AG VERNETZTE SYSTEME
FACHBEREICH INFORMATIK
AN DER TECHNISCHEN UNIVERSITÄT
KAISERSLAUTERN

Bachelorarbeit

Mikroprotokoll-basierte Restrukturierung, toolgestützte Dokumentation und Evaluation von MacZ

Dennis Christmann

---

17. März 2008

Mikroprotokoll-basierte Restrukturierung, toolgestützte Dokumentation und Evaluation von MacZ

Bachelorarbeit

Arbeitsgruppe Vernetzte Systeme
Fachbereich Informatik
Technische Universität Kaiserslautern

Dennis Christmann

Tag der Ausgabe : 07. Januar 2008
Tag der Abgabe : 17. März 2008

Themensteller : Prof. Dr. Reinhard Gotzhein
weiterer Prüfer : Prof. Dr. Jens Schmitt
Betreuer : Philipp Becker, Thomas Kuhn

Ich erkläre hiermit, die vorliegende Bachelorarbeit selbständig verfasst zu haben. Die verwendeten Quellen und Hilfsmittel sind im Text kenntlich gemacht und im Literaturverzeichnis vollständig aufgeführt.

Kaiserslautern, den 17. März 2008

(Dennis Christmann)

Zusammenfassung / Abstract

Diese Bachelorarbeit behandelt die Restrukturierung und Evaluation von *MacZ*, einem MAC-Layer für mobile Ad-Hoc-Netzwerke im Bereich der Ambient Intelligence. Das Ziel der Restrukturierung war die Identifikation und Spezifikation von Komponenten als Mikroprotokolle und deren toolgenerierte Dokumentation. Mikroprotokolle sind besondere Komponenten mit einer einzelnen, verteilten Funktion. Zur Spezifikation wurde die Specification and Description Language (SDL) verwendet. Mit Mikroprotokollen wird Wiederverwendbarkeit gefördert und die Qualität von Protokollspezifikationen verbessert. Sie helfen insbesondere auch bei der Strukturierung eines Systems, da sie die Einteilung in abgeschlossene Komponenten erzwingen. Teile von *MacZ* wurden bereits vor dieser Arbeit mit dem mikroprotokollbasierten Ansatz entworfen. Daher betrafen die Änderungen der Restrukturierung (hauptsächlich) den Service-Layer, ein Sub-Layer von *MacZ*, der für die wettbewerbsbasierte und wettbewerbsfreie Übertragung von Rahmen zuständig ist. Durch die Vergabe von Prioritäten bei wettbewerbsbasiertem Zugriff auf das Medium und durch reservierungsbasierte Übertragungen bei wettbewerbsfreiem Zugriff bietet *MacZ* Unterstützung für Dienstgüte. Diese Möglichkeiten sorgen für eine komplexe Spezifikation. Mikroprotokolle können dabei helfen, die Spezifikation übersichtlich und wohlstrukturiert zu halten.
Nach der Restrukturierung wurde *MacZ* evaluiert und die Ergebnisse wurden mit früheren Ergebnissen verglichen. Dadurch wurde gezeigt, dass die Funktionalität von *MacZ* nicht geändert wurde.

In this thesis, the restructuring of *MacZ*, a MAC-Layer for mobile ad-hoc networks in the domain of Ambient Intelligence, is presented. The intent of the restructuring was the identification and specification of components as micro protocols with a tool generated documentation. Micro protocols are special components with a single and distributed functionality. The Specification and Description Language (SDL) was used to specify them. With micro protocols, one can enhance reuse and improve the quality of protocol specifications. They also give implicit help with the structuring of a system, because they force the classification of self-contained components.
Parts of *MacZ* were already designed with the micro protocol-based approach before this work. Thus, the modifications of the restructuring affected (mainly) the Service-Layer, a sublayer of *MacZ*, which is responsible for the contention-based and contention-free sending of frames. With the assignment of priorities in a contention-based medium access and the reservation-based transmission in a contention-free medium access, *MacZ* provides capabilities for quality of service (QoS). These capabilities make the specification complex. Micro procotols can help to keep the specification clear and well-structed.
After the restructuring, *MacZ* was evaluated, and the results were compared to prior results. Thus, it was demonstrated that the functionality of *MacZ* remains unchanged.

Inhaltsverzeichnis

1. Einleitung

Die drahtlose Vernetzung hat einen immer größeren Einfluss auf unseren Alltag. Ob nun durch das Mobiltelefon mit Bluetooth-Technologie oder den WLAN-Access-Point in den eigenen vier Wänden: Täglich kommen wir mit drahtlosen Kommunikationstechnologien in Kontakt. Mit dem Trend zu Ambient Intelligence, die sich die Integration von Technologie in unseren Alltag und unsere Umgebung zur Aufgabe genommen hat, werden drahtlose ambiente Ad-Hoc-Netzwerke auch in Zukunft eine immer größere Bedeutung erhalten.

In vielen dieser ambienten Ad-Hoc-Netzwerke gibt es hohe Anforderungen bezüglich Garantien, die für eine Dienstgüte gegeben werden können. Hierunter fallen zum Beispiel Sprachübertragungen, welche hohe Anforderungen bezüglich der Ende-zu-Ende-Verzögerung haben. Solche Anforderungen resultieren häufig in komplexen Systemspezifikationen und großen Protokollentwürfen. Um so wichtiger ist es, bei der Spezifikation von Kommunikationsprotokollen ein strukturiertes Vorgehen zu wählen. Mit dem mikroprotokollbasierten Ansatz wird dieses Vorgehen unterstützt. Die Idee hinter Mikroprotokollen ist das Identifzieren von Komponenten mit einer einzigen – nicht mehr weiter zerteilbaren – Protokollfunktionalität [Got07]. Solche Ansätze können nicht nur Geld sparen, indem sie Wiederverwendbarkeit fördern, sondern helfen ebenso, die Qualität von Protokoll-Spezifikationen zu verbessern [FGGS04].

In der vorliegenden Arbeit wird *MacZ* [BGK07, Bec06, Kuh06], ein MAC-Layer für mobile Ad-Hoc-Netzwerke, welcher von der AG Vernetzte Systeme [AG] entwickelt wurde, mit dem mikroprotokollbasierten Ansatz restrukturiert und evaluiert. *MacZ* bietet Dienstgüteunterstützung, indem die Bevorzugung von Rahmen (und damit von speziellen Anwendungsdaten) bei wettbewerbsbasiertem Zugriff auf das Medium durch Vergabe von Prioritäten möglich ist. Neben dieser wettbewerbsbasierten Übertragung unterstützt *MacZ* ebenso eine wettbewerbsfreie Übertragung, die eine Nutzung zum reservierungsbasierten Versand durch entsprechenden Reservierungsmechanismen erlaubt.

Im Laufe dieser Arbeit wurden Mikroprotokolle in *MacZ* identifiziert und die Struktur der Spezifikation modifiziert. Dazu musste die Funktionalität von *MacZ* sinnvoll zerteilt werden, um abgeschlossene Komponenten zu erhalten, die wenige Abhängigkeiten untereinander aufweisen und eine einfache Komposition erlauben. Dabei sollte das Verhalten und die Funktionsweise der bestehenden Spezifikation zu den umgebenden Protokollschichten unverändert beibehalten werden.

Die Grobarchitektur von *MacZ* hat sich durch die Restrukturierung nicht geändert. Bereits vor dieser Arbeit war *MacZ* in zwei Sub-Layer geteilt, den Basic-Layer und den Service-Layer. Dabei blieb auch die Verantwortlichkeit der Sub-Layer unverändert. Der Service-Layer ist weiterhin für die Übertragung von Daten zuständig, d.h. er steht der Netzwerkebene als Diensterbringer zur Verfügung. Der Basic-Layer kümmert sich um die Synchronisation zwischen Knoten, die für eine darauf aufbauende

Einteilung des Mediums in Zeitschlitze notwendig ist. Erst diese zeitliche Einteilung ermöglicht sowohl wettbewerbsbasierten als auch wettbewerbsfreien Zugriff auf das Medium.

Durch die Evaluation wurde gezeigt, dass sich die Performanz von *MacZ* durch die Restrukturierung nicht verschlechtert hat. Dazu wurden frühere Simulationen wiederholt und die Ergebnisse gegenübergestellt.

Bei der Restrukturierung wurde die bisherige Absicht fortgeführt, die Spezifikation plattformunabhängig zu halten. Durch die Verwendung der „Specification and Description Language“ (SDL [ITU99]), die eine formale Syntax und Semantik hat, liegt die Spezifikation in einer plattformunabhängigen Beschreibungssprache vor. Plattformspezifische Parameter wurden in Konfigurationsparameter ausgelagert, so dass *MacZ* ohne größeren Aufwand auf andere Plattformen portiert werden kann.

Zur Dokumentation der identifizierten Mikroprotokolle wurde die Dokumentationsfunktion von *ConTraST* [FGW06], ein von der AG Vernetzte Systeme [AG] entwickeltes Werkzeug zur Erzeugung von C++-Quellcode aus einer SDL-Spezifikation, eingesetzt. *ConTraST* ermöglicht es, direkt aus einer SDL-Spezifikation, die mit Anmerkungen erweitert werden kann, eine Dokumentation als LaTeX-Datei zu erstellen, die das Verhalten der betrachteten Komponente nach außen beschreibt.

Der Rest dieser Arbeit gliedert sich in folgende Teile: In Kapitel 2 wird zunächst eine Einführung in *MacZ* gegeben. In Kapitel 3 wird allgemein auf Konzepte von Wiederverwendung bei der Entwicklung von Kommunikationsprotokollen eingegangen, welche dann in Kapitel 4 konkret bei der Restrukturierung von *MacZ* angewendet werden. Kapitel 5 wiederholt zunächst frühere Simulationen von *MacZ*, um zu zeigen, dass die Funktionalität durch die Restrukturierung nicht verändert und die Performanz in Form von erreichbaren Übertragungsraten nicht verschlechtert wurde. Anschließend wird eine neue Simulation, die das Abschneiden von *MacZ* bei dem bekannten Hidden-Station-Problem [Kar90] untersuchen soll, durchgeführt. Zum Schluss wird in Kapitel 6 eine Zusammenfassung und ein Ausblick gegeben. Im Anhang befindet sich abschließend ein Beispiel einer mit *ConTraST* generierten Mikroprotokoll-Dokumentation und eine CD, welche die SDL-Spezifikation von *MacZ* vor und nach der Restrukturierung enthält und die identifizierten Mikroprotokolle inklusive deren Dokumentation beinhaltet.

2. MacZ

Das folgende Kapitel gibt einen Überblick über die Funktionen und Bestandteile von *MacZ*. Dabei wird sowohl auf konzeptionelle Ideen als auch auf die Grobstruktur des Entwurfs eingegangen und eine Grundlage für die in Kapitel 4 vorgenommene Restrukturierung gebildet.

2.1. Einführung in MacZ

MacZ [BGK07, Bec06, Kuh06] ist ein MAC-Layer, der von der AG Vernetzte Systeme an der TU Kaiserslautern [AG] entwickelt wurde. Er ist vor allem für die Verwendung in ambienten Ad-Hoc- und Sensor-Netzwerken geeignet. Ambiente Ad-Hoc-Netzwerke verwalten sich selbstständig und werden im Bereich der Ambient Intelligence eingesetzt, welche eine immer größere Bedeutung erfährt. Mit Ambient Intelligence will man Informationstechnologien in unser tägliches Leben und unsere Umgebung integrieren, um den Lebensstandard zu erhöhen und den Alltag zu erleichtern. Die Technologien sollen uns dabei allgegenwärtig in Form von Sensoren und Kommunikationsmöglichkeiten umgeben [HK04].

In ambienten Ad-Hoc- und Sensornetzwerken ist ein einzelner Knoten häufig (zum Beispiel im Vergleich zu Laptops oder PCs) mit sehr beschränkten Ressourcen ausgestattet. Dies betrifft unter anderem die Rechenleistung und die Größe des Arbeitsspeichers. Zusätzlich besitzen die Knoten oft geringe Energieressourcen (häufig batteriebetrieben), weshalb Maßnahmen zum Sparen von Energie bei dem Entwurf beachtet werden sollten.

Als Teil von *MacZ* wurde ein Verfahren zur Tick-Synchronisation über mehrere Hops entwickelt [GK08]. Bei einer Tick-Synchronisation synchronisieren die Knoten im Gegensatz zu einer Uhren-Synchronisation nicht ihre absoluten Uhrenwerte, sondern sie synchronisieren sich lediglich relativ zu einem Referenzzeitpunkt (Referenztick). *MacZ* nimmt mit Hilfe des entwickelten Verfahrens zur Tick-Synchronisation eine zeitliche Einteilung des Mediums in Bereiche vor, denen unterschiedliche Bedeutungen zugeordnet werden. Dadurch ist es möglich, dass alle Knoten, die auf den Referenztick synchronisiert sind, in bestimmten Zeitabschnitten (z.B. zwischen 20 ms und 50 ms nach dem Referenztick) in einen idle-Modus gehen und Energie sparen können. Das von *MacZ* angewandte Verfahren hat weiterhin den Vorteil, dass es für die Synchronisationsgenauigkeit und Konvergenzzeit deterministische Garantien gibt und robust gegenüber Topologieänderungen ist [GK08].

MacZ ermöglicht wettbewerbsfreien und wettbewerbsbasierten Zugriff auf das Medium. Der wettbewerbsfreie Medienzugriff wird zum Versand von Rahmen mit vorheriger Reservierung genutzt. Bei wettbewerbsbasiertem Medienzugriff können

rahmenbezogene Prioritäten angegeben werden, die für eine Bevorzugung von Rahmen sorgen.
Durch den reservierungs- und prioritätsbasierten Versand unterstützt *MacZ* enhanced-best-effort-Garantien für Dienstgüte. Diese Art der Garantie entspricht allgemein einer deterministischen oder statistischen Garantie, falls die Kanalbedingungen (z.B. die zur Verfügung stehende Übertragungsrate) dies zulassen und die Topologie nicht variiert sowie prioritätsbasierter best-effort-Garantie sonst [WGS07]. Sie ist die bestmögliche Dienstgütegarantie, die in einem Ad-Hoc-Netzwerk gegeben werden kann [WFG+04]. Die Vergabe der Prioritäten und Reservierung ist kein Bestandteil von *MacZ* und wird durch höhere Netzwerkebenen erledigt.

Anders als bei dem IEEE-Standard *802.15.4* [IEEE03], der Grundlage von *ZigBee* [Zig] ist und ebenfalls die Anwendungsdomäne von ambienten Ad-Hoc-Netzwerken adressiert, unterscheidet man in *MacZ* nicht zwischen unterschiedlichen Arten von Endgeräten. Stattdessen läuft auf jedem Knoten die gleiche Implementierung von *MacZ*.

MacZ wurde in der „*Specification and Description Language*“ (SDL-2000) mit Hilfe der *Telelogic Tau Suite* spezifiziert [ITU99, EHS97, Tel]. SDL ist eine objektorientierte Spezifikations- und Beschreibungssprache mit formaler Syntax und Semantik. Sie ermöglicht die strukturierte, plattformunabhängige Spezifikation von komplexen Kommunikationssystemen und besitzt sowohl eine grafische (SDL/GR) als auch textuelle (SDL/PR) Repräsentation.

2.2. Medium-Slotting

Ein funktionaler Bestandteil von *MacZ* ist *Black Burst Synchronization (BBS)*, ein Protokoll zur Tick-Synchronisation über mehrere Hops [GK08]. Mit *BBS* ist es – unter der Voraussetzung, dass der Durchmesser des Netzwerkes kleiner oder gleich einem angenommenen maximalen Netzwerkdurchmesser ist – möglich, ein ganzes Netzwerk auf einen Referenz-Tick zu synchronisieren. Da die Geschwindigkeit der Uhren zweier Stationen voneinander abweichen kann (Clock Skew), muss die Synchronisation in regelmäßigen Abständen wiederholt werden.

Die Dauer der Ticksynchronisation (Konvergenzzeit) ist unabhängig von der Anzahl an Stationen. Stattdessen ist die Konvergenzzeit von dem maximalen Netzwerkdurchmesser abhängig. Der maximale Netzwerkdurchmesser wird bei *BBS* angenommen, wodurch die Konvergenzzeit deterministisch genau bestimmbar ist [GK08].

Möglich ist dies nur dadurch, dass bei *BBS* keine regulären Rahmen zwischen den Knoten ausgetauscht werden, sondern sogenannte Tick-Frames. Die Besonderheit eines Tick-Frames liegt in der Kodierung durch Black Bursts. In einem Tick-Frame wird eine logische 1 mit dem Senden eines Black Burst-Rahmens kodiert. Bei einer logischen 0 wird nichts übertragen.

Black Bursts sind spezielle Rahmen, bei denen ausschließlich die Länge des Rahmens relevante Informationen enthält. Die Länge eines Black Bursts ist in der Regel sehr klein (kleiner als reguläre Datenrahmen). Sie können kollisionsresistent übertragen werden, da ein gesendeter Black Burst auf Empfangsseite ausschließlich anhand der

Dauer der Kanalbelegung erkannt wird. Durch eine Kollision von mehreren Black Bursts geht dadurch die gesendete Information nicht verloren, solange sich die Dauer der Kanalbelegung nicht signifikant ändert, d.h. solange der gesendete Black Burst als Black Burst erkannt wird. Es muss also dafür gesorgt werden, dass mehrere Black Bursts (fast) gleichzeitig übertragen werden, damit sie auf Empfangsseite als *ein* Black Burst erkannt werden. Sie dürfen sich auf keinen Fall so ungünstig überlagern, dass die erkannte Medienbelegung der Belegung eines regulären Datenrahmens entspricht. Des Weiteren muss darauf geachtet werden, dass Black Bursts nie gleichzeitig mit normalen Übertragungen stattfinden, da sie reguläre Übertragung stören würden [Kuh06].

Wegen der kollisionsresistenten Übertragung ist keine Arbitrierungsphase vor dem Senden eines Black-Bursts notwendig, wodurch lediglich eine konstante Verzögerung durch das Umschalten des Transceivers in den Sendemodus und das Senden des Rahmens selbst entsteht. Bei dem *CC2420*-Transceiver, der für die *MicaZ*-Plattform verwendet wird [Crob], beträgt die Umschaltzeit vom Empfangs- in den Sendemodus 192 μs [Tex]. Diese Zeiten sind konstant, d.h. bei Knoten mit gleichem Transceiver gleich. Dadurch ist die Zeitspanne zwischen der Sendeabsicht und dem tatsächlichen Sendebeginn bekannt und spielt bei der Genauigkeit der Synchronisation keine Rolle.

Der einzige Ungenauigkeitsfaktor der Synchronisation kommt durch einen Jitter bei der Erkennung einer Medienbelegung zustande. Je später eine Medienbelegung erkannt wird, desto größer ist der zeitliche Abstand zu dem Sendebeginn. *BBS* geht davon aus, dass der Beginn einer Medienbelegung identisch mit dem Sendebeginn ist, d.h. je später eine Belegung erkannt wird, desto größer ist die Ungenauigkeit. Da der Jitter eine Obergrenze hat, kann *BBS* eine deterministische Genauigkeit, d.h. eine Obergrenze des Tick Offsets (Abweichung der Uhren relativ zu dem Referenztick), garantieren [GK08]. Um eine Medienbelegung zu erkennen, führt der Transceiver ein *Clear Channel Assessment (CCA)* durch. Dieses berechnet anhand der Signalstärke der letzten 8 empfangenen Symbole und einem Grenzwert, ob das Medium belegt ist [Tex, RR06]. Je schwächer ein empfangenes Signal ist, desto länger wird für die Erkennung einer Übertragung benötigt. Es kann auch passieren, dass Hintergrundrauschen fälschlicherweise als Mediumbelegung angesehen wird (*false positive*) oder eine Belegung nicht erkannt wird (*false negative*).

In *MacZ* wird *BBS* verwendet, um das Medium in Zeitabschnitte einzuteilen. Der Zeitabschnitt zwischen zwei Referenzticks wird *Macro-Slot* genannt. Er hat zu Beginn eine Synchronisations- bzw. Resynchronisationsphase mit *BBS* und unterteilt sich anschließend in kleine Zeitschlitze, welche *Micro-Slots* heißen und relativ zu dem Referenztick durchnummeriert werden. Aufgrund der Synchronisation haben alle Stationen die gleiche Sicht auf den aktuellen Micro-Slot. Dadurch können Micro-Slots zu verschiedenen Slot-Regionen zusammengefasst werden, denen eine unterschiedliche Funktion zugeordnet werden kann. Zum Beispiel können die Micro-Slots 3,4,5 und 6 eine Slot-Region bilden, in welcher sich alle Knoten passiv verhalten und Energieeinsparungen möglich sind. Die Einteilung ist zur Zeit statisch und für alle Macro-Slots identisch. In kommenden Arbeiten soll die Einteilung dynamisch veränderbar werden, um die Anzahl und Größe der unterschiedlichen Slot-Regionen an aktuelle Ansprüche anpassen zu können. So wäre es zum Beispiel möglich, weitere

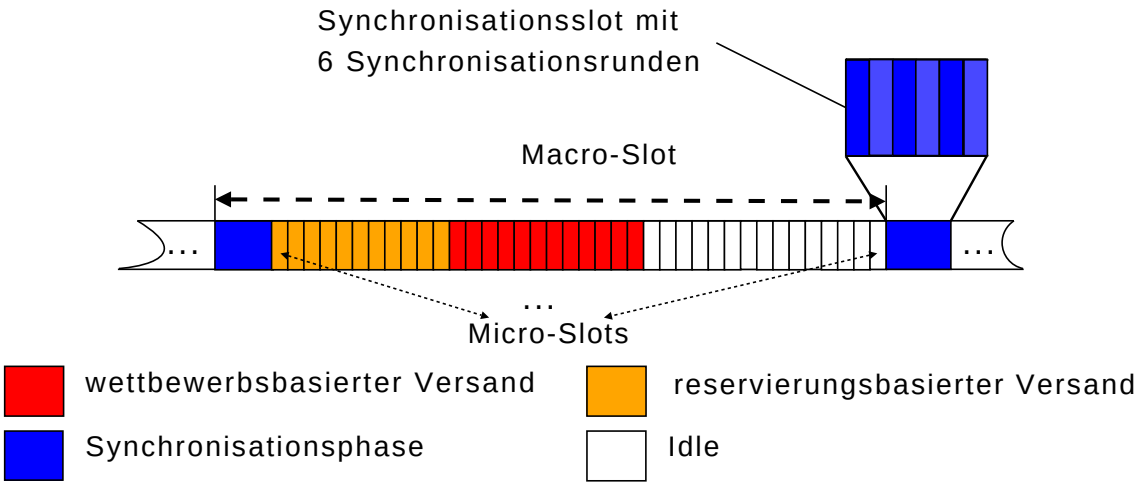


Abbildung 2.1.: Medium-Slotting: Ein Macro-Slot unterteilt in mehrere Micro-Slots

reservierbare Micro-Slots hinzuzufügen, falls die vorhandenen reservierbaren Slots ausgebucht wären.

In Abbildung 2.1 ist ein einfacher Aufbau eines Macro-Slots gegeben. Das Beispiel besitzt pro Macro-Slot jeweils eine Slot-Region für wettbewerbsbasierten und wettbewerbsfreien Versand und eine Region zum Energiesparen.

BBS führt die Synchronisation in mehreren Runden durch. Eine Station verhält sich nach dem Einschalten für die Dauer von mindestens einem Macro-Slot passiv, um zu prüfen, ob bereits ein synchronisiertes Netzwerk besteht. Falls dies nicht der Fall ist, bildet die Station ein neues synchronisiertes Netzwerk und beginnt mit dem Synchronisationsalgorithmus, indem sie einen Tick-Frame sendet, in welchem die aktuelle Rundennummer, d.h. eins, kodiert ist. Für die Station ist die Synchronisation ab diesem Zeitpunkt bereits abgeschlossen. Andere Stationen, die den Tick-Frame der ersten Station empfangen haben, kennen durch den Empfangszeitpunkt und die Rundennummer – unter Vernachlässigung der Ausbreitungsverzögerung – den Sendezeitpunkt der ersten Station: Sie sind mit dem ersten Knoten synchronisiert. Um eine netzwerkweite Synchronisation zu erreichen, wird der Tick-Frame nach Warten einer konstanten Rundendauer mit inkrementierter Rundennummer weitergeleitet. Hierdurch wird ein weiterer Hop synchronisiert und die neu synchronisierten Stationen verfahren wie die eben betrachteten Stationen. Es wird somit pro Runde ein weiterer Hop synchronisiert und die Dauer der Synchronisation ist abhängig von der Rundendauer und der Anzahl an Runden, welche der maximalen Hop-Anzahl entspricht. Da die Werte bekannt sind, garantiert das Verfahren eine deterministische Konvergenz.

Aufgrund von unterschiedlichen Taktraten zwischen den Uhren der Stationen (Clock Skew) kann der Tick Offset während eines Macro-Slots immer größer werden, wobei weiterhin – bei der Annahme eines maximalen Clock Skews – eine deterministische Genauigkeit gilt. Aus diesem Grund findet zu Beginn jedes Macro-Slots eine Resynchronisation statt, welche von der gleichen Station begonnen wird, die auch im vorherigen Macro-Slot angefangen hatte. Sollte diese Station ausfallen oder aufgrund von Mobilität außerhalb des Sendebereichs gekommen sein, merken die anderen Sta-

tionen das Ausbleiben der erwarteten Synchronisationsphase und gehen in einen Modus mit geringerer Genauigkeit [GK08]. Obwohl die garantierte Genauigkeit abnimmt, bricht das Netz nicht zusammen. *BBS* ist dadurch sehr robust gegenüber Knotenausfällen.

Die Nachrichtenkomplexität ist wie die Konvergenzzeit nicht von der Anzahl an Stationen abhängig, sondern vom maximalen Netzwerkdurchmesser und beträgt $O(d)$. Um Missverständnisse zu vermeiden, sei darauf hingewiesen, dass in der Regel deutlich mehr Tick-Frames gesendet werden, welche durch die gleichzeitige Übertragung die Komplexität nicht erhöhen.

2.3. Versandarten und Rahmenformate

Auf Grundlage der Formatierung des Mediums ermöglicht *MacZ* sowohl einen wettbewerbsbasierten Versand von Rahmen als auch eine wettbewerbsfreie Übertragung in reservierbaren Zeitabschnitten. Die Konzeption hinter den Versandarten und den möglichen Rahmenformaten wurde in [Bec06] veröffentlicht und wird im folgenden zusammengefasst wiedergegeben.

MacZ stellt als Diensterbringer die Möglichkeit zur rahmenbezogenen Vergabe von Prioritäten in wettbewerbsbasierten Slot-Regionen und zur wettbewerbsfreien Rahmenübertragung in reservierten Zeitabschnitten bereit. Die Vergabe von Prioritäten bzw. Reservierungen sind jedoch kein Teil von *MacZ* und müssen von der darüberliegenden Netzwerkebene erledigt werden.

Bei wettbewerbsbasiertem Zugriff auf das Medium kann vor der Übertragung der Daten eine Reservierungsphase mit dem RTS/CTS-Verfahren stattfinden. Das Verfahren ist auch bei WLAN (802.11) DCF [IEEE99] zu finden und verringert Hidden Station Probleme [Kar90].

2.3.1. Wettbewerbsbasierte Übertragung

Ähnlich wie bei der „Distributed Coordination Function“ (*DCF*) von WLAN [IEEE99] können sich Stationen an einem Wettbewerb um die Sendeerlaubnis auf dem Medium in Single-Hop-Reichweite beteiligen. Das folgende Unterkapitel soll die möglichen Rahmenformate in wettbewerbsbasierten Slot-Regionen vorstellen und die zeitliche Abfolge des Wettbewerbs darstellen.

