, werden in dem IEEE 802.1BA Standard erläutert.

- IEEE 802.1Qat - Stream Reservation Protocol (SRP)

  Der IEEE 802.1Qat Standard beschreibt welche Protokolle für eine AVB Stream Reservierung benötigt werden. Der Standard hat für diese Arbeit eine hohe Relevanz, da ohne eine Stream Reservierung keine dynamische Stream-Übertragung realisiert werden kann. In Abschnitt 2.1.4 wird darauf ausführlicher eingegangen.

- IEEE 802.1Qav - Forwarding and Queuing Enhancements for Time-Sensitive Streams

  Um dynamische Streams priorisieren und die Pakete ohne Engpässe in dem Netzwerk übertragen zu können, wird in dem IEEE 802.1Qav Standard festgelegt, wie mithilfe des Credit Based Shaper (CBS) Algorithmus und einer für jeden Stream reservierten Bandbreite, eine Übertragung ermöglicht werden kann. Da AVB und Time Triggered Ethernet Verkehr nebeneinander funktionieren sollen, ist dieser Standard von hoher Relevanz für diese Arbeit.

- IEEE 1722 - Layer 2 Transport Protocol for Time-Sensitive Applications in Bridged Local Area Networks

  Der IEEE 1722 Standard beschreibt ein Protokoll zur Übertragung von Audio/ Video Daten und Zeit Informationen die zur Synchronisierung der Daten dienen. Dieser Standard ist der Vollständigkeit halber aufgeführt und wird im weiteren Verlauf dieser Arbeit nicht näher erläutert.

- IEEE 1733 - Layer 3 Transport Protocol for Time-Sensitive Applications in Local Area Networks

  Um medienspezifische Kontroll- und Informationsmöglichkeiten für AVB Streams ermöglichen zu können, wurde der IEEE 1733 Standard festgelegt. Dieser Standard ist der Vollständigkeit halber aufgeführt und wird im weiteren Verlauf dieser Arbeit nicht näher erläutert.

In Abbildung 2.2 wird die Verteilung der Standards auf die Netzwerkschichten (Layer) verdeutlicht. Protokolle, die in der Ethernet Sektion eingeteilt sind, beziehen sich auf die Schichten 1 und 2. Das 1733 Protokoll sowie die Streaming-Media-API, setzen auf den Schichten 3 und 4 auf, welche unter anderem TCP/IP beinhalten.

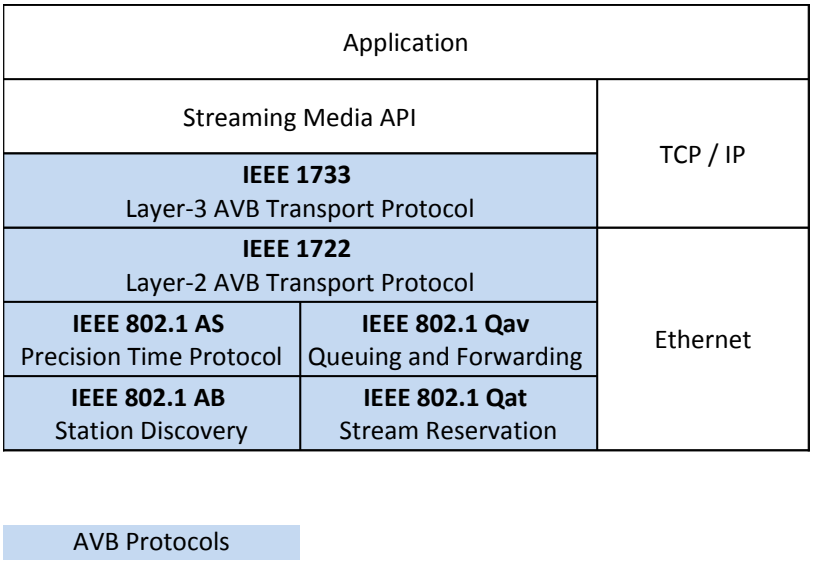

Abbildung 2.2.: Verteilung der AVB Protokolle auf die Layer-Schichten.

2.1.2. Erkennung von AVB-fähigen Teilnehmern durch das Link Layer Discovery Protocol

Das von AVB garantierte Verhalten gilt innerhalb eines Netzwerks nur für AVB-fähige Geräte. Um diese Endknoten oder Bridges erkennen zu können, wird das Link Layer Discovery Protocol (LLDP) aus IEEE 802.1AB (vgl. Institute of Electrical and Electronics Engineers (2009a)) verwendet. Damit ein Gerät als AVB-fähig gilt, ist die minimale Anforderung eine Übertragungsgeschwindigkeit von 100 Mbit/s, sowie Full-duplex Betrieb. Beim Start eines AVB Geräts sendet es Informationen über seine Ressourcen an seine Nachbarn. Wenn ein Gerät die minimalen Anforderungen erfüllt, wird es Teil der AVB-Wolke, die in Abbildung 2.3 beispielhaft visualisiert ist. Die Datenübertragung eines AVB-Streams verläuft dann von einem Talker (Sender) zu einem oder mehreren Listenern (Empfängern) und wird in regelmäßigen Zeitabständen ausgeführt. Die Garantie der Latenzzeiten gilt für Endgeräte innerhalb der AVB-Wolke und setzt eine maximale Anzahl von 7 Knoten zwischen den jeweiligen Endknoten voraus. Der Vollständigkeit halber mit aufgeführt, ist dieser Standard im weiteren Verlauf der Arbeit nicht mehr relevant.

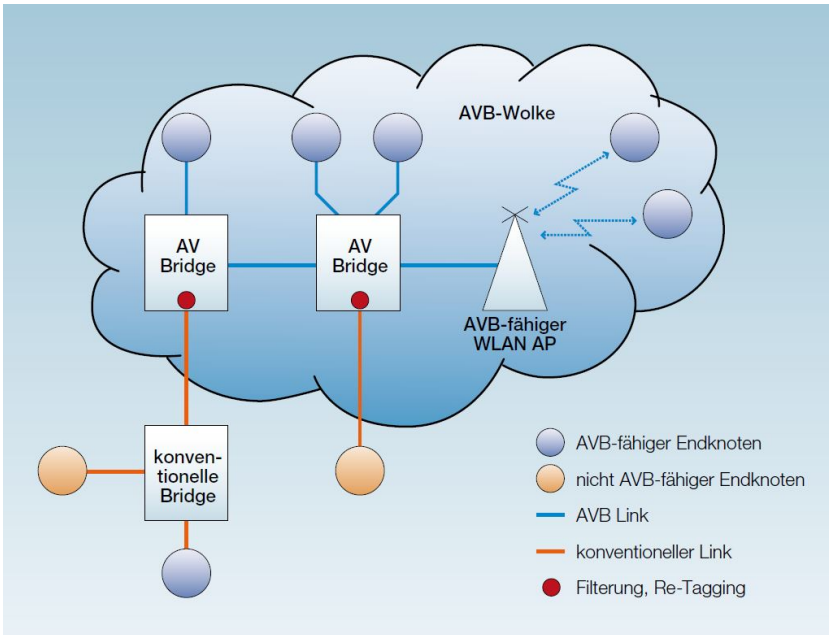

Abbildung 2.3.: Beispiel einer AVB-Wolke (Quelle: Weibel (2008)).

2.1.3. Synchronisation

Damit bei der Zeitsynchronisation eine Genauigkeit von unter 1 µs erreicht werden kann, wird eine Erweiterung des in IEEE 1588 beschriebenen Precision Time Protocol (PTP) verwendet. Diese Erweiterung ist in IEEE 802.1AS (vgl. Institute of Electrical and Electronics Engineers (2011a)) als generalized Precision Time Protocol (gPTP) definiert. Die zeitliche Genauigkeit wird auf der Anwendungsebene benötigt, um bei Audio/ Video Übertragungen, den Ton synchron zum Video wiedergeben zu können. Da die Synchronisation kein Bestandteil für den weiteren Verlauf der Arbeit ist, wird darauf nicht näher eingegangen.

2.1.4. Das Stream Reservation Protocol und das Reservieren von Bandbreite

Um eine Stream-Reservierung wie in dem Beispiel Netzwerk aus Abbildung 2.1 zu ermöglichen, werden die im IEEE 802.1Qat Standard (vgl. Institute of Electrical and Electronics Engineers (2010)) beschriebenen Protokolle verwendet:

- Multiple Registration Protocol (MRP)
- Multiple VLAN Registration Protocol (MVRP)
- Multiple MAC Registration Protocol (MMRP)

- Multiple Stream Registration Protocol (MSRP)

MRP ist die Basis für alle anderen Protokolle von SRP. Es hält sich durch zyklische Updates aktuell, läuft auf Switches und Hosts und wird in Software implementiert. Es erlaubt den Teilnehmern ihre Attribute bekannt zu geben oder diese zu verwerfen, so dass andere Teilnehmer diese Änderungen mitbekommen und sie übernehmen können. Die Änderungen werden anhand zweier State Machines ausgewertet.

Das MVRP bringt spezifische Attribute für das MRP mit sich und baut darauf auf.

Durch das MMR Protokoll wird der Einsatz von Filtering Databases in Bridges ermöglicht. Die empfangenen Pakete werden dadurch nur an Gruppenteilnehmer weitergeleitet, die sich zuvor über MMRP für die spezifische Gruppe registriert haben.

Das Multiple Stream Registration Protocol (MSRP) wird dazu verwendet, um Ressourcen in einem Netzwerk zu reservieren. Durch eine erfolgreiche Reservierung kann die Übertragung und der Empfang von Stream Daten garantiert werden.

Das Stream Reservation Protocol verwendet die zuvor erläuterten Protokolle, um die benötigten Ressourcen für einen Stream zu reservieren, einen Stream bekannt zu geben und um einen verfügbaren Stream zu empfangen.

Szenarien zum Reservieren von Bandbreite

Damit ein Talker einen Stream übertragen kann, muss die dafür benötigte Bandbreite über den Pfad vom Talker zum Listener verfügbar sein. Wenn ein Talker seinen Stream bekannt geben möchte, tut er dies indem er ein „Talker Advertise“-Paket mit Informationen über seinen Stream an eine dafür spezifizierte Multicast-Adresse sendet. In diesem Paket sind Informationen

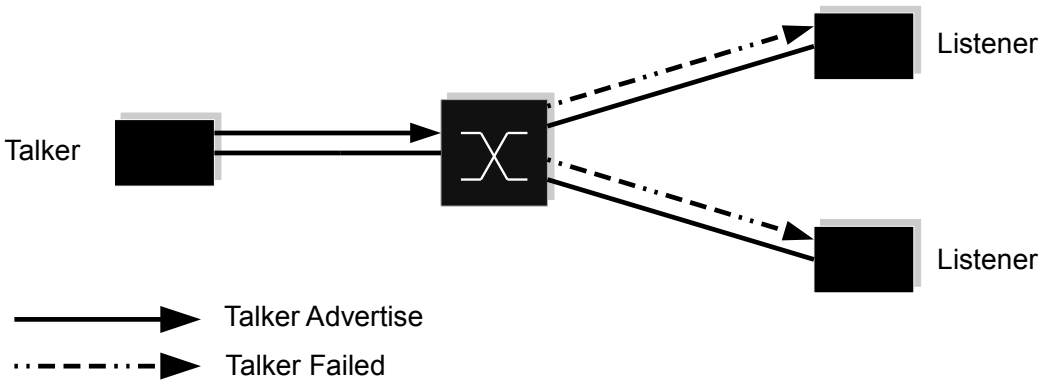

Abbildung 2.4.: Nicht erfolgreiches Talker Advertise.

über die Größe der Stream-Pakete, die Intervalllänge („Class Measurement“ Intervall) und die Häufigkeit, mit der Pakete innerhalb eines solchen Intervalls gesendet werden sollen. Anhand dieser Informationen kann die dafür benötigte Bandbreite berechnet werden. Die nächsten Bridges, die diese Pakete erhalten, überprüfen, ob auf dem Weg zu den nächsten Knoten beziehungsweise den Listenern genügend Ressourcen vorhanden sind, um die Bandbreite reservieren zu können. Ist dies nicht der Fall, so wird aus dem „Talker Advertise“-Paket ein „Talker Failed“-Paket, welches an die Listener gesendet wird. In diesem „Talker Failed“-Paket sind Fehlerinformationen wie die Identifikationsnummer der Bridge welche die Anfrage abgewiesen hat sowie ein Fehler-Code. Dieser Vorgang ist in Abbildung 2.4 dargestellt.

Das folgende Szenario bezieht sich auf Abbildung 2.5. Wenn die benötigten Ressourcen verfügbar sind und der Listener das „Talker Advertise“-Paket erhalten hat, kann dieser entscheiden, ob er den Stream empfangen möchte. Wenn er dies möchte, sendet der Listener ein „Listener Join“ an die Bridge zurück. Diese Bestätigung wandelt die Bridge in ein „Listener Ready“-Paket. Dieses Paket wird von der Bridge zurück zum Talker übertragen und die für den Stream benötigten Ressourcen werden reserviert. Durch das Empfangen des „Listener Ready“-Pakets erhält der Talker die Information, dass ein oder mehrere Listener bereit sind, seinen Stream zu empfangen. Anschließend kann der Talker mit der Übertragung beginnen.

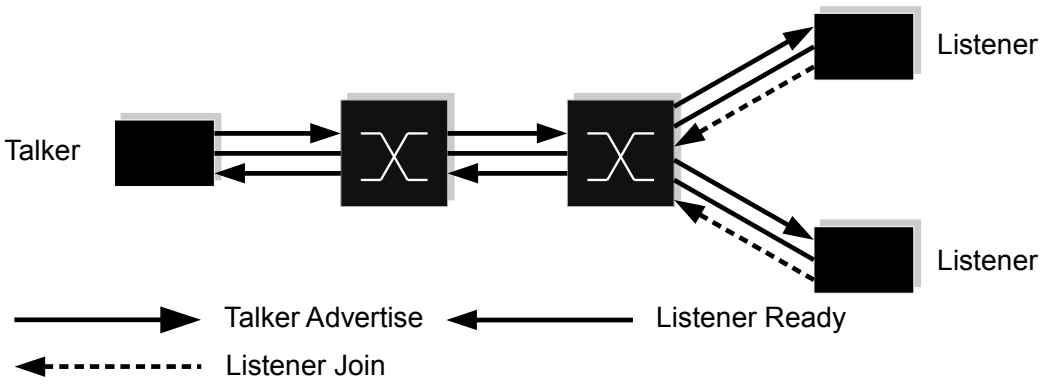

Abbildung 2.5.: Erfolgreiches Talker Advertise.

Das Netzwerk in Abbildung 2.6 zeigt eine nur teilweise erfolgreiche „Talker Advertise“-Propagierung. Der Pfad vom Talker zu Listener 1 weist die benötigten Ressourcen auf. Listener 1 kann also mit einem „Listener Join“ signalisieren, dass er den Stream empfangen möchte, was die Bridge 2 mit einem „Listener Ready“-Paket an Bridge 1 weiterreicht. Auf der Strecke zu Listener 2 gibt es jedoch ungenügende Ressourcen, weshalb Bridge 1 das „Talker Advertise“-Paket in ein

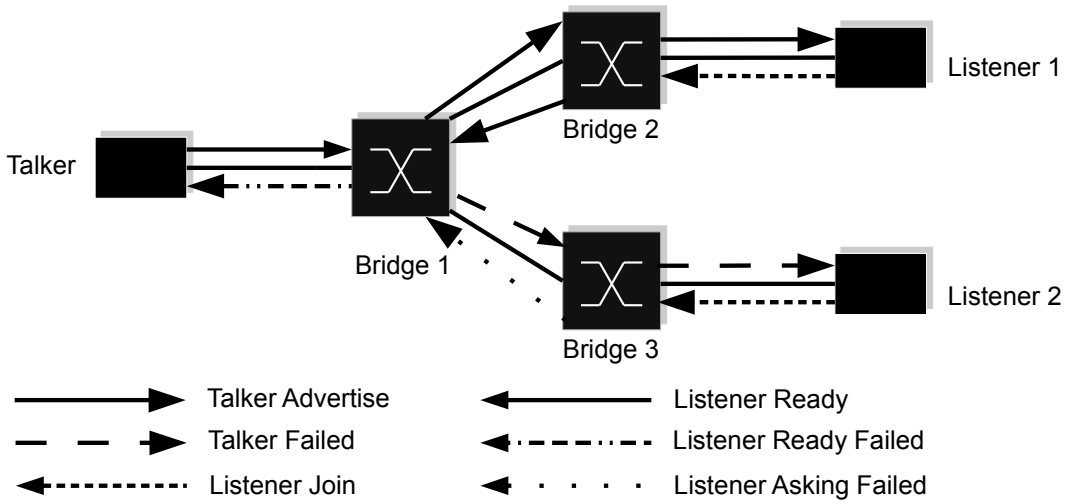

Abbildung 2.6.: Bedingt erfolgreiches Talker Advertise.

„Talker Failed“-Paket ändert, welches bis zu Listener 2 durchgereicht wird. Dieser kann durch ein „Listener Join“ an Bridge 3 zwar bekannt geben, dass er den Stream trotzdem empfangen möchte, jedoch wandelt Bridge 3 dies in ein „Listener Asking Failed“ um und leitet es an Bridge 1 weiter. Da Bridge 1 nun ein „Listener Ready“ und ein „Listener Asking Failed“ erhalten hat, führt es die beiden Nachrichten zusammen. Daraus resultiert ein „Listener Ready Failed“, welches dem Talker signalisiert, dass mehrere Listener seinen Stream empfangen möchten, jedoch nicht auf allen Pfaden ausreichend Ressourcen vorhanden sind. Da jedoch mindestens ein Listener seinen Stream empfangen kann, wird die Bandbreite reserviert und der Talker kann mit der Übertragung beginnen. Für den Talker macht es keinen wirklichen Unterschied ob er ein „Listener Ready“ oder ein „Listener Ready Failed“ erhält. Diese Informationen können jedoch für höhere Protokollebenen nützlich sein, um die Vorgänge nachzuvollziehen und gegebenenfalls darauf reagieren zu können. In einem AVB Netzwerk sollte nicht mehr als 75% der zur Verfügung stehenden Bandbreite für AVB-Übertragungen reserviert werden. Damit das Netzwerk nicht unnötig durch inaktive Endgeräte, die Bandbreite reserviert haben, ihre Registrierung jedoch nicht rückgängig gemacht haben blockiert wird, ist es erforderlich, dass in bestimmten Zeitabständen die Reservierungen erneuert werden, bevor diese vom MRP verworfen werden.

2.1.5. Weiterleiten und Queuing von AVB und Best Effort Paketen

Um beim Queuing und Scheduling in einem AVB Netzwerk den Nachrichtenklassen A und B die höchsten Prioritäten zuweisen zu können, werden von den acht Prioritätsstufen aus IEEE 802.3 die beiden höchsten Prioritäten dem Klasse A und B Verkehr zugeteilt. Um Best Effort Pakete untereinander verschieden zu priorisieren, können die verbleibenden Prioritätsstufen 5-0 genutzt werden. Jeder Nachrichtenpriorität wird, wie im IEEE 802.1Qav Standard (vgl.

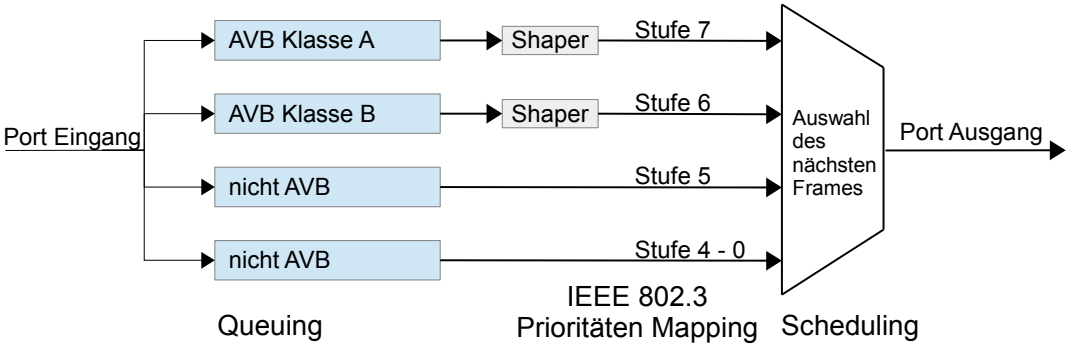

Abbildung 2.7.: Priorisierungsschema für Queuing und Scheduling.