2.3.1.1. Arbeitsweise

In wettbewerbsbasierten Slot-Regionen findet vor der Übertragung eines Rahmens eine Arbitrierungsphase mit „carrier-sense multiple access with collision avoidance“ (CSMA-CA) statt. Ziel des Verfahren ist es, Kollisionen durch gleichzeitige Sendevorgänge zu vermeiden, da diese bei regulären Rahmen zu einem Verlust der übertragenen Information führen. Es wird eine physikalische Abtastung des Kanals vorgenommen, um Übertragungen zu erkennen, die exakt zu diesem Zeitpunkt stattfinden. Diese physikalische Abtastung reicht alleine nicht aus, um Kollisionen zu

verhindern, da nicht die Interferenzen auf Senderseite von Bedeutung sind, sondern auf Empfängerseite [BDSZ94]. Deshalb gibt es zusätzlich eine virtuelle Abtastung, die in Form eines „Network Allocation Vectors“ (*NAV*) die Mediumbelegung durch Auswertung von erhaltenen RTS- bzw. CTS-Rahmen angibt. Da die Verwendung von RTS/CTS-Rahmen nicht zwingend ist, kann es allerdings auch sein, dass der *NAV* nie gesetzt ist und ausschließlich die physikalische Abtastung zur Vermeidung von Kollisionen benutzt werden kann.

Ein Arbitrierungsvorgang lässt sich in folgende Zeitabschnitte unterteilen:

1. **Warten auf freies Medium**
   Das Kommunikationsmedium wird erst als nicht belegt angenommen, wenn es für eine gewisse Dauer ungenutzt ist. Die Dauer muss größer als der zeitliche Abstand in einer laufenden Sequenz (wie z.B. RTS-CTS-DATA) sein, da eine Sequenz niemals durch eine neue Übertragung unterbrochen werden darf. Die Zeitspannen, die eine Station warten muss, bevor das Medium als frei angenommen wird, werden Interframe-Spacings (*IFS*) genannt und man unterscheidet sie nach ihrer Länge: Kurze *IFS* werden in einer Rahmensequenz (z.B. zwischen RTS und CTS) gewartet, lange vor einer Neuübertragung. Diese sind unabhängig von der Priorität eines Rahmens und müssen bei jedem Rahmen gewartet werden. Die Realisierung der unterschiedlichen Längen erfolgt über eine unterschiedliche Anzahl von Pulse-Slots. Pulse-Slots sind Zeitabschnitte mit einem fest konfigurierten Wert.

2. **Warten eines zufälligen Backoff-Intervalls**
   Um die Wahrscheinlichkeit zu verringern, dass zwei Sender gleichzeitig senden, muss jede Station vor Sendebeginn eine zufällige Zeitspanne warten. Die Zeitspanne wird durch eine zufällige Anzahl an Pulse-Slots, die aus einem Contention-Window zwischen einer Untergrenze CW_{min} und einer Obergrenze CW_{max} bestimmt wird, berechnet. D.h. ein Contention-Window hat in diesem Zusammenhang die gleiche Bedeutung wie in WLAN [IEEE99]. Wegen dieser durch Zufall bestimmten Zeit spricht man von nicht-persistentem CSMA [Tan96].
   CW_{min} und CW_{max} werden paketbezogen durch die Netzwerkebene vergeben und erlauben eine rahmenbezogene Vergabe von Prioritäten, die sich auf die Bearbeitungsreihenfolge in einem Knoten und auf die Sendereihenfolge zwischen mehreren Knoten auswirkt. In einem Knoten hat der Rahmen mit kleinstem CW_{max} die höchste Priorität und wird als erstes gesendet. Die Obergrenze CW_{max} hat aber zusammen mit der Untergrenze CW_{min} auch Einfluss auf die Reihenfolge der Rahmen-Übertragungen mehrerer Knoten in Single-Hop Reichweite. Denn ein Backoff-Intervall, das vor dem Beginn einer neuen Übertragung gewartet werden muss, wird aus einer Anzahl Pulse-Slots zwischen CW_{min} und CW_{max} bestimmt. Wenn man die Intervalle $[CW_{min}, CW_{max}]$ von verschiedenen Rahmen disjunkt voneinander wählt, kann man auf diese Weise strikte Prioritäten vergeben, die für eine Bevorzugung einzelner Rahmen sorgen. In diesem Punkt unterscheidet sich die Berechnung des Backoffs von dem WLAN Standard 802.11 [IEEE99] bzw. auch von der Dienstgüte-Erweiterung

802.11e [IEEE05], da dort immer aus dem Intervall [0, [CW_{min}, CW_{max}]] gewählt wird. Die Vergabe von strikten Prioritäten ist hierdurch nicht möglich. Falls während dem Warten eine Übertragung durch eine andere Station begonnen wird, wird das Backoff-Intervall zurück gesetzt und der Arbitrierungsvorgang neu gestartet. Ohne das Zurücksetzen des Backoff-Intervalls könnten keine strikten Prioritäten vergeben werden, da bei der Arbitrierung nicht nur die Priorität, sondern auch die vergangenen Arbitrierungsvorgänge eine Rolle spielen würden.

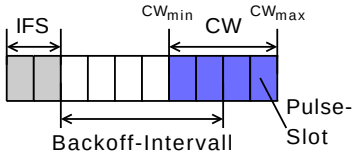


Abbildung 2.2.: Arbitrierungsvorgang

Abbildung 2.2 zeigt beispielhaft einen Arbitrierungsvorgang mit einem aus 2 Pulse-Slots bestehendem *IFS*, einem Contention-Window zwischen 4 und 8 Pulse-Slots und einem zufällig bestimmtem Backoff-Intervall von 6. Falls das Medium zwischenzeitlich nicht durch eine andere Station belegt wird, beginnt der Sendevorgang damit nach 8 Pulse-Slots.

Obwohl die wettbewerbsbasierte Slot-Region aus einer Anzahl von Micro-Slots besteht, orientiert sich der Sendebeginn eines Rahmens nicht an einer Micro-Slot Grenze.

2.3.1.2. Verwendbare Rahmenformate

Bei der wettbewerbsbasierten Übertragung mit Prioritäten sind alle vier – von *MacZ* unterstützten – Rahmenformate möglich:

- **Data**
  Rahmen mit diesem Format bieten die Möglichkeit, die Adressen des Senders und Empfängers im Rahmen zu spezifizieren. Dadurch ist durch Angabe einer konkreten Adresse eine eins-zu-eins Kommunikation und durch Angabe der Broadcast-Adresse (0xFFFFFFFF) eine eins-zu-alle Kommunikation in 1-Hop-Reichweite möglich. 1 Byte des Headers ist zusätzlich für Flags reserviert, mit denen in Zukunft Empfangsbestätigungen (ACKs) für den Rahmen angefordert werden können.

- **RTS**
  Ein **RTS**-Rahmen (Request To Send) leitet das RTS/CTS-Verfahren ein, wodurch Hidden-Station Probleme verringert werden [Kar90]. Er kündigt einen Datenrahmen an und enthält die Länge des anstehenden Rahmens und die Sender- und Empfängeradresse. Alle Stationen in Reichweite (ausgeschlossen der Empfänger) setzen nach Erhalt des **RTS** ihren *NAV* mit Hilfe der angekündigten Rahmenlänge und stören dadurch die folgende Kommunikation nicht.

- **CTS**
  CTS (Clear To Send) ist die Antwort des Empfängers eines **RTS**, falls der Kanal bei dem Empfänger weder physikalisch noch virtuell belegt ist. Der Kanal ist virtuell belegt, wenn der *NAV* gesetzt ist. Stationen, die das **CTS** empfangen, setzen ihren *NAV* auf Basis der im **CTS** enthaltenen angekündigten Rahmenlänge. Dadurch stören auch Stationen (hidden stations), die das **RTS** nicht gehört haben, die folgende Übertragung des Datenrahmens nicht.

- **SIMPLE**
  SIMPLE-Rahmen bieten die Möglichkeit der Datenübertragung mit minimalem Overhead. Neben dem Typ des Rahmens ist kein weiterer Platz im Header für Adressen oder ähnliches vorgesehen, d.h. die Kommunikation entspricht auf MAC-Ebene einer Broadcast-Kommunikation. Der Rahmentyp ist zum Beispiel sinnvoll, falls die Empfänger-/Senderadresse in dem Netzwerkpaket bereits gesetzt ist.

Der konkrete Aufbau der Rahmen wurde bei der Restrukturierung beibehalten und ist in [Bec06] beschrieben.

2.3.2. Wettbewerbsfreie Übertragung

In wettbewerbsfreien Regionen bietet *MacZ* die Möglichkeit, Rahmen mit Reservierungen zu senden. Das Reservierungsprotokoll selbst ist kein Teil von *MacZ* und wird in aktuellen Arbeiten entwickelt.

2.3.2.1. Arbeitsweise

Die Reservierung erfolgt auf Micro-Slot-Ebene, d.h. die Übertragung beginnt immer zu Beginn eines Micro-Slots. Wegen möglichen Tick-Offsets kann die Wahrnehmung des Anfangs eines Micro-Slots zwischen unterschiedlichen Stationen zwar leicht abweichen, jedoch gibt es weiterhin eine deterministische Obergrenze für diesen Tick-Offset. Falls die Zeit eines Micro-Slots nicht ausreicht, um einen Rahmen zu senden, kann die Übertragung über mehrere (reservierte) Micro-Slots hinweg erfolgen.

Bei der wettbewerbsfreien Übertragung wird davon ausgegangen, dass vor dem Sendebeginn eine Reservierung stattgefunden hat, d.h. es wird keine Mediumarbitrierung durchgeführt. Falls parallel eine Übertragung stattfände, käme es zu einer Kollision, deren Vermeidung jedoch nicht die Aufgabe des MAC-Layers ist. Stattdessen muss die Netzwerkebene für eine disjunkte Reservierung der Micro-Slots sorgen.

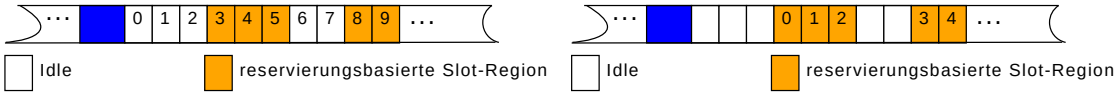
2.3.2.2. Änderung der Micro-Slot Zählung

Im Zuge dieser Arbeit wurde eine Änderung an der Zählweise der Micro-Slots durchgeführt. In [Bec06] fand für Reservierungen eine absolute Zählung der Micro-Slots pro Macro-Slot statt. Sowohl die Netzwerkebene als auch die Komponente im MAC-Layer, die für den Versand von Rahmen mit Reservierung verantwortlich ist, arbeiteten auf Basis der Micro-Slot-Nummer relativ zu dem Macro-Slot. Dadurch war es

möglich, dass Micro-Slots reserviert werden konnten, die nicht zu einer reservierungsbasierten Slot-Region gehörten. Im ungünstigsten Fall konnte es passieren, dass in wettbewerbsbasierten Regionen bestehende Übertragungen gestört wurden, weil ein Rahmen „reservierungsbasiert“ ohne vorheriges CCA übertragen wird. Obwohl dies ein Fehler des Reservierungsprotokolls wäre, wurde *MacZ* abgeändert, um Reservierungen von Micro-Slots außerhalb der wettbewerbsfreien Slot-Region zu verhindern.

Nach Änderung werden Micro-Slots nun relativ zu den wettbewerbsbasierten Bereichen innerhalb eines Macro-Slots gezählt. Dies wird durch Spezifikation einer Kontrollkomponente (siehe Kapitel 4.3.1.3) möglich, die ein Mapping zwischen der absoluten und relativen Micro-Slot-Zählung vornimmt. Außerdem sorgt die Kontroll-Instanz dafür, dass die Komponenten, die für den reservierungsbasierten und wettbewerbsbasierten Versand verantwortlich sind, nie gleichzeitig aktiv sein können. In Abbildung 2.3 sind die alten und neuen Zählweisen dargestellt.

Um Verwirrungen zu vermeiden, werden die Zählweisen im Folgenden begrifflich unterschieden. Unter der *Micro-Slot-Zählung* versteht man die absolute Zählweise, die mit Beginn jedes Macro-Slots bei 0 startet und bei jedem Micro-Slot inkrementiert wird. Unter der *Reservierungs-Slot-Zählung* ist die relative Zählung zu verstehen, die ebenfalls zu Beginn eines Macro-Slots zurückgesetzt wird, aber sich nur in wettbewerbsfreien Slot-Regionen erhöht.



(a) (b)

Abbildung 2.3.: (a) zeigt beispielhaft die Zählweise vor der Restrukturierung und (b) nach den Änderungen. Bei (a) findet die Zählweise unabhängig von den Slot-Regionen statt.

Zunächst hat die veränderte Zählweise die Vorteile, dass die Netzwerkebene in großen Teilen von dem Medium-Slotting abstrahieren kann und nur an einer einzigen Stelle im System die Slot-Regionen verwaltet werden müssen. Dies vereinfacht die zukünftig geplante dynamische Konfiguration von Macro-Slots in Slot-Regionen. Aber sie hat auch Nachteile: Den höheren Schichten ist nicht bekannt, aus wievielen Micro-Slots einzelne wettbewerbsfreie Regionen bestehen oder wie groß der Abstand zwischen diesen ist. In der Netzwerkschicht ist es dadurch kaum möglich, die Verzögerung bis zu einem bestimmten Slot in einer wettbewerbsfreien Slot-Region vorherzusagen. Ebenso kann man auf der Netzwerkebene nicht abschätzen, ob ein Rahmen komplett in einer wettbewerbsfreien Region übertragen werden kann, da die Übertragung über mehrere Reservierungs-Slots andauern könnte. Wenn z.B. der Beginn einer Rahmenübertragung – die mehrere Reservierungs-Slots benötigt – in dem Reservierungs-Slot x stattfindet, muss der nächste Reservierungs-Slot $x+1$ nicht unbedingt direkt nach Reservierungs-Slot x folgen.
Zur Lösung dieses Problems muss durch zukünftige Arbeiten die Netzwerkebene über die aktuelle Slot-Konfiguration informiert werden. Dies ist sowieso spätestens nach Einführung von dynamischen Slot-Konfigurationen notwendig (siehe Kapitel 2.2).

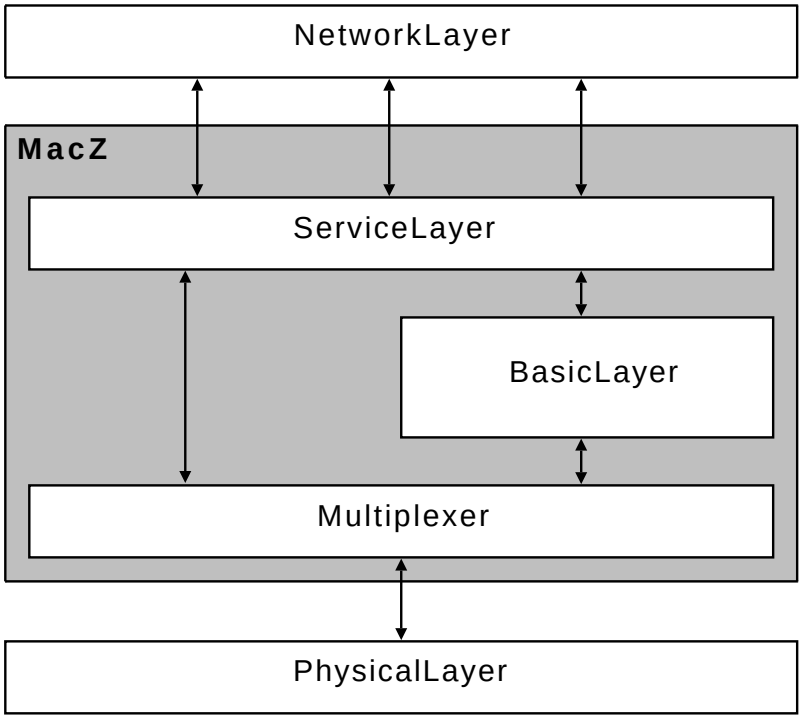


Abbildung 2.4.: Architektur von MacZ zwischen Netzwerk- und Hardware-Ebene

2.3.2.3. Verwendbare Rahmenformate

Die möglichen Rahmenformate in wettbewerbsfreien Zeitabschnitten beschränken sich auf **DATA** und **SIMPLE**. Ein RTS/CTS-Verfahren ist weder notwendig noch möglich, da der Versand von Daten ausschließlich mit vorheriger Reservierung geschieht.

2.4. Architektur

MacZ befindet sich als MAC-Layer zwischen Netzwerk- und Hardware-Ebene. Die Grobstruktur besteht aus zwei funktional getrennten Komponenten. Durch einen Multiplexer werden die Signale, die von beiden Komponenten kommen, zusammengeführt und zur physikalischen Schicht weitergegeben. In der anderen Richtung werden Signale von der Hardwareebene getrennt und zur zuständigen Komponente geleitet. In Abbildung 2.4 ist die Grobstruktur mit den genannten Ebenen dargestellt.

2.4.1. Basic-Layer

Der Basic-Layer bildet die Grundlage für das Medium-Slotting, wie es in Kapitel 2.2 vorgestellt wurde. Er ist für die Tick-Synchronisation zwischen den Stationen zuständig und meldet dem Service-Layer nach erfolgreicher Synchronisation die Position in einem Macro-Slot in Form der aktuellen Micro-Slot-Nummer. Da die Synchronisation mit Hilfe von Black Bursts geschieht und Black Bursts ausschließlich über die Dauer der Kanalbelegung erkannt werden, muss der Basic-Layer mit dem

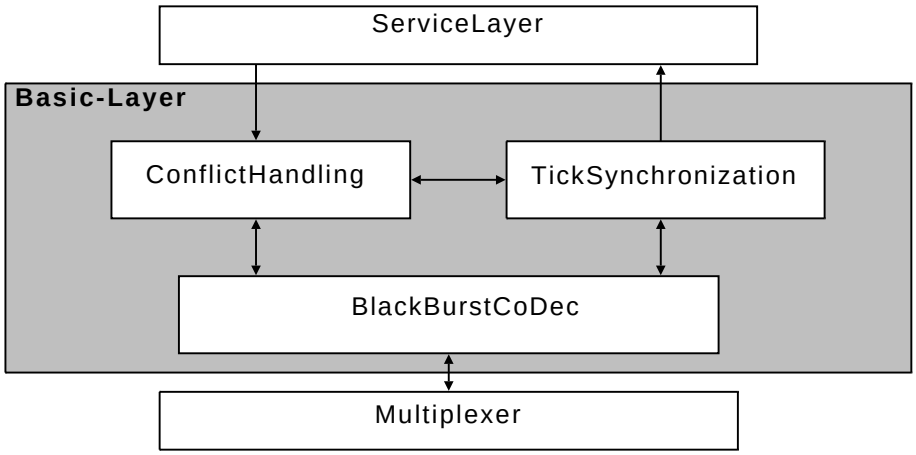


Abbildung 2.5.: Die Unterkomponenten im Basic-Layer

CCA-Mechanismus des Transceivers Black Bursts von normalen Datenrahmen unterscheiden.

In Abbildung 2.5 sind die Subkomponenten des Basic-Layer abgebildet. Es gibt insgesamt drei Unterkomponenten:

- **TickSynchronization**
  Das Medium-Slotting, wie es in Kapitel 2.2 beschrieben wurde, wird von dieser Subkomponente erledigt. Die Komponente ist für die Einteilung in Macro- und Micro-Slots verantwortlich und führt die dafür notwendige Tick-Synchronisation in den Synchronisations-Slots durch. Dabei ist hier sowohl die Funktionalität der Master-Stationen enthalten, die mit der Synchronisation beginnt, falls kein bestehendes Netz entdeckt wird, als auch die Funktionalität der Slave-Stationen, die sich mit einem bereits bestehenden Netz synchronisieren.

- **BlackBurstCoDec**
  Die Aufgabe dieser Komponente besteht darin, einen Black Burst über die Dauer einer Kanalbelegung zu dekodieren und einen Black Burst als physikalischen Rahmen, dessen einzig relevantes Kriterium die Länge ist, kodiert zu übertragen. Dabei geschieht die Erzeugung der physikalischen Repräsentation eines Black Bursts einmalig, da bei jeder Black Burst-Übertragung der gleiche physikalische Rahmen gesendet wird.

- **ConflictHandling**
  In Ad-Hoc-Netzen kann es häufig zu Topologieänderungen kommen, durch welche es zu Partitionsbildungen oder Vermischung zweier Netze kommen kann. Da bei einer Mischung von zwei Netzwerken die Netzwerke nur in sich synchronisiert sind, ist eine Neusynchronisation notwendig. Aufgabe dieser Komponente ist es, einen solchen Konflikt in der Synchronisation zu finden, zu signalisieren und eine Neusynchronisation auszulösen.

In Kapitel 4.2 wird im Zusammenhang mit der mikroprotokollbasierten Restrukturierung auf die einzelnen Komponenten etwas genauer eingegangen.

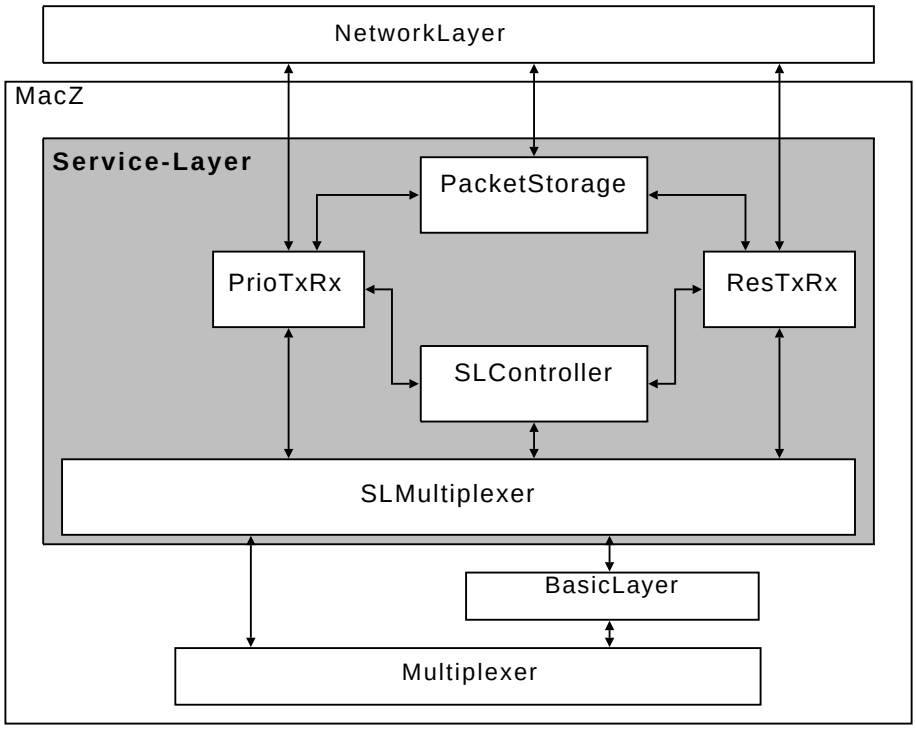


Abbildung 2.6.: Unterarchitektur des Service-Layers

2.4.2. Service-Layer

Die Schnittstelle zwischen *MacZ* und der Netzwerkebene wird durch den Service-Layer gebildet. Der Service-Layer ermöglicht die in Kapitel 2.3 vorgestellten Versandarten mit den jeweils möglichen Rahmenformaten. Er ist dabei sowohl für den beschriebenen Arbitrierungsvorgang in wettbewerbsbasierten Slot-Regionen als auch für den reservierungsbasierten Versand in vorgeschriebenen Micro-Slots verantwortlich. Die Komponente unterteilt sich in fünf Unterkomponenten, die entsprechend Abbildung 2.6 in Zusammenhang stehen:

- **PacketStorage**
  Mit der Zwischenspeicher-Komponente ist es möglich, dass die Netzwerkebene mehrere Pakete für den Versand in Auftrag stellen kann. Dies ist insbesondere sinnvoll, weil dadurch Pakete für den reservierungsbasierten Versand unabhängig von der aktuellen Slot-Region in Auftrag gegeben werden können.

- **SLController**
  Die Kontroll-Komponente wurde durch die Restrukturierung neu hinzugefügt und ist für die Gruppierung von Micro-Slots in die unterschiedlichen Slot-Regionen verantwortlich. Die Komponente übernimmt auch das Mapping zwischen der absoluten und relativen Slot-Zählung (siehe Kapitel 2.3.2.2).

- **PrioTxRx**
  In wettbewerbsbasierten Slot-Regionen wird diese Komponente von der Kontroll-Komponente aktiviert und hat das Senden (mit vorheriger Arbitrierung) und Empfangen von Rahmen zur Aufgabe. Dabei wird der Netz-

werkebene der erfolgreiche Versand eines Pakets bestätigt und empfangene Rahmen werden als Pakete nach oben gegeben. Die Komponente fragt nach Aktivierung durch den **SLController** und nach erfolgreichem Versand die **PacketStorage**-Komponente nach neuen Aufträgen und erhält das Paket mit der höchsten Priorität zurück.

- **ResTxRx**
  Diese Komponente ist für das Senden und Empfangen in wettbewerbsfreien Slot-Regionen zuständig. Sie bekommt von **PacketStorage** Informationen über neue Aufträge und beginnt mit dem Sendevorgang, wenn ein Auftrag für den aktuellen Reservierungs-Slot vorliegt. Der **SLController** aktiviert die Komponente zu Beginn einer wettbewerbsfreien Slot-Region (bzw. deaktiviert sie am Ende) und teilt ihr während einer wettbewerbsfreien Slot-Region mit, welcher Reservierungs-Slot zur Zeit an der Reihe ist.

- **SLMultiplexer**
  Der Multiplexer im Service-Layer leitet Signale zwischen Unterkomponenten des Service-Layers und dem Basic-Layer bzw. Multiplexer von *MacZ* weiter. Er sorgt dafür, dass Signale nur zu Komponenten geleitet werden, welche die entsprechenden Signale verarbeiten und zur Zeit aktiv sind.

3. Wiederverwendung

Bei der Entwicklung von Kommunikationsprotokollen lassen sich mehrere Ansätze für Wiederverwendbarkeit unterscheiden. Zwei dieser Ansätze werden im Folgenden näher betrachtet: Mikroprotokolle und SDL-Entwurfsmuster (SDL-Patterns). Sie unterscheiden sich vor allem in der Stärke der Abstraktion von einer Protokollfunktionalität.

In diesem Kapitel wird zunächst auf die Vorteile und Ziele von Wiederverwendung eingegangen und anschließend eine Definition für Mikroprotokolle und SDL-Patterns gegeben.

3.1. Ziele der Wiederverwendung

In der Softwareentwicklung wird Wiederverwendung bereits intensiv betrieben. Ein Beispiel für wiederverwendbare Softwarekomponenten sind Klassenbibliotheken für moderne objektorientierte Programmiersprachen. Es gibt aber auch generischere Entwurfs-Lösungen mit musterbasierten Ansätzen (Design-Patterns), wie sie zum Beispiel in [GHJV95] beschrieben sind. Diese Ansätze lassen sich auch bei der Entwicklung von Kommunikationsprotokollen wiederfinden.

Wiederverwendung hat zum Ziel, die Qualität eines Kommunikationsprotokolls zu verbessern [FGGS04]. Durch sie ist es möglich, Wissen aus früheren Entwicklungen weiterzugeben und bereits gewonnene Erfahrungen zu nutzen. Fehler, die bei der Erstentwicklung einer wiederverwendbaren Komponente gemacht und verbessert wurden, können bei Wiederverwendung vermieden werden. Zusätzlich zu der Qualitätssteigerung wird die Produktivität erhöht [FGGS04], weil Lösungen für Probleme nicht mehrfach entwickelt werden müssen und eine Wiederverwendung der Designlösung auch über Projektgrenzen hinaus möglich ist. Voraussetzung ist, dass wiederverwendbare Komponenten identifiziert und in einem Repository zur Verfügung gestellt werden. Je generischer ein wiederverwendbares Element gehalten ist, desto häufiger finden sich Anwendungsmöglichkeiten. Allerdings ist in der Regel auch der Zusatzaufwand, der zur Verwendung des Elementes in einem Projekt nötig ist, größer.

3.2. SDL-Patterns

SDL-Patterns beschreiben generische Lösungen für wiederkehrende Design-Probleme mit SDL als Entwurfssprache [Got07]. Die Lösungen sind sehr allgemein gehalten, weswegen sie an konkrete Anwendungen angepasst werden müssen. Neben SDL-Fragmenten werden zur Beschreibung von SDL-Patterns weitere Informationen, wie

zum Beispiel Sequenzdiagramme, benötigt, die dabei helfen, Anwendungsfälle von SDL-Patterns bei der Entwicklung von Kommunikationsprotokollen zu finden. Anhand der Dokumentation der SDL-Patterns kann der Entwickler auf Grundlage des Analysemodells Einsatzmöglichkeiten für SDL-Patterns erkennen und die SDL-Patterns nach einer Anpassung an den konkreten Kontext nutzen.

3.3. Mikroprotokolle

Bevor der Begriff Mikroprotokoll definiert werden kann, muss zunächst eine Definition für SDL-Komponenten gegeben werden. SDL-Komponenten sind Designlösungen, welche direkt verwendet werden können [Got07]. Sie können – im Gegensatz zu den bereits vorgestellten SDL-Patterns – mit deutlich weniger Aufwand verwendet werden, da sie spezifischer sind und weniger Anpassungen nötig sind.

Unter einem Mikroprotokoll versteht man eine spezielle SDL-Komponente eines Kommunikationsprotokolls, die eine *einzelne* und *verteilte* Protokollfunktionalität bereitstellt, und mit weiteren Kommunikationskomponenten zusammenarbeitet [FGGS04]. Eine Protokollfunktionalität beschreibt dabei nur eine Operation, d.h. sie lässt sich nicht weiter in Unterkomponenten zerlegen [Got07]. Beispiele sind Flußkontrolle oder das kodierte Übertragen von Daten mit Black Bursts.

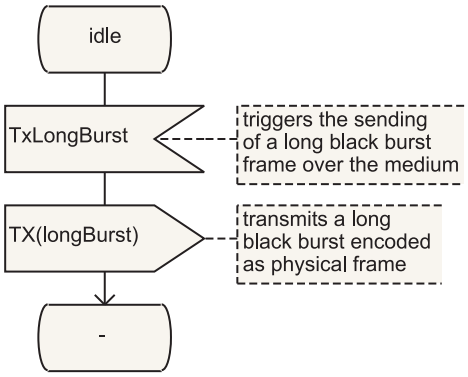
Ein Mikroprotokoll ist als verteilte SDL-Komponente in sich abgeschlossen und kann direkt verwendet werden. Bei der Verwendung ist lediglich die Einbettung der Protokollfunktionalität in den Kontext des Kommunikationssystems notwendig. Die Einbettung wird mit Hilfe von so genanntem „glue code“ vorgenommen [FGGS04]. Hierzu gehören zum Beispiel ein- und ausgehende SDL-Kanäle mit zugehörigen Signalen.

Die Komposition mehrerer Mikroprotokolle wird als Makroprotokoll bezeichnet [Got07]. Ein Makroprotokoll besteht also aus mehreren verteilten Protokollfunktionalitäten.

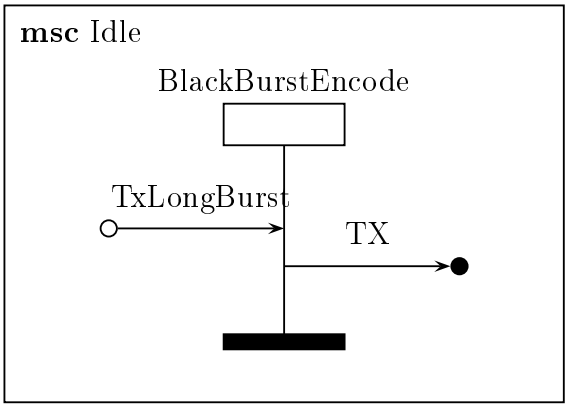
Zur Spezifikation von Mikroprotokollen bietet SDL mehrere Möglichkeiten. Neben SDL-Blocktypen, SDL-Prozesstypen und SDL-Servicetypen können Mikroprotokolle ebenso mit SDL-Prozeduren spezifiziert werden [FGGS04]. Allerdings konnten bei der in Kapitel 4 vorgestellten Restrukturierung von *MacZ* keine SDL-Servicetypen verwendet werden, da Servicetypen von dem verwendeten Code-Generator Cmicro, der Teil der Telelogic Tau Suite ist [Tel], nicht unterstützt werden. Prozess- bzw. Blocktypen können an jeder Stelle in einem SDL-System instanziiert werden, an dem auch ein Prozess bzw. Block stehen kann. Der große Vorteil gegenüber einfachen Blöcken und Prozessen liegt bei Typen darin, dass sie in SDL-Systemen mehrfach an unterschiedlichen Stellen instanziiert werden können. Dies erhöht einerseits die Lesbarkeit, da direkt sichtbar ist, welcher Typ hinter der Instanz steckt, und andererseits ist eine einfache (projektübergreifende) Verwendung des gleichen Mikroprotokolls mit exakt derselben Spezifikation sehr schnell möglich. Einfache Blöcke oder Prozesse müssten evtl. kopiert werden, um an unterschiedlichen Stellen verwendet zu werden. Dadurch wäre der Aufwand größer, um z.B. Fehler in der Spezifikation zu beseitigen oder allgemeine Änderungen zu Wartungszwecken durchzuführen.

Mikroprotokolle können in SDL-Packages gekapselt werden [FGGS04]. Dadurch kann man in größeren SDL-Komponenten, in denen ein Mikroprotokoll verwenden werden

Annotationen in SDL



resultierende LaTeX -Ausgabe



Description:

The signal `TxLongBurst` triggers the sending of a long black burst frame over the medium. The signal `TX` transmits a long black burst encoded as physical frame.

Abbildung 3.1.: Dokumentationsbeispiel mit *ConTraST*

soll, das als Typ definierte Mikroprotokoll durch Benutzung des zugehörigen SDL-Packages instanziieren. Mikroprotokolle sind verteilte Komponenten [Got07] und werden auf mehreren Netzwerkknoten instanziiert. Es müssen nicht zwangsläufig auf jedem Knoten die gleichen Typen instanziiert oder Prozeduren verwendet werden. Es sind auch asymmetrische Mikroprotokolle möglich, wie zum Beispiel bei dem Mikroprotokoll zur Übertragung von Black Bursts. Bei diesem Mikroprotokoll ist es möglich, dass ein Knoten Bursts sendet und ein anderer Knoten Bursts dekodieren und empfangen kann, d.h. in diesem Fall wäre die Kommunikation nur unidirektional.

Durch ihre Abgeschlossenheit können Mikroprotokolle in der Regel einfacher getestet werden, weil sie ein klar definiertes Verhalten aufweisen. Da sie nur eine einzige Protokollfunktionalität beinhalten, ist die Komplexität der Komponente meist geringer und ein einfaches Debugging ist möglich.

3.4. Toolgestützte Dokumentation

Zur Dokumentation der identifizierten Mikroprotokolle wurde die Dokumentationsfunktion von *ConTraST* verwendet [FGW06, Fli07]. *ConTraST* wurde ursprünglich als Transpiler entwickelt, der eine SDL-Spezifikation, die in der textuellen Repräsentation SDL-PR vorliegt, in C++-Quellcode übersetzt.

Die Dokumentationsfunktion von *ConTraST* erwartet, dass ein Mikroprotokoll in einem SDL-Package gekapselt ist und dass die SDL-Spezifikation um spezielle Annotationen erweitert wurde. Sie extrahiert Annotationen an SDL-Elementen (z.B. an Signal-Inputs) und generiert eine LaTeX -Datei, bei welcher die Annotationen kontextbezogen, d.h. bezogen auf den internen Zustand, ausgegeben werden.

In Abbildung 3.1 ist der Auszug aus einem minimalen Beispiel des Dokumentationsgenerators dargestellt. Das Beispiel umfasst eine Transition mit nur einem Signal-Output. Der linke Teil der Abbildung zeigt den Ausschnitt aus der SDL-Spezifikation. Rechts ist die zugehörige Ausgabe durch die Dokumentationsfunktion von *ConTraST* abgebildet.

Die Dokumentation bezieht sich ausschließlich auf die Interaktion der betrachteten Komponente mit der Umgebung. Dies betrifft vor allem Signal-Inputs und Signal-Outputs. Das interne Verhalten wird nur beachtet, wenn ein- oder ausgehende Signale davon abhängen. Von der konkreten Spezifikation (z.B. interne Berechnungen) wird abstrahiert, da sich die Dokumentation auf Mikroprotokolle bezieht und nicht auf die SDL-Spezifikationen. Für Mikroprotokolle, bei denen bei Verwendung keine Änderungen der Spezifikation nötig sind, ist ausschließlich das Verhalten nach außen, d.h. die Funktionalität, von Belang. Dies ist der große Vorteil von Mikroprotokollen, weil der Entwurf von Kommunikationsprotokollen durch Komposition von Mikroprotokollen auf einer höheren Abstraktionsebene möglich ist.

Die Dokumentationsfunktion von *ConTraST* ist ein junges Projekt und befindet sich noch in der Erprobungsphase. Es werden zur Zeit nicht von allen SDL-Konstrukten fehlerfreie Dokumentationen erzeugt. Insbesondere mehrfache Verzweigungen in einer Transition sind noch fehleranfällig. Bis zum Abschluss dieser Bachelorarbeit konnten mit Hilfe der neu dokumentierten Komponenten einige Fehler gefunden werden, von denen viele bereits beseitigt wurden. Jedoch sind weitere Bemühungen erforderlich, um die restlichen Fehler auszumerzen. In Anhang A ist ein komplettes Beispiel einer Dokumentation von dem Mikroprotokoll zur wettbewerbsfreien Übertragung zu finden, das in Kapitel 4.3.1.1 vorgestellt wird. Bei diesem Beispiel wurden nach der toolgestützten Erzeugung der Dokumentation (wenige) manuelle Änderungen durchgeführt, um kleinere Fehler zu beheben. Auf der CD in Anhang B ist dieses Beispiel zusätzlich in der unmodifizierten Version, sowie die Dokumentation der weiteren spezifizierten Mikroprotokolle verzeichnet.

4. Restrukturierung

Die Restrukturierung von *MacZ* hatte vor allem das Identifizieren und Gruppieren von Mikroprotokollen innerhalb der Protokollspezifikation zum Ziel, wodurch *MacZ* übersichtlicher und besser wartbar wird. Durch die Definition der Mikroprotokolle als SDL-Typen in eigenen SDL-Packages können sie nun an anderen Stellen wiederverwendet und einzelne Komponenten unabhängig von anderen Komponenten ausgetauscht werden. Somit kann *MacZ* sehr schnell an unterschiedliche Kontexte angepasst werden. Zum Beispiel kann im Service-Layer die in Kapitel 2.4.2 vorgestellte PrioTxRx-Komponente ersetzt werden, falls ein anderer Arbitrierungsmechanismus gewünscht ist.

Im folgenden Kapitel wird zunächst auf eine allgemeine Möglichkeit eingegangen, die Übersichtlichkeit zu verbessern. Die weiteren Unterkapitel beschreiben konkrete Veränderungen im Basic- und Service-Layer. Dabei liegt der Schwerpunkt der Ausführungen auf Komponenten, bei denen größere Änderungen durchgeführt wurden.

4.1. Signallisten

Während der Restrukturierung wurde auf eine gute Lesbarkeit der SDL-Diagramme geachtet. Deshalb wurden bei SDL-Kanälen und Signal-Routen konsequent Signal-Listen verwendet, falls mehr als zwei Signale mit dem Kanal bzw. der Route assoziiert wurden.

Signallisten tragen zur Lesbarkeit von SDL-Systemen bei, weil SDL-Diagramme weniger überfüllt aussehen. Des Weiteren haben sie den Vorteil, dass einfach und schnell Änderungen möglich sind. Ändert sich der Name eines Signals, müssen neben der Signaldefinition, lediglich die Definition der Signal-Liste und die SDL-Inputs/-Outputs in den Prozessen geändert werden. Änderungen an den Kanälen und Routen sind nicht nötig. Gerade bei langen Signal-Pfaden reduziert dies einigen Aufwand.

4.2. Basic-Layer

Alle drei in Kapitel 2.4.1 beschriebenen Komponenten sind entsprechend der Definition aus Kapitel 3.3 Mikroprotokolle. Zu großen Teilen waren diese bereits vor dieser Arbeit identifiziert und in SDL-Packages aufgeteilt, so dass neben der Dokumentation der bereits identifizierten Mikroprotokolle im Basic-Layer nur wenig verändert wurde. Im Folgenden wird deshalb ausschließlich auf das BlackBurst-Mikroprotokoll eingegangen.

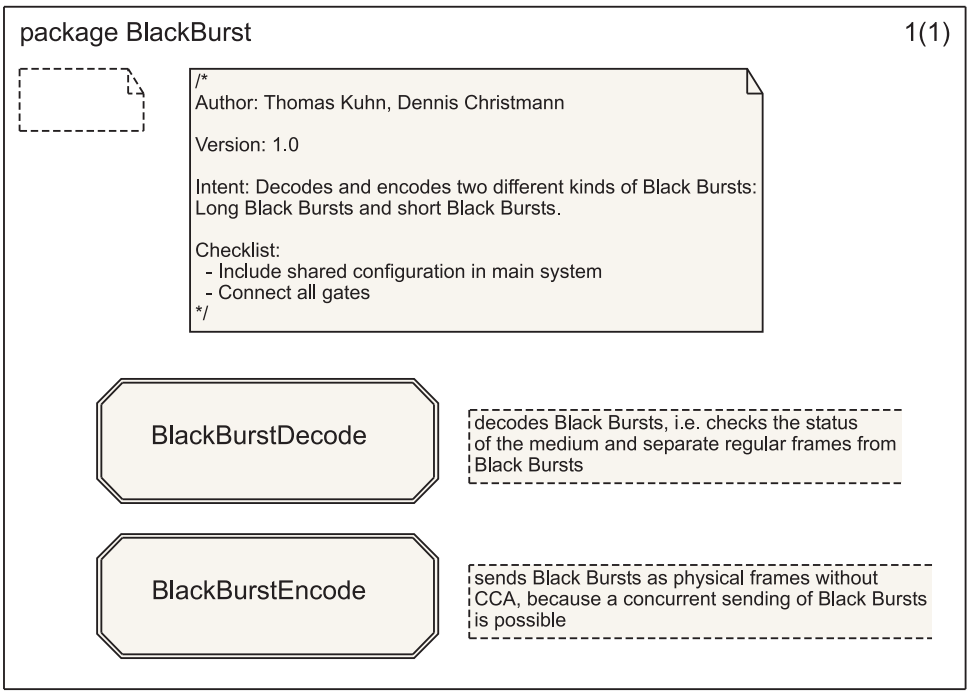


Abbildung 4.1.: Mikroprotokoll BlackBurst zum (De-)Kodieren von Black Bursts.

Die Aufgabe des Mikroprotokolls BlackBurst ist das Senden eines speziellen Rahmens, der als Black Burst definiert wird, sowie das Erkennen eines gesendeten Black Bursts (siehe Kapitel 2.4.1). Während das Senden eines Black Bursts lediglich aus der Übertragung eines kurzen Rahmens besteht, ist das Erkennen deutlich schwieriger. Bei der Erkennung müssen reguläre Datenrahmen von Black Burst-Rahmen unterschieden werden. Diese Unterscheidung erfolgt über die Belegungsdauer des Mediums. Dazu muss vorausgesetzt werden, dass reguläre Datenrahmen (deutlich) länger als Black Bursts sind. Die Länge muss sich mindestens so stark unterscheiden, dass es auch zu minimal zeitversetzten Überlagerungen mehrerer Black Bursts kommen kann, die nicht fälschlicherweise als Datenrahmen interpretiert werden.

Vor dieser Arbeit wurden Black Bursts in zwei getrennten Komponenten erzeugt und erkannt. Dies hatte den Nachteil, dass in zwei Systemteilen Änderungen notwendig waren, wenn die Länge der Black Bursts verändert wurde.

In dieser Arbeit wurden die Komponenten zusammengefasst und bilden nun ein asymmetrisches Mikroprotokoll. Das Mikroprotokoll ist asymmetrisch, da es sich aus zwei Prozess-Typen zusammensetzt, bei denen einer für das Senden, d.h. Kodieren von Black Bursts als physikalische Rahmen, verantwortlich ist und der andere für das Erkennen, d.h. Dekodieren von physikalischen Rahmen als Black Bursts. Dadurch kann ein Knoten den Kodier-Anteil des Mikroprotokolls instanziieren und ein anderer Knoten den Dekodier-Anteil. In diesem Fall ist nur eine unidirektionale Übertragung mit Black Bursts möglich.

Abbildung 4.1 zeigt das SDL-Package BlackBurst, welches das BlackBurst-Mikroprotokoll zusammen mit den beiden definierten Prozess-Typen kapselt.

Das Zusammenspiel zwischen den Instanzen der beiden Prozess-Typen ist in der an einem Message Sequence Chart (*MSC*) angelehnten Abbildung 4.2 dargestellt.

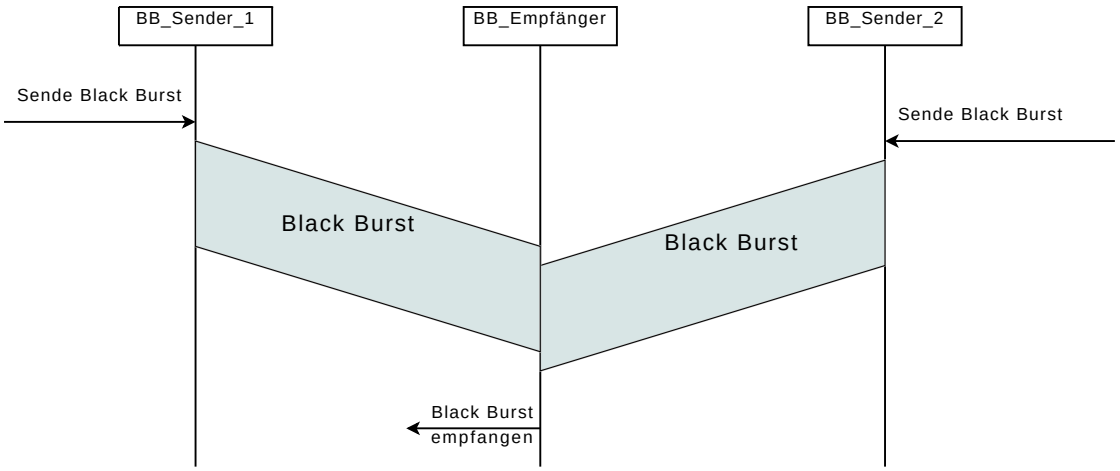


Abbildung 4.2.: Zusammenhang zwischen den beiden Prozess-Typen. Die beiden Sender instanziieren den Prozess-Typ BlackBurstDecode und der Empfänger BlackBurstEncode.

Die beiden Sender instanziieren jeweils den BlackBurstEncode-Prozess-Typ und der Empfänger den Prozess-Typ BlackBurstDecode. Das Szenario zeigt, dass die beiden Sender den Auftrag bekommen, einen Black Burst zu senden. Dadurch soll nochmal die kollisionsresistente Übertragung verdeutlicht werden, indem mehrere Sender parallel einen Black Burst übertragen und der Empfänger diesen dennoch dekodieren kann. Nach dem Erkennen des Black Bursts meldet der Empfänger den Black Burst seinem Dienstnutzer.

4.3. Service-Layer

Der Service-Layer, wie er in Kapitel 2.4.2 vorgestellt wurde, besteht aus fünf Komponenten. Von diesen fünf Komponenten realisieren vier verteilte Protokollfunktionalitäten und erfüllen die Eigenschaften eines Mikroprotokolls.

4.3.1. Mikroprotokolle

Im folgenden werden die vier identifizierten Mikroprotokolle vorgestellt.

4.3.1.1. Wettbewerbsfreier Versand mit Reservierungen

Das Mikroprotokoll zum wettbewerbsfreien Versand wurde in dem SDL-Package ResTxRx spezifiziert (siehe Abbildung 4.3). Es ist symmetrisch und beinhaltet einen Prozesstyp. Die Aufgabe des Mikroprotokolls besteht aus der wettbewerbsfreien Übertragung von Rahmen. Diese kann für einen reservierungsbasierten Versand benutzt werden. Das Mikroprotokoll setzt die Einteilung des Mediums in durchnummerierte Slots voraus, wie es bei *MacZ* vom Basic-Layer erledigt wird. Rahmen werden auf Basis einer angegebenen Slot-Nummer gesendet, d.h. dem Mikroprotokoll

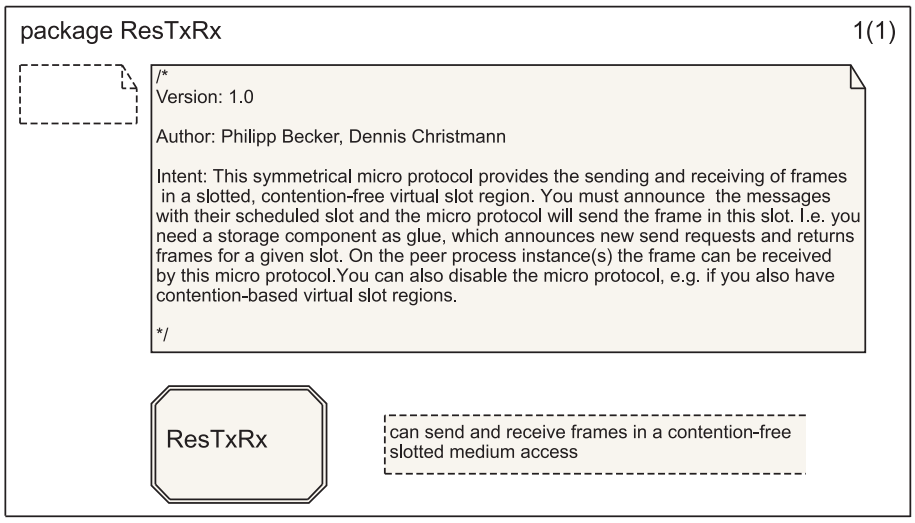


Abbildung 4.3.: Komponente zur Übertragung in wettbewerbsfreien Slot-Regionen

muss die aktuelle Slot-Nummer mitgeteilt werden und Rahmen, die wettbewerbsfrei übertragen werden, müssen mit der zugehörigen Slot-Nummer angekündigt werden. In *MacZ* werden diese Aufgaben von der Kontroll- und Storage-Komponente (siehe Kapitel 4.3.1.3 und 4.3.2) übernommen. Die Kontroll-Komponente erledigt dabei zusätzlich ein Mapping zwischen der absoluten Micro-Slot-Zählweise und der Reservierungs-Slot-Zählweise.

Bei der Restrukturierung wurde die Komponente um die Möglichkeit zum Empfang von Rahmen erweitert. In [Bec06] wurde diese Aufgabe von einem Teil der wettbewerbsbasierten Übertragungskomponente erledigt, was zur Folge hatte, dass die wettbewerbsfreie Übertragungskomponente alleine (ohne die wettbewerbsbasierte Übertragungskomponente) nicht verwendbar war.

Diese Erweiterung hat allerdings den Nachteil, dass in *MacZ* an zwei Stellen empfangene Rahmen als Pakete an die Netzwerkebene gegeben werden. Damit keine Daten doppelt zu den höheren Netzwerkschichten gelangen, wurde der Multiplexer des Service-Layers erweitert, so dass empfangene Rahmen entweder an die Komponente zum wettbewerbsfreien Versand geleitet werden oder zur Komponente zum wettbewerbsbasierten Versand. In Kapitel 4.3.1.4 wird näher darauf eingegangen.

Damit das gleiche Verhalten nicht mehrfach spezifiziert wird, wurden die Aktionen, die bei dem Empfang eines Rahmens von beiden Übertragungskomponenten durchgeführt werden, in eigenen Prozeduren spezifiziert und in einem gesonderten SDL-Package gekapselt. Die Komponente zum wettbewerbsbasierten Versand bedient sich dieser Prozeduren ebenso, wie die betrachtete Komponente zum wettbewerbsfreien Versand. Die Prozeduren beschreiben das Auslesen und Schreiben des MAC-Headers von/in einen Rahmen und wie Rahmen, die nicht an den eigenen Knoten adressiert sind, gefiltert und nicht an die Netzwerkschicht weitergeleitet werden.

In Abbildung 4.4 ist die Interaktion zwischen zwei Knoten dargestellt, die den ResTxRx-Prozesstyp instanziiert haben. Beide Instanzen erhalten Informationen über die aktuelle Reservierungs-Slot-Nummer. Der Knoten $ResTxRx\_1$ auf der linken Seite erhält den Auftrag einen Rahmen in Reservierungs-Slot x zu übertragen und

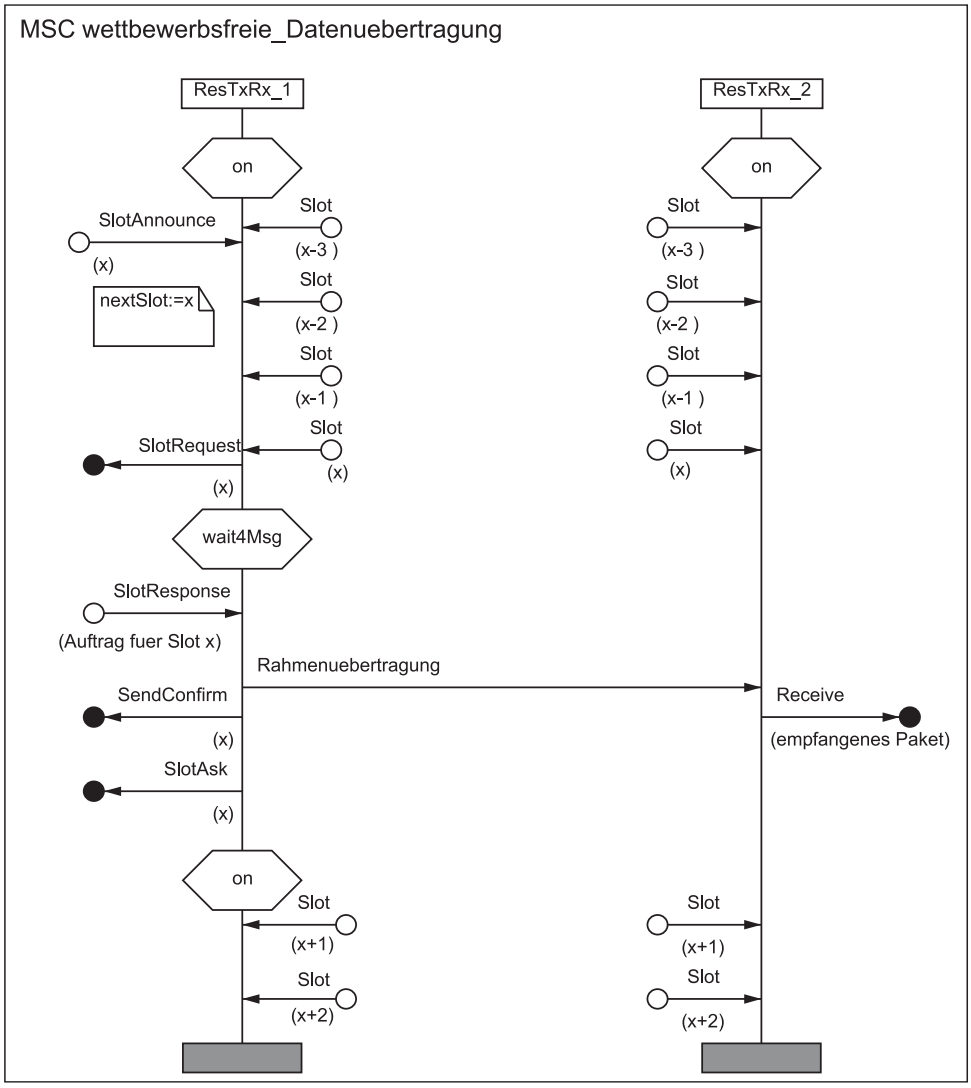


Abbildung 4.4.: Das MSC zeigt die wettbewerbsfreie Rahmenübertragung zwischen zwei ResTxRx-Instanzen.

vermerkt sich diesen Auftrag. Bei Eintreten von Slot x fordert er den Auftrag aus dem Zwischenspeicher an und überträgt ihn zu dem anderen Knoten $ResTxRx\_2$. Knoten $ResTxRx\_1$ bestätigt der Netzwerkebene den Versand und erfragt die Reservierungs-Slot-Nummer des nächsten Auftrages von der Zwischenspeicherkomponente. Der andere Knoten $ResTxRx\_2$ empfängt den Rahmen und leitet ihn als Paket zur Netzwerkschicht weiter.