Institute of Electrical and Electronics Engineers (2009b)) beschrieben, eine seperate Queue zugewiesen. Dies gilt für AVB- und Best-Effort- (BE) Nachrichten. Abbildung 2.7 zeigt das Priorisierungsschema des Queuing und Scheduling einer AVB Bridge. Die AVB Frames werden von einem Credit Based Shaper (im weiteren Verlauf CBS genannt) Algorithmus gescheduled. Nach dem Shaper werden alle Frames basierend auf ihren Prioritäten ausgewählt, bevor letztlich die Nachricht mit der höchsten Priorität auf den Port Ausgang gelangt.

Das „Class Measurement“-Intervall (im weiteren Verlauf CMI genannt), spezifiziert die Länge eines Intervalls, in dem die reservierte Bandbreite nicht überschritten werden darf. Damit dies nicht geschieht, wird mithilfe des CBS-Algorithmus dafür gesorgt, dass innerhalb des CMI die Bandbreitenreservierung eingehalten wird. Das CMI hat eine Länge von 125 µs bei Klasse A und 250 µs bei Klasse B. Je nach Übertragungsgeschwindigkeit bedeutet dies, dass innerhalb eines CMI nur 75% der verfügbaren Bits übertragen werden dürfen, welches die maximale AVB-Frame-Größe festlegt.

Durch den „Credit Based Shaper“-Algorithmus wird sichergestellt, dass die vom „Stream Reservation Protocol“ reservierten Bandbreiten eingehalten werden. Ausgehende Pakete werden in die in Abbildung 2.7 dargestellte AVB Klasse A Queue gepackt.

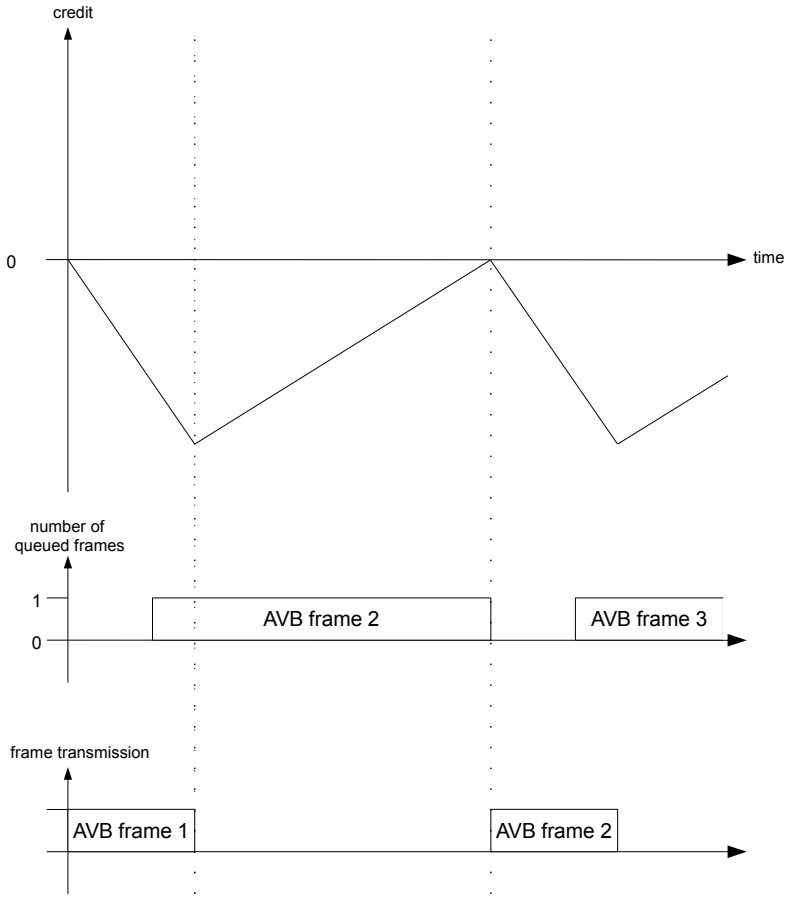

Abbildung 2.8.: Funktionsweise des CBS Algorithmus.

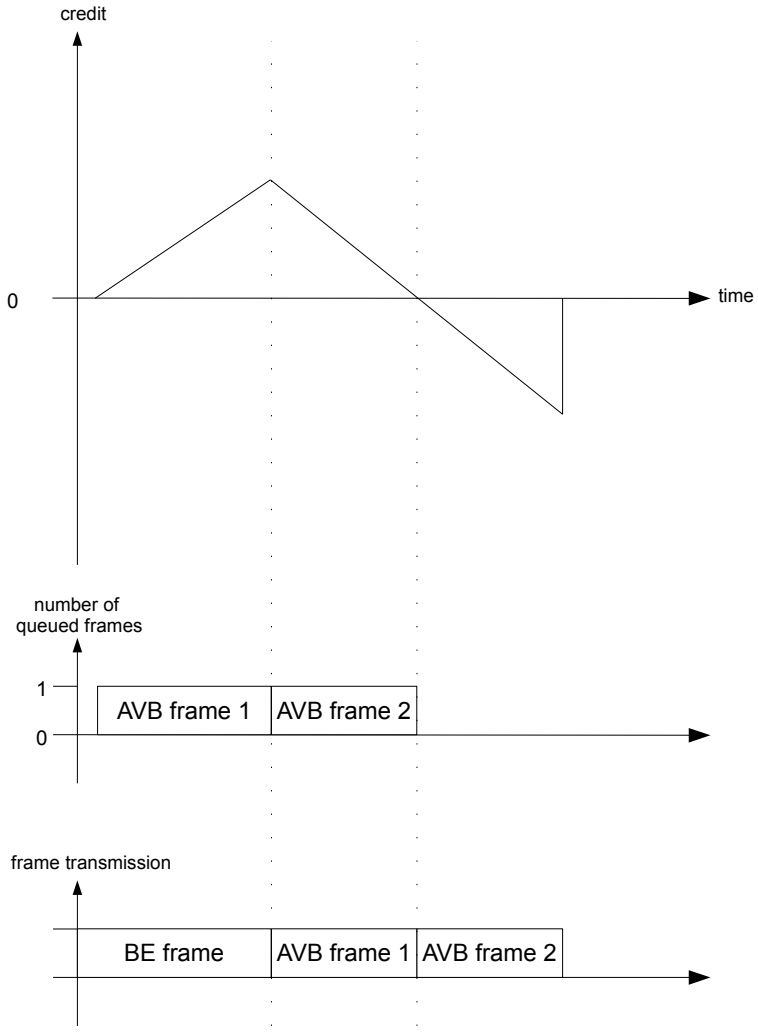

Abbildung 2.9.: CBS Algorithmus bei Verzögerung durch BE Frame.

Der Shaper hinter der jeweiligen Queue funktioniert nach einem Art Kreditsystem, welches anhand Abbildung 2.8 nachfolgend erklärt wird. Wenn ein AVB Paket gesendet wird, fällt der Zähler des Kreditsystems ins Negative. Wenn das gesendete Paket übertragen wurde und ein weiteres Paket darauf wartet gesendet zu werden, inkrementiert sich der Zähler des Kreditsystems bis er bei null angelangt ist und das wartende Paket übertragen werden kann. Das Fallen des Kreditwerts bezeichnet man als $sendSlope$ und das Steigen als $idleSlope$. Die beiden Formeln kann man aus der physikalischen Bandbreite (B) und der reservierten Bandbreite (RB) errechnen.

$$sendSlope = RB - B \tag{2.1}$$

$$idleSlope = RB \tag{2.2}$$

Anstehende AVB Pakete in der Queue dürfen gesendet werden, wenn der Zähler des Kreditsystems $\geq$ null ist. Müssen AVB Pakete, wie in Abbildung 2.9 veranschlaulicht, warten, weil ein Best Effort Paket gerade gesendet wird, erhöht sich der Zähler des Kreditsystems um $idleSlope$. Wenn das BE Paket versendet wurde, wird das AVB Paket gesendet und der Zähler des Kreditsystems um $sendSlope$ verringert. In dem Beispiel aus der vorher erwähnten Abbildung kann aufgrund des Zählers, der nach dem Übertragen von AVB Frame 1 genau auf null ist, direkt danach AVB Frame 2 versendet werden, wodurch der Zähler ins Negative fällt. Weil sich nach dem Versenden von AVB Frame 2 kein Paket mehr in der Warteschlange befindet, wird der Zähler, unabhängig davon ob er gerade im positiven oder negativen Bereich ist, auf null gesetzt.

2.2. Time Triggered Ethernet

Neben mehreren Echtzeit Erweiterungen für Standard Ethernet, wie zum Beispiel PROFINET (PROFIBUS & PROFINET International), ist Time-Triggered Ethernet (im weiteren Verlauf auch TTEthernet genannt) ein von der Firma TTTech Computertechnik AG entwickelte Echtzeit Erweiterung für Ethernet-basierte Kommunikation. Diese wurde in dem SAE AS6802 Standard (vgl. Society of Automotive Engineers - AS-2D Time Triggered Systems and Architecture Committee (2011)) standardisiert. Es ermöglicht Übertragungen mit niedriger Latenz und Jitter.

2.2.1. Nachrichtenklassen

TTEthernet bringt drei verschiedene Nachrichtenklassen mit sich: Time-Triggered (TT), Rate Constrained (RC) und Best-Effort (BE). Den Inhalt von Nachrichten aus höheren Netzwerk-

schichten behandelt TTEthernet transparent. Dies ermöglicht zum Beispiel die Verwendung von TCP und IP. TTEthernet Nachrichten werden über den Standard IEEE 802.3 übertragen.

Time-Triggered-Nachrichten

Time-Triggered Kommunikation hat die höchste Priorität. Die Sende- und Empfangszeitpunkte werden offline Mikrosekunden-genau geplant und zyklisch verschickt. Jedem Teilnehmer wird ein dedizierter Übertragungsslot zugewiesen. Diese koordinierte time-division-multiple-access (TDMA) multiplexing Strategie erlaubt eine deterministische Übertragung mit vorhersehbarer Verzögerung. Bei einem korrekten Schedule vermeidet die Strategie Anstauungen auf der Sendeseite und ermöglicht eine synchrone Kommunikation mit niedriger Latenz und Jitter. Um das TDMA-Konzept umsetzen zu können, ist ein ausfallsicheres Synchronisationsprotokoll mit einer Genauigkeit von $\pm 1\,\mu s$ nötig, welches als globale Zeit von allen Teilnehmern implementiert wird. Die Routen über die physikalischen Links sind statisch in Hosts und Switches festgelegt.

Rate-Constrained-Nachrichten

Rate-Constrained (RC) Nachrichten haben nach TT Nachrichten die zweithöchste Priorität. Bei dieser Nachrichtenklasse werden die Routen und die Bandbreite offline geplant. Die Übertragung ist event-basiert und damit unabhängig von der globalen Zeit. Abhängig vom Nachrichtenaufkommen erhöht sich Latenz und Jitter. Da RC Nachrichten bandbreiten-basiert sind und eine Bandbreite garantiert wird, sind die Charakteristiken von RC Nachrichten vergleichbar mit AVBs Stream Reservation (SR)-Klassen A und B.

Best-Effort-Nachrichten

Best-Effort (BE) entspricht den Standard IEEE 802.3 Ethernet Nachrichten, die mit der niedrigsten Priorität übertragen werden. Es gibt keine Garantie dafür, ob eine BE-Nachricht übertragen wird. Zusätzlich kann aufgrund der niedrigen Priorität keine feste Aussage über Latenz und Jitter getroffen werden. BE-Nachrichten werden mit der restlichen, von TT und RC nicht genutzten, Bandbreite übertragen. Dies ermöglicht andere Hosts in dem selben Netzwerk, die das Echtzeit Protokoll nicht kennen und dadurch unsynchronisiert bleiben. Diese Hosts können ihre Nachrichten über Best-Effort übertragen. Abbildung 2.10 zeigt die Medienzugriffsstrategie für Nachrichten verschiedener Klassen.

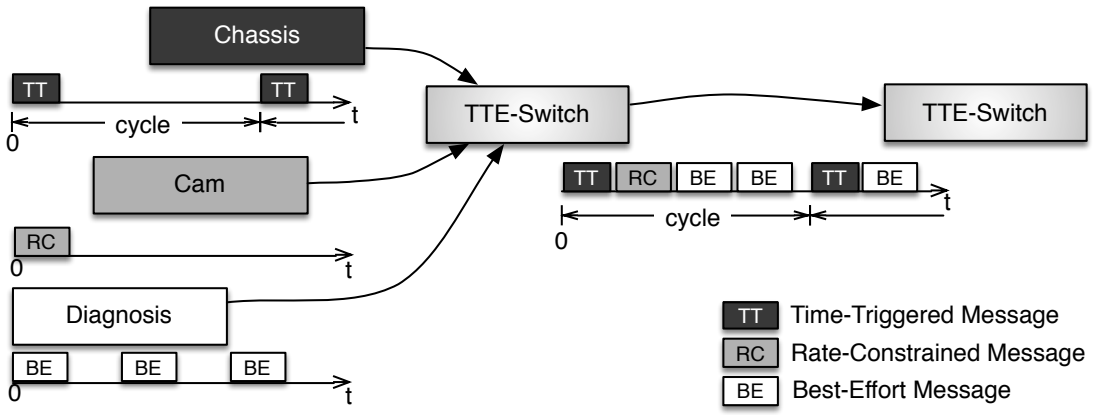

Abbildung 2.10.: Priorisierung und zeit-basierter Medienzugriff von Time-Triggered Ethernet.

3. Hardware

Da diese Arbeit auf einer bestehenden Implementation von TTEthernet aufbaut, wird die dafür verwendete Hardware in den folgenden Abschnitten grundlegend beleuchtet.

3.1. Aufbau des NXHX500-ETM

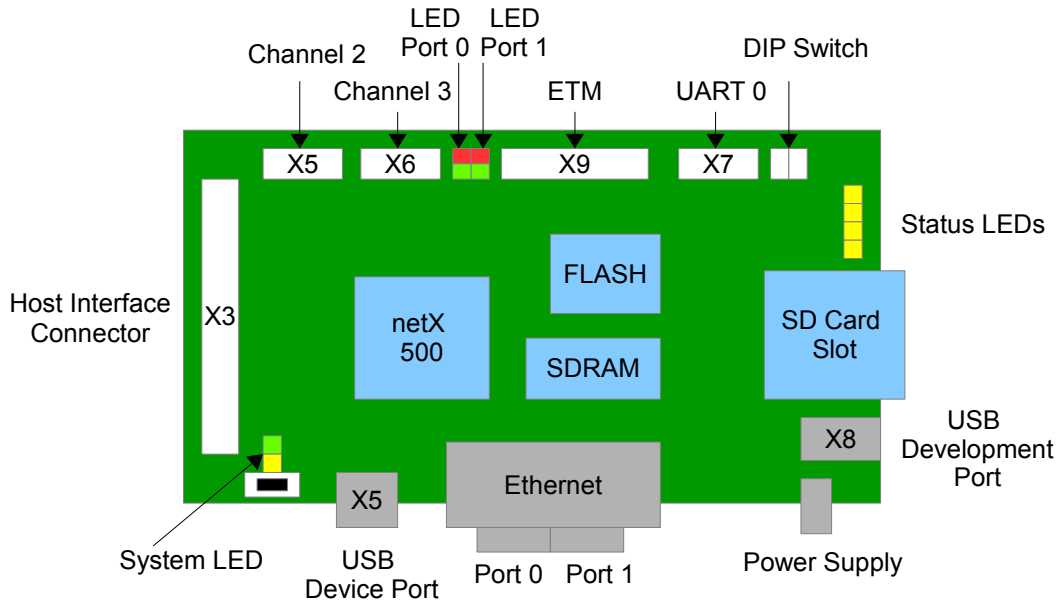

Abbildung 3.1.: Vereinfachte Ansicht des NXHX500-ETM

Das System-on-Chip-Design der netX CPU ermöglicht unabhängige Kommunikation über mehrere Kanäle zur gleichen Zeit und bildet das Herzstück der Architektur des NXHX500-ETM. Das Board ist mit zwei 100 Mbit Ethernet-Kanälen mit integrierten PHYs ausgestattet, welche in Abbildung 3.1 zu sehen sind. Über die Modulsteckplätze Channel 2 und 3 lässt sich das Board um weitere Kommunikationsschnittstellen erweitern. Es verfügt außerdem über eine serielle Schnittstelle, einen USB Port mit Kompatiblität zum USB-2.0-Standard und

einer Dual Port Memory-Schnittstelle, um hohe Übertragungsbandbreiten an ein Endsystem liefern zu können. Weiterhin bringt es einen 8 MB SDRAM und 16 MB Flashspeicher mit. Über die ETM-Schnittstelle oder den USB-Development-Port kann in Kombination mit Entwicklungswerkzeugen ein Debugging ermöglicht werden. Mit einer SD Karte kann ausführbarer Programmcode auf das Board geladen und ausgeführt werden (vgl. Lipfert (2008)).

3.2. netX500

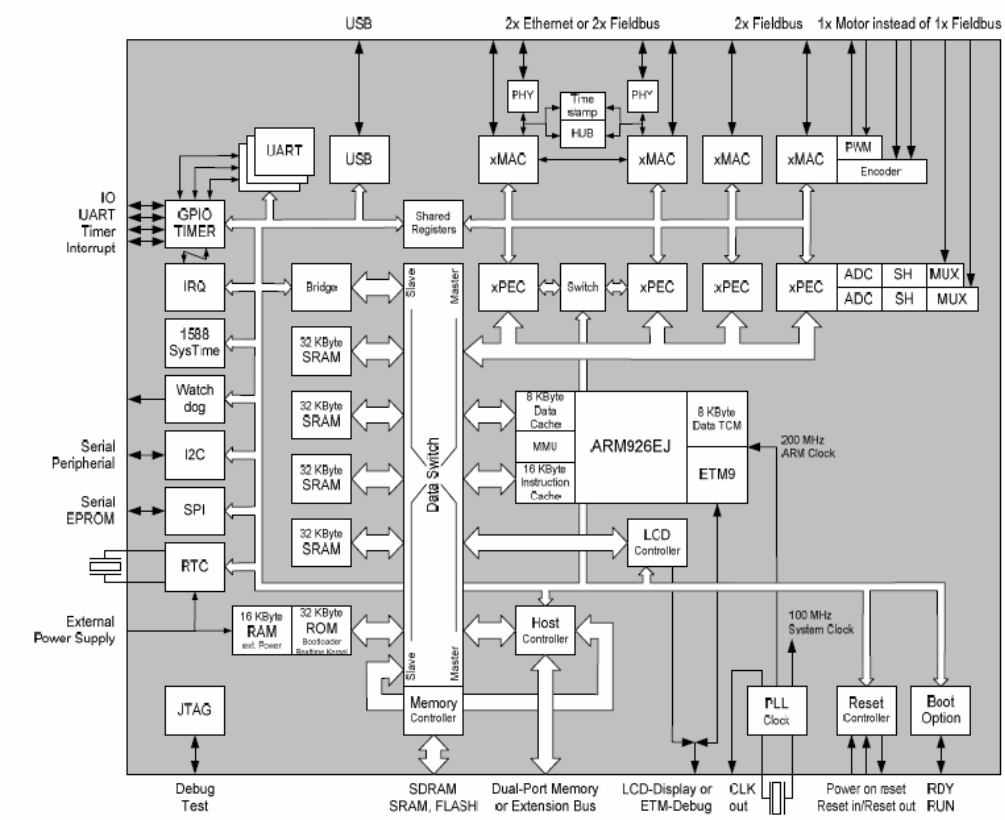

Abbildung 3.2.: netX500 Blockdiagramm (Quelle: Lipfert (2008))

Der netX500 ist ein hoch integrierter Netzwerkcontroller mit einer für Kommunikation und maximalen Datendurchsatz optimierten Systemarchitektur. Basierend auf der „ARM 926EJ-S 32-Bit CPU“-Architektur bringt er unter anderem eine Memory Management Unit sowie einen Echtzeit-Kernel mit. Über vier unabhängige Kommunikationskanäle, die durch jeweils drei voneinander unabhängig frei konfigurierbare ALUs betrieben werden, wird zeitgleiches Senden

und Empfangen ermöglicht, was den Kanal Full-Duplex-fähig macht. Die empfangenen Pakete werden nach IEEE 1588 zeitgestempelt und in den internen SRAM gespeichert. Der ARM 926EJ-S läuft mit einem Takt von 200 MHz und ist mit einem 8 kB Daten-Cache sowie einem 16 kB Instruktions-Cache gekoppelt. Weiterhin ist in Abbildung 3.2 der 8 kB große Tightly Coupled Memory (TCM) zu sehen, der an die ARM CPU gekoppelt ist und als interner Speicher realisiert wurde. Durch die geringen Latenzen beim Zugriff auf den TCM können zeitkritische Programmteile, wie zum Beispiel Interrupserviceroutinen, ausgeführt werden. In einem 32 kB ROM sind Bootloader und Echtzeit Kernel untergebracht. Der interne SRAM ist in vier 32 kB große Blöcke unterteilt, zusätzlich kann ein 16 kB RAM über eine externe Stromversorgung getrieben werden. Der Data Switch verbindet die ARM CPU, die Kommunikations- und Host Controller sowie die Speicherblöcke und Peripheriegeräte über seine fünf Datenpfade untereinander. Dadurch kann auf unterschiedliche Speicherblöcke parallel und individuell zugegriffen werden. So kann die CPU zum Beispiel auf einen Block des internen SRAM zugreifen, während die Kommunikationskanäle eingehende Pakete in einen anderen Block schreiben.

4. Verwandte Arbeiten

In diesem Kapitel wird auf Arbeiten eingegangen, die sich bereits mit dem Verhalten von Audio/Video Bridging auseinander gesetzt haben oder anderweitig in direktem Zusammenhang mit dem Thema dieser Arbeit stehen. Der Inhalt und das Ergebnis dieser Arbeiten werden kurz zusammengefasst und von den Zielen dieser Arbeit abgegrenzt.

4.1. AVB Implementierungen

4.1.1. XMOS AVB Referenzdesign

Die Firma XMOS bietet ein eigens entwickeltes AVB Referenzdesign XMOS Ltd. (2014) für ihre Mikrocontroller an. XMOS implementiert sämtliche Spezifikationen, die zu den AVB Standards gehören. Dadurch ermöglichen sie Mehrkanal Tonübertragungen, Talker und Listener einzeln oder simultan laufen zu lassen sowie bis zu 32-Kanal Übertragungen. Die verwendeten Mikrocontroller laufen im Mehrkernbetrieb mit bis zu vier Kernen, wobei jeder Kern bis zu 32 logische Threads unterstützt. Über das Kommunikationskonzept, das die Kerne und deren einzelne Threads untereinander über „Channels“ verbindet, können Aufgaben auf mehrere Threads verteilt und Informationen übermittelt werden. Dies ermöglicht eine hohe Leistungsfähigkeit des Systems. Das Design ist in einer für XMOS-Mikrocontroller angepassten und abgewandelten C-Sprache entwickelt worden und auf Parallelisierung ausgelegt. Das MRP-, MMRP-, MVRP- und SRP-Konzept wurde übernommen und angepasst. Weitere Details des Referenzdesigns wurden aufgrund der Parallelisierung und der auf den Mikrocontroller angepassten Programmiersprache nicht in die Arbeit integriert.

4.1.2. Open AVB

Open AVB (vgl. AVnu Alliance) ist ein von der AVnu Alliance gesponsertes Open Source Projekt. Das Projekt bietet Netzwerk-Block-Komponenten wie Treiber, Bibliotheken, Beispiel-Code und Daemon Source Code, um Teile eines AVB Systems zusammenzustellen. Diese sind konform zu den AVB Standards und wurden in C entwickelt. Das Konzept wurde aufgrund der Verwendung

von speziellen Bibliotheksfunktionen, die von dem NXHX500-ETM Compiler nicht unterstützt werden, nicht in die Arbeit integriert.

4.2. Aus AVB wird TSN

Als die AVB Task Group im Jahre 2011 mit der Herausgabe des IEEE 802.1AB Standards, AVB komplettierte, wurde die Gruppe im November 2012 zur Time-Sensitive Networking (TSN) Group umbenannt. Die Hauptziele sind es, noch niedrigere Latenzen zu erreichen, sowie das Erweitern des Stream Reservation Protokolls. Da der AVB Verkehr beim Scheduling noch immer von großen zeit-unkritischen Paketen verzögert werden kann und dies in einer nicht unwesentlichen Latenzerhöhung resultiert, soll mit 802.1Qbu (vgl. IEEE 802.1 TSN Task Group (a)) der Präemptions-Ansatz spezifiziert werden. Während einer Unterbrechung (Präemption) eines zeit-unkritischen Pakets, sollen mehrere zeitkritische Pakete gesendet werden können. Anschließend soll das unterbrochene unkritische Paket übertragen werden. Mit der Erweiterung des IEEE 802.1Q-Standards um 802.1Qbv (vgl. IEEE 802.1 TSN Task Group (b)) soll das Behandeln von geschedultem Traffic ermöglicht werden. Um dies zu erreichen soll ein „Transmission Gate“ bei der Auswahl der zu übertragenden Frames festlegen, ob die jeweilige Queue der Nachrichtenklasse für die Übertragung freigegeben ist oder nicht. Die einzelnen „Gates“ werden über eine Kontrollliste zu zeitlich definierten Punkten geöffnet oder geschlossen, um zeitlich kritischen oder unkritischen Traffic zu ermöglichen. Über einen solchen Ansatz könnte zum Beispiel AVB- und TTE-Verkehr in einem Netzwerk ermöglicht werden.

Aufgrund inadäquater Hardwareanforderungen wird der Präemptions-Ansatz nicht übernommen. Da sich der TSN Standard noch in der Entwicklungsphase befindet, wird eine Testimplementierung zum Unterstützen von geschedultem Traffic entwickelt, um die Auswirkungen in Hardware sehen zu können. Das Konzept dieser Probeimplementierung wird dadurch ein Vorreiter für den Präemptions-Ansatz.

4.3. AVB in Simulation

Meyer (2013) entwickelt im Zuge seiner Bachelorarbeit eine Implementierung von AVB für ein Framework einer Simulationsumgebung. Da dieses Framework bereits TTEthernet Verkehr simulieren konnte, war das Ziel seiner Arbeit AVB Komponenten zu implementieren, die sowohl TT- als auch AVB-Pakete behandeln können. Bei der Simulation verschiedener Netzwerke gelangte er unter anderem zu dem Ergebnis, dass sich bei Verkleinerung der TT-Paketgrößen

eine maximale Latenz für Klasse A Nachrichten garantieren lässt. Diese beträgt statt der von AVB garantierten 2 ms über 7 Hops die doppelte Dauer von 4 ms über 7 Hops.

In Meyer u. a. (2013) zeigt er außerdem Scheduling Ansätze für AVB und TTEthernet Verkehr. Der erste Ansatz ist ein kompaktes Scheduling des TTEthernet Verkehrs. In seinem Worst-Case-Fall konfiguriert er den TT Verkehr so, dass die Lücken zwischen den TT Paketen nur 23 µs groß sind, was für das Versenden von AVB Paketen zu klein ist. Durch die Medienreservierung für TT Nachrichten können die AVB Pakete erst gesendet werden, nachdem vier TT Nachrichten übertragen wurden. Da in der Simulationskonfiguration die AVB Nachrichten über 3 Switches übertragen werden müssen, kommt er unter Berücksichtigung der 23 µs Lücken zwischen den TT Nachrichten, der Verarbeitungszeit der Switche und des Endknotens auf eine Worst-Case-Latenz von 2.4 ms bei Verwendung des kompakten Schedules. Den zweiten Scheduling-Ansatz bildet das angepasste Scheduling. Durch die Vergrößerung der Lücke zwischen den TT-Paketen auf 123 µs passen nun AVB-Pakete zwischen die TT-Pakete. Die Ergebnisse beschreibt er als eine signifikante Verbesserung für die Performanz der AVB-Streams. Der angepasste Schedule reduziert die maximale Latenz von AVB und Best Effort Traffic um 50%. Außerdem verringerte sich die durchschnittliche Latenz von AVB um zwei Drittel.

Des weiteren präsentiert er einen Traffic Shaping Ansatz für AVB und TT Verkehr, der die zuvor beschrieben Schedules von AVB-, TT- und BE-Nachrichten unterstützt. Dieser Ansatz wird mit in diese Arbeit übernommen und angepasst. Außerdem wird das Verhalten der von ihm beschrieben Tests in abgewandelter Form in Hardware getestet.

4.4. Time-Triggered Ethernet auf einem echtzeitfähigen Netzwerk-Stack

Müller (2011) entwickelt im Zuge seiner Bachelor Arbeit einen echtzeitfähigen Netzwerkstack mit TTEthernet Implementierung für einen Mikrocontroller. Die Implementierung wird hinsichtlich Latenz und Jitter evaluiert. Das Resultat zeigt bei einem Testaufbau von zwei Teilnehmern lediglich einen Jitter von maximal 300 ns, statt den laut Spezifikation erlaubten 1 µs, bei dem Empfang von TT-Nachrichten.

In Müller u. a. (2011) werden Ergebnisse eines Testaufbaus mit drei Prototypen und einem Standard PC als Traffic Generator, die alle über einen TTEthernet Switch miteinander verbunden sind, gezeigt. Ein Controller ist als Synchronisierungsmaster und die anderen beiden als Clients konfiguriert. Einer der Clients sendet TT-Nachrichten, der andere RC Nachrichten an den Master. Die Zyklus Dauer des TT-Nachrichten-Schedules beträgt 3 ms und die RC Nachrichten wurden mit einer bandwidth allocation gap (BAG) von 2 ms definiert. Der

Standard-PC erzeugt dazu Best Effort Traffic mit einer minimalen Frame-Größe. Diese Pakete wurden an alle Prototypen geroutet, um Auslastung im Netzwerk zu erzeugen. Mit einem Oszilloskop wurde die Synchronisationsgenauigkeit der Prototypen verifiziert und dadurch, dass jedes empfangene Paket von jedem Controller zeit-gestempelt wurde, konnten Jitter und Latenzen ermittelt werden. In der Analyse der Ergebnisse zeigte sich, dass beim Empfangen von TT-Nachrichten der maximale Jitter unter 1 µs lag. Das Scheduling-Empfangsfenster betrug 10 µs. Die Prototypen liefen mehr als 10 000 Perioden im Testbetrieb. Innerhalb dieser Zeit wurde kein Event außerhalb der geschedulten Zeit ausgeführt und keine Anzeichen dafür sichtbar, dass diese Bedingung in weiteren Testbetrieben verletzt werden könnte.

Seine Arbeit bildet die Basis für diese Arbeit, um das bestehende TTEthernet durch AVB erweitern und zusammen in Hardware implementieren und verifizieren zu können. Des weiteren wird in dieser Arbeit verifiziert, dass TT-Nachrichten vom AVB-Verkehr unbeeinflusst bleiben und die zeitkritischen Anforderungen von TTEthernet nicht verletzt werden.

5. Anforderungen und Konzept

In diesem Kapitel werden die generellen und protokolltechnischen Eigenschaften analysiert. Aus den resultierenden Anforderungen wird das Konzept, das zum Ergebnis führt, erläutert. Um das Verhalten der in diesem Kapitel beschriebenen Fälle nachvollziehen zu können, wird ein TT-Schedule vorausgesetzt, der genug Platz für die Übertragung von möglichen AVB-Paketen lässt.

5.1. Anforderungen

Dieses Kapitel betrachtet die von AVB und TTEthernet ausgehenden protokolltechnischen Eigenschaften/ Problemstellungen, sowohl an die Hard- als auch an die Software. Die daraus resultierenden Anforderungen werden formuliert und im Konzept umgesetzt.

5.1.1. Generelle und protokolltechnische Eigenschaften von AVB und TTEthernet

1. Durch zyklische Updates und um den vollen Funktionsumfang sowie den korrekten Betrieb des Stream Reservation Protocols (SRP) gewährleisten zu können, sind die in Tabelle 5.1 aufgeführten Timer erforderlich.

2. Die Spezifikationen von AVB und TTEthernet setzen jeweils voraus, dass AVB Klasse A Traffic als auch TT-Nachrichten die alleinige höchste Priorität im Netzwerk besitzen. Da die TT-Nachrichten zu vorgegebenen festen Zeitpunkten versendet werden und nicht unterbrochen werden dürfen, müssen die TT-Nachrichten in Kombination mit AVB die höchste Priorität im Netzwerk haben. Dies stellt durch die Überschneidungen der Protokollanforderungen jedoch einen Konflikt dar, welcher im Konzeptteil gelöst wird.

3. Es wird laut Spezifikation empfohlen, dass AVB maximal 75% der physikalischen Bandbreite benutzt, damit Best Effort (BE)-Nachrichten die restlichen 25% verwenden können. Da neben den niedrig priorisierten BE-Nachrichten auch höher priorisierte TT-

Nachrichten im Netzwerk auftreten, müssen diese bei der Berechnung der verfügbaren Bandbreite für AVB berücksichtigt werden.

4. Basierend auf einem bestehenden echtzeitfähigen Netzwerkstack der TTEthernet unterstützt (vgl. Müller (2011)) soll das vorhandene Projekt durch AVB erweitert und mit der bestehenden TTEthernet-Implementierung zusammengeführt werden.

5.1.2. Aus den Eigenschaften resultierende Anforderungen

Die resultierenden Anforderungen für die Implementierung ergeben sich aus den in Abschnitt 5.1.1 aufgeführten Punkten.

- Zu Punkt 1: Damit der einwandfreie Betrieb des Stream Reservation Protocols (SRP) gewährleistet werden kann, muss eine Menge von mindestens vier Timern unterstützt werden. Aufgrund der in Tabelle 5.1 aufgeführten Timer, die eine vernachlässigbare Genauigkeit im Sekundenbereich haben, ist ein logischer Timer ausreichend. Durch die Verwendung eines Software Timer, können mit nur einem Hardware Timer mehrere Timer realisiert werden, wodurch die Ressourcen effizient genutzt werden.
- Zu Punkt 2: Das neue Scheduling von AVB- und TT-Nachrichten muss gewährleisten, dass bei der Übertragung von TT-Nachrichten, Latenz und Jitter keine Abweichungen gegenüber des Einzelbetriebs aufweisen. Dadurch muss bei der Auswahl der zu sendenden Pakete im CBS Algorithmus berücksichtigt werden, dass TT-Pakete immer die höchste Priorität haben, da sie am zeit-kritischsten sind.
- Zu Punkt 3: Bei der Berechnung der für AVB zur Verfügung stehenden Bandbreite muss gewährleistet sein, dass die durch TT-Pakete verbrauchte Bandbreite mit in die Berechnungen der noch verfügbaren Bandbreite von AVB einfließt. Das bedeutet, dass von den 75% der physikalischen Bandbreite, die von TT benötigte Bandbreite plus ein Paket mit maximaler Größe vor dem jeweiligen TT-Paket abgezogen wird, bevor durch das Stream Reservation Protocol Bandbreite für AVB-Streams reserviert werden kann. Wenn ein AVB-Paket übertragen werden soll, jedoch nicht mehr genug Zeit für die Übertragung bis zum nächsten TT-Paket verfügbar ist, wird durch eine Erhöhung des Kreditsystem Werts die Verzögerung bis zum TT Paket notiert.
- Zu Punkt 4: Da diese Arbeit auf einer bereits bestehenden TTEthernet-Implementierung aufbaut, müssen die noch zur Verfügung stehenden Ressourcen effizient genutzt und die weiteren konzeptionellen Überlegungen an die bestehende Architektur angepasst werden.

Timer	Intervall in s
Periodic	1
Join	2
Leave	8
Leaveall	100

Tabelle 5.1.: AVB Timer und deren Intervalllänge gemäß IEEE 802.1Q (vgl. Institute of Electrical and Electronics Engineers (2011c))

5.2. Konzept

In diesem Kapitel wird ein Konzept aus den Anforderungspunkten in Kapitel 5.1 erarbeitet, welches die Zusammenführung und Implementation von AVB und TTEthernet ermöglichen soll.

5.2.1. Verändertes Queueing

Das ursprünglich in Abbildung 2.7 gezeigte Queuing und Scheduling wird um eine Eigenschaft erweitert, die in Abbildung 5.1 dargestellt ist. Diese Erweiterung gewährleistet, dass der TTEthernet-Verkehr berücksichtigt und von dem restlichen Verkehr nicht beeinträchtigt wird. Dazu muss der CBS Algorithmus vor dem Versenden von nicht-TT-Verkehr bei der Auswahl des nächsten Frames überprüfen, ob er sein Paket senden darf. Der Algorithmus fragt zum aktuellen Zeitpunkt die Sendezeiten und Paketgrößen der TT-Nachrichten vom TTE Scheduler ab und berechnet das Zeitfenster bis zur nächsten TT-Nachricht. Ist das Fenster für die zu sendende AVB- oder BE-Nachricht groß genug, kann der Scheduler die Nachricht versenden. Andernfalls wird das Versenden der Nachricht verzögert, bis eine Lücke gefunden wird, in die das Paket hinein passt. In dem Fall, dass ein AVB-Paket aufgrund seiner Größe nicht mehr bis zum nächsten TT-Paket versendet werden darf, kann ein BE-Paket mit einer passenden Größe noch vor dem AVB-Paket versendet werden. Der TT-Verkehr wird dabei nicht beeinflusst.

5.2.2. Anpassung des Credit Based Shaper Algorithmus

Durch das Hinzufügen von TT-Verkehr muss das Verhalten des CBS-Algorithmus anpasst werden. Da TT-Pakete eine höhere Priorität als AVB und damit die höchste Priorität im Netzwerk besitzen, haben sie vor allen anderen Paketen Vorrang. Dieses Verhalten wird anhand Abbildung 5.2 verdeutlicht. Wenn ein AVB-Paket gesendet werden soll, wird überprüft, ob das Senden des AVB-Pakets das Zeitfenster eines TT-Pakets verletzen würde. Ist dies der Fall, so wird das Senden des AVB-Pakets verzögert. Während der Verzögerung und des Sendevorgangs

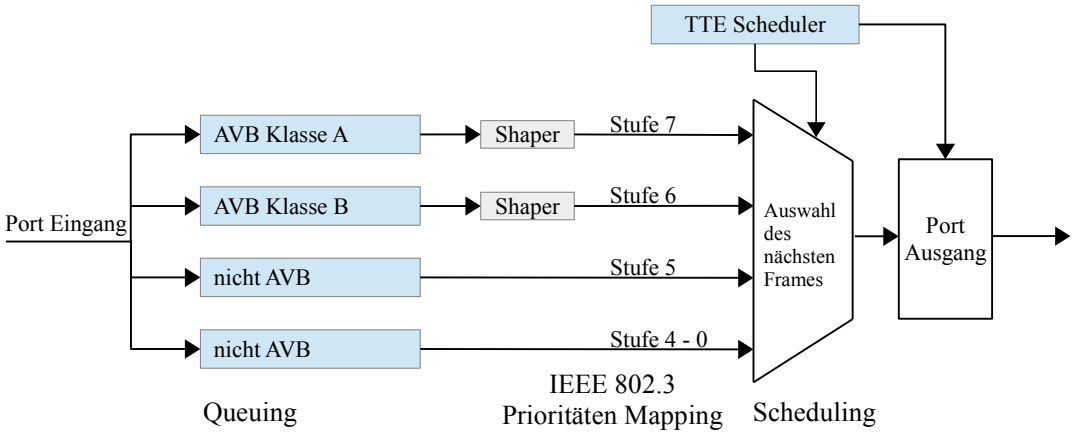

Abbildung 5.1.: Angepasstes Queueing und Scheduling für TTEthernet und AVB

des TT Pakets steigt der Wert des Kreditsystems für die Queue des zu sendenden AVB-Pakets an. Wenn das TT-Paket mit dem Erreichen des Zeitpunktes „send window end“ die Übertragung beendet hat und ausreichend Zeit bis zum nächsten TT-Paket ist, wird das AVB-Paket versendet.

Der beschriebene Scheduling Ansatz bietet auf Kosten der AVB-Latenz ein reibungsloses Zusammenspiel von TT- und AVB-Nachrichten. Je nach Konfiguration des TT-Schedules, kann die AVB Latenz erhöht oder verringert werden. Nachfolgend werden weitere mögliche Scheduling-Ansätze beleuchtet und erklärt, warum diese nicht übernommen wurden.