4.3.1.2. Wettbewerbsbasierter Versand mit Prioritäten

Das symmetrische Mikroprotokoll zum wettbewerbsbasierten Versand ist in dem SDL-Package PrioTxRx gekapselt, das in Abbildung 4.5 dargestellt ist. Seine Aufgabe besteht aus der Übertragung von Rahmen bei wettbewerbsbasiertem Zugriff auf das Medium. D.h. diese Komponente realisiert die Arbitrierungsphase, die Kollisionen bei parallelen Sendewünschen unterschiedlicher Stationen vermeiden soll.

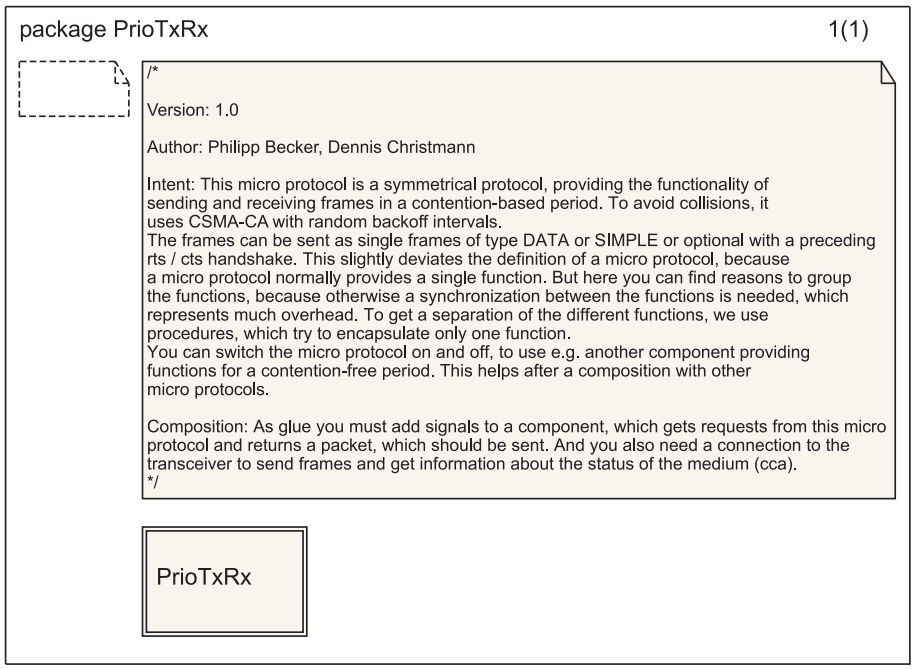


Abbildung 4.5.: Mikroprotokoll PrioTxRx zur wettbewerbsbasierten Übertragung

Im engeren Sinne erfüllt diese Komponente nicht die Definition eines Mikroprotokolls, da sie mehrere Protokollfunktionen (Übertragung alleine mit *CSMA/CA*-Mechanismus und Übertragung mit zusätzlichem RTS/CTS-Mechanismus) realisiert. Allerdings gibt es Gründe, die Komponente weiterhin als „*ein* Mikroprotokoll mit der Option einer vorangehenden RTS/CTS-Sequenz“ zu bezeichnen. Folgende Begründungen sprechen gegen eine Trennung der Komponente in mehrere Mikroprotokolle:

- **Gemeinsame Funktionen**
  Die Übertragung eines RTS-Rahmens oder unabhängigen Datenrahmens stellt den Beginn eines Sendevorgangs dar. Vor einer Neuübertragung muss jedoch eine Arbitrierungsphase stattfinden. D.h. bei getrennter Spezifikation der Mikroprotokolle müsste die Arbitrierungsphase sowohl in der Komponente realisiert werden, die ein RTS sendet, als auch in der Komponente, die einfache Daten ohne RTS sendet. Die Auslagerung in eine gemeinsam genutzte (oder kopierte) Komponente wäre zwar möglich, um doppelten Entwicklungsaufwand zu sparen, allerdings stellt bei genauerer Betrachtung der unabhängige Versand von Datenrahmen eine Untermenge der benötigten Aktivitäten für den Versand von Daten mit vorherigem RTS/CTS dar. Diese Eigenschaft kann man nutzen, um das komponierte System klein zu halten, was letztendlich auch den beschränkten Ressourcen der Hardwareplattformen zu Gute kommt.

- **Hohe Abhängigkeiten**
  Das Eintreffen eines RTS-Rahmens soll unbeteiligte Stationen daran hindern, das Medium zu belegen. Ebensowenig darf aber auch der Empfänger des RTS-Rahmens nach dem Senden eines CTS-Rahmens das Medium belegen, bis die angekündigten Daten eingetroffen sind (oder ein Timeout aufgetreten ist). D.h.

einfache Datenrahmen müssen zurückgehalten werden, falls eine andere Station mit einem RTS-Rahmen einen Datenrahmen angekündigt hat. Das erfordert jedoch, dass die Komponente, die einen einfachen Datenrahmen versenden möchte, den internen Zustand der Komponente kennt, die das RTS empfangen und mit dem CTS geantwortet hat. Anders ausgedrückt: Das Verhalten in einer Komponente würde von einer zweiten Komponente und deren Zustand abhängen. Eine strikte Trennung der Komponenten in zwei Mikroprotokolle ist – zumindest mit den später beschriebenen Einschränkungen durch den Code-Generator hinsichtlich der SDL-Services – nicht zweckmäßig.

Wenn man also zwei getrennte Mikroprotokolle entwickelt, benötigt man Funktionalitäten an zwei Stellen und erhält bei der Komposition sehr viel „glue code“ zur Synchronisation der Komponenten. Die Komplexität würde (unnötig) groß werden. Aus diesem Grund wurde keine *unabhängige* Trennung vorgenommen, aber es wurde die Tatsache genutzt, dass RTS/CTS-Reservierungen eine Erweiterung des einfachen Versendens eines Datenrahmens darstellen.

Als einziges vorgestelltes Mikroprotokoll wurde PrioTxRx in einem SDL-Block-Typ spezifiziert (siehe Abbildung 4.6 (b)). In dem Block-Typ existieren drei Prozesse, auf welche die Komplexität der Funktionalität verteilt ist.

Im Vergleich zu [Bec06] wurde die Komponente grundlegend verändert. In [Bec06] bestand sie aus insgesamt neun SDL-Prozessen (siehe Abbildung 4.6 (a)). Diese Anzahl wurde auf drei reduziert. Dieser Schritt erhöht zwar leicht die innere Komplexität der einzelnen Prozesse, aber die Gesamtkomplexität konnte stark verringert werden. Bei der alten Spezifikation war das Problem, dass sehr viele Signale zwischen den Prozessen innerhalb der Komponente ausgetauscht wurden, wenn zum Beispiel am Ende einer wettbewerbsbasierten Slot-Region ein Sendevorgang, der sich noch in der Arbitrierungsphase befand, abgebrochen werden musste. Der Abbruch eines Sendeversuches kann notwendig sein, um zu verhindern, dass eine Übertragung über die Grenze der wettbewerbsbasierten Slot-Region hinweg andauert. Da die Signalwege durch die Restrukturierung verkürzt wurden, ist zu erwarten, dass ein Performanzgewinn auf realen Plattformen erzielt wird.

Abbildung 4.6 stellt die alte Spezifikation der Spezifikation nach der Restrukturierung gegenüber. Durch unterschiedliche Farben soll gezeigt werden, wie die Funktionalität der ursprünglichen Version in der neuen Spezifikation zusammengefasst wurde. Man kann erkennen, dass durch die Restrukturierung sechs Prozesse zu einem (PrioTxRx) zusammengefasst wurden.

Zur Verwendung des Mikroprotokolls ist das Instanziieren eines SDL-Blocktyps notwendig. Von den inneren Prozessen kann dabei abstrahiert werden.

Im Folgenden werden zunächst die einzelnen Prozesse des SDL-Blocktyps vorgestellt:

FrameTx ist für die Arbitrierung verantwortlich. Der Prozess erhält von dem Prozess NAV Informationen über den Belegungszustand des Mediums (Signal VCCA). Er bekommt von dem Prozess PrioTxRx ein Signal PrioTx, welches Teil der Signalliste PrioTxRx2FrameTx ist (vgl. Abbildung 4.6). Das Signal hat als Parameter eine neu eingeführte SDL-Datenstruktur SingleFrameStruct mit folgenden Inhalten:

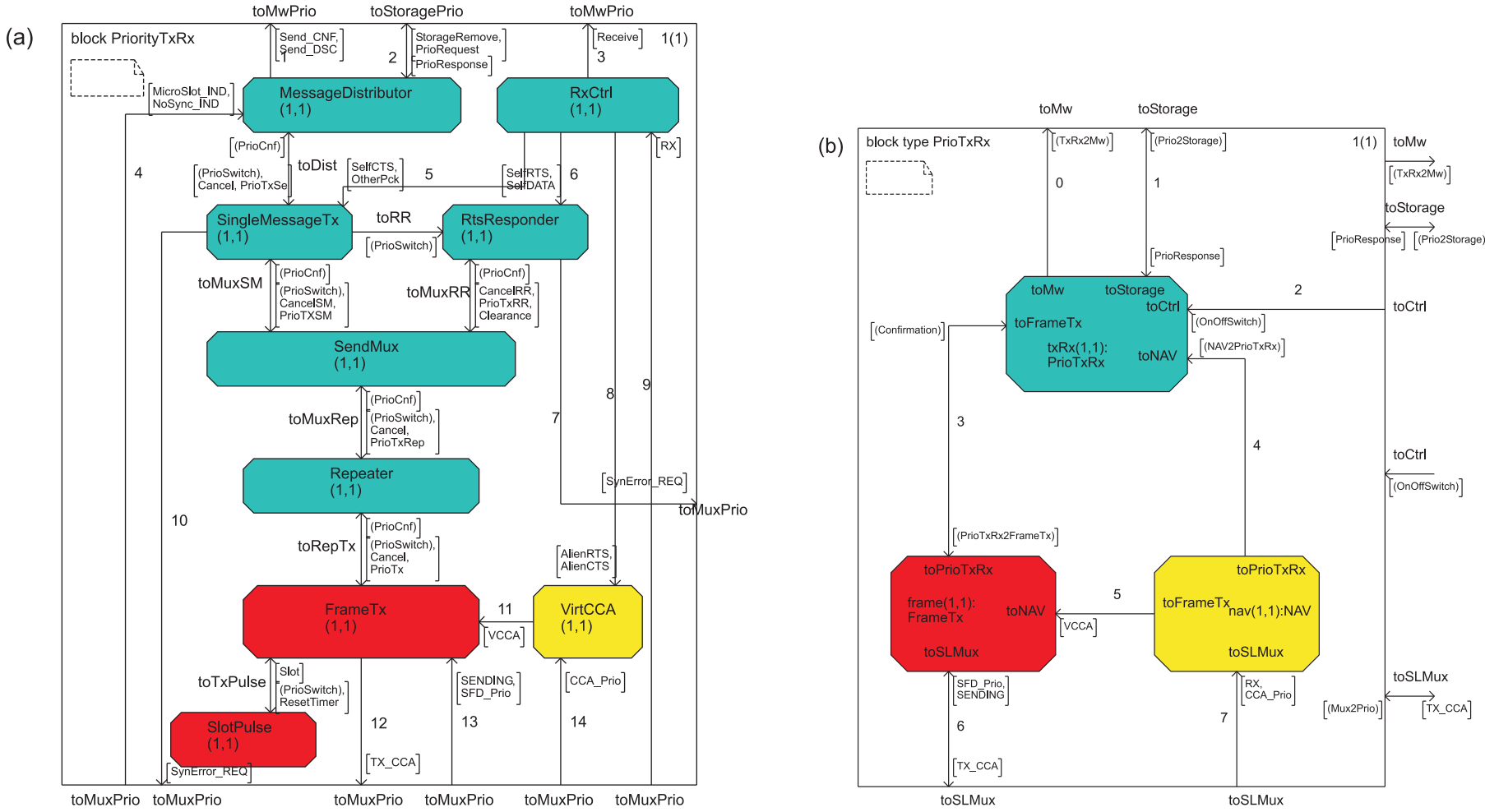


Abbildung 4.6.: SDL-Spezifikation der Komponente zur Übertragung bei wettbewerbsbasiertem Zugriff auf das Medium. (a) zeigt die Spezifikation aus [Bec06] vor der Restrukturierung und (b) die Spezifikation nach der Restrukturierung

- **data:** Rahmen, der übertragen werden soll (Octet_StringFixed)
- **cwPrefix:** Anzahl an Pulse-Slots, die als Interframe-Spacing gewartet werden soll (Octet)
- **cwMin:** Untergrenze des Contention-Windows (Octet)
- **cwMax:** Obergrenze des Contention-Windows (Octet)
- **ccaAbort:** Angabe, ob bei Belegung des Mediums gewartet werden soll, bis Medium frei ist oder ob ein Abbruch erfolgen soll (Boolean)

SDL-Strukturen bieten allgemein eine gute Möglichkeit, Parameterlisten von Signalen (zumindest optisch) klein zu halten. Sie vereinfachen aber auch Änderungen, wenn sich zum Beispiel der Datentyp eines Elements der SDL-Struktur ändert, da keine Änderungen an den SDL-Inputs notwendig sind, sondern nur an der Definition der Struktur.

Alle Rahmen passieren unabhängig von ihrem Typ den FrameTx-Prozess. Dabei unterscheiden sie sich in den Werten der Parameter. Zum Beispiel werden CTS-Rahmen mit sehr kleinem **cwPrefix** und einem Backoff-Intervall von 0 (d.h. **cwStart** = **cwStop** = 0) gesendet und müssen abgebrochen werden, falls das Medium belegt ist (d.h. **ccaAbort** = true).

NAV überwacht den physikalischen und virtuellen Status des Mediums und teilt FrameTx Änderungen des Status mit (Signal VCCA). Die physikalische Überwachung geschieht mit dem *CCA* des Transceiver, der anhand der Signalenergie prüft, ob das Medium aktuell belegt ist (Signal CCA_Prio). Der virtuelle Status leitet sich zusätzlich aus empfangenen RTS- und CTS-Rahmen ab, denn diese enthalten die Länge des angekündigten Datenrahmens. Dadurch kann berechnet werden, wie lange das Medium belegt ist, und auch Stationen, die nur das CTS gehört haben (*Hidden Stations*), schweigen. Dazu müssen empfangene RTS/CTS-Rahmen (Signal RX) von NAV ausgewertet werden: Anhand der RTS/CTS-Rahmen, die nicht an den eigenen Knoten adressiert sind, berechnet NAV die Dauer der virtuellen Belegung. Alle anderen Rahmen werden zu PrioTx weitergeleitet. Darunter fallen auch RTS/CTS-Rahmen, die an den eigenen Knoten adressiert sind.

PrioTxRx verwaltet aktuelle Übertragungen und Übertragungssequenzen. In diesem Prozess ist unter anderem die Logik enthalten, die zur Bearbeitung von RTS/CTS-Sequenzen notwendig ist, wie zum Beispiel das Warten auf ein CTS nach Senden eines RTS. Insbesondere löst er auch Konflikte, die zum Beispiel auftreten, wenn ein Datenrahmen gerade in der Arbitrierungsphase ist, d.h. bereits zu dem FrameTx-Prozess gegeben wurde, und ein RTS eintrifft. Da der Prozess etwas umfangreicher ist und sich in weitere Teile (speziell Prozeduren) untergliedert, wird er im Folgenden näher betrachtet.

Der Prozess kann deaktiviert werden (siehe Signalliste (OnOffSwitch) in Abbildung 4.6 (b)), wenn kein wettbewerbsbasierter Versand gewünscht ist, wie es zum Beispiel bei wettbewerbsfreien Slot-Regionen der Fall ist. Falls bei der Deaktivierung

eine Übertragung stattfindet, versucht der Prozess, den Übertragungsversuch abzubrechen. Wenn ein Rahmen bereits an die physikalische Ebene gegeben wurde, d.h. FrameTx verlassen hat, kann die Übertragung nicht mehr gestoppt werden. Aus diesem Grund ist es die Aufgabe der später vorgestellten Kontrollkomponente (siehe Kapitel 4.3.1.3), den Prozess früh genug vor Ablauf der wettbewerbsbasierten Slot-Region zu deaktivieren, damit bei Verlassen der wettbewerbsbasierten Slot-Region tatsächlich keine Übertragung mehr stattfindet. Wenn der Prozess deaktiviert wurde, ist er immer noch empfangsbereit, um Rahmen, die durch bereits begonnene Übertragungen von anderen Stationen gesendet werden, empfangen zu können.

Wenn der Prozess aktiviert wird, fordert er Sendeaufträge aus einem Zwischenspeicher, der später in Kapitel 4.3.2 vorgestellt wird, an (siehe Signalliste Prio2Storage in Abbildung 4.6 (b)). Falls keine Aufträge vorliegen, lässt sich über die Konfiguration einstellen, ob der Zwischenspeicher regelmäßig nach neuen Aufträgen abgefragt werden soll (Polling), wobei das Intervall für die Abfragen ebenfalls konfigurierbar ist. Dadurch können Sendeaufträge schneller bearbeitet werden, die in Auftrag gegeben wurden, nachdem alle anderen wettbewerbsbasierten Aufträge bereits abgearbeitet wurden. Durch Deaktivierung des Pollings bietet *MacZ* die Möglichkeit Energie zu sparen und den Transceiver – nach Versand des letzten Auftrages – zu deaktivieren. Dazu muss zusätzlich sichergestellt sein, dass auch die anderen Station in der aktuellen Slot-Region nichts mehr senden.
Falls im Zwischenspeicher Aufträge sind, sendet der Zwischenspeicher nach der Anfrage einen Auftrag mit dem zu sendenden Rahmen zurück (Signal PrioResponse). Nach einer erfolgreichen Übertragung meldet der Prozess der Netzwerkebene den erfolgreichen Versand (in der Signalliste TxRx2Mw enthalten) und löscht den zugehörigen Auftrag aus dem Zwischenspeicher. Falls der Rahmen nicht erfolgreich verschickt werden konnte, wird ebenfalls die Netzwerkebene informiert (Signal der Signalliste TxRx2Mw), aber der Auftrag nicht aus dem Zwischenspeicher gelöscht. Bei einer erfolglosen Übertragung werden jedoch zunächst weitere Sendeversuche unternommen, bis eine maximale Anzahl an Versuchen erreicht ist, die sich über die Konfiguration einstellen lässt. Erst wenn auch nach mehreren Wiederholungen der Rahmen nicht erfolgreich übertragen werden konnte, wird der Netzwerkebene der Misserfolg gemeldet.

Der Prozess beginnt eigene Sendevorgänge, indem er Daten- oder RTS-Rahmen an den FrameTx-Prozess gibt, der daraufhin eine Arbitrierungsphase startet. Ebenso kann PrioTxRx auf RTS-Rahmen reagieren, die von einem anderen Knoten gesendet wurden und an den eigenen Knoten adressiert sind.

Durch Zusammenfassung von ursprünglich sechs Prozessen zu einem wurde die Komplexität in dem Prozess unweigerlich größer, wobei jedoch die Komplexität, die durch die Abhängigkeiten zu den anderen Prozessen bestand, wegfiel. Zur Strukturierung wären SDL-Service(typen) optimal geeignet, weil sie ohne Signalaustausch über gemeinsame Variablen synchronisiert werden können und eine übersichtliche Komposition in einem Prozess erlauben. Allerdings stößt man dabei an die Grenzen des Code-Generators, der später aus der SDL-Spezifikation Code generiert. Wegen den beschränkten Ressourcen der verwendeten *MicaZ*-Plattform, insbesondere in Bezug auf Speicherressourcen, wird zur Codeerzeugung der *Cmicro*-Codgenerator verwen-

det, der zwar sparsamer mit Ressourcen umgeht, aber nur eine Untermenge aller SDL-Konstrukte unterstützt.

Da Servicetypen von *Cmicro* nicht unterstützt werden, wurden in PrioTxRx SDL-Prozeduren verwendet, um das Verhalten der Funktionalitäten (RTS/CTS und einfache Übertragung) zu trennen. Allerdings müssen auch bei Verwendung von SDL-Prozeduren die Beschränkungen des Codegenerators beachtet werden, denn dieser erlaubt keine Verwendung von SDL-Zuständen in den Prozeduren.

Die Prozeduren synchronisieren sich über gemeinsame Variablen, die in PrioTxRx definiert sind. Der Zustand des Prozesses wird daher sowohl von den SDL-Zuständen bestimmt, als auch durch den Wert der Variablen. Eigentlich bestimmen die Werte von Variablen immer den Zustand eines Prozesses mit, werden hier jedoch in der Bedeutung von SDL-Zuständen verwendet, um eine der Beschränkungen von *Cmicro* zu umgehen.

Zunächst wird auf die SDL-Zustände im Prozess PrioTxRx eingegangen, anschließend werden die Prozeduren und Synchronisationsvariablen näher erläutert.

Die folgenden SDL-Zustände sind im Prozess PrioTxRx definiert. Sie geben im Prinzip nur an, ob der Prozess aktiviert oder deaktiviert ist, d.h. über die SDL-Zustände wird bestimmt, ob auf ein Ereignis reagiert wird. Der aktuelle Status von Rahmenübertragungen ist durch die später vorgestellten Synchronisationsvariablen festgelegt.

- off: Der Prozess ist deaktiviert und beginnt keine neuen Übertragungen. Ebenfalls werden eingehende RTS-Rahmen nicht bearbeitet.
- on: Der Prozess ist aktiviert, d.h. er fordert Aufträge aus dem Zwischenspeicher an und beginnt neue Übertragungen. Er reagiert auch auf eingehende RTS-Rahmen.
- oldPending: Der Prozess wurde deaktiviert, während ein Rahmen gesendet wurde, d.h. nachdem ein Rahmen an den Prozess FrameTx weitergegeben wurde, und wurde wieder aktiviert, bevor eine Bestätigung für den Rahmen vorlag. Dieser Zustand ist sehr selten und kann nur auftreten, wenn zwei wettbewerbsbasierte Slot-Regionen sehr dicht aufeinander folgen.

Abbildung 4.7 zeigt die Prozeduren, die in PrioTxRx definiert sind. Diese lassen sich in vier Klassen unterteilen:

- **Allgemeine Hilfs-Prozeduren**: Verhalten, das an mehreren Stellen im Prozess oder in den Prozeduren benötigt wird, wurde hier spezifiziert, um eine mehrfache Spezifikation der gleichen Funktionalität zu verhindern.
- **Prozeduren zum Lesen/Schreiben von/in Rahmen**: Die Prozeduren beinhalten Zugriffe auf Rahmen, wie zum Beispiel das Schreiben der Länge eines vorliegenden Datenrahmens in einen RTS-Rahmen. Ausgenommen sind Zugriffe, die ebenfalls von der Komponente zum wettbewerbsfreien Versand durchgeführt werden. Diese sind mit Prozeduren in einem gesonderten Package spezifiziert.

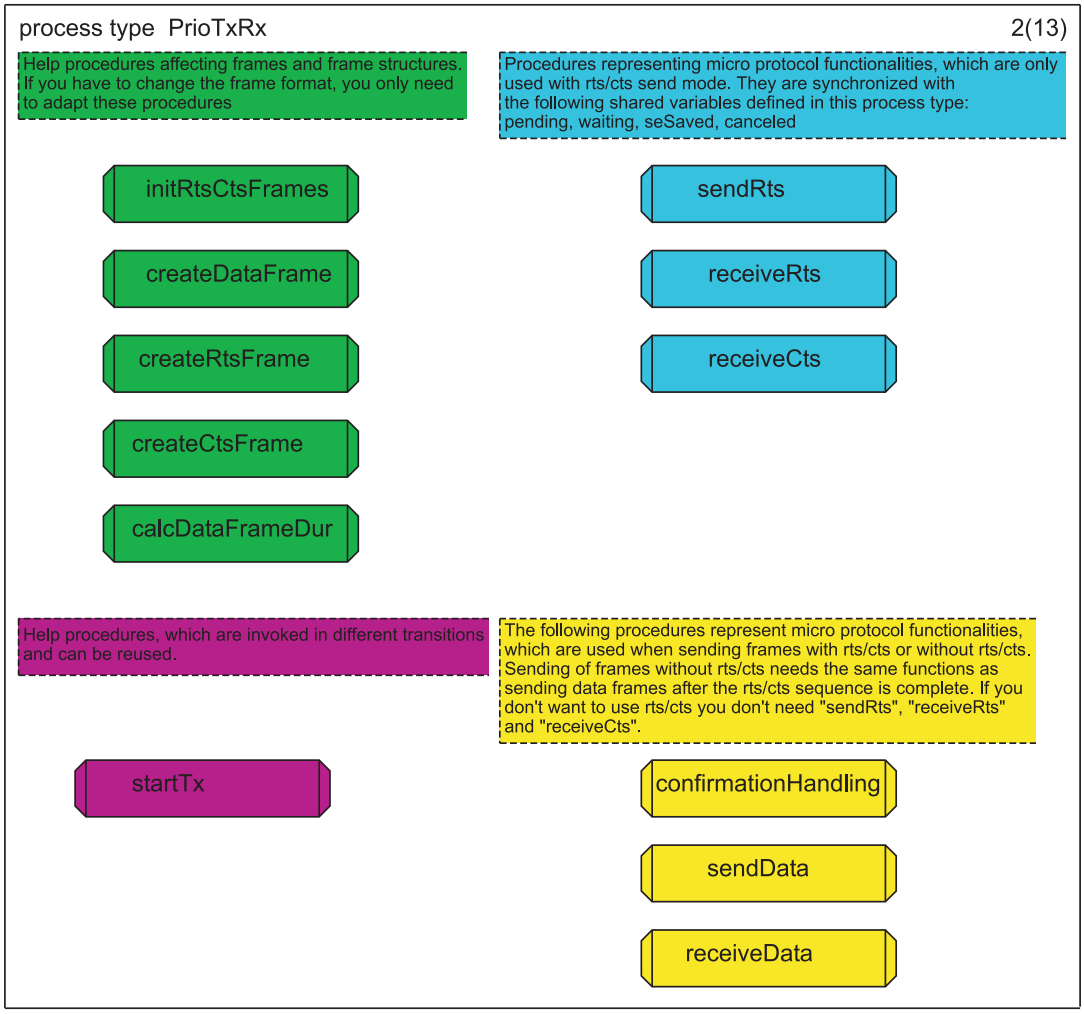


Abbildung 4.7.: Definition der Prozeduren im SDL-Prozess PrioTxRx. Die Prozeduren lassen sich in vier Klassen unterteilen, die farblich gekennzeichnet sind.