Präemption von BE Verkehr

Wie bereits in Kapitel 4.2 erwähnt, spezifiziert die TSN Task Group derzeit einen Ansatz zur Unterbrechung von zeit-unkritischen Paketen. Dadurch wäre es möglich BE-Pakete zu unterbrechen, um Latenz und Jitter von AVB-Nachrichten zu verbessern. Wenn ein Paket durch ein zeitkritischeres unterbrochen wurde, wird der restliche Teil des Pakets nach der Übertragung des kritischen Pakets versendet. Dabei wird der Header der unterbrechbaren Pakete angepasst, damit der Empfänger weiß, dass er noch weitere Pakete erhält, die zu dem empfangenen Paket gehören. Wenn der letzte Paketteil angekommen ist, kann der Empfänger die Teil-Pakete wieder zusammensetzen. Dieser Ansatz wurde aufgrund von inadäquaten Hardwareanforderungen nicht übernommen.

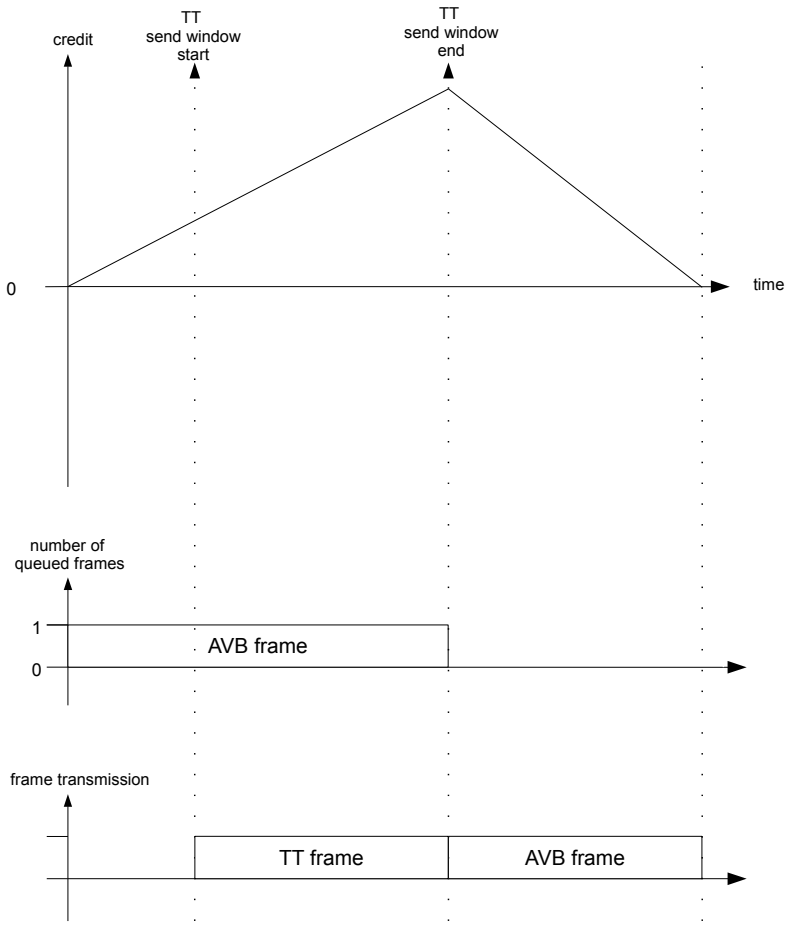

Abbildung 5.2.: Verhalten des CBS bei Verzögerung eines AVB Pakets durch ein TT Paket.

Anpassung der Paketgrößen

Eine weitere Möglichkeit ist die Verringerung der Größe von TT-, AVB- und BE-Paketen (vgl. Imtiaz u. a. (2012)). Dadurch würde die Übertragungsdauer gemindert und Latenz und Jitter verringert werden. Dies entspräche allerdings nicht mehr den Spezifikationen von AVB, wodurch die Kompatibilität zu anderen Teilnehmern im Netzwerk nicht mehr vollständig gegeben wäre.

Betrachtung eines Worst-Case-Szenarios

Durch die Veränderungen des CBS-Algorithmus ergibt sich ein neues Worst-Case-Szenario, das anhand Abbildung 5.3 erklärt wird. Die Queue für AVB-Pakete ist leer und es wird ein

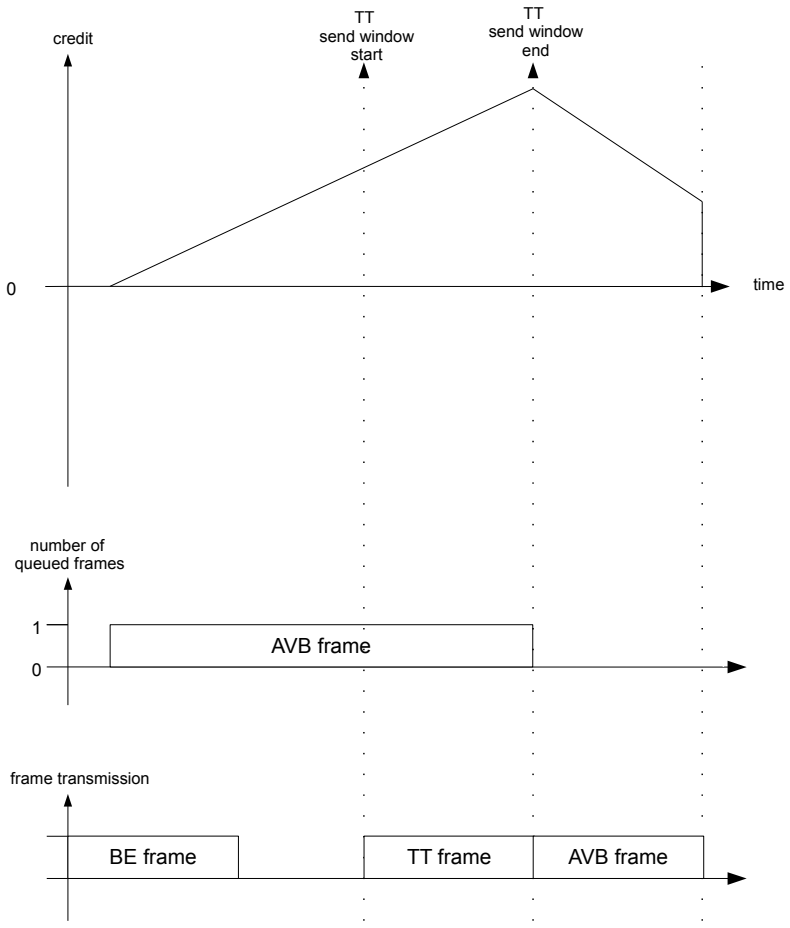

Abbildung 5.3.: Worst-Case-Szenario für den angepassten CBS Algorithmus.

BE-Paket mit maximaler Länge übertragen. Kurz darauf steht ein AVB-Paket in der Queue an, wodurch die Übertragung verzögert wird, bis das BE-Paket versendet wurde und der Port Ausgang frei ist. Da nach dem BE-Paket die Übertragung eines TT-Pakets ansteht, welche das AVB-Paket aufgrund seiner Größe interferieren würde, wird es weiter verzögert. Ab dem Zeitpunkt zu dem das BE-Paket mit der Übertragung begonnen hat und bis das TT-Paket die Übertragung mit dem Erreichen des Zeitpunkts „send window end“ beendet hat, steigt der Wert des Kreditsystems der AVB Queue gemäß $idleSlope$ (siehe Formel 2.2). Nach der Übertragung des TT-Pakets kann das AVB-Paket versendet werden. Der Wert des Kreditsystems fällt dabei um $sendSlope$ (siehe Formel 2.1). Da nach der Übertragung des AVB-Pakets kein Paket mehr in der Queue ansteht, wird der Wert des Kreditsystems auf Null gesetzt.

5.2.3. Der modulare Aufbau des SRP und seine architektonische Integration

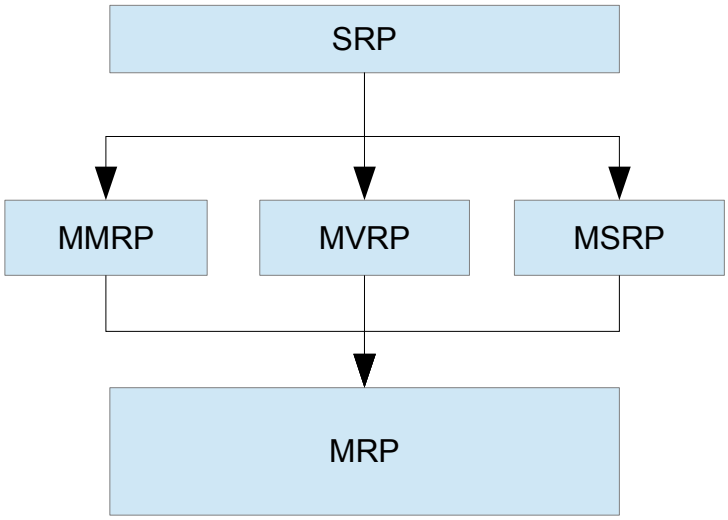

Abbildung 5.4.: Modularer Aufbau vom SRP.

In Abbildung 5.4 wird verdeutlicht, welche Protokoll-Teile vom Stream Reservation Protocol (SRP) verwendet werden, um einen Stream im Netzwerk zu reservieren. Wie in Abschnitt 2.1.4 bereits erwähnt wurde, dient das MRP dabei als grundlegende Basis um Attribute im Netzwerk bekannt zu geben und sie wieder zu verwerfen. Nachfolgend werden die darauf aufbauenden Protokolle beleuchtet, die für den Betrieb des SRP verwendet werden. SRP benutzt das Multiple MAC Registration Protocol (MMRP), um optional die Propagierung von

Talker-Registrierungen innerhalb des Netzwerks zu kontrollieren. Standardmäßig werden Talker-Registrierungen von den AVB Bridges an alle Portausgänge weitergeleitet. Wenn diese Registrierungen jedoch nur an bestimmte Teilnehmer in einem Netzwerk gesendet werden sollen, wird eine Funktion namens „Talker Pruning“ verwendet. Um bei aktivem „Talker Pruning“ die Talker-Registrierungen empfangen zu können, registrieren sich die Teilnehmer über ein „MMRP Join“ auf die Zieladresse des Streams. Diese Anmeldung wird von den AVB Bridges in einer Tabelle vermerkt und gegebenenfalls an andere Bridges weitergereicht. Das Multiple VLAN Registration Protocol (MVRP) wird von Endstationen und Bridges verwendet, um einem VLAN beizutreten, in dem ein Stream vom Talker gesendet wird. Der Beitritt in ein VLAN wird von jeder Bridge oder den Endstationen in einer Filtering Database vermerkt. Da MSRP das Erstellen von Streams über Bridge Ports, die nicht auf die jeweilige VLAN ID registriert sind, nicht zulässt, müssen Teilnehmer, die einen Stream beziehen möchten, für das selbe VLAN registriert sein wie der Talker. Das Multiple Stream Registration Protocol (MSRP) ermöglicht die Reservierung von Netzwerkressourcen, damit das Empfangen und Senden von Stream Daten garantiert werden kann.

Die aufgeführten Protokolle erweitern das MRP um ihre eigenen Attribute, damit diese über das MRP im Netzwerk bekanntgegeben werden können.

Verwendung des SRP als Endstation

Um eine Endstation als Talker oder Listener fungieren zu lassen, verwendet das SRP die auf dem MRP basierenden Protokolle. Ein Talker gibt über das Versenden eines Domain-Pakets bekannt, ob er die Stream Reservation Klasse A, Klasse B oder beide unterstützt. Dies geschieht über das MSRP. Nachfolgend registriert er sich über das MVRP für ein VLAN. Optional kann das erwähnte MMRP für das „Talker Pruning“ benutzt werden. Zuletzt gibt der Talker über sein „Talker Advertise“ seinen Stream bekannt. Erhält er von den Bridges ein „Listener Ready“ oder ein „Listener Ready Failed“ kann er mit der Übertragung beginnen, da zu dem Zeitpunkt die benötigten Ressourcen für den Stream reserviert wurden.

Der Listener gibt über das MSRP seine Kompatibilität zur Stream Reservation Klasse bekannt und wartet auf das „Talker Advertise“ eines Streams. Möchte er den bekanntgegebenen Stream empfangen, so registriert er sich über das MVRP für das VLAN und sendet über das MSRP ein „Listener Ready“.

Die architektonische Integration des SRP

Um die Attribute des MRP aktuell zu halten, werden gemäß des „Periodic“-Timers in Tabelle 5.1 die Attribute jede Sekunde aktualisiert. Dabei werden ebenfalls die restlichen Timer überprüft

und je nach Situation erneut gestartet. Da die Intervalllänge der Timer im Sekundenbereich liegt und Intervalle solcher Größe nicht als zeitkritisch angesehen werden, wird die periodische Überprüfung von einer Routine ausgeführt, die nur aktiv wird, wenn keine höher priorisierten Funktionen, wie zum Beispiel das Senden von TT-Nachrichten oder das Empfangen von Paketen ausgeführt werden. In der in Abschnitt 6.6 beschriebenen Architektur wird das MRP, MMRP, MVRP und MSRP unter dem Begriff Stream Reservation Protocol (SRP) dargestellt.

6. Implementierung

In diesem Kapitel wird die Umsetzung der Anforderungs- und Konzeptpunkte beschrieben sowie weitere essentielle Aspekte der Implementierung betrachtet.

6.1. Erweiterung des Bufferpools

Um empfangene und zu sendende Pakete schnell und effizient verarbeiten zu können, hat Müller (2011) einen Bufferpool zur Unterstützung von TTEthernet- und Best-Effort-Paketen entwickelt. Der Bufferpool besteht aus einer getrennten Datenstruktur für Input- und Output-Buffer. Diese

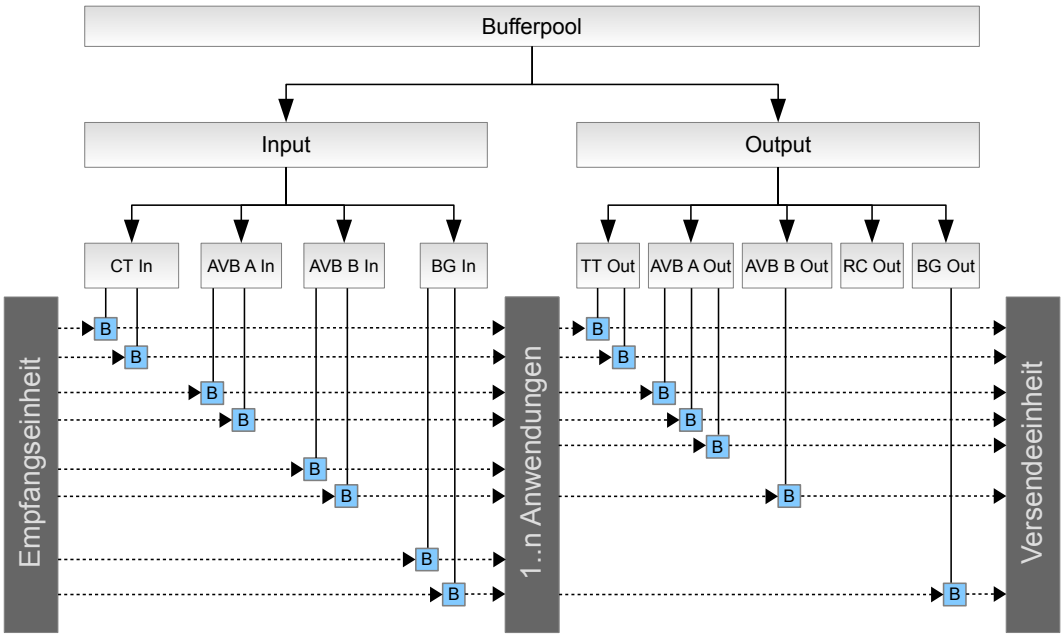

Abbildung 6.1.: Erweiterte Version des Bufferpools um AVB.

Datenstrukturen wurden um AVB-Buffer erweitert. In Abbildung 6.1 ist die daraus entstandene Version des Bufferpools zu sehen. Für AVB Klasse A und Klasse B wurden Input- und Output-

Bufferpools hinzugefügt. Ein solcher Bufferpool kann mehrere Buffer für jeweils eine Stream ID verwalten. Die empfangenen AVB-Pakete werden je nach Stream Reservation-Klasse und anhand der Stream ID in den dafür vorgesehenen Buffer gespeichert. Mittels Angabe der Stream ID können Anwendungen direkt auf den jeweiligen Buffer zugreifen um Pakete verarbeiten oder senden zu können. Weiterhin bietet das System die Möglichkeit, den jeweiligen Buffer so zu konfigurieren, dass beim Empfangen oder Versenden eines Pakets eine Callback-Funktion aufgerufen wird, um beispielsweise ein empfangenes Paket durch Aufrufen einer Funktion gezielt verarbeiten zu können. Die AVB Buffer werden nach dem FIFO-Prinzip betrieben.

6.2. Logischer Timer / Background Task

Da in einer AVB-Implementierung mehrere Events vom System zu unterschiedlichen Zeitpunkten verarbeitet werden müssen, wurde ein logischer Timer implementiert. Dieser benutzt einen Hardware-Timer, der mit einer Auflösung von 10 ns pro Tick seinen Zählwert in einem vorzeichenlosen 32 Bit Register erhöht. Abbildung 6.2 zeigt die Struktur des logischen Timers. Um die Timer effizient verwalten zu können, wurden zwei Arrays verwendet. Das erste Array

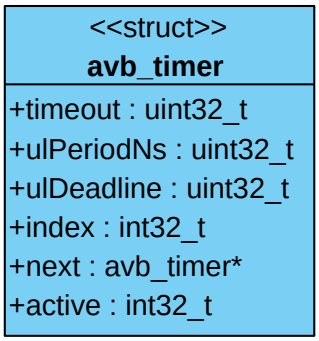

Abbildung 6.2.: AVB Timer Struktur.

speichert Pointer auf die einzelnen „avb_timer“-Strukturen. Das zweite Array ermöglicht anhand von Booleans die Verwaltung der Plätze im ersten Array. Im Folgenden wird der Vorgang zum Starten eines Timers, der in Abbildung 6.3 dargestellt ist, beschrieben: Mit der „`init_avb_timer()`“-Funktion wird unter Angabe der Adresse der „avb_timer“-Struktur die Initialisierung der einzelnen Variablen durchgeführt. Anschließend kann der Timer durch Aufrufen der „`start_avb_timer()`“-Funktion und unter erneuter Angabe der Adresse des Timers sowie der Dauer in Nanosekunden gestartet werden. Beim Aufrufen der Funktion wird anhand der aktuellen Systemzeit und der Dauer des Timers die Deadline berechnet, wann

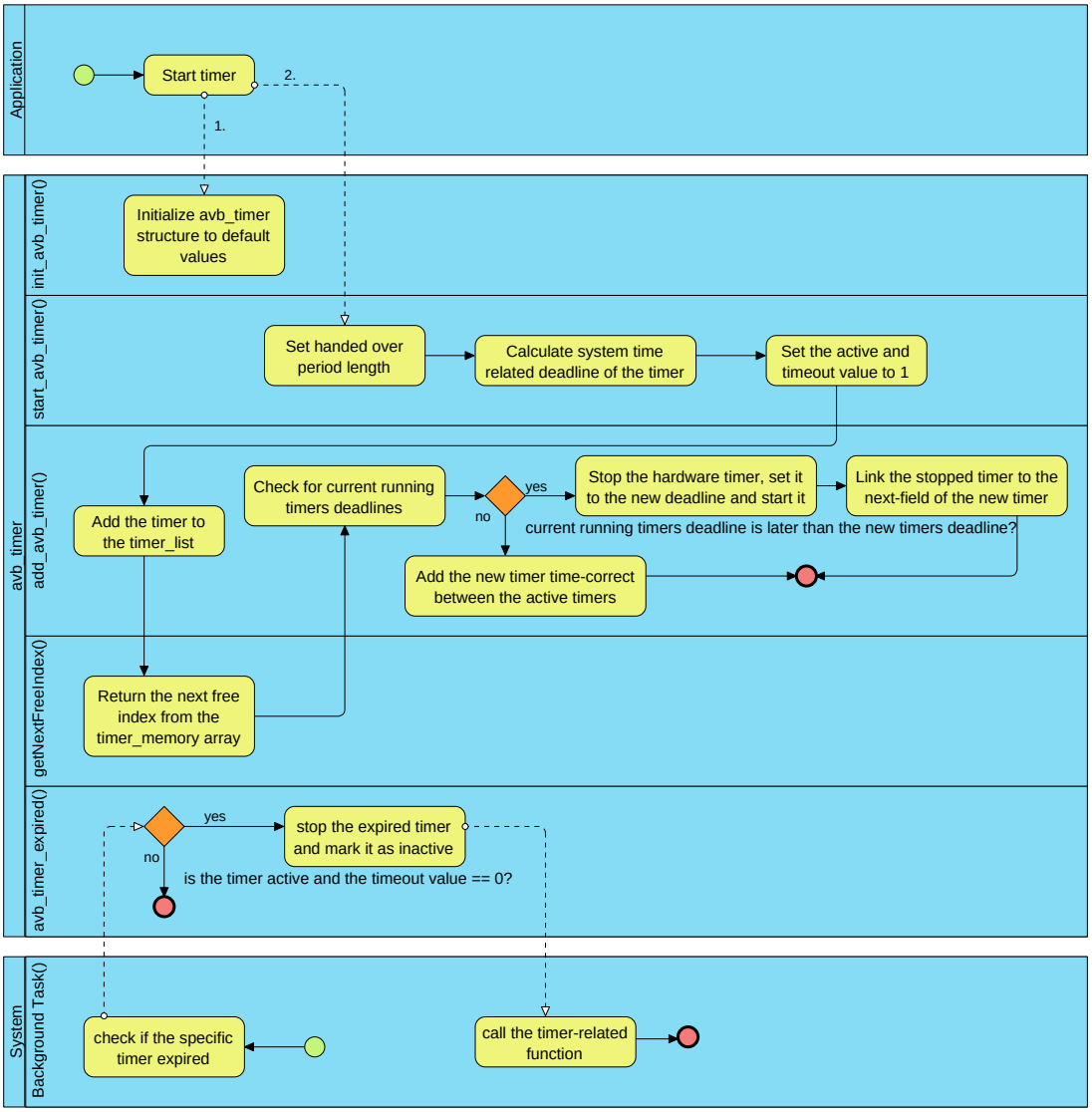

Abbildung 6.3.: Ablauf beim Starten eines Timers sowie das Bearbeiten der Timer-Funktion durch die Background Task.

der Timer ablaufen soll, und in der Timer-Struktur gespeichert. Die „Active“- und „Timeout“-Variable werden auf 1 gesetzt und die Funktion „`add_avb_timer()`“ mit der Adresse der Timer-Struktur aufgerufen. Die übergebene Adresse wird in ein leeres Feld des Timer-Arrays gespeichert, welches die „`getNextFreeIndex()`“-Funktion aus dem Boolean-Array liefert. Nun wird das Timer-Array auf bereits vorhandene Timer-Einträge überprüft und der neue Timer an der zeitlich-korrekten Stelle eingefügt. Wenn beim Einfügen des neuen Timers ein Timer mit einer späteren Deadline an erster Stelle im Array steht, was bedeutet, dass der Hardware-Timer gerade auf die spätere Deadline eingestellt ist, wird der Hardware-Timer angehalten und anschließend auf die frühere Deadline des neuen Timers gesetzt. Damit der unterbrochene Timer nach Ablauf des neu eingefügten Timers wieder aufgenommen werden kann, wird über den „next“-Pointer in dem aktuellen Timer die Adresse des unterbrochenen Timers gespeichert. So kann der abgelaufene Hardware-Timer erkennen, ob er einen Folgetimer aktivieren soll oder nicht.

Da die Ausführungszeit innerhalb von Interrupt Service Routinen (ISR) kurz gehalten werden muss, wird bei Ablauf des aktuellen Timers lediglich die „Timeout“-Variable auf 0 gesetzt. Um die mit dem Timer verbundene Funktion abarbeiten zu können, wird die Background Task benutzt. Die Background-Task-Funktion wird automatisch vom System aufgerufen, wenn die CPU gerade keine anderen Funktionen ausführt. Dadurch wird gewährleistet, dass keine wichtigen Programmabläufe unterbrochen werden können. Mittels der Funktion „`avb_timer_expired()`“ kann durch das Übergeben der Timer-Struktur-Adresse aus der Background Task heraus geprüft werden, ob der Timer abgelaufen ist (vgl. Abbildung 6.3). Ist dies der Fall, so kann die Funktion, die mit dem Timer verknüpft wurde, aus der Background Task heraus ausgeführt werden.

6.3. Stream Reservation Protocol

Aus dem von der Firma XMOS Ltd. (2014) entwickelten AVB Referenzdesign wurde das Stream Reservation Protocol in diese Arbeit übernommen, an die verwendete Prozessorarchitektur des ARM926J-CPUs angepasst und gemäß den Spezifikationen des Standards IEEE 802.1Q (vgl. Institute of Electrical and Electronics Engineers (2011c)) modifiziert. Wie bereits in Abbildung 5.4 dargestellt wurde, bildet das MRP die Grundlage des Stream Reservation Protocols und verwaltet die protokollspezifischen Attribute von MMRP, MVRP und MSRP.

6.3.1. Architektur des MRP und Verarbeitungsablauf von empfangenen Paketen

Ein Multiple Registration Protocol-Teilnehmer besteht aus einer Anwendungs- und einer „MRP Attribute Declaration“ (MAD)-Komponente. Die Anwendungskomponente ist für das korrekte Behandeln von Attributwerten und deren Registrierung verantwortlich. Um ein Attribut bekanntzugeben oder zu verwerfen nutzt sie die „`MAD_join()`“ oder „`MAD_leave()`“-Funktion, über die der jeweilige Attribut-Typ und der dazugehörige Attribut-Wert übergeben wird. Die MAD-Komponente registriert, versendet und empfängt die Änderungen der Anwendungskomponente sowie die Änderungen von anderen MRP-Teilnehmern aus dem Netzwerk. Erhält es eine Bekanntgabe oder eine Verwerfung eines Attribut-Typs, so wird dies über die „`MAD_join_indication()`“- oder „`MAD_leave_indication()`“-Funktion an die Anwendungskomponente weitergegeben. Über zwei State Machines werden die Attribute aktuell gehalten und auf Änderungen reagiert. Dies geschieht über die „`MRP_update_state()`“-Funktion unter Angabe des „State Machine Events“, des Attribut Containers, der Informationen zu den jeweiligen Attributen beinhaltet und eines optionalen Informationsparameters, der als „Four Packed Event“ bezeichnet wird. Die Registrar State Machine verzeichnet lediglich Deklarationen der Attribute, die von anderen Teilnehmern im Netzwerk gesendet wurden, und versendet keine Nachrichten. Die Applicant State Machine hingegen verzeichnet das Verhalten sämtlicher Teilnehmer und von sich selbst, um zu erkennen, ob folglich eine Bekanntgabe oder Verwerfung für das Attribut gesendet werden muss. In Abbildung 6.4 wird der

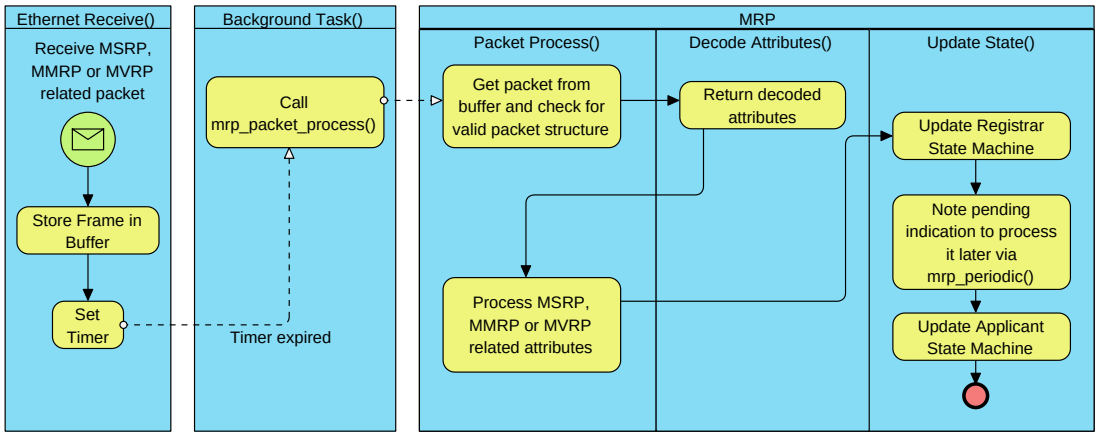

Abbildung 6.4.: Verarbeitungsablauf bei Erhalt eines MRP-bezogenen Pakets.

Verarbeitungsablauf der Implementierung von einem empfangenen SRP Paket in der Business Process Model Notation (im weiteren Verlauf auch BPMN genannt) dargestellt. Ein von der

„EthernetReceive()“ erhaltenes MSRP-, MMRP- oder MVRP-Paket wird in dem dafür vorgesehenen Buffer gespeichert und ein Timer zur anschließenden Verarbeitung des Pakets durch die „Background Task()“ gesetzt. Wenn das System keine wichtigen Ereignisse verarbeitet, ruft es die „Background Task()“-Funktion auf und der abgelaufene Timer wird erkannt. Daraufhin wird die „MRP_packet_process()“-Funktion aufgerufen und das Paket aus dem Buffer verarbeitet. Sofern das Paket eine korrekte Struktur gemäß des Standards aufweist, wird über die „decode_attributes()“-Funktion das in dem Paket übermittelte Attribut dekodiert. Anhand des Attributs kann der Verarbeitungsprozess erkennen, ob es sich dabei um ein Talker-, Listener-, Domain-, VLAN- oder MMRP-Paket handelt und es attributspezifisch verarbeiten. Anschließend wird der Attribut-Typ und der Attribut-Wert an die „update_state()“-Funktion weitergereicht. Dort notiert zuerst die Registrar State Machine Änderungen des Teilnehmers in dem „Registrar State“ und vermerkt je nach aktuellem Zustand der State Machine eine „Pending_indication“, welche von dem nächsten „MRP_periodic()“-Aufruf aus der „Background Task“ heraus bearbeitet wird. Zuletzt gleicht die Applicant State Machine die empfangenen Attribut-Werte mit ihren lokalen ab und ändert ihren „Applicant State“. Je nach Zustand der State Machines werden Timer gestartet oder MRP-Deklarationen in das Netzwerk versendet.

6.3.2. Anmelden eines Talkers und bekanntgeben seines Streams

Abbildung 6.5 zeigt den Ablauf der System-Initialisierung und den Anwendungsfall wenn ein Talker seinen Stream bekanntgeben möchte. Während der Initialisierung des Systems wird die „avb_init()“-Funktion aufgerufen. Neben der Initialisierung von MRP wird aus der Funktion heraus die „avb_register_talkers()“-Funktion aufgerufen, in der die vorher festgelegte Anzahl von Talkern vorbereitet wird. Es wird in der Talker-Struktur eine Multicast-Adresse zugewiesen, eine Stream ID vergeben, die VLAN ID gesetzt, der Status des Talkers auf „deaktiviert“ gesetzt und das Talker-Attribut beim MRP initialisiert. Damit ist die System-Initialiserung abgeschlossen.

Nun kann die Anwendung der „talker_register_stream()“-Funktion unter Angabe der Talker-Nummer und den übertragungsspezifischen Talker-Informationen wie Paket-Größe, Anzahl der Inverall-Pakete und der SR-Klasse die Stream Bekanntgabe initiieren. Da diese Funktion zu der AVB API gehört, werden die übergebenen Parameter an die „register_talker()“-Funktion weitergereicht. Diese aktualisiert anhand der Talker-Nummer die übertragungsspezifischen Informationen und ruft die „set_source_state()“-Funktion mit dem Parameter „SOURCE_STATE_POTENTIAL“ auf. Daraufhin wird über die „join_vlan()“-Funktion ein neuer VLAN-Eintrag im MRP registriert und ein VLAN-Join-

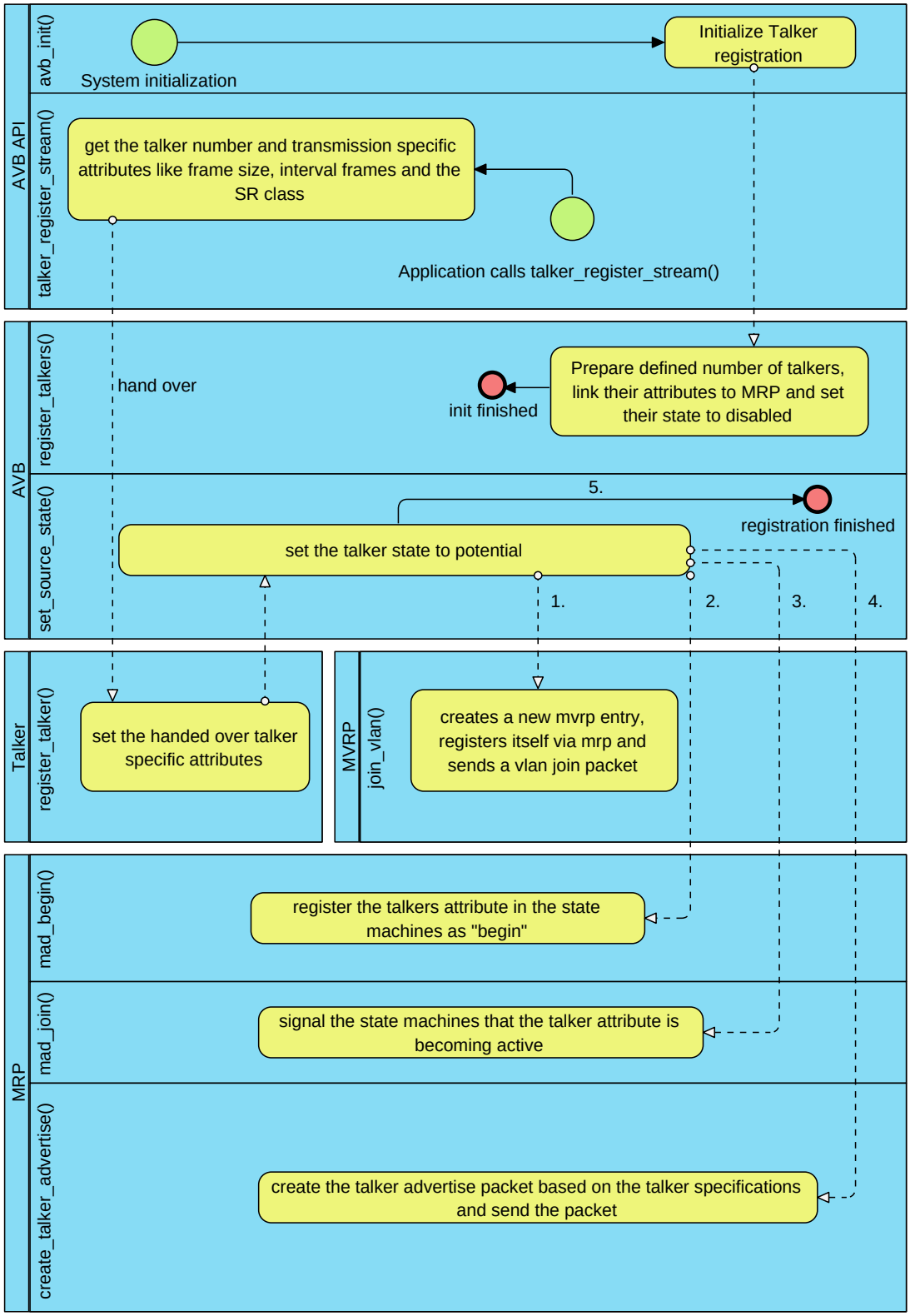

Abbildung 6.5.: Ablauf der Talker Initialisierung und Anmeldung eines Streams.

Paket versendet. Über ein „`mrp_mad_begin()`“ wird das Talker-Attribut in den State Machines als „BEGIN“ registriert. Anschließend folgt das „`mrp_mad_join()`“, um den State Machines mitzuteilen, dass das Talker-Attribut aktiv wird. Zuletzt wird über die „`create_talker_advertise()`“-Funktion anhand der vorher gesetzten Talker-Informationen das Talker Advertise-Paket erstellt und versendet. Damit wurde der Stream im Netzwerk bekanntgegeben und der Talker kann auf eine Antwort von seiner Bridge warten, die ihm mitteilt, ob die Bandbreite für seinen Stream reserviert wurde oder nicht.

6.4. Umsetzung des CBS

Der von Meyer (2013) übernommene Ansatz des CBS-Algorithmus wurde an die Architektur angepasst und durch die in den folgenden Kapiteln beschriebenen Strukturen erweitert.

6.4.1. Verwaltungsstruktur für Nachrichtenklassen

Um den Credit Based Shaper im Bezug auf Erweiterbarkeit flexibel zu halten, wurden klassenbezogene Parameter in einer Struktur untergebracht. Dies hat den Vorteil, dass der Algorithmus auf einfache Art und Weise um eine zusätzliche Nachrichtenklasse erweitert werden kann. Die Struktur ist in Abbildung 6.6 dargestellt. Der „Credit“ repräsentiert den Wert des Kreditsys-

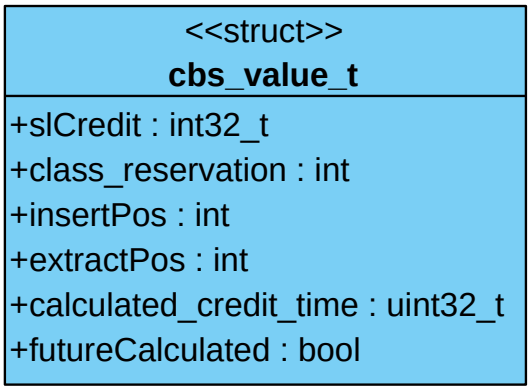

Abbildung 6.6.: Verwaltungsstruktur der Nachrichtenklassen des CBS.

tems, der von der idleSlope oder sendSlope erhöht beziehungsweise verringert wird. Da dieser auch einen negativen Wert haben kann, wird der Wert in einem 32 Bit vorzeichenbehafteten Integer dargestellt. „Class Reservation“ enthält die für die AVB Klassen reservierte Bandbreite. Die „Calculated Credit Time“ gibt an, bis zu welchem Zeitpunkt der Credit vom Algorithmus berechnet wurde. Das Boolean „Future Calculated“ signalisiert, ob der berechnete Credit in

der Zukunft liegt. Dies ist der Fall, wenn zum Beispiel die sendSlope für die Übertragung des Pakets den Credit im Voraus berechnet. Die „Insert“- und „Extract“-Position zeigt die Einfüge- und Entnahme-Position des nächsten Pakets für die Verwaltungsstruktur der Queues, worauf im folgenden Abschnitt näher eingegangen wird.

6.4.2. Queues & Buffer

Für jede Nachrichtenklasse existiert eine in Abbildung 6.7 dargestellte Verwaltungs-Queue. Die Queue wird als Array deklariert und verfügt über eine konfigurierbare Anzahl an „node_t“-

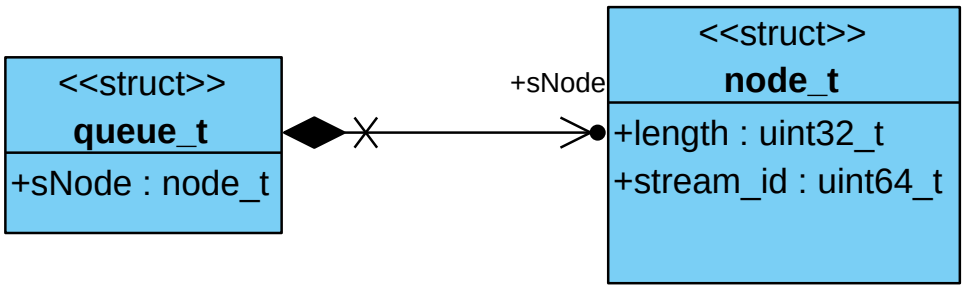

Abbildung 6.7.: Struktur der Verwaltungs-Queues des CBS.