- **Prozeduren zur Datenübertragung**: In dieser Klasse ist das Verhalten definiert, das allgemein Datenrahmen betrifft. Dazu gehört sowohl das Senden, als auch das Empfangen von Daten und ebenso die Reaktionen auf Übertragungsbestätigungen für Rahmen, die zuvor an FrameTx gegeben wurden. Bei negativen Bestätigungen wird hier entschieden, ob ein neuer Sendeversuch unternommen wird; gegebenenfalls wird die Netzwerkebene über den Misserfolg benachrichtigt. Bei positiven Bestätigungen wird hingegen die Netzwerkebene über den erfolgreichen Versand informiert.
- **RTS/CTS-Prozeduren**: Das Verhalten für den RTS/CTS-Mechanismus wurde in diesen Prozeduren spezifiziert. Darin enthalten sind sowohl der aktive Start einer RTS/CTS-Sequenz, als auch das Reagieren auf einen eintreffenden RTS-Rahmen.

Obwohl einfache Datenübertragungen und RTS/CTS-Übertragungen in unterschiedlichen Prozedur-Klassen zusammengefasst wurden, sind die Klassen nicht vollständig

unabhängig. Unter anderem wird die Prozedur confirmationHandling bei erfolgreichem Versand eines RTS-Rahmens und bei erfolgreichem Versand eines Datenrahmens aufgerufen. Grund dafür ist, dass bei der Schnittstelle zu dem FrameTx-Prozess nicht zwischen unterschiedlichen Rahmentypen unterschieden wird, d.h. das gleiche SDL-Signal für die Bestätigung des Datenrahmens und des RTS-Rahmens verwendet wird. Die Rahmentypen unterscheiden sich zwar in der Anzahl an Pulse-Slots, die als Interframe-Spacing gewartet werden sollen oder die die Ober- und Untergrenze des Contention-Windows angeben, aber in FrameTx wird zwischen diesen Werten und dem Rahmentyp kein Zusammenhang hergestellt. Diese Abstraktion ist auch gewünscht, um die Schnittstelle klein zu halten und die für alle Rahmentypen gleiche Komponente für das Arbitrierungsverfahren zu verwenden.

Zur Synchronisation zwischen den Prozeduren wurden aus diesem Grund vier gemeinsame Variablen in PrioTxRx definiert. Anhand der Werte kann zum Beispiel – bei Erhalt einer Bestätigung für einen erfolgreichen Versand – unterschieden werden, welcher Rahmentyp versandt wurde:

- pending: Gibt an, ob ein Rahmen an den Prozess FrameTx übergeben wurde und wenn ja, von welchem Typ der Rahmen ist. Mit der Variable kann bei einer Bestätigung von FrameTx erkannt werden, für welchen Rahmentyp die Bestätigung gilt. Für die Variable wurde der Datentyp PendingType eingeführt, der folgende Literale besitzt:
  - no: kein Rahmen in Bearbeitung
  - rts: ein RTS-Rahmen wurde an FrameTx gegeben
  - cts: ein CTS-Rahmen wurde an FrameTx gegeben
  - data: ein Datenrahmen wurde an FrameTx gegeben
- waiting: Diese Variable wird für den RTS/CTS-Mechanismus benötigt und gibt an, ob auf einen Datenrahmen oder einen CTS-Rahmen gewartet wird. Sie ist vom neu spezifizierten Typ WaitingType, welcher aus folgenden Literalen besteht:
  - no: aktuell läuft keine RTS/CTS-Sequenz
  - data: ein Datenrahmen wird erwartet, d.h. es wurde zuvor ein CTS-Rahmen gesendet
  - cts: ein CTS-Rahmen wird erwartet, d.h. es wurde zuvor ein RTS-Rahmen versandt
- seSaved: Diese Variable vom Typ Boolean speichert, ob ein Rahmen für eine spätere Übertragung gespeichert wurde. Bei einer laufenden RTS/CTS-Sequenz muss mit der Übertragung eines neuen Rahmens gewartet werden, bis die Sequenz beendet ist, d.h. der neue Rahmen muss so lange gespeichert werden. Erst nach dem Ende der Sequenz kann die Übertragung starten.
  seSaved wird unter anderem auf true gesetzt, wenn die Prozedur sendData (siehe Abbildung 4.7) aufgerufen wird, aber die Variable waiting auf data gesetzt ist. In diesem Fall wurde auf ein RTS mit einem CTS geantwortet und der

Prozess wartet auf die angekündigten Daten des RTS-Senders, bevor der gespeicherte Rahmen übertragen wird.

- canceled: Gibt an, ob der Übertragungsversuch eines bereits an FrameTx übergebenen Rahmens abgebrochen wurde. In diesem Fall wird eine negative Übertragungsbestätigung von FrameTx nicht als regulärer Sendeversuch gewertet, d.h. die Anzahl an Versuchen wird nicht verringert.
  Ein Übertragungsversuch wird zum Beispiel abgebrochen, wenn der Prozess am Ende der wettbewerbsbasierten Slot-Region deaktiviert wird oder wenn ein RTS-Rahmen übertragen werden soll (pending = RTS) und ein RTS-Rahmen empfangen wurde. In dem zweiten Fall würde es wenig Sinn machen, den RTS-Rahmen zu senden, da mit einer hohen Wahrscheinlichkeit der Empfänger des eigenen RTS-Rahmens das fremde RTS ebenfalls empfangen hat, und somit nichts sendet, da der *NAV* gesetzt ist.

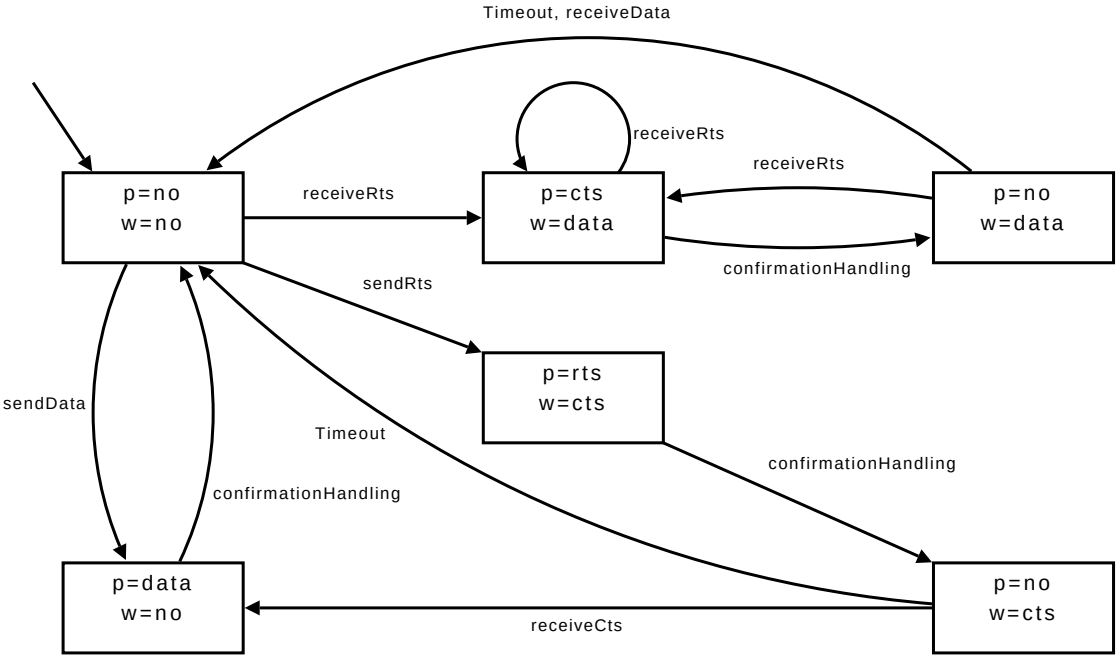


Abbildung 4.8.: vereinfachtes Zustandsdiagramm des Prozesses PrioTxRx mit wichtigen Zuständen und Übergängen. p und w stehen abkürzend für die Variablennamen pending und waiting

Von den theoretisch 48 ($4 \cdot 3 \cdot 2 \cdot 2$) möglichen Wert-Kombinationen der vier Zustandsvariablen können viele in der Praxis nicht auftreten. Zum Beispiel kann nicht gleichzeitig ein Rahmen für eine spätere Übertragung gespeichert (seSaved=true) und auf die Bestätigung der Übertragung eines anderen Datenrahmens von FrameTx gewartet werden (pending=data), da PrioTxRx maximal eine von dem eigenen Knoten initiierte Übertragung verwaltet. In dem Zustandsdiagramm in Abbildung 4.8 sind einige der wichtigsten Kombinationen mit Übergängen aufgeführt. Das Diagramm ist stark vereinfacht und betrachtet nur Zustände der Variablen pending (mit p abgekürzt) und waiting (mit w abgekürzt) im SDL-Zustand on. An den Kanten stehen

die entsprechenden SDL-Prozeduren, die zu den Zustandsübergängen führen. Die beiden Timeout-Übergänge bilden dabei Ausnahmen.

In der Abbildung werden nur erfolgreiche Sendeversuche dargestellt, d.h. die Prozedur confirmationHandling, die in der SDL-Spezifikation positive *und* negative Bestätigungen behandelt, wird hier nur bei positiven aufgerufen. Bei negativen Bestätigungen wird, wenn ein Daten- oder RTS-Rahmen erfolglos übertragen wurde, der Sendeversuch wiederholt. Bei erfolgloser Übertragung eines CTS-Rahmens wird hingegen die Variable waiting zurück gesetzt. Diese Zustandsübergänge sind in der Abbildung nicht dargestellt. Es werden auch keine Zustandsübergänge betrachtet, die einen Fehlerfall darstellen, wie zum Beispiel der Übergang durch Aufruf der Prozedur sendData im Zustand (p=data,w=no). Des Weiteren werden keine Fälle betrachtet, die durch Nebenläufigkeit mehrerer Knoten entstehen können, wie zum Beispiel der Empfang eines RTS-Rahmens, während auf die Sendebestätigung eines Datenrahmens gewartet wird.

Das gewünschte Verhalten ist – ausgenommen von der Entscheidung, ob überhaupt auf ein Ereignis reagiert wird – komplett in den Prozeduren realisiert. Bei Empfangen eines Signals wird die verantwortliche Prozedur aufgerufen, die anhand der Variablenwerte entscheidet, wie darauf reagiert wird. In Abbildung 4.9 sind zwei Transitionen herausgegriffen, die zeigen, wie die Reaktion auf Signale an Prozeduren delegiert wird.

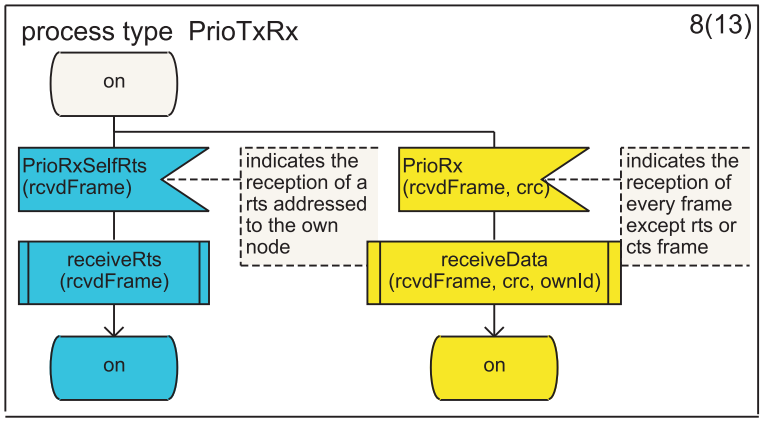


Abbildung 4.9.: Beispiel einer Transition.

Durch die Verteilung der Funktionalität auf Prozeduren wurde trotz der großen Abhängigkeit eine Trennung erreicht. Mit diesem Ansatz sind einfache Änderungen am Verhalten möglich. Auch das Verhalten des RTS/CTS-Mechanismus wurde separat spezifiziert und besteht aus den entsprechenden Prozeduren und den Prozeduraufrufen. Dadurch kann zum Beispiel der Mechanismus durch Löschen der korrespondierenden Prozeduraufrufe entfernt werden.

Abbildung 4.10 zeigt mit einem Beispielszenario in Form eines *MSC*s den Zusammenhang zwischen FrameTx, NAV und PrioTxRx auf zwei Stationen. Das Szenario zeigt vereinfacht die Übertragung eines Datenrahmens. Die Zustände an den vertikalen Achsen entsprechen den SDL-Zuständen. Die Kommentarboxen neben den Achsen zeigen die Werte der beiden Variablen pending und waiting.

Das Szenario beginnt damit, dass der Prozess PrioTxRx einen Auftrag mit PrioRequest anfordert und mit PrioResponse erhält. Er sendet den Rahmen in einer SingleFrameStruct zu dem Prozess FrameTx, der die Arbitrierung durchführt, d.h. Interframe-Spacing und Backoff-Intervall abwartet, und anschließend den Rahmen überträgt, da keine andere Belegung des Mediums festgestellt wurde. Die erfolgreiche

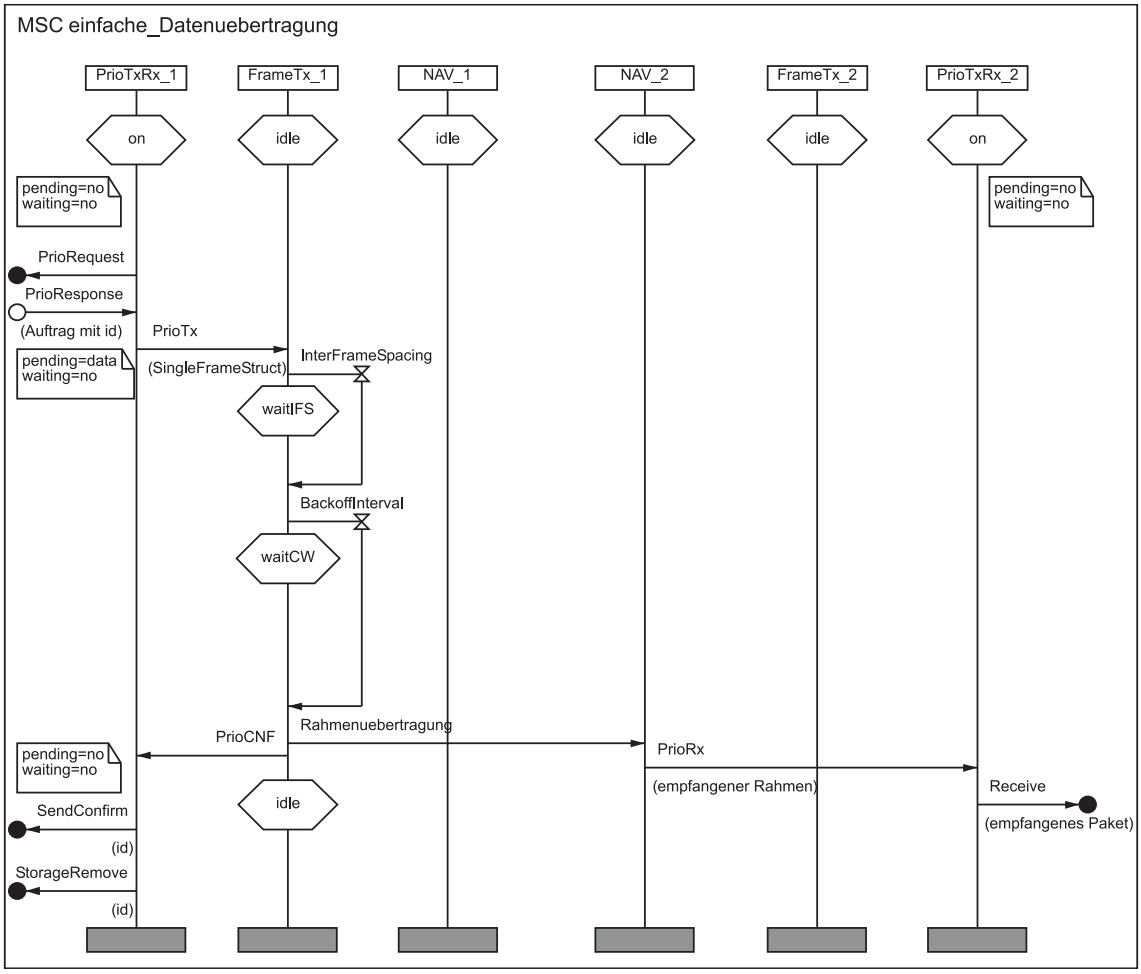


Abbildung 4.10.: MSC, das den Ablauf in dem Blocktyp PrioTxRx zeigt. Es wird eine einfache Übertragung eines Datenrahmens zwischen 2 Instanzen des Mikroprotokolls dargestellt.

Übertragung wird PrioTxRx gemeldet, der daraufhin der Netzwerkebene die erfolgreiche Übertragung bestätigt und den Auftrag aus dem Zwischenspeicher löscht. Auf der anderen Seite empfängt der Prozess NAV den Rahmen und leitet ihn zu dem Prozess PrioTxRx weiter. Dort wird der Rahmen als Paket an die Netzwerkebene weitergegeben.

4.3.1.3. Kontroll-Komponente

Das SDL-Package ModeCtrl in Abbildung 4.11 stellt ein Mikroprotokoll dar, welches ein zentraler Bestandteil von *MacZ* ist. Es sorgt für die Einteilung in virtuelle Slot-Regionen auf Basis von Micro-Slots. Dadurch können die Komponenten zur wettbewerbsbasierten und wettbewerbsfreien Übertragung von der konkreten Micro-Slot-Zählung abstrahieren. Die Kontroll-Komponente (de-)aktiviert die beiden Übertragungskomponenten abhängig von der aktuellen Micro-Slot-Nummer und erreicht dadurch, dass maximal eine der beiden Komponenten aktiv ist. Für den wettbewerbsfreien Zugriff auf das Medium übernimmt sie ein Mapping zwischen

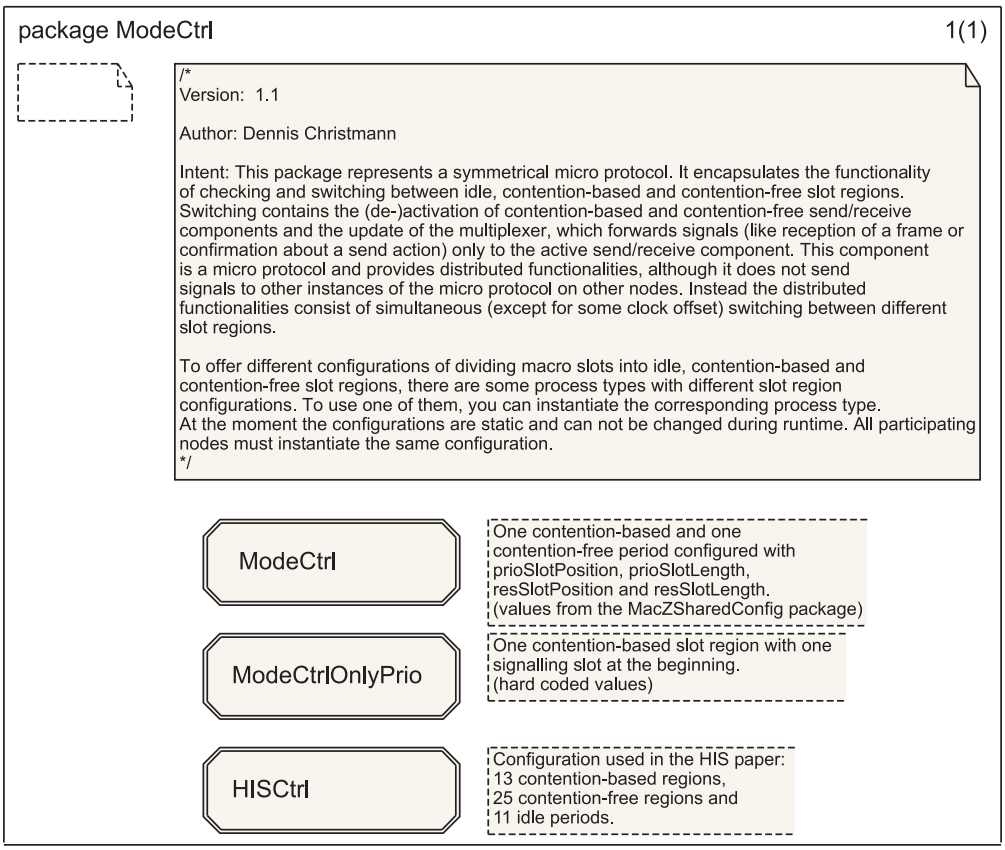


Abbildung 4.11.: Komponente zur Einteilung des (mit Micro-Slots geslotteten) Mediums in Slot-Regionen und zum (De-)Aktivieren der beiden Versandkomponenten

der Micro-Slot-Zählweise und der Reservierungs-Slot-Zählweise und teilt der wettbewerbsfreien Übertragungskomponente den aktuellen Reservierungs-Slot mit. Bei der wettbewerbsbasierten Komponente muss bei der Spezifikation beachtet werden, dass rechtzeitig vor Ende der wettbewerbsbasierten Slot-Region die Komponente deaktiviert wird, um sicherzustellen, dass eine bereits begonnene Übertragung bis zum Ende der Slot-Region abgeschlossen ist. Dies wird dadurch erreicht, dass die Komponente einige Micro-Slots vor Ende der wettbewerbsbasierten Slot-Region deaktiviert wird, wobei die Anzahl der Micro-Slots von der zeitlichen Länge eines Micro-Slots abhängt. Die wettbewerbsbasierte Übertragungskomponente versucht eventuell nach der Deaktivierung, eine aktuelle Übertragung abzubrechen, kann aber immer noch empfangene Datenrahmen bearbeiten und zur Netzwerkebene weiterleiten.

Zur vollständigen Separation der beiden Übertragungskomponenten, sendet die Kontroll-Instanz Signale an den Multiplexer des Service-Layers, welche dem Multiplexer angeben, welche Übertragungskomponente aktiv ist, d.h. an welche Komponente eingehende Rahmen und Statusmeldungen des Transceivers (wie zum Beispiel das erfolgreiche Ende einer Rahmenübertragung) weitergeleitet werden sollen. Dadurch wird erreicht, dass Meldungen des Transceivers und empfangene Rahmen nur an die Komponente geleitet werden, die für die Übertragung in der aktuellen Slot-Region zuständig ist.

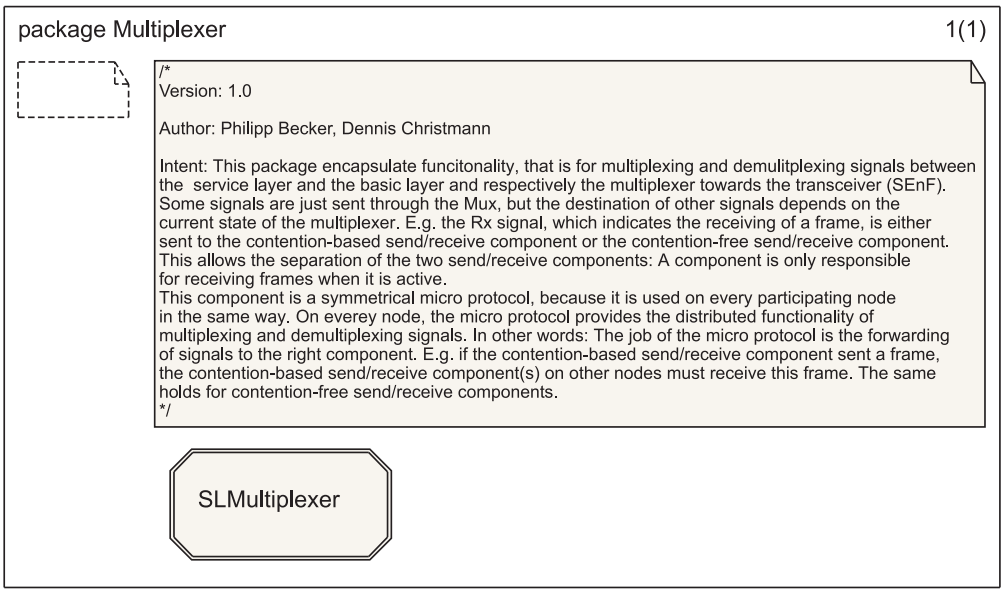


Abbildung 4.12.: Zustandsbehaftete Multiplexer-Komponente zwischen Service-Layer und Basic-Layer/physikalischem Layer

Die drei dargestellten Prozesstypen in Abbildung 4.11 stellen unterschiedliche Konfigurationen der Einteilung in virtuelle Slot-Regionen dar, wie zum Beispiel die Einteilung in eine einzige wettbewerbsbasierte Slot-Region. Pro Knoten muss der gleiche Prozesstyp instanziiert werden, damit alle Knoten die gleiche Einteilung des Mediums vornehmen. Die Spezifikation in unterschiedlichen Prozesstypen erlaubt das schnelle Wechseln der Sloteinteilung durch Instanziieren des entsprechenden Typs. Zur Zeit muss man sich bereits zur Kompilierzeit für eine Konfiguration entscheiden, d.h. diese ist statisch.

Obwohl eine Instanz der Komponente kein Signal mit Instanzen auf anderen Knoten austauscht, fällt die Komponente dennoch unter den Begriff Mikroprotokoll, da sie eine verteilte Funktionalität bereitstellt. Diese Funktionalität besteht aus dem synchronen Wechsel (mit kleineren Abweichungen aufgrund des Tick Offsets) zwischen virtuellen Slot-Regionen auf allen Stationen des Netzwerkes.

4.3.1.4. Multiplexer des Service-Layer

Im Laufe der Restrukturierung wurde der Multiplexer des Service-Layer um eine Zustandsvariable erweitert, die angibt, an welche Übertragungskomponente Signale von dem Transceiver, wie z.B. empfangene Rahmen, weitergeleitet werden. Dadurch unterscheidet sich der Multiplexer von einem einfachen zustandslosen Multiplexer, der Signale immer an die gleichen Komponenten leitet. Der Zustand des Multiplexers wird von der Kontroll-Komponente bestimmt, da diese entscheidet, welche Übertragungskomponente aktiv ist, d.h. an welche Komponente empfangene Rahmen geleitet werden.

Bei der Komponente handelt es sich um ein Mikroprotokoll, da die nötigen Eigenschaften erfüllt sind. Insbesondere ist auch die Eigenschaft der verteilten Funktio-

nalität erfüllt, da der Multiplexer auf jeder Station in dem Netzwerk Signale auf die gleiche Art und Weise (de-)multiplext und durch seinen Zustand dafür sorgt, dass ein Rahmen, der von der wettbewerbsbasierten Übertragungskomponente gesendet wurde auch von der wettbewerbsbasierten Übertragungskomponente empfangen wird. Das Mikroprotokoll wurde in dem Package Multiplexer spezifiziert, das in Abbildung 4.12 abgebildet ist.