Feldern. In den Feldern wird die Paketlänge und zugehörige Stream ID, sofern vorhanden, gespeichert. Wenn eine Anwendung die erstellten und zu versendenden Pakete in einen Buffer gespeichert hat, wird der CBS unter Angabe der Traffic Klasse, Stream ID und Paketgröße darüber informiert, dass er ein Paket versenden soll. Diese Informationen werden über die API-Funktion „`avb_send_frame()`“ an die CBS-Funktion „`cbs_process_packet()`“ weitergereicht. Dort werden die Informationen in ein neues Array-Feld gespeichert. Die in Abbildung 6.6 gezeigte „Extract“-Position wird beim Einfügen inkrementiert. Über die „Insert“- und „Extract“-Positionen kann der CBS-Algorithmus überprüfen, ob noch weitere Pakete darauf warten versendet zu werden und anhand der Stream ID auf den zugehörigen Buffer zugreifen, um die Pakete versenden zu können.

6.4.3. Abfragen der TT-Sendezeitpunkte

Da der TT-Verkehr von AVB- und BE-Paketen nicht beeinflusst werden darf, muss der CBS-Algorithmus die Versendezeitpunte sowie die Länge der TT-Pakete kennen. Während der Initialisierungsphase wird beim Starten des CBS-Algorithmus die Funktion „`Initial_tte_msg_check()`“ aufgerufen. Diese initialisiert die Array-Felder, die aus der in Abbildung 6.8 gezeigten Struktur bestehen. Die Adresse des Arrays wird der Funktion

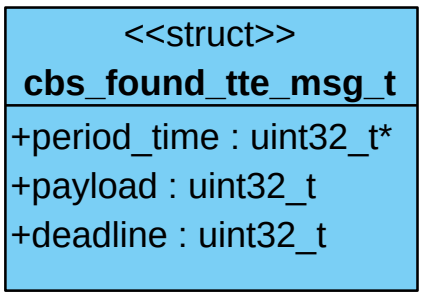

Abbildung 6.8.: Struktur für geschedulte TTEthernet Einträge.

„`scheduler_check_tt_send_msg()`“ übergeben, welche die im Scheduler eingetragenen Events auf zu sendende TT-Nachrichten überprüft. Wenn ein Eintrag gefunden wurde, wird die Adresse der Periodendauer in die „period_time“-Pointer Variable gespeichert. Da in den Scheduler-Einträgen die Periodendauer immer zur absoluten Systemzeit angegeben ist, wird dieser Wert mit jedem Zyklus vom Scheduler aktualisiert und muss deshalb nicht bei jedem neuen Scheduler-Zyklus erneut ausgelesen werden. Die Paketlänge der TT-Pakete wird in der „payload“-Variable gespeichert. Da diese über den Schedule vorher statisch konfiguriert wurde und daher immer gleich bleibt, muss sie nicht mehrfach ausgelesen werden. Die „deadline“-Variable wird zum Übermitteln der zeitlichen Sende- und Verarbeitungsdauer von TT-Synchronisierungspaketen benutzt, um dies beim Senden von AVB- und BE-Paketen berücksichtigen zu können.

Anhand der Informationen über die Sendezeitpunkte und die Paketlängen kann der CBS-Algorithmus beim Versenden der Pakete errechnen wie lange die Übertragung der TT-Pakete benötigt und ob er davor noch ein AVB- oder BE-Paket senden darf. Dies wird in der „`check_tt_messages()`“-Funktion durch Übergabe der Nachrichtenklasse und Verwaltungsstruktur berechnet.

6.4.4. Beispiel des CBS Algorithmus

Abbildung 6.9 zeigt den Ablauf des CBS-Algorithmus anhand der BPMN. In der Abbildung wird der Ablauf des Algorithmus beim Versenden eines Pakets der AVB-Klasse A erläutert. Für dieses Beispiel wird angenommen, dass vorher keine Pakete übertragen wurden, weshalb der Credit gleich Null ist. Des weiteren wird zum Sendezeitpunkt kein anderes Paket übertragen, welches den Ausgangsport blockiert und es ist genug Zeit, um das AVB-Paket bis zum nächsten TT-Paket versenden zu können.

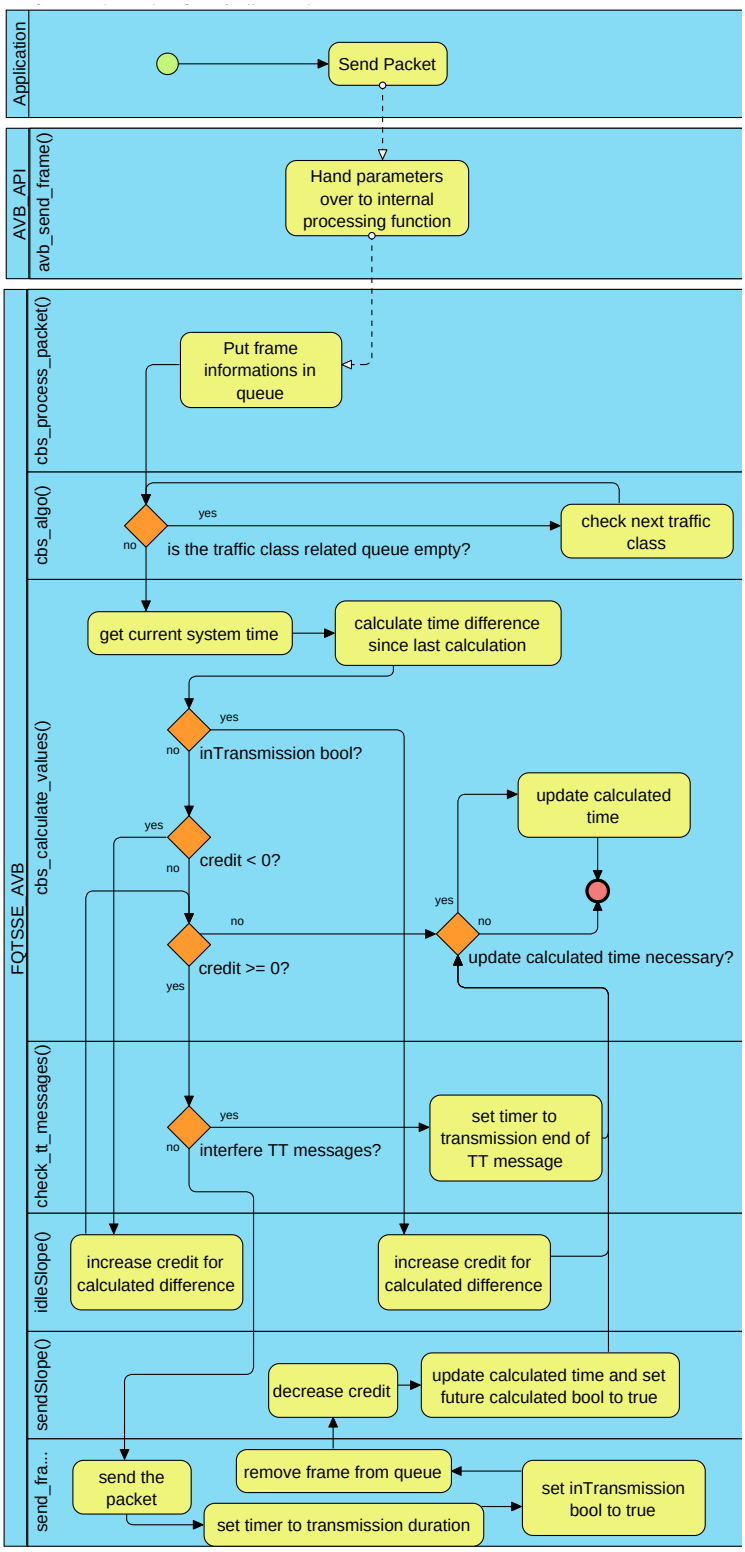

Abbildung 6.9.: Darstellung des CBS-Algorithmus in der BPMN.

Wenn die Anwendung ein zuvor erstelltes und in einem Buffer gespeichertes Paket versenden möchte, ruft sie unter Angabe der Nachrichtenklasse, Paketgröße und Stream ID die Funktion „`avb_send_frame()`“ auf. Da die Funktion zu der AVB API gehört, gibt sie die Parameter an die „`cbs_process_packet()`“-Funktion weiter. Dort wird einem Eintrag der Verwaltungs-Queue für die jeweilige Nachrichtenklasse die Paketlänge und Stream ID zugewiesen. Anschließend wird die „`cbs_algo()`“-Funktion aufgerufen, welche die Nachrichtenklassen der Priorität nach überprüft, ob deren Verwaltungs-Queues Pakete vermerkt haben, die gesendet werden sollen. Ist in der Queue von Klasse A beispielsweise ein Eintrag vorhanden, wird die Verwaltungsstruktur der Klasse A an die Funktion „`cbs_calculate_values()`“ übergeben. Der Algorithmus berechnet den Credit durch das Ermitteln der zeitlichen Differenz von der letzten Credit-Berechnung bis hin zum aktuellen Zeitpunkt des Funktionsaufrufs. Aus diesem Grund wird die aktuelle Systemzeit mit der Verwaltungsstruktur-Variable „calculated_credit_time“ verglichen und die Differenz errechnet. Zum Zeitpunkt des Funktionsaufrufs findet keine Paketübertragung von AVB- oder BE-Paketen statt, weshalb „inTransmission“ auf „false“ gesetzt ist und folglich überprüft wird, ob der Credit kleiner als Null ist. Da der Credit auf Null steht, wird die Funktion „`check_tt_messages()`“ aufgerufen und überprüft, ob das Senden des AVB-Pakets die Übertragung von zuvor geschedulten TT-Paketen stören oder verzögern würde. Die Überprüfung liefert das Ergebnis, dass das AVB-Paket übertragen werden kann. Durch Aufruf der „`send_frame()`“-Funktion unter Angabe der Nachrichtenklasse wird aus der Verwaltungs-Queue die Paketgröße und Stream ID des Pakets ermittelt. Anhand der Stream ID kann der Buffer, in dem das vorbereitete Paket liegt, ermittelt und das Paket versendet werden. Damit während des Zeitraums der Paketübertragung kein anderes Paket gesendet werden kann, wird „inTransmission“ auf „true“ gesetzt und ein Timer mit der Länge der Sendedauer gestartet. Läuft dieser Timer aus, wird dies in der Background Task erkannt und „inTransmission“ wieder auf „false“ gesetzt. Der Eintrag in der Verwaltungs-Queue wird entfernt und der Credit durch Aufrufen der „`sendSlope()`“-Funktion verringert. Zuletzt wird die Variable „calculated_credit_time“ aktualisiert, da mit der Berechnung der sendSlope der Credit über die aktuelle Zeit hinaus berechnet wurde.

6.5. AVB API, Konfigurationsmöglichkeiten und Testfälle

Um Anwendungen den Zugriff auf die Implementierung zu vereinfachen, werden die für eine Anwendung relevanten Funktionen über die eigens entworfene AVB API-Datei bereitgestellt. Neben der Initialisierungs- und Startfunktion von AVB sind des weiteren die Initialisierung der AVB Buffer sowie die Buffer spezifischen Funktionen untergebracht. Ein Talker Stream kann

über die Funktion „avb_talker_register_stream()“ seinen Stream anmelden und über die „avb_talker_deregister_stream()“-Funktion abmelden. Genauso kann ein Listener über „avb_listener_get_stream()“ einen Stream empfangen, von dem er zuvor ein Talker Advertise erhalten hat oder den Stream über ein „avb_listener_leave_stream()“ verlassen. Zum Senden von Paketen über den CBS-Algorithmus kann die Funktion „avb_send_frame()“ aufgerufen werden. Zuletzt ist für Debuggingzwecke die „avb_error_report()“-Funktion enthalten, welche Fehlerwerte von Funktionsrückgaben über die Terminal-Schnittstelle ausgibt.

Weitere Konfigurationsmöglichkeiten, wie zum Beispiel das Ein- oder Ausschalten von TT-Schedules oder Debuggingausgaben über die Terminal-Schnittstelle, wurden über „#define“-Einträge in der „AVB.h“ Datei realisiert. So können Code-Abschnitte durch „#ifdef“-Blöcke bei Bedarf aktiviert werden. Die folgende Auflistung gibt einen Überblick der möglichen Defines und deren Funktionen:

- AVB_ACTIVE : Aktiviert die Initialisierung und das Starten genereller Funktionen von AVB
- SENDMODE : Ermöglicht das generelle Versenden von Frames über den CBS, sowie die Ausführung von Testfällen
- RECEIVEMODE : Pakete werden lediglich empfangen und verarbeitet
- MEASUREMODE : Aktiviert GPIOs (General Purpose Input/Outputs) zum Messen von Ereignissen im Programmablauf
- AVB_NUM_OF_SOURCES : Konfiguriert die Anzahl der zu initialisierenden Talker
- AVB_NUM_OF_SINKS : Konfiguriert die Anzahl der zu initialisierenden Listener
- TT_ACTIVE : Aktiviert das Versenden von Synchronisierungspaketen sowie TT-Scheduler-Einträgen und die Überprüfung der TT-Nachrichten im CBS-Algorithmus
- AVB_DEBUG : Generelle AVB Debugging-Informationen über Terminal-Ausgaben
- AVB_TALKER_DEBUG : Talker-bezogene Terminal-Ausgaben

Über die Funktionen „avb_test_1()“ bis „avb_test_4()“ können bestimmte Szenarien zum Versenden von AVB- und BE-Paketen über den CBS getestet werden. Durch Übergabeparameter kann die Anzahl der Pakete sowie die Paketgröße individuell für jeden Testfall konfiguriert werden. AVB-Pakete werden als Klasse A-Pakete und Talker Advertise-Pakete als BE-Pakete versendet.

6.6. Stack-Architektur

In Abbildung 6.10 wird die aus der Implementierung resultierende Stack-Architektur dargestellt. Wie in Abschnitt 6.5 bereits erwähnt wurde, lässt sich der AVB Stack einzeln betreiben sowie in Verbindung mit dem Time-Triggered Stack. Je nach Konfiguration benutzt der AVB Stack Teil-Funktionen des Time-Triggered Stacks. So werden BE-Buffer, ein Scheduler für Mixed

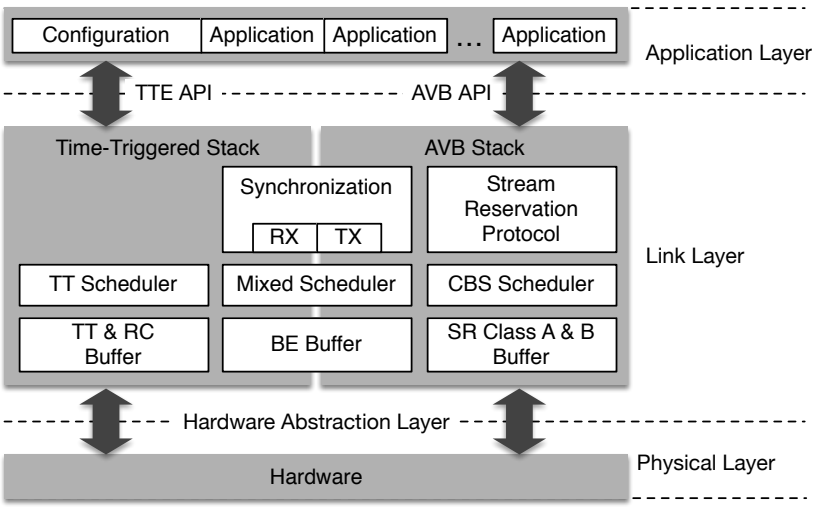

Abbildung 6.10.: Stack-Architektur, AVB und time-triggered Ethernet.

Traffic (TT, AVB und BE) und der Synchronisations-Teil stackübergreifend verwendet. Der AVB Stack erweitert die vorherige Time-Triggered-Implementierung um das Stream Reservation Protocol, die Buffer für Klasse A & B Nachrichten sowie den CBS Scheduler. Die bereits bestehende Time-Triggered-Implementierung wurde um das Stream Reservation Protocol, den CBS Scheduler, den Mixed Scheduler und die Buffer für die Stream Reservation Klassen A und B erweitert. Die beiden API-Schnittstellen bieten den Anwendungen getrennten Zugriff auf die jeweiligen Stack-Funktionen. Über den Hardware Abstraction Layer (HAL) wird der Zugriff auf die Hardware ermöglicht, wodurch eine gewisse Flexibilität bei der Verwendung des Quelltextes auf anderen Plattformen gegeben ist.

6.7. Erweiterbarkeit

In Hinsicht auf die Weiterentwicklung des AVB-Standards zum TSN-Standard wird im Folgenden kurz auf die Erweiterbarkeit der Implementierung eingegangen. Wie in Abschnitt bereits 4.2 erwähnt wurde, soll der TSN-Standard unter anderem das Behandeln von gesche-

dultem Traffic ermöglichen, was durch sogenannte „Transmission Gates“ realisiert werden soll. Da diese Gates festlegen, ob die jeweilige Queue der Nachrichtenklasse für die Übertragung freigegeben ist, ließe sich dies durch Erweitern der in Abschnitt 6.4.1 beschriebenen Verwaltungsstruktur der Nachrichtenklassen um beispielsweise ein Boolean realisieren. Der Zustand des Boolean könnte beim Aufruf des CBS-Algorithmus abgefragt werden, ob die Queue sendeberechtigt ist. Die Implementierung bietet in Bezug auf den kommenden TSN-Standard somit eine gewisse Erweiterungsmöglichkeit, um auf protokollspezifische Änderungen angepasst werden zu können.

7. Evaluierung

In diesem Kapitel wird die Implementierung anhand von Testfällen geprüft und eine generelle Qualitätssicherung durchgeführt.

Da es zum Zeitpunkt der Messungen keine Bridge gab, die TTEthernet und AVB unterstützt, wurden die Messergebnisse mit zwei NXHX500-ETM Boards durchgeführt, die als Endgeräte fungierten und über ein Ethernet-Kabel miteinander verbunden waren. Das Verifizieren des AVB-Teils über eine AVB-Bridge war aufgrund der nicht vorhandenen Implementierung des gPTP und 1722-Protokolls nicht möglich. Weitherhin wurden durch die GPIO-Stifte an dem jeweiligen Board mit einem digitalen Messgerät die zeitlichen Aufrufe entscheidender Funktionen aufgezeichnet sowie die Ethernet-Pakete über das Abgreifen der Signale von einem Patchfeld mit einer Messspitze protokolliert.

Aufgrund der unterschiedlichen Hardware-Uhren zwischen Messgerät und dem NXHX500-ETM Board ist in den Messungen ein systematischer Fehler erkennbar.

Für den Betrieb von TTEthernet sind Synchronisierungsnachrichten notwendig, die am Anfang der jeweiligen Scheduling-Periode aufgrund der Vorbereitungs- und Versendezeit die Übertragung jeglicher Pakete für eine Dauer von 290 µs blockieren. Da diese Dauer in Anbetracht eines AVB-Klasse-A-Intervalls von 125 µs nicht unerheblich ist, muss dies bei den Messungen, die TT-Nachrichten beinhalten, berücksichtigt werden.

7.1. Qualitätssicherung

In den nachfolgenden Unterkapiteln werden generelle Funktionen der Implementierung getestet und deren Ergebnisse dargestellt.

7.1.1. TT-Funktionstest mit und ohne AVB-Verkehr

Wie in Kapitel 5.1.2 in Punkt zwei bereits erwähnt wurde, soll sicher gestellt werden, dass der TT-Verkehr von der AVB-Implementierung weder verzögert, noch in anderer Weise beeinflusst werden darf. Im ersten Test wurde die Latenz eines TT-Schedules, der alle 2 ms ein 60 Byte TT-Paket versendet, überprüft. Im zweiten Test kam zu dem vorherigen TT-Schedule die

Versendung von AVB-Klasse-A-Paketen hinzu, die eine Paketgröße von 64 Byte hatten. In Abbildung 7.1 ist das Ergebnis der beiden Tests dargestellt. Sie zeigt, dass die TT-Latenzen bei zusätzlichem AVB-Verkehr geringfügig größer werden, was auf die Verarbeitungszeit der AVB-Funktionen zurückzuführen ist. Die Latenzen liegen unter den Spezifikationsvorgaben von 1 µs, was bedeutet, dass kein Empfangszeitpunkt der TT-Pakete eine zu hohe Latenz aufweist.