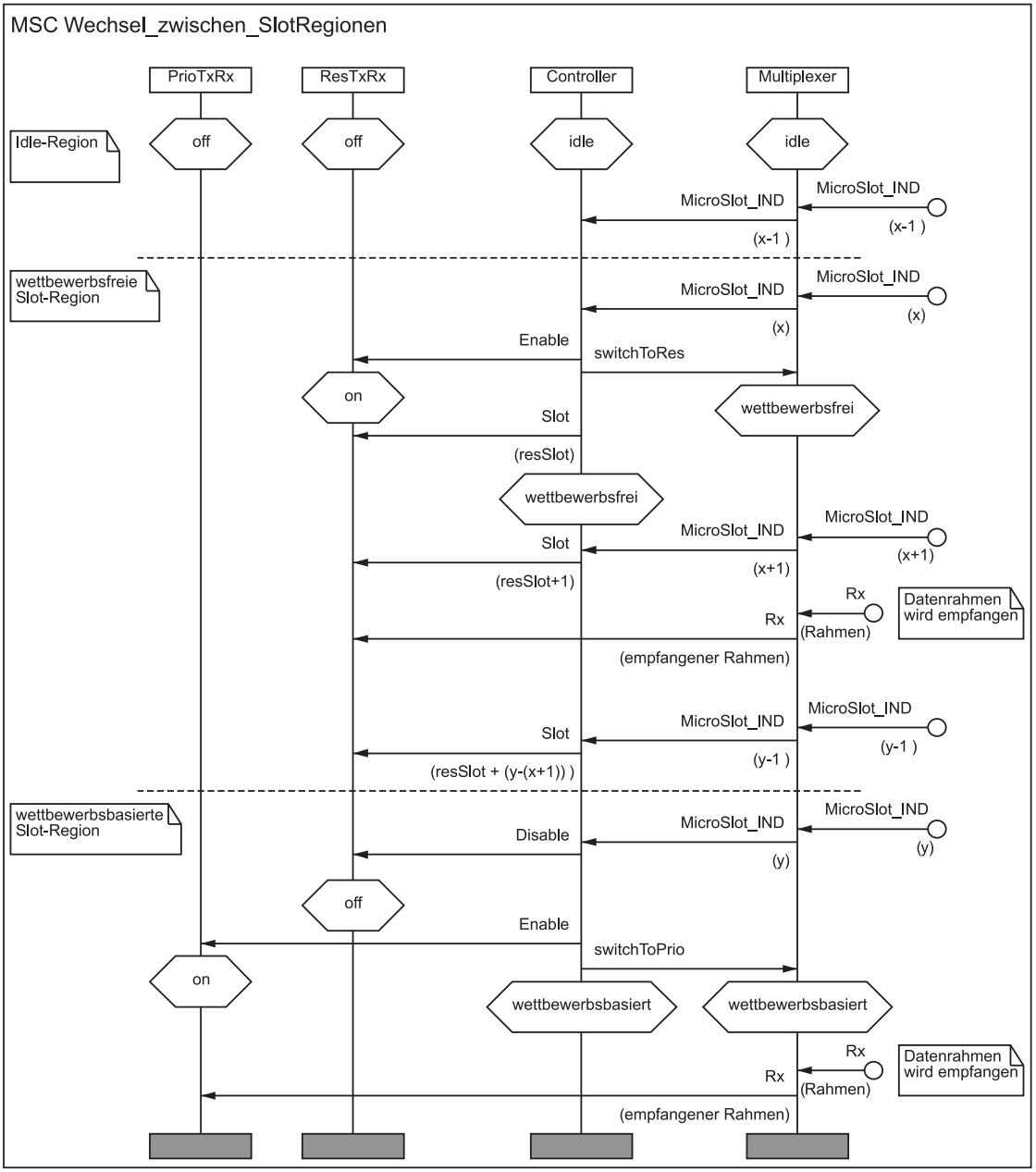


Abbildung 4.13.: Interaktion zwischen den Komponenten Multiplexer, Controller, wettbewerbsbasierter und wettbewerbsfreier Übertragungskomponente. Das mit einem MSC dargestellte Beispiel zeigt den Wechsel zwischen idle, wettbewerbsfreien und wettbewerbsbasierten Slot-Regionen.

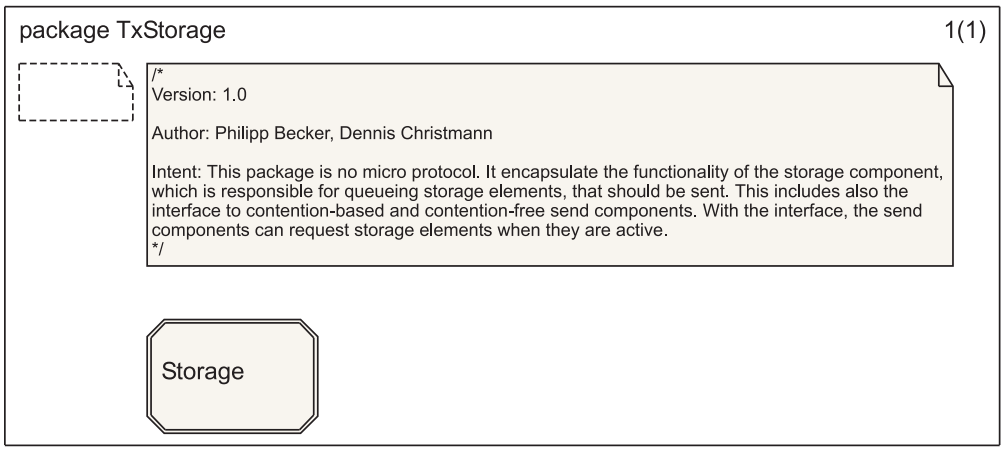


Abbildung 4.14.: Komponente zur Zwischenspeicherung von Elementen, die von der Netzwerkschicht zur Übertragung in Auftrag gegeben wurden

Das Zusammenspiel zwischen Multiplexer, Kontroll-Komponente und den beiden Übertragungskomponenten ist in dem *MSC* in Abbildung 4.13 abgebildet. Der Controller erhält über den Multiplexer vom Basic-Layer Informationen über die aktuelle Micro-Slot-Nummer und aktiviert bei Micro-Slot-Nummer x die wettbewerbsfreie Übertragungskomponente. Verbunden damit wird die Reservierungs-Slot-Nummer zu der wettbewerbsfreien Komponente gesendet und der Zustand im Multiplexer aktualisiert, so dass einkommende Rahmen an die wettbewerbsfreie Übertragungskomponente weitergeleitet werden. Bei Micro-Slot-Nummer y wird die wettbewerbsfreie Komponente deaktiviert, die wettbewerbsbasierte Komponente aktiviert und ebenfalls der Zustand im Multiplexer aktualisiert.

4.3.2. Weitere SDL-Komponente: Zwischenspeicher

Neben den genannten Mikroprotokollen wurde im Service-Layer der Zwischenspeicher als weitere Komponenten identifiziert, der nach objektorientierten Ansätzen zusammengefasst wurde. Die Komponente verletzt die Eigenschaft der Verteiltheit und fällt deswegen nicht unter die Kategorie eines Mikroprotokolls. Sie hat dennoch eine starke Bindung (Cohesion) und eine schwache Kopplung (Coupling), die eine Wiederverwendung fördern und zur Wartbarkeit des Systems beitragen. Unter einer Bindung versteht man ein Maß dafür, wie aufgabenspezifisch eine Komponente ist, d.h. eine Komponente mit einer starken Bindung erledigt exakt eine Aufgabe. Kopplung bezieht sich auf die Schnittstelle zwischen zwei Komponenten. D.h. eine Kopplung ist schwach, wenn zum Beispiel als Signal-Parameter nur Parameter angegeben werden, die von dem empfangenden Prozess benötigt werden [Joh96].

In den folgenden Unterabschnitten ist die Komponente beschrieben, wobei primär das sichtbare Verhalten nach außen im Mittelpunkt steht.

Die Komponente wurde in dem SDL-Package TxStorage gekapselt, das in Abbildung 4.14 dargestellt ist. Sie wurde durch die Restrukturierung kaum verändert. Alle Pakete, die von der Netzwerkebene zur Übertragung in Auftrag gegeben werden, werden in dem Zwischenspeicher gespeichert, bis sie bearbeitet werden. Der

Zwischenspeicher hat jeweils eine Schnittstelle zur wettbewerbsbasierten und zur wettbewerbsfreien Übertragungskomponente.
Bei jedem neuen Auftrag für einen wettbewerbsfreien Versand bekommt die wettbewerbsfreie Übertragungskomponente Informationen über den Auftrag mit der zugehörigen Reservierungs-Slot-Nummer. Bei Eintritt einer angekündigten Slot-Nummer, fragt die wettbewerbsfreie Übertragungskomponente nach dem speziellen Auftrag und bekommt diesen vom Zwischenspeicher als Antwort. Der Auftrag wird direkt, nachdem er an die wettbewerbsfreie Übertragungskomponente gesendet wurde, aus dem Zwischenspeicher gelöscht, da die wettbewerbsfreie Übertragung – außer in Ausnahmefällen (Hardwarefehlern oder Ähnliches) – nicht scheitert.
Aufträge für den wettbewerbsbasierten Versand werden nicht angekündigt. Stattdessen fragt die wettbewerbsbasierte Übertragungskomponente den Zwischenspeicher nach wettbewerbsbasierten Aufträgen ab und bekommt von dem Zwischenspeicher den Auftrag mit der höchsten Priorität zurück, d.h. den Auftrag, dessen Contention-Window die niedrigste Obergrenze hat. Das Löschen des Auftrages aus dem Speicher erfolgt nicht direkt, sondern wird von der wettbewerbsbasierten Übertragungskomponente erst nach erfolgreicher Übertragung veranlasst.

Das MSC in Abbildung 4.15 zeigt die Interaktion zwischen den Übertragungskomponenten und dem Zwischenspeicher:
Der erste Teil zeigt einen möglichen Ablauf in einer wettbewerbsfreien Slot-Region. Die wettbewerbsfreie Übertragungskomponente fordert bei Eintreten des Reservierungs-Slot x einen Auftrag, der vorher für den Reservierungs-Slot x angekündigt wurde, aus dem Zwischenspeicher an und überträgt ihn.
Im zweiten Teil ist ein möglicher Ablauf in einer wettbewerbsbasierten Slot-Region dargestellt, in der die wettbewerbsbasierte Übertragungskomponente wettbewerbsbasierte Aufträge anfordert und den Auftrag mit der höchsten Priorität erhält, der anschließend übertragen wird. Nach erfolgreicher Übertragung wird der Zwischenspeicher angewiesen, den Auftrag zu löschen.

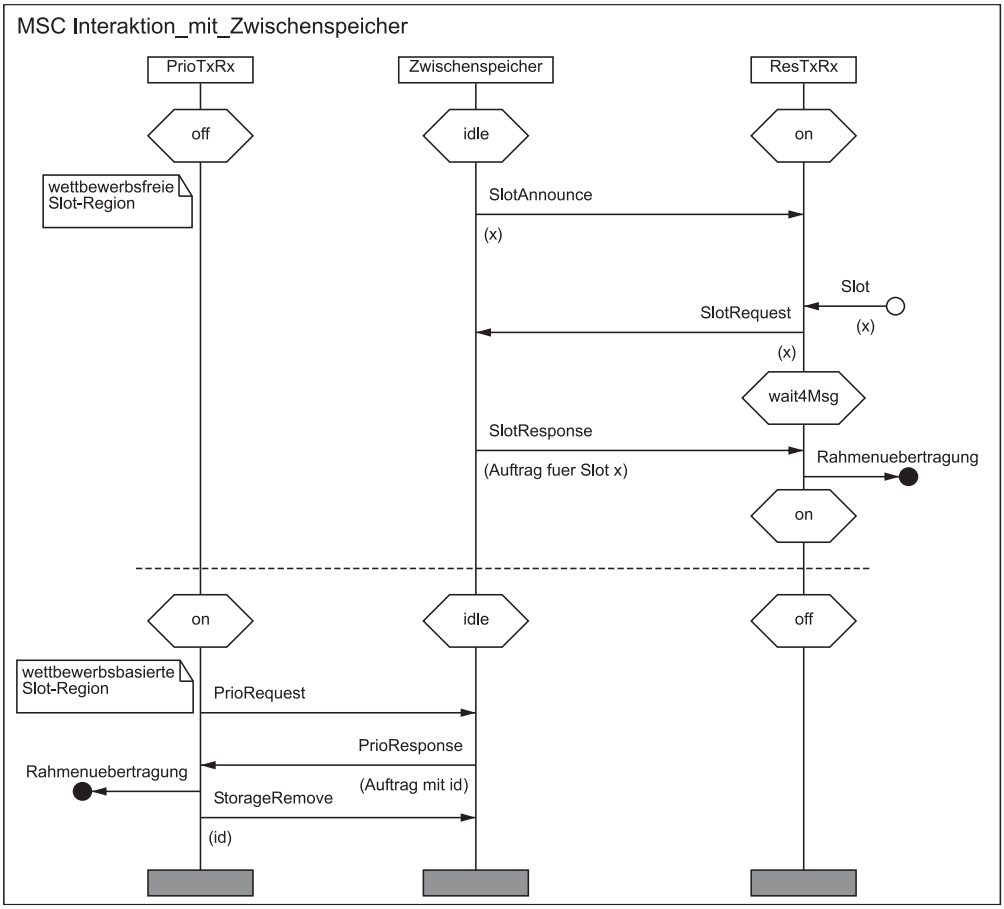


Abbildung 4.15.: Interaktion zwischen den Komponenten Zwischenspeicher, wettbewerbsbasierter und wettbewerbsfreier Übertragungskomponente. In dem ersten Teil findet die Kommunikation in einer wettbewerbsfreien Slot-Region statt, im zweiten Teil in einer wettbewerbsbasierten Slot-Region.

5. Evaluation

Um die Ergebnisse der Restrukturierung zu bewerten, wurden mehrere Simulationen mit unterschiedlichen Szenarien durchgeführt. Da sich die Funktionalität und die Schnittstelle von *MacZ* weder zur Middleware noch zur physikalischen Ebene hin grundsätzlich verändert hat, konnte man erwarten, dass die Ergebnisse vergleichbar mit Ergebnissen früherer Simulationen sind.

Aus diesem Grund wurden zur Evaluation frühere Simulationsszenarien verwendet und die Ergebnisse verglichen. Die Simulationen, die als Vergleichsbasis dienten, wurden in [BGK07] und [Bec06] veröffentlicht.

Die Simulationen können keinen formalen Beweis für die Korrektheit des Entwurfs liefern. Die Resultate zeigen aber, dass bei gängigen Verwendungszwecken – wie zum Beispiel der regelmäßigen Übertragung von Rahmen – keine Fehler auftreten und sich *MacZ* entsprechend den Erwartungen verhält.

Das folgende Kapitel gibt eine Einführung in den verwendeten Simulator PartsSim, bevor in den weiteren Kapiteln die Ergebnisse der einzelnen Simulationen vorgestellt werden.

5.1. Der Simulator PartsSim

Der verwendete Simulator PartsSim ist eine Weiterentwicklung des Netzwerksimulators ns+SDL, welcher wiederum auf ns-2 basiert [BGK08, KGGR05, USC].

ns+SDL, der Vorgänger von PartsSim, wurde von der AG Vernetzte Systeme entwickelt, um eine SDL-Spezifikation sowohl für die automatische Erstellung von Code zu Simulationszwecken als auch für die Codegenerierung für den Produktiveinsatz verwenden zu können. In beiden Fällen können die gleichen SDL-zu-C Code-Generatoren *Cadvanced* bzw. *Cmicro* der Telelogic Tau Suite verwendet werden [Tel]. Da für den Produktiveinsatz und die Simulation die gleiche SDL-Spezifikation als Grundlage dient, ist zu erwarten, dass das Verhalten von Testsystemen in der Simulation dem Verhalten im Produktiveinsatz entspricht [KGGR05].

Ohne die Erweiterungen durch ns+SDL wären weitere Implementierungsschritte notwendig, um ein SDL-System mit ns-2 zu simulieren. D.h. durch ns+SDL kann Entwicklungsaufwand gespart werden. Zusammen mit dem *SDL Environment Framework* (SEnF) der AG Vernetzte Systeme, einer Schnittstelle zwischen dem ns+SDL Simulator und SDL-System, muss der generierte Code lediglich kompiliert werden, um anschließend mit ns+SDL geladen und ausgeführt zu werden.

Da ns+SDL auf ns-2 basiert, werden Netzwerkcharakteristiken wie Netzwerkverzögerung, Sendereichweite und Rahmenkollisionen bei den Simulationen beachtet.

Allerdings werden Eigenschaften der Hardwareplattform nicht beachtet. Aus diesem Grund wurde ns+SDL weiterentwickelt, um eine realistischere Simulation von Hardwareplattformen zu erreichen. Mit PartsSim können nun auch auf manchen Plattformen Hardwareeigenschaften wie Berechnungszeiten durch den Prozessor, Zugriffszeiten auf den Speicher und Verzögerungen durch den Transceiver simuliert werden. Dadurch wird der Tatsache Rechnung getragen, dass neben den Netzwerkressourcen häufig auch die Hardwareplattform einen Engpass darstellt [BGK08].

Für die Simulationen wurde das SDL-System mit dem Code-Generator *Cmicro* in C-Code übersetzt. Da *MacZ* für die *MicaZ*-Plattform entwickelt wird und dort die Ressourcen (insbesondere Speicher) sehr knapp sind, kann *Cadvanced* aufgrund zu großzügigem Umgang mit Speicherressourcen nicht verwendet werden. Anschließend wurden die SDL-Systeme mit PartsSim geladen und unterschiedliche Szenarien simuliert, die im Folgenden vorgestellt werden.

5.2. Simulations-Reihe 1

In der ersten Simulations-Reihe wurden die Szenarien aus [BGK07] verwendet. In diesen werden sehr viele Möglichkeiten von *MacZ* genutzt, wie z.B. prioritätsbasierter und reservierungsbasierter Versand von Rahmen.

Bei den Szenarien wurde – wie für die weiteren Simulations-Reihen ebenfalls – der *CC2420*-Transceiver von *Texas Instruments* für die physikalische Schicht verwendet [Tex]. Der *ZigBee*-kompatible Transceiver hat eine Brutto-Übertragungsrate auf physikalischer Ebene von 250 kBit/s und eine Blind-Periode von 320 μs zum Umschalten von Sende- in Empfangsmodus und von 192 μs zum Umschalten von Empfangs- in Sendemodus.

5.2.1. Topologie

Die in [BGK07] verwendete Topologie besteht aus sechs Knoten, deren Anordnung in Abbildung 5.1 abgebildet ist. Hierbei handelt es sich um ein Multi-Hop-Szenario mit einem maximalen Netzdurchmesser von zwei Hops. Die eingezeichneten Kanten zeigen die Sendereichweiten der einzelnen Knoten. Man kann erkennen, dass fünf Knoten sich gegenseitig hören können. Dadurch können Kollisionen auftreten, falls zwei oder mehr Knoten gleichzeitig senden.

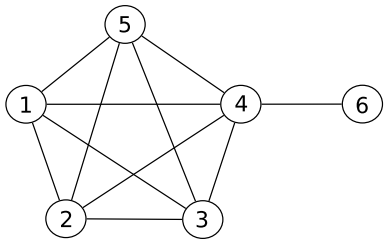


Abbildung 5.1.: Topologie

MacZ versucht Kollisionen in prioritätsbasierten Slot-Regionen zu vermeiden, indem vor der Übertragung eines Daten- bzw. RTS-Rahmens eine zufällige Anzahl an Pulse-Slots aus einem Contention-Window gewartet wird. Falls das Medium während der Wartezeit von einer anderen Station belegt wird, informiert die physikalische Schicht, welche über Clear Channel Assessment den Status des Mediums überwacht, über diese Belegung. Der Arbitrierungsvorgang ist dann gescheitert und wird neu gestartet. Ob das Backoff-Intervall neu bestimmt oder an der unterbrochenen Stelle fortgesetzt

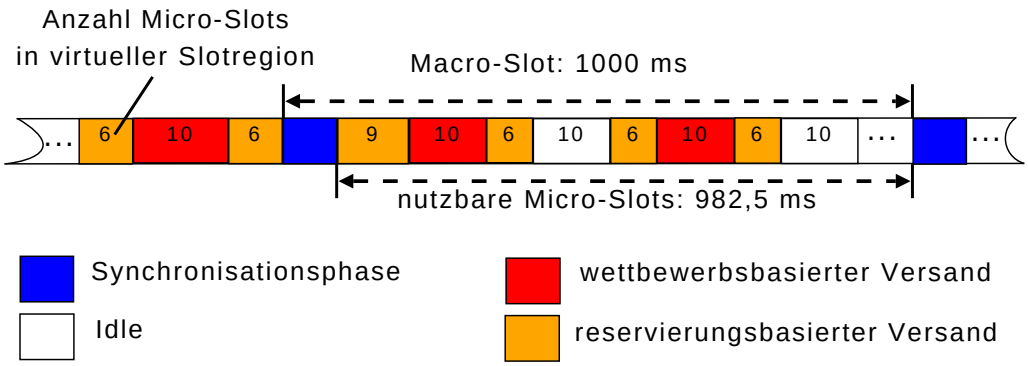


Abbildung 5.2.: Einteilung des Macro-Slots in Regionen für die 1. Simulations-Reihe.

wird, lässt sich in der Konfiguration einstellen. In reservierungsbasierten Perioden ist es Aufgabe des Reservierungsprotokolls, die Sendezeiten benachbarter Stationen disjunkt zu halten.

In der vorgestellten Topologie werden folgende Nachrichten ausgetauscht:

- **Audio-Daten:** Ein Audiostream wird von Knoten 1 erzeugt und an Knoten 4 gesendet. Dieser leitet die Daten anschließend weiter zu Knoten 6, wobei lediglich die Zieladresse im Rahmen geändert wird.
  Eine Audio-Nachricht transportiert 40 ms Audio-Daten und besteht aus 82 Byte, welche auf der MAC-Ebene um einen 10 Byte großen Header ergänzt wird. Insgesamt besteht ein Rahmen mit Audio-Daten somit aus 92 Byte. Daraus ergibt sich, dass Knoten 1 pro Sekunde 25 Rahmen mit Audio-Daten sendet, d.h. pro Hop werden $25\,s^{-1} \cdot 82\,Byte = 2050\,Byte/s$ an Übertragungsrate (netto) benötigt. Für den Transport von Audio-Informationen gelten hohe Ansprüche bezüglich Ende-zu-Ende-Verzögerung und Verzögerungsschwankung. So sollte die Ende-zu-Ende-Verzögerung 250 ms nicht überschreiten und die Verzögerungsschwankung sollte möglichst klein (idealerweise 0) sein.

- **Events:** In zufälligen Abständen (im Durchschnitt ein mal pro Sekunde) sendet Knoten 3 Events an Knoten 5. Ein Event-Rahmen besteht aus insgesamt 14 Byte (10 Byte MAC-Header und 4 Byte Payload). Im Vergleich zu den Audio- und Hintergrund-Daten sind Events wichtiger.

- **Hintergrund-Daten:** Rahmen mit 50 Byte Payload und 10 Byte zusätzlichen MAC-Header werden in prioritätsbasierten Slot-Regionen ununterbrochen von Station 2 zu Station 5 übertragen.

5.2.2. Aufteilung des Macro-Slots

In Kapitel 2.2 wurde die Möglichkeit der Einteilung eines Macro-Slots in unterschiedliche virtuelle Slot-Regionen beschrieben. Für die betrachteten Szenarien wurde ein Macro-Slot in 393 Micro-Slots mit einer Länge von je 2,5 ms unterteilt. Zusammen mit einer Synchronisationsphase von 17,5 ms hat ein Macro-Slot dadurch eine Dauer von 1 s.

	Anzahl Blöcke	Blockgrößen	Anteil an Macro-Slot	Übertragungs-rate in kBit/s
wettbewerbsfrei	25	6 (9)	38,25%	95,6
wettbewerbsbasiert	13	10	32,5%	81,3
idle	11	10	27,5%	68,8

Tabelle 5.1.: Aufteilung der Micro-Slots auf die unterschiedlichen Slot-Regionen. Der erste wettbewerbsfreie Block besteht aus 9 Micro-Slots. 1,75% des Macro-Slots wird für die Synchronisation benötigt.

Die Aufteilung des Macro-Slots in idle-, wettbewerbsbasierte- und wettbewerbsfreie Bereiche ist in Abbildung 5.2 dargestellt. Insgesamt gibt es in diesem Szenario 13 Regionen für den wettbewerbsbasierten Versand, 25 Regionen mit wettbewerbsfreiem Versand und 11 idle-Bereiche bieten die Möglichkeit, Energie zu sparen. Tabelle 5.1 fasst dies zusammen und zeigt die theoretisch möglichen Übertragungsraten auf physikalischer Ebene. Wegen Blind-Perioden und Interframe-Spacings bzw. Backoff-Intervallen bei der wettbewerbsbasierten Übertragung können diese Werte in der Praxis nicht erreicht werden, stellen aber eine obere Schranke dar.

5.2.3. Wahl der Versandart

Für die Evaluation kamen in mehreren Simulationen fast alle Versandarten zum Einsatz, die MacZ bereitstellt.

- In der ersten Simulation werden alle Rahmen in den wettbewerbsbasierten Regionen versandt. Hierzu bekommen alle Stationen das gleiche Contention-Window [0,10], d.h. jede Station bestimmt zur Medium-Arbitrierung eine zufällige Zahl zwischen 0 und 10 und wartet die gewählte Anzahl an Zeitschlitzen (Backoff-Intervall), bevor sie mit der Übertragung beginnt.
  In diesem Szenario stellen die Hintergrund-Daten das größte Problem für die Dienstgüteanforderungen der Audio-Daten dar. Jedesmal, wenn Station 1 Audio-Daten übertragen möchte, versucht auch Station 2 Hintergrund-Daten zu senden. Die beiden Stationen sind dadurch in *ständigem* Wettbewerb. In Folge dessen kann es zu großen Ende-zu-Ende-Verzögerungen der Audio-Informationen kommen, falls Station 1 die Arbitrierung verliert. Zusätzlich können Kollisionen auftreten, falls zwei Stationen das gleiche Backoff-Intervall bestimmen und gleichzeitig mit dem Sendevorgang beginnen.

- Die zweite Simulation nutzt die Möglichkeit des prioritätsbasierten Versendens in wettbewerbsbasierten Slot-Regionen. Die Rahmen werden weiterhin in den wettbewerbsbasierten Slot-Regionen versandt, allerdings unterscheiden sich die Contention-Windows der Stationen. Station 1 sendet mit einem Contention-Window von [4,4], Station 2 mit [6,6], Station 3 mit [0,0] und Station 4 mit [2,2]. Dies hat zur Folge, dass bei parallelen Arbitrierungsvorgängen, wie zum Beispiel am Anfang einer wettbewerbsbasierten Slot-Region, die Events von Station 3 die höchste Priorität erhalten. Die Hintergrund-Daten von Station 2

erhalten die niedrigste Priorität und stören die Stationen 1 und 4 bei der Übertragung von Rahmen mit Audioinformationen nicht. Voraussetzung hierfür ist, dass das Backoff-Intervall bei fehlgeschlagener Arbitrierung neu – mit dem ursprünglich bestimmten Backoff-Intervall – gezählt wird (siehe Kapitel 5.2.1). Es sei darauf hingewiesen, dass die Ober- und Untergrenze des Contention-Windows paketbezogen angegeben wird. Da in der Simulation jede beteiligte Station nur eine Art von Nutzdaten überträgt, ist das Contention-Window hier stationsbezogen.

- In der dritten Simulation werden Rahmen mit Audio-Daten in reservierungsbasierten Slot-Regionen versendet. Events und Hintergrund-Daten werden weiterhin in prioritätsbasierten Regionen gesendet. Somit ist zu erwarten, dass der Audio-Stream ungestört von Station 1 zu Station 6 gelangt und sich die Gesamtübertragungsrate erhöht, da nun auch wettbewerbsfreie Slot-Regionen benutzt werden, in denen in den ersten beiden Simulationen nichts übertragen wurde. Die Einteilung der Macro-Slots in Slot-Regionen wurde bewusst so vorgenommen, dass alle 40 ms ein wettbewerbsfreier Bereich ist, der zeitlich so dimensioniert wurde, dass Rahmen von Station 1 über Station 4 zu Station 6 gelangen können. Dadurch müssen Rahmen nur sehr kurz bis zum nächsten reservierten Micro-Slot zwischengespeichert werden und eine konstante Ende-zu-Ende-Verzögerung ist zu erwarten.

5.2.4. Übertragungsrate

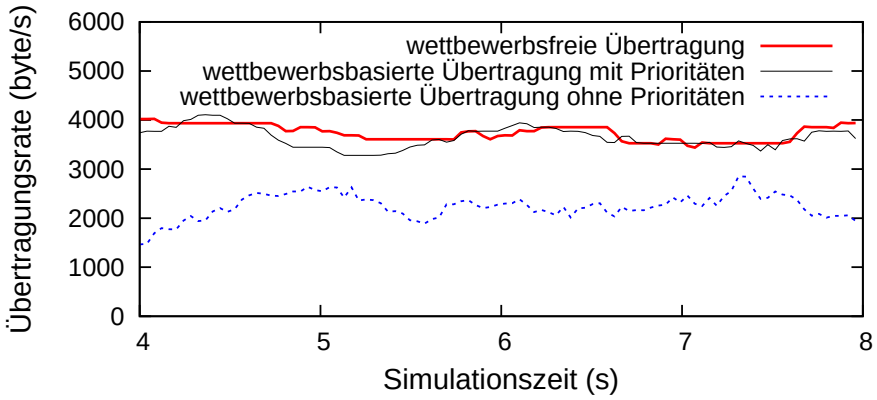
In einem ersten Schritt wird die Übertragungsrate des Audio-Streams untersucht. Die Beispiel-Anwendung erzeugt alle 40 ms Daten mit einer Länge von 82 Byte, d.h. es wird eine Nettodatenrate von 2,05 kByte/s benötigt. Wegen der Übertragung über 2 Hops verdoppelt sich die benötigte Übertragungsrate auf 4,1 kByte/s. Diese Größe stellt die maximale durchschnittliche Übertragungsrate dar, die durch Kollisionen oder Verluste geringer sein kann.

Abbildung 5.3 stellt die Auswirkungen der Versandarten auf die erreichte Nettodatenrate dar. Im Durchschnitt kann maximal eine Übertragungsrate von 4,1 kByte/s erreicht werden. Allerdings kann es durch Bursts bei wettbewerbsbasiertem Versand kurzzeitig zu einer Überschreitung der Grenze kommen.

- Bei wettbewerbsbasiertem Senden ohne Prioritäten ist die verfügbare Übertragungsrate weit von der benötigten entfernt. Ursache hierfür sind Verluste und Kollisionen, die auftreten, da die Stationen 1, 2, 3 und 4 häufig Rahmen gleichzeitig übertragen. Die variable Rate zeigt, dass es vom Zufall abhängt, ob die Mediumarbitrierung gewonnen wird. Damit ein einzelner Rahmen beide Hops überbrücken kann, muss zuerst Station 1 die Arbitrierung gewinnen und anschließend Station 4. Da Station 1 nach erfolgreichem Versand bereits mit der Bearbeitung eines weiteren Rahmens beginnen kann, gibt es zwischen Station 1 und Station 4 ebenfalls einen Wettbewerb.

- Durch Einführung von Prioritäten erreicht die effektiv genutzte Bandbreite des Audiostreams die Grenze von 4,1 kByte/s. Es findet kein direkter Wettbewerb

a)



b)

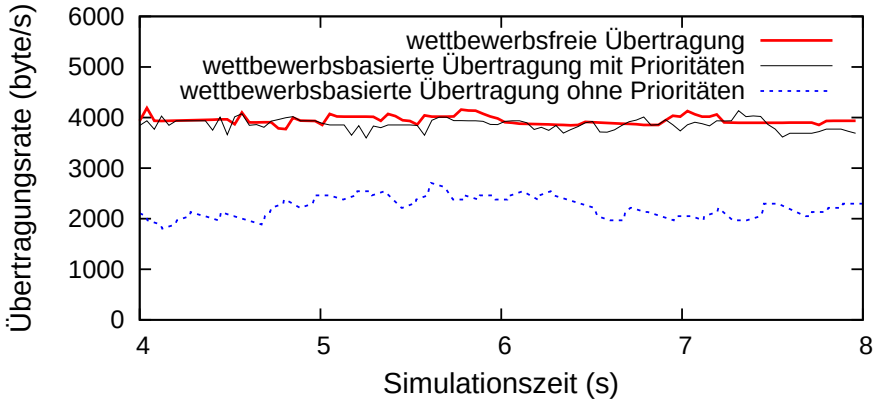


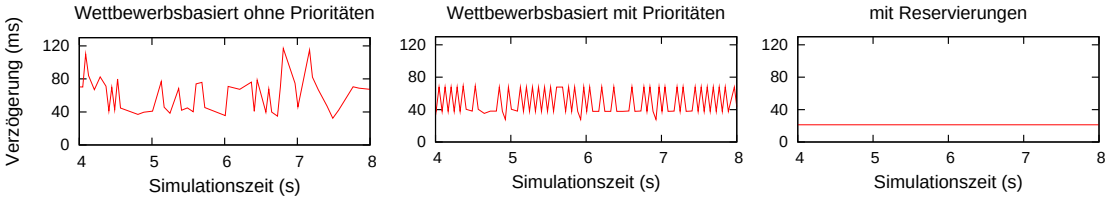
Abbildung 5.3.: Auslastung des Mediums durch Audiodaten. Ergebnisse in (a) stammen aus [BGK07] vor der Restrukturierung, (b) zeigt Ergebnisse nach der Restrukturierung,

mehr statt, da die gewählten Contention-Windows der einzelnen Stationen disjunkt sind. Lediglich das sporadische Senden der Events kann sich auf die Übertragungsrate des Audio-Datenstroms auswirken, da die Events eine höhere Priorität haben. Dies hat aber aufgrund der geringen Häufigkeit der Events nur kleine Auswirkungen.

- Bei wettbewerbsfreiem Senden können Audio-Rahmen direkt in den reservierten Micro-Slots übertragen werden. Da die Audio-Daten im gleichen Intervall erzeugt und gesendet werden, wird das Medium gleichmäßig mit 2 Rahmen pro wettbewerbsfreier Slot-Region ausgelastet. Die erkennbaren Schwankungen in der Übertragungsrate kommen ausschließlich durch Rahmenverluste aufgrund der Entfernung zwischen den Knoten zustande. Kollisionen treten wegen dem wettbewerbsfreien Versand keine auf.

Die Unterschiede zwischen den früheren Ergebnissen und den Ergebnissen dieser Arbeit lassen sich unter anderem durch die Verwendung anderer Parameter für das in den Simulationen verwendete Shadowing-Ausbreitungsmodells erklären. Bei dem Modell ist die Stärke eines Signals nicht nur von der Entfernung des Senders, sondern

a)



b)

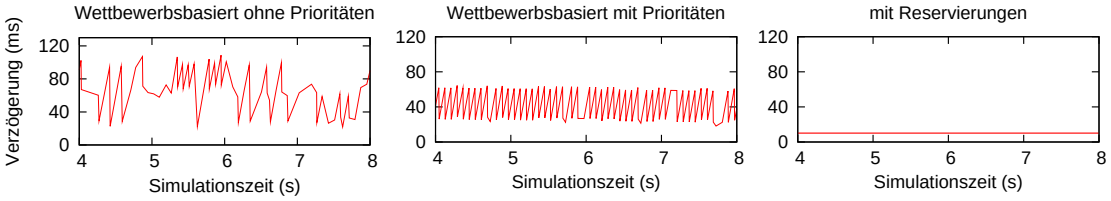


Abbildung 5.4.: Ende-zu-Ende-Verzögerungen von Rahmen mit Audio-Daten über 2 Hops bei unterschiedlichen Versandarten. Die oberen Grafiken stammen aus [BGK07], die unteren Grafiken wurden durch Simulationen nach der Restrukturierung erstellt.

auch von einem distanzbezogenen zufälligen Wert abhängig [USC]. Die Parameter wurden zwischen den Simulationen von [BGK07] und denen dieser Bachelorarbeit empirisch neu bestimmt. Die Änderungen wirken sich z.B. auf die Sendereichweite aus, die nun deutlich geringer ist.
Bei wettbewerbsbasiertem Versand ohne Prioritäten trägt zusätzlich noch die Wahl der Zufallszahlen zur Bestimmung des Backoff-Intervalls zu den Abweichungen bei. Vor allem Zufallszahlen können große Auswirkungen auf den Verlauf der Übertragungsrate haben. Aus diesem Grund kann auch bei wettbewerbsbasiertem Versand ohne Prioritäten keine Dienstgüte garantiert werden.

5.2.5. Ende-zu-Ende-Verzögerung

Bei Audio-Anwendungen gibt es neben den Anforderungen an die Übertragungsraten auch hohe Anforderungen an Verzögerungsschwankungen und die Ende-zu-Ende-Verzögerung. Zum Beispiel ist eine bidirektionale Sprach-Kommunikation mit einer Ende-zu-Ende-Verzögerung größer als 250 ms nicht akzeptabel. Wenn Verzögerungsschwankungen auftreten können, wird ein Playout-Buffer benötigt, um beim Empfänger eine unterbrechungsfreie Ausgabe zu erhalten. Allerdings wirkt sich ein Playout-Buffer negativ auf die Ende-zu-Ende-Verzögerung aus, da Audiodaten zusätzlich gepuffert werden. Damit die Verzögerungen gering bleiben, muss der Playout-Buffer möglichst klein gehalten werden, d.h. die Schwankungen dürfen nicht groß sein.

Abbildung 5.4 fasst die Verzögerungen der Simulationen zusammen.

- Bei der wettbewerbsbasierten Übertragung *ohne Prioritäten* ist die Ende-zu-Ende-Verzögerung sehr variabel. Sie bewegt sich zwischen 20 ms und 120 ms, d.h. der Jitter ist sehr hoch. Ein Grund für die Schwankungen ist die Arbitrierungsphase vor dem Senden eines Rahmens, denn sowohl Station 1 als

auch Station 4 müssen die Arbitrierung gegeneinander und gegen die konkurrierenden Stationen 2 bzw. 3 gewinnen. Zusätzlich entsteht der Jitter durch das relativ große Contention-Window von [0,10], das selbst ohne konkurrierende Übertragungen für Verzögerungsschwankungen zwischen $0 \cdot t_{Pulse-Slots}$ und $10 \cdot t_{Pulse-Slots}$ sorgen, wobei $t_{Pulse-Slots}$ für die Dauer eines Pulse-Slots steht, wie in Kapitel 2.3.1.1 vorgestellt. Falls der Wettbewerb gegen Ende einer wettbewerbsbasierten Region verloren wird, kann sich das Senden eines Rahmens sogar bis zur nächsten wettbewerbsbasierten Region verzögern. Ein weiterer Grund ist die Aufteilung des Macro-Slots, da in jedem 80 ms Zeitabschnitt nur ein 25 ms großes Intervall für den wettbewerbsbasierten Versand zur Verfügung steht. Audiodaten, die kurz vor dem Beginn einer wettbewerbsbasierten Region generiert werden, können dadurch potentiell schneller versendet werden als Daten, die direkt nach einer wettbewerbsbasierten Region generiert werden.

- Mit der Einführung von Prioritäten wird der Jitter deutlich geringer und die Ende-zu-Ende-Verzögerungszeiten nehmen nur noch Werte zwischen 20 ms und 65 ms an. Die Verzögerung durch die Arbitrierung wird kleiner, da es nur noch zu zusätzlichen Verzögerungen kommt, falls ein Audiodatenrahmen während der Übertragung von Hintergrunddaten generiert wird. Die Schwankungen kommen hauptsächlich durch die ungleichmäßige Verteilung der wettbewerbsbasierten Regionen in einem Macro-Slot zustande.

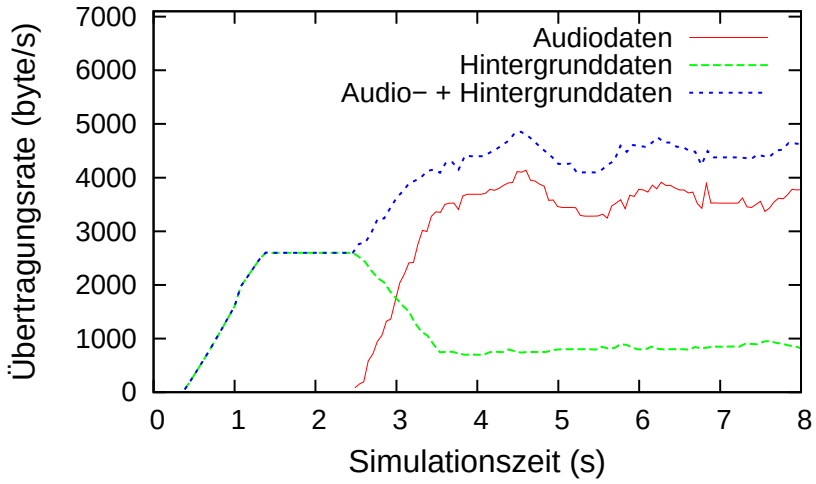
- Die besten Resultate werden erzielt, wenn Audiorahmen in wettbewerbsfreien Slot-Regionen mit Reservierungen gesendet werden. Durch die gleichmäßige Verteilung der wettbewerbsfreien Regionen können Rahmen im gleichen Intervall, in dem sie generiert werden, auch geschickt werden. Wegen dem Senden mit Reservierungen entfällt die Arbitrierungsphase für die Audiodaten. Die Verzögerung besteht lediglich aus der konstanten Wartezeit bis zu dem nächsten reservierten Micro-Slot zuzüglich der Sende- und Ausbreitungsverzögerung (sowohl bei Hop 1 als auch bei Hop 2). Bei den Ergebnissen nach der Restrukturierung beträgt die Ende-zu-Ende-Verzögerung ca. 6 ms, wobei die Ergebnisse stark davon abhängen, ob Audio-Daten kurz nach oder vor einer wettbewerbsfreien Slot-Region generiert werden. D.h. der Unterschied zu den Ergebnissen aus [Bec06] weißt nicht auf eine erreichte Performanzsteigerung von *MacZ* hin, sondern darauf, dass in [Bec06] der zeitliche Abstand zwischen Generierung von Audiodaten und Übertragung etwas größer war.

Zwischen den Ergebnissen aus [BGK07] und den aktuellen Resultaten fallen vor allem die Unterschiede bei dem wettbewerbsbasierten Versand ohne Prioritäten auf. Hier ist der Zufall bei Bestimmung des Backoff-Intervalls Hauptursache für die Differenzen. Bei den anderen beiden Versandarten kommen die Unterschiede durch unterschiedliche Zeitabstände zwischen der Generierung von Audiodaten und der nächsten wettbewerbsbasierten / wettbewerbsfreien Region zustande.

5.2.6. Veränderung des Verkehrsmusters

In einer weiteren Simulation wurden Auswirkungen auf die zur Verfügung stehende Übertragungsrate in wettbewerbsbasierten Slot-Regionen bei Veränderung des Ver-

a)



b)

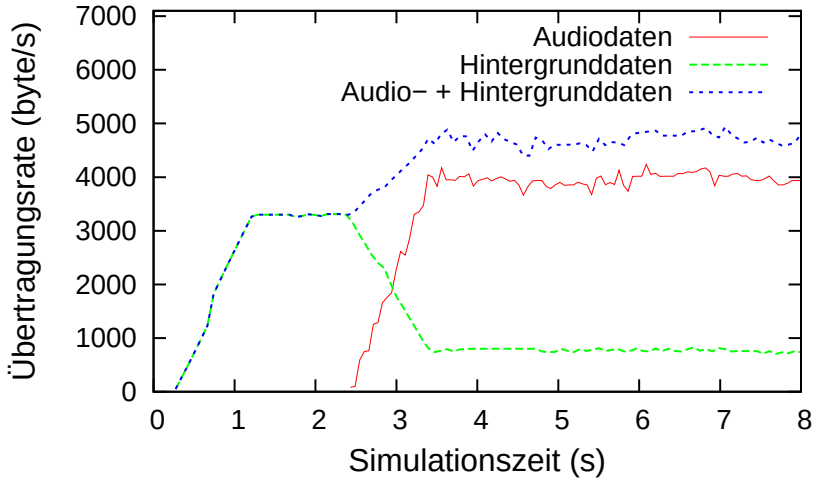


Abbildung 5.5.: Veränderung der Auslastung des Mediums bei wettbewerbsbasierter Übertragung mit Prioritäten. Die obere Grafik ist aus [BGK07] entnommen, die untere entstand nach der Restrukturierung.

kehrsaufkommens untersucht. Hierzu wurden zunächst ausschließlich Datenrahmen mit den in Kapitel 5.2.3 beschriebenen Prioritäten von der 2. Station gesendet. Nach 2,4 s Simulationszeit kam die Übertragung von Audiodaten hinzu, die ebenfalls mit Prioritäten aus Kapitel 5.2.3 gesendet wurden.

Der Verlauf der effektiv genutzten Übertragungsrate ist in Abbildung 5.5 dargestellt. Die Hintergrund-Daten können zu Beginn mit ihrer maximalen Übertragungsrate gesendet werden, welche durch eine theoretischen Obergrenze limitiert ist. Diese berechnet sich aus der Länge der wettbewerbsbasierten Region, der Größe eines Datenrahmens (und damit der Sendedauer), dem zu wartenden Inter-Frame-Spacing und Backoff-Intervall und einer Blindperiode, die durch das Wechseln des Transceivers vom Empfangs- in den Sendemodus entsteht. Da das Backoff-Intervall aufgrund

der Verwendung von Prioritäten konstant ist und aus 6 Pulse-Slots besteht, beträgt die Sendedauer t_{send} eines Datenrahmens mit 60 Byte:

$$\begin{aligned} t_{send} &= t_{ifs} + t_{BackoffInterval} + t_{BlindPeriodRX} + t_{Tx} \\ &= 2 \cdot t_{PulseLength} + 6 \cdot t_{PulseLength} + t_{BlindPeriodRX} + 60 \cdot t_{ByteTx} \\ &= 2 \cdot 200\,\mu\text{s} \; + 6 \cdot 200\,\mu\text{s} \; + 192\,\mu\text{s} \; + 60 \cdot \frac{8Bit}{250.000Bit/s} \\ &= 3,712\,ms \end{aligned}$$

Obwohl die wettbewerbsbasierten Regionen jeweils eine Dauer von 25 ms haben, wird nur in den ersten 20 ms eine neue Übertragung begonnen. Dies ist notwendig, weil eine Übertragung nach Übergabe eines Rahmens an den Transceiver nicht mehr abgebrochen werden kann und verhindert werden muss, dass eine Übertragung über die Grenze der wettbewerbsbasierten Region hinaus andauert. Dadurch werden bei einem Backoff-Intervall von 6 Pulse-Slots maximal $n = \lfloor \frac{20.000\,\mu\text{s}}{3.712\,\mu\text{s}} \rfloor = 5$ Datenrahmen mit eine Größe von 60 Byte pro wettbewerbsbasierter Region übertragen. Der Sendeversuch des 6. Rahmens wird während der Arbitrierung abgebrochen. Somit kann Station 2 in 13 wettbewerbsbasierten Regionen pro Sekunde maximal eine Nettodatenrate von $5 \cdot 13 s^{-1} \cdot 50\,Byte = 3.250\,Byte/s$ nutzen.

Mit Beginn der Übertragung von Audiodaten bricht die von Station 2 genutzte Übertragungsrate ein, da die Audiorahmen mit einer höheren Priorität übertragen werden. Damit ist es möglich, dass Audiodaten mit der benötigten Übertragungsrate von 4,1 kByte/s gesendet werden können. Ebenfalls wie in [BGK07] nimmt die genutzte Gesamtübertragungsrate zu, was sich durch die geringeren Contention-Windows in den Stationen 1 und 4 erklären lässt, die eine bessere Ausnutzung des Mediums erreichen. Dadurch haben die Stationen bei der Arbitrierungsphase eine geringere Wartezeit und es ist möglich, in einem 25 ms Block neben den Audiodaten zusätzlich Hintergrund-Daten zu übertragen.

In der folgenden Rechnung soll die maximal mögliche Gesamtübertragungsrate bei Übertragung von Audiodaten und Hintergrunddaten bestimmt werden. Zuerst wird betrachtet, wie lange die Übertragung eines Audiorahmens über zwei Hops andauert. Station 1 hat ein Contention-Window von [2,2] und Station 4 von [4,4], d.h. die Übertragungsdauer für den ersten bzw. zweiten Hop beträgt:

$$\begin{aligned} t_{send\_Hop1} &= 2 \cdot t_{PulseLength} + 4 \cdot t_{PulseLength} + t_{BlindPeriodRX} + 92 \cdot t_{ByteTx} \\ &= 2 \cdot 200\,\mu\text{s} \; + 4 \cdot 200\,\mu\text{s} \; + 192\,\mu\text{s} \; + 92 \cdot \frac{8Bit}{250.000Bit/s} \\ &= 4,336\,ms \\ t_{send\_Hop2} &= 2 \cdot t_{PulseLength} + 2 \cdot t_{PulseLength} + t_{BlindPeriodRX} + 92 \cdot t_{ByteTx} \\ &= 2 \cdot 200\,\mu\text{s} \; + 2 \cdot 200\,\mu\text{s} \; + 192\,\mu\text{s} \; + 92 \cdot \frac{8Bit}{250.000Bit/s} \\ &= 3,936\,ms \end{aligned}$$

Für die Übertragung eines Audiorahmens über 2 Hops werden demnach $4,336\,ms + 3,936\,ms = 8,272\,ms$ benötigt. Da pro Sekunde 25 Audiorahmen erzeugt werden,

aber nur 13 wettbewerbsbasierte Slot-Regionen pro Sekunde in einem Macro-Slot sind, werden in 12 wettbewerbsbasierten Slot-Regionen jeweils 2 Audiorahmen über 2 Hops gesendet und in einer wettbewerbsbasierten Slot-Region wird nur ein Audiorahmen über 2 Hops übertragen (eventuell wird ein Audiorahmen auch in einer wettbewerbsbasierten Slot-Region über den ersten Hop gesendet und in der nächsten wettbewerbsbasierten Slot-Region über den zweiten Hop; diese Möglichkeit ändert jedoch am folgenden Ergebnis nur wenig). Die Übertragungsdauer von 2 Rahmen über 2 Hops beträgt $8,272\,ms \cdot 2 = 16,544\,ms$, d.h. es bleiben von den $20\,ms$, in denen eine Übertragung in einer wettbewerbsbasierten Slot-Region beginnen kann, noch $3,456\,ms$ für die Übertragung weiterer Rahmen. Diese Zeit reicht um einen Rahmen mit Hintergrunddaten zu senden, denn für diesen werden $3,712\,ms$ benötigt (genauer betrachtet wird die $20\,ms$ Grenze während der Übertragung überschritten, allerdings befindet sich der Rahmen zu dem Zeitpunkt bereits bei dem Transceiver und der Sendevorgang kann nicht mehr abgebrochen werden).
Für die Slot-Region, in der nur ein Rahmen über 2 Hops gesendet wird, bleiben $20\,ms - 8,272\,ms = 11,728\,ms$ für weitere Rahmen. Diese Zeit reicht, um 3 Rahmen mit Hintergrunddaten zu versenden (in diesem Fall kann der Rahmen, der bei der Überschreitung der $20\,ms$ in Bearbeitung ist, abgebrochen werden). Insgesamt folgt damit, dass pro Sekunde 25 Audiorahmen über 2 Hops und 15 Rahmen mit Hintergrunddaten übertragen werden. Das ergibt folgende Netto-Gesamtübertragungsrate:

$$\begin{aligned} R_{Gesamt} &= 2 \cdot 25\,s^{-1} \cdot 82\,Byte + 13\,s^{-1} \cdot 50\,Byte \\ &= 4.750\,Byte/s \end{aligned}$$

Dieses Ergebnis stimmt mit dem in Abbildung 5.5 dargestellten Simulationsergebnis überein.

5.2.7. Vergleich der Ergebnisse

Die Ergebnisse zeigen zwar einige kleine Unterschiede zwischen den Simulationen aus [BGK07] und den Simulationen dieser Arbeit, sind aber dennoch vergleichbar. Alle Differenzen lassen sich durch Verwendung anderer Parameter des Ausbreitungsmodells bzw. durch Zufall bei der Bestimmung des Backoff-Intervalls erklären. Dies zeigt, dass die Performanz durch die Restrukturierung von MacZ – zumindest für die betrachteten Szenarien – nicht beeinträchtigt wurde. Ob sich der Rechen- oder Speicheraufwand verändert hat, lässt sich durch die Simulationen nicht zeigen, da der Simulator diese Größen bei der simulierten Hardware nicht beachtet und für die Bearbeitungszeit einzelner Transitionen Nullzeit annimmt.

Rahmenlänge in Byte	benötigte Micro-Slots pro Paket	mögliche Rahmen pro Macro-Slot
10-15	1	999
16-46	2	499
47-77	3	333
78-108	4	249
109-121	5	199

Tabelle 5.2.: Anzahl benötigter Micro-Slots in Abhängigkeit der Rahmenlänge und Anzahl möglicher Rahmen pro Macro-Slot bei gegebener Rahmenlänge.

5.3. Simulations-Reihe 2

Die zweite Simulations-Reihe wiederholt die Simulationen aus [Bec06], welche insgesamt aus zwei Szenarien bestehen. Die Simulationen wurden ursprünglich nach der Spezifikation des Service-Layers durchgeführt, um die Funktionalität des Service-Layers zu überprüfen.

5.3.1. Topologie

In dem ersten Szenario besteht die Topologie aus zwei Knoten, ein Sender und ein Empfänger. Dadurch kann es zu keinen Kollisionen kommen. Zusätzlich sind die beiden Stationen so dicht beieinander, dass Rahmenverluste (zumindest in der Simulation) nicht auftreten. Bei dieser Topologie gibt es somit für die Simulationen keine Grundlage für Indeterminismus. Daher sollten die Simulationsergebnisse exakt mit denen aus [Bec06] übereinstimmen.

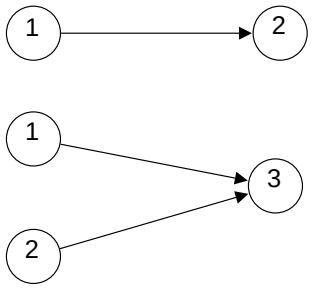


Abbildung 5.6.: Topologien

In der zweiten Topologie kommt im Vergleich zur ersten Topologie ein weiterer Sendeknoten hinzu, so dass hier zwei Sender und ein Empfänger ein Netzwerk bilden. Hier kann es also durch wettbewerbsbasierten Versand zu Kollisionen kommen.

Die beiden Topologien sind in Abbildung 5.6 abgebildet.