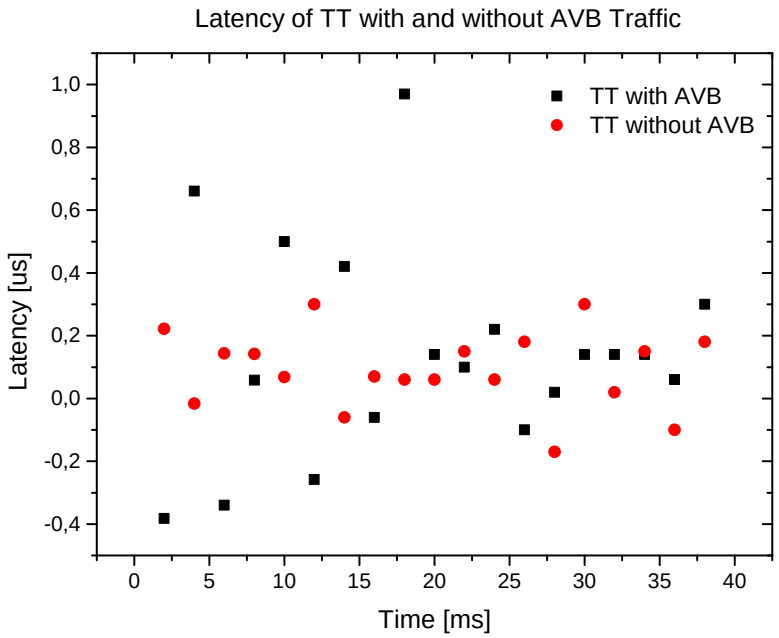

Abbildung 7.1.: TT-Latenz ohne und mit AVB-Verkehr.

7.1.2. AVB-Intervallzeiten mit und ohne TT-Verkehr

In Abbildung 7.2 werden die Empfangszeiten von AVB-Klasse-A-Nachrichten mit und ohne TT-Nachrichten gezeigt. Im ersten Test der AVB-Intervallzeiten wurden AVB-Klasse-A-Nachrichten mit einer Paketgröße von 64 Byte versendet. Im zweiten Test der AVB-Intervallzeiten kam ein TT-Schedule dazu, der alle 2 ms ein 60 Byte TT-Paket versendete. Abbildung 7.2 zeigt, dass die meisten AVB-Nachrichten - mit und ohne TT-Verkehr - Empfangszeitpunkte von etwa 130 µs aufweisen. Diese Abweichungen zu den eigentlichen 125 µs sind zum einen auf den systematischen Fehler der Uhrendifferenzen zurückzuführen, zum anderen jedoch auf die Verarbeitungszeit des Systems, da der Aufruf zum Ausführen des CBS-Algorithmus aus der

Background Task heraus aufgerufen wird. Die erkennbaren Spitzenwerte der Messung von AVB-mit TT-Nachrichten sind darauf zurückzuführen, dass die Messung der Empfangszeitpunkte von AVB nicht die Übertragung der erwähnten Synchronisierungsnachrichten für den TT-Betrieb mit einbezieht. Aus diesem Grund treten bei der Berechnung der AVB-Differenzzeiten alle 2 ms zu Beginn der neuen Scheduling-Periode sowie 1 ms nach Beginn der neuen Scheduling-Periode, wenn das TT-Paket versendet wird, Spitzen auf. Die Verzögerungen der AVB-Nachrichten unterstreichen das korrekte Verhalten des CBS-Algorithmus, da dieser sicherstellt, dass keine Synchronisierungs- und TT-Pakete von AVB-Paketen maßgeblich verzögert oder unterbrochen werden.

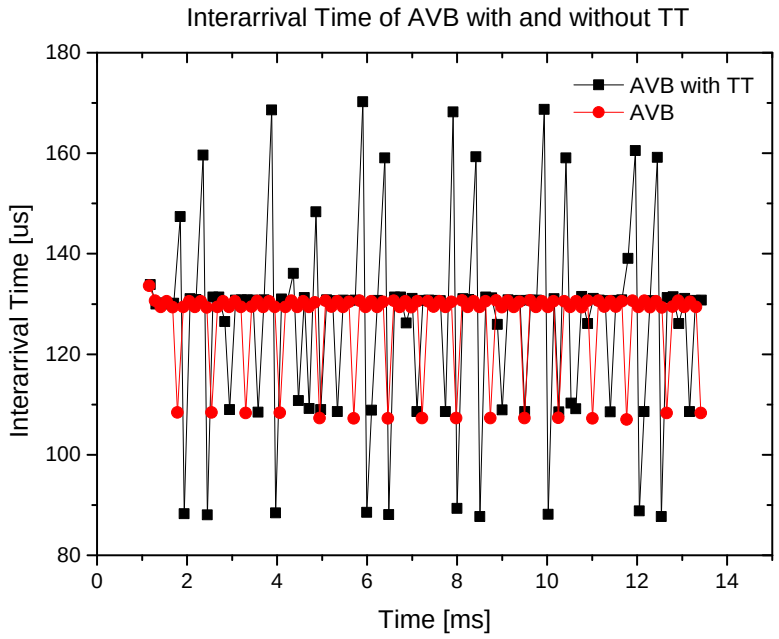

Abbildung 7.2.: AVB-Empfangszeiten mit und ohne TT-Verkehr.

7.2. Fallbeispiele

In diesem Unterkapitel werden die Ergebnisse der Tests von unterschiedlichen Konfigurationsszenarien dargestellt und erläutert.

7.2.1. Verzögerung von AVB-Verkehr aufgrund eines zu engen TT-Schedules

Im Folgenden Szenario werden zwei auf das Szenario bezogene Tests erläutert. Der erste Test des Szenarios zeigt, dass durch einen TT-Schedule zu wenig Platz für AVB-Pakete die AVB-Pakete nicht mehr versendet werden können. Der zweite Test zeigt, dass bei einer Lücke von 38 µs ein einziges AVB-Paket innerhalb einer Scheduling-Periode versendet werden kann. Für die Tests wurde ein TT-Schedule mit einer Periodenlänge von 2 ms verwendet. Der Schedule

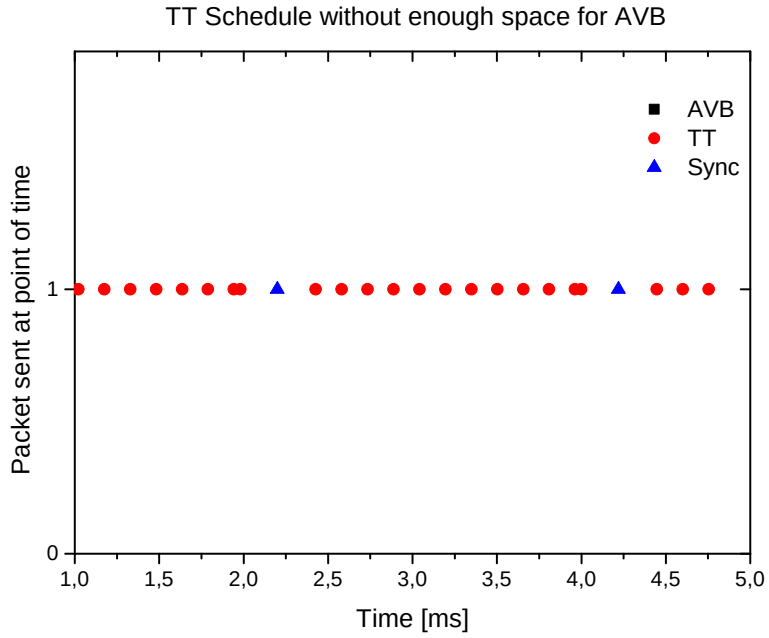

Abbildung 7.3.: TT-Schedule lässt nicht genügend Platz für AVB-Pakete.

setzt sich aus dem Synchronisierungspaket und 11 darauf folgenden TT-Paketen mit einer Paketgröße von jeweils 1500 Byte, die in 152 µs-Abständen versendet werden, zusammen. Damit sind von den möglichen 2 ms noch 38 µs für andere Nachrichtenklassen wie AVB oder BE verfügbar. Um für den ersten Test die verbleibenden 38 µs zu füllen, so dass kein AVB-Paket versendet werden kann, wird ein TT-Paket mit einer Paketgröße von 60 Byte an den Schedule angehängt. Da die Übertragung eines 60 Byte TT-Pakets 6.72 µs benötigt, bleiben nur noch 31.28 µs, die zum Versenden eines 400 Byte großen AVB-Pakets nicht ausreichen. Die 152 µs Abstände beinhalten nach der Versendedauer des TT-Pakets eine Lücke von 30 µs. Diese Lücke macht es dem CBS-Algorithmus unmöglich sein 400 Byte großes AVB-Paket zu versenden, da dieses mit einer Übertragungsdauer von 35 µs die TT-Nachricht verletzen würde und somit

nicht versendet werden kann. Das Ergebnis dieses Schedules ist in Abbildung 7.3 dargestellt. Damit der zweite Test zeigen kann, dass das Versenden eines einzelnen AVB-Pakets innerhalb einer Scheduling-Periode möglich ist, wird das 60 Byte TT-Paket aus den Scheduling-Einträgen genommen. Dadurch bietet der Schedule eine Lücke von 38 µs, die gerade ausreichend für die

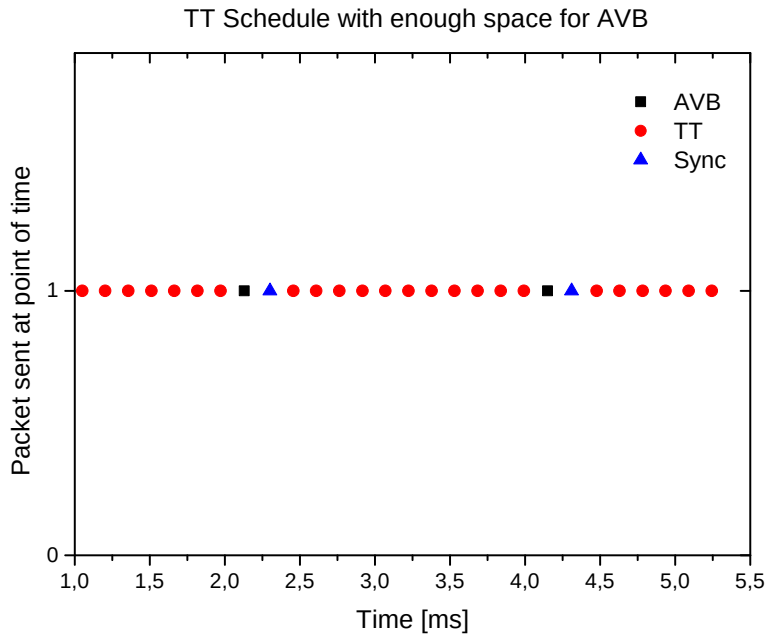

Abbildung 7.4.: TT-Schedule lässt genügend Platz für ein AVB-Paket.

Übertragung eines AVB-Pakets mit einer Größe von 400 Byte ist. In Abbildung 7.4 wird das Ergebnis dargestellt.

7.2.2. Versenden von TT-, AVB- und BE-Paketen

In diesem Szenario wird das Versenden von TT-, AVB- und BE-Paketen unter Berücksichtigung eines bestimmten TT-Schedules dargestellt. Der Schedule setzt sich aus dem Synchronisierungspaket mit einer Dauer von 290 µs (T_{Sync}) und sechs darauf folgenden 1500 Byte TT-Paketen zusammen. Die TT-Pakete werden in 248 µs ($T_{TTScheduleLength}$) Abständen versendet. Diese Abstände beinhalten die Übertragungsdauer des TT-Pakets mit ∼122 µs sowie einer 126 µs (T_{Gap}) Lücke. Nach dem TT-Paketen bleiben innerhalb der Scheduling-Periode von 2 ms noch ∼222 µs Zeit für AVB- und BE-Pakete. Es werden AVB-Klasse-A-Pakete mit einer Paketgröße von 64 Byte sowie BE-Pakete mit einer Paketgröße von 128 Byte versendet.

Das in Abbildung 7.5 dargestellte Ergebnis zeigt die Empfangszeiten der Pakete über einen Zeitraum von 4 ms. Die in der Abbildung erkennbaren Verzögerungen der Empfangszeiten von TT-Paketen sind in Formel 7.1 erläutert.

$$T_{delay} = T_{Gap} + T_{Sync} + T_{TTScheduleLength} = 126\,\mu s + 290\,\mu s + 248\,\mu s = \underline{\underline{664\,\mu s}} \quad (7.1)$$

In ähnlicher Weise variieren zu Beginn der Scheduling-Periode die Empfangszeiten der AVB- und BE-Pakete. Zwischen den TT-Paketen findet der CBS-Algorithmus Platz, um ein BE- und AVB-Paket oder zwei AVB-Pakete zu versenden, je nachdem wie der Kredit der Klasse A Queue steht. Des Weiteren wird sichtbar, dass der CBS-Algorithmus im Falle eines hohen Kredits für Klasse A die AVB-Pakete innerhalb kürzerer Abstände versendet, was in Abbildung 2.9 bereits erläutert wurde.

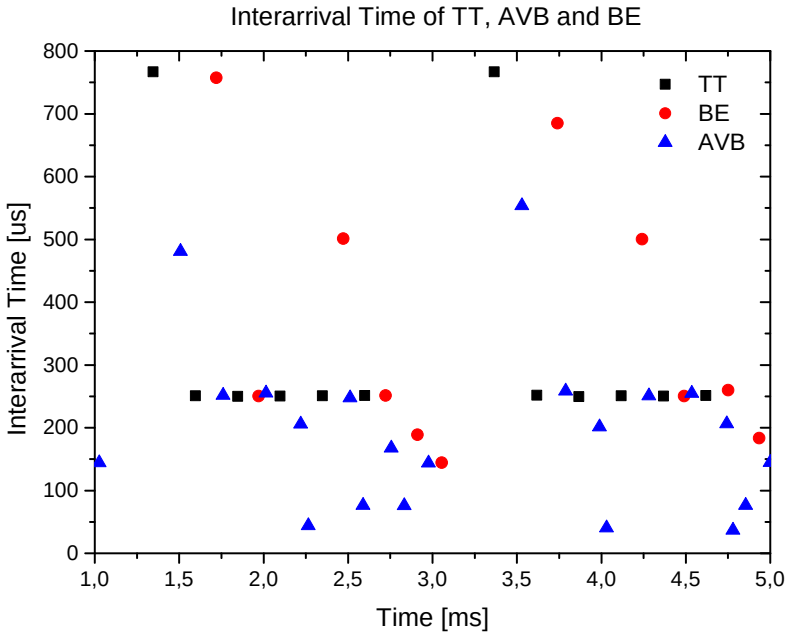

Abbildung 7.5.: Empfangszeiten von TT-, AVB- und BE-Paketen.

7.3. CPU-Auslastung und Zeitmessungen

In diesem Abschnitt werden Messergebnisse wie CPU-Auslastung, die Dauer von Funktionsaufrufen sowie die Speicherbelegung des Systems erläutert. Die CPU-Auslastung gemessen

an der Background-Task beläuft sich während der Übertragung von 100 AVB-Paketen über einen Zeitraum von ∼13 ms auf maximal 70%. Die Verarbeitungsdauer des CBS-Algorithmus während der Übertragung eines AVB-Pakets beläuft sich auf 64 µs.
Die Verarbeitungszeit von MRP-Paketen über den Funktionsaufruf „`mrp_packet_process()`" wurde mit einem Minimum von 400 ns und einem Maximum von 1.4 µs gemessen.

7.4. PICS proforma — End station implementations

Ein Protocol Implementation Conformance Statement (PICS) in Form eines sechsseitigen Fragebogens ist in Anhang A beigefügt. Der PICS zeigt die implementierten Fähigkeiten und Optionen der Endstationen, gemäß der Spezifikation des Standards IEEE 802.1Q (vgl. Institute of Electrical and Electronics Engineers (2011c)).

8. Fazit und Ausblick

Den abschließenden Teil der Arbeit bildet das Fazit, die Zusammenfassung der Ergebnisse sowie ein Ausblick über zukünftige Arbeiten.

8.1. Zusammenfassung und Ergebnisse

Da in heutigen Automobilfahrzeugen, unter Zunahme der darin verbauten technischen Geräte, die Komplexität der Kommunikationsnetze steigt, wird aufgrund der hohen Bandbreitenanforderungen ein Übertragungsnetz erforderlich, das den Anforderungen gerecht werden kann. TTEthernet ist ein auf dem Standard Ethernet basierendes Echtzeit-Protokoll, das diesen Anforderungen gerecht wird. Mit diesem Protokoll jedoch geht ein Konfigurationsaufwand einher, der bei zunehmender Kommunikationsnetzkomplexität erheblich steigt. Um den Aufwand zu verringern, wurde eine bestehende TTEthernet-Implementierung um das Audio/ Video Bridging-Protokoll erweitert, das eine dynamische Konfiguration ermöglicht. Damit das AVB-Protokoll gemäß der Spezifikationen implementiert werden konnte, wurden umfangreiche Informationen über die IEEE Standards 802.1Qat und 802.1Qav erlangt. Das Ziel dieser Arbeit einen ressourcenschonenden Stack zur Unterstützung von AVB und Time-Triggered-Ethernet Verkehr zu entwickeln, wurde durch die Anforderungen sowie das Erarbeiten eines Konzepts formuliert. Bei der Evaluation konnte gezeigt werden, dass der Anforderungspunkt, der voraussetzt, dass der TT-Verkehr durch die Erweiterung von AVB-Verkehr nicht maßgeblich beeinflusst wird, eine Latenz von unter 1 µs aufweist. Weiterhin konnte gezeigt werden, dass der CBS-Algorithmus dahingehend funktioniert, dass TT-, AVB- und BE-Pakete korrekt versendet werden.
Während der Bachelorarbeit wurde ein Paper im Rahmen der ICCE-Konferenz 2014 veröffentlicht (vgl. Rumpf u. a. (2014)).

8.2. Ausblick

Die Arbeiten der CoRE-Arbeitsgruppe werden sich auch in Zukunft mit der weiteren Entwicklung von AVB zu TSN beschäftigen sowie der Kommunikation von TTEthernet. Nachfolgend sind die Themen aufgelistet:

- **Evaluationsframework für eine Kombination von Realtime-Ethernet-Konzepten in einer Ethernet-Bridge**
  Es wird ein Framework entworfen durch den eine AVB- und TTEthernet-fähige Bridge ermöglicht werden soll. Dadurch können die Ergebnisse dieser Arbeit innerhalb eines AVB- und TTEthernet-fähigen Netzes evaluiert werden.
- **Auswirkung von Frame-Präemption auf das Echtzeitverhalten in Simulation**
  Gemäß des TSN-Standards der den AVB-Standard erweitert, wird die Unterbrechbarkeit von Paketen und deren Auswirkung auf das Echtzeitverhalten überprüft.
- **Worst-Case Timing-Analyse von TSN anhand verschiedener Analyse-Techniken**
  Als Ergebnis soll eine analysierte und evaluierte Topologie mit TSN entstehen.

Des weiteren könnte das korrekte Verhalten von AVB-Klasse-B-Nachrichten geprüft werden, da diese bereits in der Implementierung enthalten sind, jedoch voranging Klasse-A-Verkehr aufgrund seiner höheren Priorität evaluiert wurde. In Hinblick auf die Erweiterung von AVB zu TSN, bietet die Implementierung die Möglichkeit, um weitere Nachrichtenklassen erweitert zu werden. Außerdem besteht die Möglichkeit, die von der TSN-Gruppe in dem Standard IEEE 802.1Qbv (vgl. IEEE 802.1 TSN Task Group (c)) beschriebene Einführung eines „Transmission Gates“ zu implementieren, da die Verwaltungsstruktur des CBS flexibel und anpassbar ist.

A. Protocol Implementation Conformance Statement (PICS) proforma — End station implementations

In diesem Kapitel wird anhand des PICS Fragebogens für die Implementierung von Endstationen aus dem Standard IEEE 802.1Q (vgl. Institute of Electrical and Electronics Engineers (2011c)) gezeigt, welche Fähigkeiten und Optionen der Spezifikation implementiert wurden. Nachfolgend werden die für das PICS benutzten Symbole erklärt sowie der sechsseitige Fragebogen dargestellt:

Symbole

M : Mandatory (Pflicht)

O : Optional

$O.n$: Optional, jedoch muss zumindest eine Option der Gruppe, die durch n gekennzeichnet ist, unterstützt werden

N/A : Not Applicable (Nicht andwendbar)

B.5 Major capabilities

Item	Feature	Status	References	Support	
MRPAP	Does the implementation support any MRP applications? If "No" is marked, continue at FQTSSE	O	5.16.1	Yes [✓]	No [ ]
MMRP	Is the operation of MMRP supported?	O.1	5.16.1, B.6	Yes [✓]	No [ ]
MVRP	Is automatic configuration and management of VLAN topology using MVRP supported?	O.1	5.16.1, B.7	Yes [✓]	No [ ]
MSRP	Is the operation of MSRP supported?	O.1	5.16.3, B.10	Yes [✓]	No [ ]
MRP	Is the Multiple Attribute Registration Protocol (MRP) implemented in support of MRP Applications?	M	10 B.6, B.7, B.8	Yes [✓]	
SPRU	Does the implementation support Source Pruning?	O	5.16, 10.10.3, 11.2.1.1	Yes[ ]	No [✓]
MRP1	Does the MRP implementation support operation of the Full Participant?	SPRU:O.2 ¬SPRU:O.3	10 B.8	Yes [✓]	No [ ]
MRP2	Does the MRP implementation support operation of the Full Participant, point-to-point subset?	SPRU:O.2 ¬SPRU:O.3	10 B.8	Yes [ ]	No [✓]
MRP3	Does the MRP implementation support operation of the Applicant-Only Participant?	¬SPRU:O.3	10 B.8	Yes [✓]	No [ ]
MRP4	Does the MRP implementation support operation of the Simple-Applicant Participant, point-to-point subset?	¬SPRU:O.3	10 B.8	Yes [ ]	No [✓]
FQTSSE	Does the implementation support forwarding and queuing for time-sensitive streams?	O	5.18, 34.6	Yes [✓]	No [ ]
CN	Is congestion notification implemented?	O	5.19, 30, 31, 32, 33	Yes [ ]	No [✓]

B.6 MMRP

Item	Feature	Status	References	Support
	If MMRP is not supported, mark N/A and continue at B.7			N/A [ ]
MMRP1	Does the implementation support the exchange of MMRPDUs, using the generic MRPDU format defined in 10.8 to exchange MMRP-specific information, as defined in 10.12,?	M	5.4.1.3, 10.8, 10.12	Yes [✓]
MMRP2	Is the MMRP Application supported as defined in 10.12?	M	5.4.1.3, 10.12	Yes [✓]
MMRP5	Is the MAP Context Identifier used to identify a VLAN Context equal to the VID used to identify the corresponding VLAN?	M	10.12.1.1	Yes [ ]

B.6 MMRP *(continued)*

Item	Feature	Status	References	Support
MMRP7	Is the group MAC address used as the destination address for MRPDUs destined for MMRP Participants the group MAC address identified in Table 10-1 as "Customer and Provider Bridge MMRP address"?	M	10.12.1.3	Yes [✓]
MMRP8	Is the EtherType used for MRPDUs destined for MMRP Participants the MMRP EtherType identified in Table 10-2?	M	10.12.1.4, Table 10-2	Yes [✓]
MMRP9	Does the ProtocolVersion used for the implementation of MMRP take the hexadecimal value 0x00?	M	10.12.1.5	Yes [✓]
MMRP10	Are the Attribute Type values used in the implementation as specified in 10.12.1.6?	M	10.12.1.6	Yes [✓]
MMRP11	Does the implementation encode the values in FirstValue fields in accordance with the definition in 10.12.1.7?	M	10.12.1.7	Yes [✓]
MMRP12	Is management of the Restricted_MAC_Address_Registration control parameter supported?	O	10.12.2	Yes [ ] No [✓]
MMRP13	If management of the Restricted_MAC_Address_Registration control parameter is not supported, is the value of this parameter FALSE for all Ports?	¬MMRP12:M	10.12.2	Yes [ ] N/A [✓]
MMRP14	Does the implementations maintain state information for all attribute values that support the Group service requirement registration?	SPRU:M	10.10.2	Yes [✓] N/A [ ]
MMRP15	Is the implementation capable of supporting any attribute value in the range of possible values that can be registered using Group membership and individual MAC address registration?	M	10.12.4	Yes [✓]
MMRP16	State the number of Group membership and individual MAC address state values that can be supported on each Port.	M	10.12.4	Number 1

B.7 MVRP

Item	Feature	Status	References	Support
	If MVRP is not supported, mark N/A and continue at B.8			N/A [ ]
MVRP1	Does the implementation support the exchange of MMRPDUs, using the generic MRPDU format defined in 11.2 to exchange MMRP-specific information, as defined in 10.12?	M	5.4.2, 10.8, 11.2	Yes [✓]
MVRP2	Is the MMRP Application supported as defined in 11.2?	M	5.4.2, 11.2	Yes [✓]
MVRP7	Is the group MAC address used as the destination address for MRPDUs destined for MVRP Participants as defined in 11.2.3.1.3?	M	11.2.3.1.3	Yes [✓]
MVRP8	Is the EtherType used for MRPDUs destined for MVRP Participants the MVRP EtherType identified in Table 10-2?	M	11.2.3.1.4, Table 10-2	Yes [✓]
MVRP9	Does the ProtocolVersion used for the implementation of MVRP take the hexadecimal value 0x00?	M	11.2.3.1.5	Yes [✓]

B.7 MVRP *(continued)*

Item	Feature	Status	References	Support
MVRP10	Are the Attribute Type values used in the implementation as specified in 11.2.3.1.6?	M	11.2.3.1.6	Yes [✓]
MVRP11	Does the implementation encode the values in FirstValue fields in accordance with the definition in 11.2.3.1.7?	M	11.2.3.1.7	Yes [✓]
MVRP12	Is the implementation of MVRP capable of supporting all attribute values in the range of possible values that can be registered using MVRP?	SPRU:M	11.2.6	Yes [✓] N/A [ ]
MVRP13	Is the implementation capable of maintaining current state information for all attributes in the range of possible values?	SPRU:M	11.2.6	Yes [✓] N/A [ ]

B.8 MRP

Item	Feature	Status	References	Support
MRP5	Does the implementation of MRP meet all of the requirements for interoperability stated in 10.5 that apply to end station operation?	M	10.5	Yes [✓]
MRP6	Does the implementation support the operation of the complete Applicant state machine?	MRP1:M	10.7, 10.7.7	Yes [✓] N/A [ ]
MRP7	Does the implementation support the operation of the point-to-point subset of the Applicant state machine?	MRP2:M	10.7, 10.7.7	Yes [ ] N/A [✓]
MRP8	Does the implementation support the operation of the Applicant-Only subset of the Applicant state machine?	MRP3:M	10.7, 10.7.7	Yes [✓] N/A [ ]
MRP9	Does the implementation support the operation of the Simple-Applicant subset of the Applicant state machine?	MRP4:M	10.7, 10.7.7	Yes [ ] N/A [✓]
MRP10	Does the implementation support the operation of the Registrar state machine?	MRP1:M MRP2:M	10.7, 10.7.8	Yes [✓] N/A [ ]
MRP11	Does the implementation support the operation of the LeaveAll state machine?	MRP1:M MRP2:M	10.7, 10.7.9	Yes [✓] N/A [ ]
MRP12	Does the implementation support the operation of the PeriodicTransmission state machine?	M	10.7, 10.7.10	Yes [✓]

B.9 Forwarding and queuing for time-sensitive streams

Item	Feature	Status	References	Support
FQTSSE1	Support a minimum of two traffic classes, of which one supports the strict priority algorithm and the other is an SR class?	FQTSSE:M	5.18, 8.6.8.1, 34.6	Yes [✓] N/A [ ]
FQTSSE2	Support the credit-based shaper algorithm as the transmission selection algorithm for frames transmitted for each stream associated with the SR class.	FQTSSE:M	5.18, 8.6.8.2, 34.6	Yes [✓] N/A [ ]
FQTSSE3	Support the operation of the credit-based shaper algorithm on all Ports as the transmission selection algorithm used for the SR class.	FQTSSE:M	5.18, 8.6.8.2, 34.6	Yes [✓] N/A [ ]
FQTSSE4	Use the default priority associated with SR class "B" as shown in Table 6-6 as the priority value carried in transmitted SR class "B" data frames.	FQTSSE:M	5.18, Table 6-6, 34.6	Yes [✓] N/A [ ]
FQTSSE5	Support two or more SR classes (a maximum of seven), and support the operation of the credit-based shaper algorithm on all Ports as the transmission selection algorithm used for those SR classes. The number of SR classes supported shall be stated in the PICS.	FQTSSE:O	5.18, 8.6.8.2, 34.6	Yes [✓] No [ ] Number 2
FQTSSE6	Use the default priority associated with SR class "A" as shown in Table 6-6 as the priority value carried in transmitted SR class "A" data frames. If more than two SR classes are supported, the priority value carried in transmitted data frames for the additional SR classes shall be stated in the PICS.	FQTSSE:O	5.18, Table 6-6, 34.6	Yes [✓] No [ ] Priority override values_____

B.10 SRP (Stream Reservation Protocol)

Item	Feature	Status	Reference	Support	
	If SRP is not supported, mark N/A and ignore the remainder of this table.				N/A [ ]
SRP-1	Does the implementation support the exchange of MSRPDUs, using the generic MRPDU format defined in 10.8 to exchange MSRP-specific information, as defined in 35.2.2.8.1?	M	5.4.4, 10.8, 35.2.2.8.1	Yes [✓]	
SRP-2	Is the MSRP Application supported as defined in Clause 35?	M	5.4.4, 35	Yes [✓]	
SRP-3	Is the group MAC Address used as the destination address for MRPDUs destined for MSRP Participants the group MAC address identified in Table 8-1, Table 8-2, Table 8-3 as "Individual LAN Scope group address, Nearest Bridge group address"?	M	35.2.2.1, Table 8-1, Table 8-2, Table 8-3	Yes [✓]	
SRP-4	Is the EtherType used for MRPDUs destined for MSRP Participants the MSRP EtherType identified in Table 10-2?	M	35.2.2.2, Table 10-2	Yes [✓]	

B.10 SRP (Stream Reservation Protocol) *(continued)*

Item	Feature	Status	Reference	Support	
SRP-5	Does the ProtocolVersion used for the implementation of MSRP take the hexadecimal value 0x00?	M	35.2.2.3	Yes [✓]	
SRP-6	Are the Attribute Type values used in the implementation as specified in 35.2.2.4 and Table 35-1?	M	35.2.2.4, Table 35-1	Yes [✓]	
SRP-7	Are the Attribute Length values used in the implementation as specified in 35.2.2.5 and Table 35-2?	M	35.2.2.5, Table 35-2	Yes [✓]	
SRP-8	Are the MSRP Vector FourPackedEvents values used in the implementation as specified in 35.2.2.7.2 and Table 35-3?	M	35.2.2.7.2, Table 35-3	Yes [✓]	
SRP-9	Does the implementation encode the values in FirstValue fields in accordance with the definition in 35.2.2.8?	M	35.2.2.8	Yes [✓]	
SRP-10	Does the Talker implementation populate the Accumulated Latency with a reasonable, nonzero value?	M	35.2.2.8.6	Yes [✓]	
SRP-11	Does the implementation update the Failure Information Bridge ID and Code in the event of insufficient bandwidth or resources through a Bridge?	M	35.2.2.8.7	Yes [✓]	
SRP-12	Does the implementaiton create a Talker Failed in the event of insufficient bandwidth or resources through a Bridge?	M	35.2.4.3, 35-10	Yes [ ]	
SRP-13	Is talkerPruning and MMRP supported?	O	35.2.4.3.1	Yes [ ]	No [✓]
SRP-14	Are MSRPDUs transmitted on all ports?	M	35.2	Yes [ ]	
SRP-15	State the number of Talker registration values that can be supported on each Port.	M	35.2.7	Number 2	
SRP-16	State the number of Listener registration values that can be supported on each Port.	M	35.2.7	Number 2	
SRP-17	Does the Listener issue an appropriate MVRP VLAN membership request before attaching to a Stream?	M	35.1.2.2	Yes [✓]	
SRP-18	When MAC_Operational transitions to TRUE does the device declare the default SR class priority value as the SRclassPriority prior to receving an SRclassPriority declaration from its neighbor?	M	6.6.2, 35.2.2.9.3, Table 6-6	Yes [ ]	

B.10 SRP (Stream Reservation Protocol) ***(continued)***

Item	Feature	Status	Reference	Support	
SPR-19	Does the device set SRclassPriority to the value declared by the neighboring device?	M	35.2.2.9.3	Yes [ ]	
SRP-20	When MAC_Operational transitions to TRUE does the device declare the default SR_PVID value as the SRclassVID prior to receiving an SRclassVID declaration from its neighbor?	M	6.6.2, 35.2.2.9.4, Table 9-2	Yes [ ]	
SPR-21	Does the device set SRclassVID to the value declared by the neighboring device?	M	35.2.2.9.4	Yes [ ]	
SRPMDCSN	Does this device support media dependent Coordinated Shared Networking (CSN) functionality on one or more ports?	SRPMDMOCA:M OR: SRPMDDOT11:M	C	Yes [ ]	No [✓]
SRPMDMOCA	Does this device support media dependent MoCA functionality on one or more ports?	O:1	C.2	Yes [ ]	No [✓]
SRPMDDOT11	Does this device support media dependent IEEE 802.11 Access Point functionality on one or more ports?	O:1	C.3	Yes [ ]	No [✓]
SRPMDCSN-1	Does this device support a single Designated MSRP node (DMN)?	SRPMDCSN:M	35.1.1	Yes [ ]	
SRPMDMOCA-1	Does this device support DMN Device Attribute Information Element to L2ME message?	SRPMDMOCA:M	C.2.1.2	Yes [ ]	
SRPMDMOCA-2	Does this device support DMN selection?	SRPMDMOCA:M	C.2.1.3	Yes [ ]	
SRPMDMOCA-3	Does this device support MSRP Attribute Declaration as specified in Table C-1?	SRPMDMOCA:M	C.2.2 Table C-1	Yes [ ]	
SRPMDDOT11-1	Does this device support EDCA-AC?	SRPMDDOT11:M	Table C-5	Yes [ ]	
SRPMDDOT11-2	Is the DMN and the QAP of the IEEE 802.11 BSS co-located in the same device?	SRPMDDOT11:M	C.3.2	Yes [ ]	
SRPMDDOT11-3	Does the device support MLME primitives specified in Table C-4?	SRPMDDOT11:M	C.3.3, Table C-4	Yes [ ]	
SRPMDDOT11-4	Does the device support VLAN tag encapsulation/de-encapsulation on the 802.11 interface?	SRPMDDOT11:M	C.3.3.1	Yes [ ]	
SRPMDDOT11-5	Is the reservation process an atomic operation?	SRPMDDOT11:M	C.3.1, Figure C-11, Figure C-12, Figure C-13	Yes [ ]	

Literaturverzeichnis

[AVnu Alliance] AVnu Alliance: *Open AVB Projekt.* – URL https://github.com/AVnu/Open-AVB

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

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

[IEEE 802.1 TSN Task Group a] IEEE 802.1 TSN Task Group: *IEEE 802.1Qbu - Frame Preemption.* – URL http://www.ieee802.org/1/pages/802.1bu.html

[IEEE 802.1 TSN Task Group b] IEEE 802.1 TSN Task Group: *IEEE 802.1Qbv - Enhancements for Scheduled Traffic.* – URL http://www.ieee802.org/1/pages/802.1bv.html

[IEEE 802.1 TSN Task Group c] IEEE 802.1 TSN Task Group: *IEEE 802.1Qbv - Enhancements for Scheduled Traffic.* – URL http://www.ieee802.org/1/pages/802.1bv.html

[Imtiaz u. a. 2012] Imtiaz, Jahanzaib ; Jasperneite, Jürgen ; Weber, Karl: Approaches to reduce the latency for high priority traffic in IEEE 802.1 AVB networks. In: *9th IEEE International Workshop on Factory Communication Systems (WFCS 2012)*, 2012, S. 161–164. – ISBN 978-1-4673-0693-5

[Institute of Electrical and Electronics Engineers 2009a] Institute of Electrical and Electronics Engineers: IEEE 802.1AB - IEEE Standard for Local and metropolitan area networks - Station and Media Access Control Connectivity Discovery / IEEE. September 2009 (IEEE 802.1AB-2009). – Standard. – ISBN 978-0-7381-6038-2

[Institute of Electrical and Electronics Engineers 2009b] Institute of Electrical and Electronics Engineers: IEEE 802.1Qav - IEEE Standard for Local and metropolitan area

networks - Virtual Bridged Local Area Networks - Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams / IEEE. Dezember 2009 (IEEE 802.1Qav-2009). – Standard. – ISBN 978-0-7381-6143-3

[Institute of Electrical and Electronics Engineers 2010] Institute of Electrical and Electronics Engineers: IEEE 802.1Qat - IEEE Standard for Local and metropolitan area networks - Virtual Bridged Local Area Networks - Amendment 14: Stream Reservation Protocol (SRP) / IEEE. September 2010 (IEEE 802.1Qat-2010). – Standard. – ISBN 978-0-7381-6501-1

[Institute of Electrical and Electronics Engineers 2011a] Institute of Electrical and Electronics Engineers: IEEE 802.1AS - IEEE Standard for Local and metropolitan area networks - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks / IEEE. Februar 2011 (IEEE 802.1AS-2011). – Standard. – ISBN 978-0-7381-6536-3

[Institute of Electrical and Electronics Engineers 2011b] Institute of Electrical and Electronics Engineers: IEEE 802.1BA - IEEE Standard for Local and metropolitan area networks - Audio Video Bridging (AVB) Systems / IEEE. September 2011 (IEEE 802.1BA-2011). – Standard. – ISBN 987-0-7381-7639-8

[Institute of Electrical and Electronics Engineers 2011c] Institute of Electrical and Electronics Engineers: IEEE 802.1Q - IEEE Standard for Local and metropolitan area networks - Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks / IEEE. August 2011 (IEEE 802.1Q-2011). – Standard. – ISBN 978-0-7381-6501-1

[Lipfert 2008] Lipfert, Jan: *Technical Data Reference Guide - netX500/100.* Hilscher GmbH. Dezember 2008. – URL http://www.hilscher.com

[Meyer 2013] Meyer, Philipp: *Simulationsbasierte Analyse der Integration von TDMA basierter Kommunikation in Ethernet AVB.* Oktober 2013. – URL http://core.informatik.haw-hamburg.de/images/publications/theses/m-saitk-13.pdf

[Meyer u. a. 2013] Meyer, Philipp ; Steinbach, Till ; Korf, Franz ; Schmidt, Thomas C.: Extending IEEE 802.1 AVB with Time-triggered Scheduling: A Simulation Study of the Coexistence of Synchronous and Asynchronous Traffic. In: *2013 IEEE Vehicular Networking Conference (VNC).* Piscataway, New Jersey : IEEE Press, Dezember 2013, S. 47–54. – ISBN 978-1-4799-2686-2

[Müller 2011] Müller, Kai: *Time-Triggered Ethernet für eingebettete Systeme: Design, Umsetzung und Validierung einer echtzeitfähigen Netzwerkstack-Architektur.* August 2011. – Bachelorthesis

[Müller u. a. 2011] Müller, Kai ; Steinbach, Till ; Korf, Franz ; Schmidt, Thomas C.: A Real-time Ethernet Prototype Platform for Automotive Applications. In: *2011 IEEE International Conference on Consumer Electronics - Berlin (ICCE-Berlin).* Piscataway, New Jersey : IEEE Press, September 2011, S. 221–225. – ISBN 978-1-4577-0233-4

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

[Rumpf u. a. 2014] Rumpf, Soeren ; Steinbach, Till ; Korf, Franz ; Schmidt, Thomas C.: Software Stacks for Mixed-critical Applications: Consolidating IEEE 802.1 AVB and Time-triggered Ethernet in Next-generation Automotive Electronics. In: *2014 IEEE International Conference on Consumer Electronics - Berlin (ICCE-Berlin).* Piscataway, New Jersey : IEEE Press, 2014, S. 14–18. – ISBN 978-1-4799-6165-8

[Society of Automotive Engineers - AS-2D Time Triggered Systems and Architecture Committee 2011] Society of Automotive Engineers - AS-2D Time Triggered Systems and Architecture Committee: *Time-Triggered Ethernet AS6802.* SAE Aerospace. November 2011. – URL http://standards.sae.org/as6802/

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

[Weibel 2008] Weibel, Hans: *Audio/Video Bridging das Realtime-Ethernetprotokoll des IEEE 802.* 2008

[XMOS Ltd. 2014] XMOS Ltd.: *AVB Design Guide.* 2014. – URL http://www.xmos.com/applications/avb

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

Hamburg, 14. November 2014 Soeren Rumpf