5.3.2. Aufteilung des Macro-Slots und Versandarten

Die Aufteilung des Macro-Slots wird in relativ kleine Micro-Slots vorgenommen. Ein einzelner Micro-Slot hat lediglich eine Länge von 1 ms, ein Macro-Slot hat weiterhin eine Länge von 1 s. Lediglich 1 ms des Macro-Slots wird für Synchronisation benötigt, womit in einem Macro-Slot 999 nutzbare Micro-Slots enthalten sind. Die Synchronisationsphase konnte in diesem Fall sehr kurz gehalten werden, da aufgrund der Topologie der Netzdurchmesser bekannt ist und nur ein Hop beträgt. Dadurch war es nicht nötig, die Möglichkeit der Multi-Hop-Synchronisation von *BBS* zu verwenden.

- Die Micro-Slots werden für Szenario 1 komplett für den reservierungsbasierten Versand verwendet. Es können somit 999 Micro-Slots reserviert werden. Durch die Simulationen wird untersucht, wie der Durchsatz bzw. die Mediumauslastung von der Länge eines Rahmens und damit von der Anzahl an benötigten zusammenhängenden Micro-Slots abhängt. Tabelle 5.2 zeigt den rechnerischen Zusammenhang zwischen Rahmengröße und Anzahl nötiger Micro-Slots. Es ist zu erkennen, dass bei einem reservierten Micro-Slot nur wenige Bytes gesendet werden können. Dies liegt an der Blindperiode, die vor der Übertragung des ersten Bytes als Overhead durch das Umschalten des Transceivers vom Empfangs- in den Sendemodus zustande kommt, und wurde bereits in Kapitel 5.2.6 angesprochen. Zusätzlich muss noch eine weitere Blindperiode bei dem Umschalten vom Sende- in den Empfangsmodus beachtet werden, da sichergestellt sein muss, dass ein Sender nach erfolgreicher Übertragung im folgenden Micro-Slot wieder empfangsbereit ist. Bei dem CC2420 benötigen die Blindperioden zusammen 512 μs [Tex], was 51,2% eines Micro-Slots entspricht. Da die Wartezeiten von insgesamt 512 μs unabhängig von der Rahmenlänge sind, kann durch Senden über mehrere Micro-Slots der Durchsatz gesteigert werden, weil die Wartezeit einen immer geringeren Anteil an der Übertragung ausmacht.

- Im zweiten Szenario werden Rahmen ausschließlich wettbewerbsbasiert übertragen. Pro Macro-Slot gibt es eine wettbewerbsbasierte Slot-Region, die 499 Micro-Slots umfasst. Die restlichen 500 nutzbaren Micro-Slots werden nicht verwendet. Die beiden Sendestationen versuchen während der kompletten wettbewerbsbasierten Slot-Region, Rahmen mit einem Contention-Window zwischen 0 und 5 zu übertragen. Durch dieses kleine Contention-Window ist die Wahrscheinlichkeit für Kollisionen relativ hoch.

Die Simulationen für die Diplomarbeit wurden im Oktober 2006 durchgeführt. Seit dieser Zeit wurden bereits vor dieser Arbeit viele Parameter von *MacZ* optimiert, wodurch die Performanz verbessert werden konnte. Insbesondere wurden die Interframe-Spacings auf ein Minimum verkleinert. In [Bec06] waren diese Parameter relativ groß gewählt, da durch die Simulationen nicht die Performanz von *MacZ*, sondern das Verhalten des Service-Layers überprüft werden sollte. Im Konkreten heißt dies, dass in [Bec06] das Interframe-Spacing vor der Übertragung eines Datenrahmens 8 Pulse-Slots beträgt und aktuell nur 2 Pulse-Slots. Damit die Ergebnisse des zweiten Szenarios vergleichbar sind, wurden die Parameter für das zweite Szenario zurück auf ihre Werte von Oktober 2006 gesetzt. Um die Performanzsteigerung durch die Änderungen zu verdeutlichen, wurde das zweite Szenario ebenfalls mit aktuellen Parametern durchgeführt.

5.3.3. Vergleich der Ergebnisse

In Abbildung 5.7 ist das Ergebnis des ersten Szenarios abgebildet. Es ist zu erkennen, dass die genutzte Übertragungsrate mit der Größe der Rahmen steigt, aber es an den Stellen, an denen ein weiterer Micro-Slot benötigt wird, zu einem Einbruch der Übertragungsrate kommt. Der grundsätzliche Anstieg der Übertragungsrate lässt sich zum einen dadurch erklären, dass der Overhead durch die Umschaltzeiten des

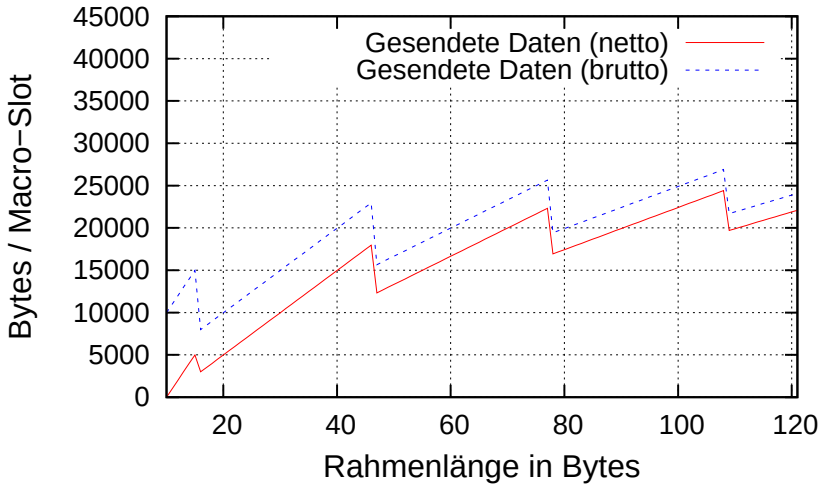


Abbildung 5.7.: Genutzte Übertragungsrate in Abhängigkeit der Rahmengrößen.

Transceivers bei steigender Rahmengröße in Relation zu der Sendezeit geringer wird. Auch die Relation von gesendeten Nutz- zu Gesamtdaten wird bei steigender Rahmengröße höher. Die Einbrüche der Raten bei Hinzunahme eines weiteren Micro-Slots kommen durch die ungenutzte Zeit im letzten reservierten Micro-Slot zustande. Die veränderten Parameter haben hier keine Auswirkungen auf die Ergebnisse, da sie nur bei wettbewerbsbasiertem Versand Einfluss nehmen. Deshalb konnten die Simulationen für das erste Szenario mit den neuen Parametern durchgeführt werden. Das Ergebnis stimmt exakt mit dem Ergebnis aus [Bec06] überein, da aufgrund der Eigenschaften des Netzwerks und der reservierungsbasierten Übertragung kein Indeterminismus auftreten kann. Aus diesem Grund wurde auf die Abbildung aus [Bec06] verzichtet.

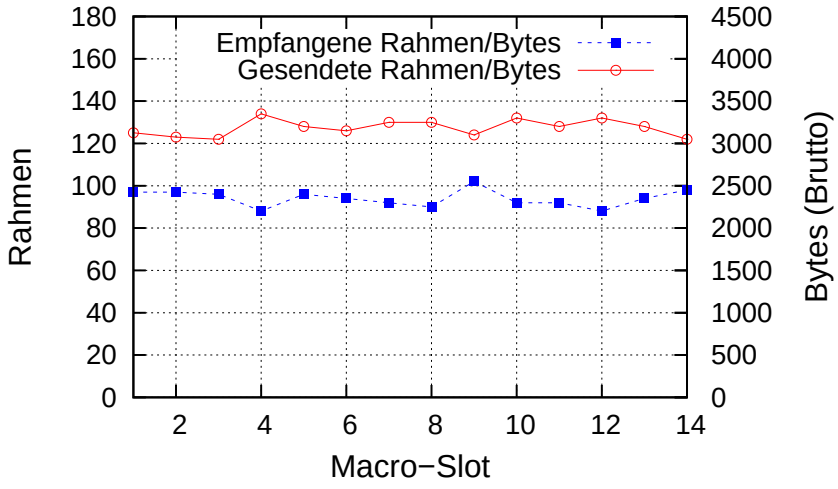


Abbildung 5.8.: Gesendete und empfangene Rahmen pro Macro-Slot und die Übertragungsrate in Anzahl an Bytes pro Macro-Slot mit den Parametern aus [Bec06]

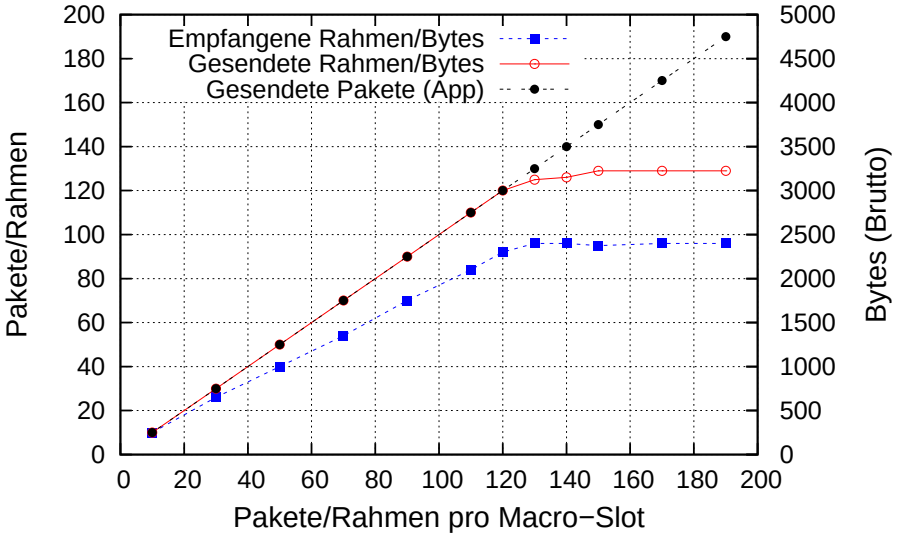


Abbildung 5.9.: Verhältnis zwischen gesendeten und empfangenen Rahmen/Paketen pro Macro-Slot. Die Abbildung zeigt ebenfalls die Übertragungsrate in Bytes pro Macro-Slot, die abhängig von der Anzahl an Rahmen/Paketen pro Macro-Slot ist.

Abbildung 5.8 stellt Ergebnisse des zweiten Szenarios dar. Hier spielt wieder der Indeterminismus bei der Bestimmung der Zufallszahlen für das Backoff-Intervall eine große Rolle. Wegen dem kleinen Contention-Window ist die Wahrscheinlichkeit für Kollisionen relativ hoch, so dass nur ca. 75% der Rahmen korrekt empfangen werden. Die Ergebnisse aus [Bec06] weichen zwar minimal von denen dieser Simulation aufgrund des Indeterminismus ab, zeigen aber keine grundsätzlichen Unterschiede. Aus diesem Grund wurde ebenfalls auf die Abbildung aus [Bec06] verzichtet.

Abbildung 5.9 zeigt für das zweite Szenario den Zusammenhang zwischen gesendeten und empfangenen Rahmen bei steigender Paketanzahl pro Macro-Slot. Wenn nicht mehr als 120 Pakete pro Macro-Slot versendet werden, kann jedes Paket als Rahmen verschickt werden. Falls die Netzwerkebenen mehr Pakete senden möchten, reicht die Dauer eines Macro-Slots nicht aus, um alle Pakete zu übertragen. Maximal können durch die MAC-Schichten knapp 130 Rahmen pro Macro-Slot versendet werden.
Bei genauer Betrachtung der Verlustraten fällt auf, dass die Verlustrate bei steigender Rahmenanzahl pro Macro-Slot leicht zunimmt. Zum Beispiel gelangen bei 50 erfolgreich versendeten Rahmen pro Macro-Slot 40 zu dem Empfänger. Das entspricht einer Erfolgsquote von $40/50 = 80\%$. Bei 170 Paketen können 130 Rahmen erfolgreich versendet werden und es gelangen ca. 97 Rahmen zu dem Empfänger. Dies entspricht einer geringeren Erfolgsquote von $97/130 = 75\%$. Die Erklärung hierfür liegt darin, dass bei wenigen Paketen nur zu Beginn eines Macro-Slots Wettbewerb stattfindet. Wenn ein Sender alle in Auftrag gegebenen Rahmen versendet hat, überträgt nur noch der andere Sender. Bei steigender Paketanzahl findet immer länger ein direkter Wettbewerb durch parallele Arbitrierungsvorgänge statt und die Wahrscheinlichkeit der Kollisionen nimmt zu. Sobald beide Sendeknoten über die komplette Dauer eines Macro-Slots versuchen, Rahmen zu übertragen, beträgt die

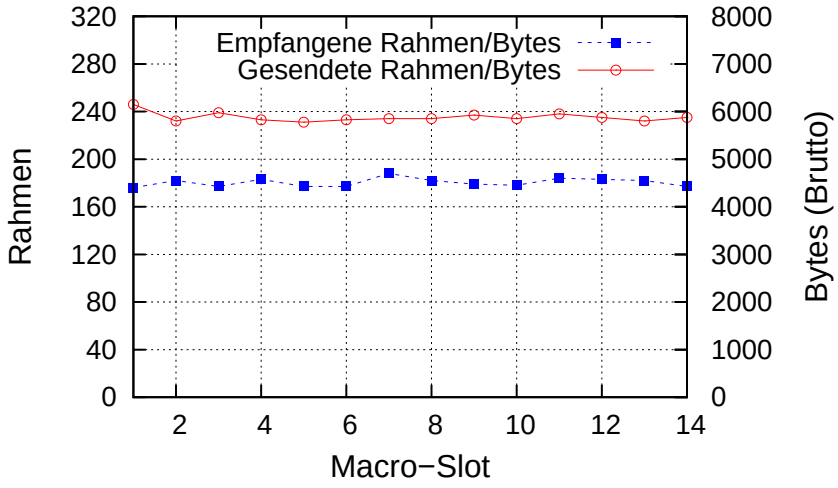


Abbildung 5.10.: Gesendete und empfangene Rahmen pro Macro-Slot mit aktuellen Interframe-Spacing-Parameter

Verlustrate wie in den vorherigen Simulationen ca. 25%.
Das Ergebnis stimmt mit dem Ergebnis aus [Bec06] überein, weshalb auch hier auf die Abbildung aus [Bec06] verzichtet wurde.

5.3.4. Ergebnisse mit optimiertem Parameter

Durch Optimierung der Interframe-Spacings konnten Wartezeiten vor dem Beginn einer Übertragung reduziert werden, ohne dass die Wahrscheinlichkeit von Kollisionen zunimmt. In der Regel treten Kollisionen zwischen Stationen, die mit der Arbitrierung gleichzeitig beginnen und in gegenseitiger Sendereichweite sind, nur auf, wenn beide Stationen das gleiche Backoff-Intervall wählen oder wenn eine Belegung des Mediums nicht oder zu spät erkannt wird, weil z.B. das Signal zu schwach ist.

In Abbildung 5.10 sind die Ergebnisse des zweiten Szenarios bei aktueller Konfiguration dargestellt. Aufgrund des kürzeren Interframe-Spacings können deutlich mehr Rahmen versendet werden. Statt durchschnittlich 120 Rahmen können nach Änderung 240 Rahmen pro Macro-Slot übertragen werden. Das unveränderte Verhältnis zwischen gesendeten und empfangenen Rahmen zeigt, dass die Kollisionswahrscheinlichkeit durch das geringere Interframe-Spacings nicht angestiegen ist.

5.4. Simulations-Reihe 3

Die bisherigen Simulations-Reihen haben von der Möglichkeit der Rahmenübertragung mit vorheriger RTS/CTS-Sequenz keinen Gebrauch gemacht. Aus diesem Grund wurde in einer dritten Simulations-Reihe ein Szenario simuliert, das von dem RTS/CTS-Handshake-Verfahren profitiert.

5.4.1. Topologie

Die Topologie, dargestellt in Abbildung 5.11, ist ein klassisches Beispiel für das Hidden-Station-Problem [KR05, Kar90]. Sie besteht aus drei Stationen, von denen zwei Stationen (1 und 2) reine Sender sind und Rahmen an Station 3 senden. Die Knoten sind so angeordnet, dass der Empfänger in Sendereichweite beider Sender ist und es zu Kollisionen kommt, wenn beide Sender gleichzeitig übertragen. In diesem Fall kann der *CSMA-CA*-Mechanismus keine Kollisionen verhindern, da die Signalstärke der Übertragungen bei den Sendern so schwach ist, dass eine Mediumbelegung nicht erkannt wird. Das RTS/CTS-Verfahren kann sich hier auszeichnen, da ein CTS-Rahmen, der von Station 2 als Antwort auf einen RTS-Rahmen gesendet wird, Station 1 und 3 erreicht und über die anstehende Belegung des Mediums informiert (siehe auch Kapitel 2.3.1).

Abbildung 5.11.: Topologie

Es besteht zwar weiterhin die Gefahr, dass ein RTS-Rahmen mit einem zweiten RTS-Rahmen oder einem CTS-Rahmen kollidiert [LG97], aber die Wahrscheinlichkeit einer solchen Kollision ist in der Regel deutlich geringer, da RTS/CTS-Rahmen sehr klein sind und CTS-Rahmen ohne Backoff-Intervall gesendet werden. In *MacZ* beträgt die Größe für RTS- und CTS-Rahmen genau 11 Bytes, während ein Datenrahmen zwischen 10 Bytes und 121 Bytes groß ist. Daran erkennt man, dass RTS/CTS-Handshakes bei sehr kleinen Datenrahmen keine Verbesserungen bringen, wenn zusätzlich ein Bestätigungsmechanismus (ACKs) verwendet wird, und nur bei Datenrahmen mit vielen Nutzdaten sinnvoll sind. Denn bei großen Datenrahmen kostet eine Kollision zwischen RTS- und/oder CTS-Rahmen weniger als zwischen zwei Datenrahmen, da die „verschwendete“ Übertragungszeit gering ist [Kar90].

5.4.2. Aufteilung des Macro-Slots und Versandarten

Ein Macro-Slot wird in 495 Micro-Slots unterteilt mit je 2 ms Länge. Mit einer Synchronisationsphase der Länge 10 ms hat ein Macro-Slot damit eine Länge von 1 s. Pro Macro-Slot gibt es 10 Slot-Regionen für den wettbewerbsbasierten Versand, die jeweils eine Länge von 25 Micro-Slots bzw. 50 ms haben. Die Slot-Regionen werden gleichmäßig auf den Macro-Slot aufgeteilt, so dass alle 100 ms eine wettbewerbsbasierte Region startet. RTS/CTS-Rahmen werden nur bei wettbewerbsbasiertem Senden benötigt, da bei wettbewerbsfreiem Senden das Medium ohnehin reserviert ist. Abbildung 5.12 zeigt einen Macro-Slot mit der beschriebenen Einteilung.

In einer ersten Simulation werden Rahmen ohne vorheriges RTS/CTS gesendet; in einer zweiten Simulation sollen die Auswirkungen bei Übertragung mit dem

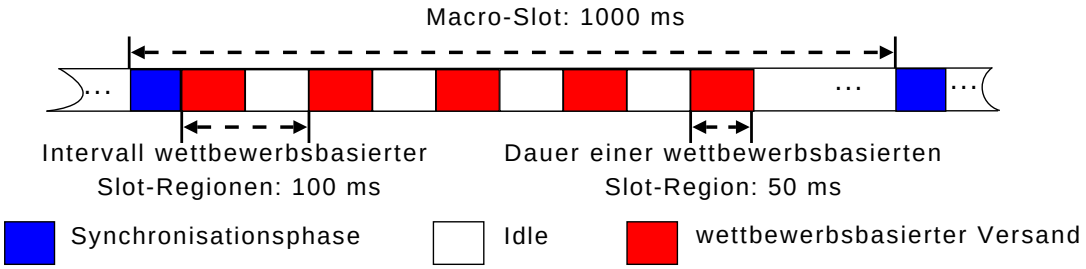


Abbildung 5.12.: Macro-Slot mit Einteilung in virtuelle Slot-Regionen für Simulations-Reihe 3

RTS/CTS-Mechanismus betrachtet werden. Die Applikation gibt alle 100 ms jeweils drei 40 Bytes umfassende Pakete an *MacZ*. Zusammen mit 10 Byte MAC-Header werden somit 50 Byte pro Rahmen übertragen. Untersucht wird die effektiv genutzte Übertragungsrate, d.h. die empfangenen Daten bei Station 3. Im Idealfall, wenn alle gesendeten Daten auch empfangen werden, beträgt die genutzte Übertragungsrate auf Applikationsebene 19,2 $kBit/s$ bzw. 2,4 $kByte/s$ (60 Pakete mit je 40 Bytes pro Sekunde). Beide Sender haben ein Contention-Window von [0,12], was einen großen Spielraum für das Backoff-Intervall zulässt.

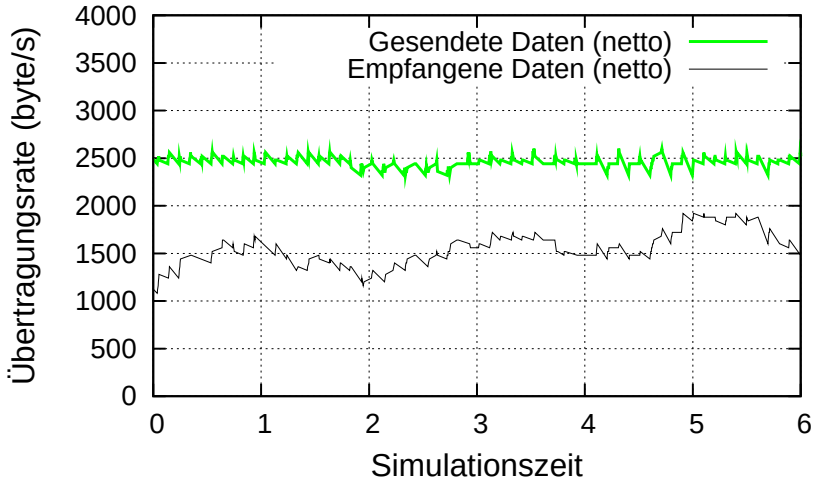
5.4.3. Ergebnisse

Abbildung 5.13 stellt die Ergebnisse der Simulationen dar. Bei der Übertragung ohne RTS/CTS (Abbildung 5.13 (a)) treten sehr viele Kollisionen auf und ca. 40% aller gesendeten Rahmen können nicht korrekt empfangen werden. Da der CSMA-Mechanismus in der Hidden-Station-Topologie komplett versagt, kommt es auch zu Kollisionen, wenn die beiden Sendestationen unterschiedliche Backoff-Intervalle wählen. Wenn die beiden Stationen in Sendereichweite wären, treten Kollisionen bei korrekter und schneller Erkennung der Mediumbelegung nur bei gleichem Backoff-Intervall auf, d.h. mit einer Wahrscheinlichkeit von 7,7% (1/13, da das Contention-Window 13 Werte umfasst).

Teil (b) von Abbildung 5.13 zeigt das Ergebnis bei Verwendung von RTS/CTS-Rahmen. Die Anzahl der Kollisionen geht deutlich zurück: Fast jeder gesendete Datenrahmen kann von dem Empfänger empfangen werden. Die geringen Abweichungen zwischen Sende- und Empfangsrate lassen sich auf Rahmenverluste aufgrund der Entfernung zwischen Sendern und Empfänger zurückführen. Das macht erkennbar, dass das RTS/CTS-Verfahren seinen Zweck erfüllt und das *Hidden Station*-Problem eingrenzt, denn Station 1 bzw. Station 2 wird mit einem CTS-Rahmen von Station 3 über Übertragungen der jeweils anderen Station informiert und wartet mit ihrem Übertragungsversuch.

Allerdings ist der Overhead durch RTS/CTS-Rahmen sehr groß, denn die zusätzliche Zeit, die für die RTS/CTS-Sequenz benötigt wird, geht für die Übertragung von Datenrahmen verloren. In diesen Simulationen ist die benötigte Übertragungsrate weit unter der verfügbaren, so dass dieser Nachteil nicht auffällt, weil die zusätzliche Kapazität für den RTS/CTS-Mechanismus verfügbar ist. Pro wettbewerbsbasierter

(a)



(b)

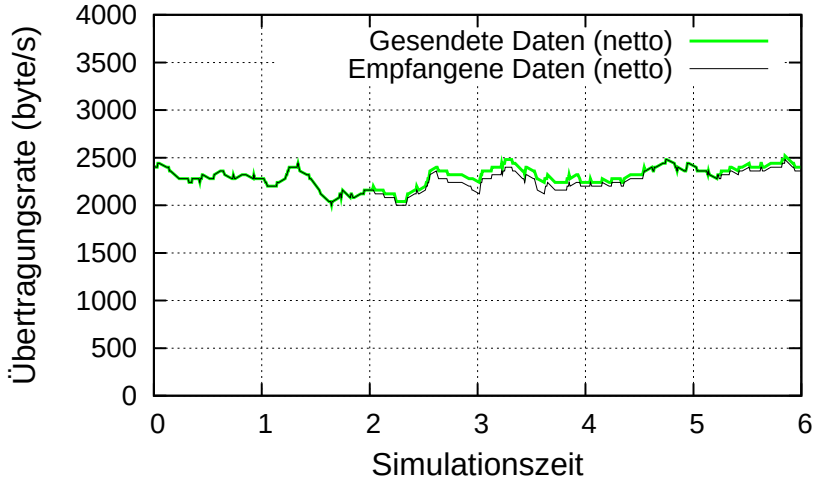


Abbildung 5.13.: Effektive Übertragungsrate in Abhängigkeit der Simulationszeit. In (a) ohne RTS/CTS-Sequenz vor der Datenübertragung und in (b) mit RTS/CTS vor Datenübertragung.

Slot-Region werden im Durchschnitt pro Sende-Station nur 3 Rahmen mit je einer Größe von 50 Byte gesendet.
Zur Verdeutlichung, dass in diesem Szenario die Kapazität einen RTS/CTS-Mechanismus erlaubt, wird im Folgenden die Zeit $t_{RtsCtsData}$ zur Übertragung eines Rahmens der Größe 50 Byte mit RTS/CTS-Reservierung bei einem durchschnittlichen Backoff-Intervall von 6,5 (Contention-Window ist [0,12]) berechnet. Dabei betragen die Interframe-Spacings bei RTS-Rahmen 2 Pulse-Slots, bei CTS-Rahmen 1 Pulse-Slot und bei Daten nach einer RTS/CTS-Sequenz ebenfalls 1 Pulse-Slot. Bei CTS- und Datenrahmen nach einer RTS/CTS-Sequenz wird kein Backoff-Intervall gewartet.

$$
\begin{aligned}
t_{RtsCtsData} &= t_{RTS} + t_{CTS} + t_{Data} \\
t_{RTS} &= t_{ifs\_RTS} + t_{BackoffInterval\_RTS} + t_{BlindPeriodRx} + t_{Tx\_RTS} \\
&= 2 \cdot 200\,\mu\text{s} \; + 6,5 \cdot 200\,\mu\text{s} \; + 192\,\mu\text{s} \; + 11 \cdot \frac{8Bit}{250.000Bit/s} \\
&= 2.244\,\mu\text{s} \\
t_{CTS} &= t_{ifs\_CTS} + t_{BlindPeriodRx} + t_{Tx\_CTS} \\
&= 1 \cdot 200\,\mu\text{s} \; + 192\,\mu\text{s} \; + 11 \cdot \frac{8Bit}{250.000Bit/s} \\
&= 744\,\mu\text{s} \\
t_{Data} &= t_{ifs\_Data} + t_{BlindPeriodRx} + t_{Tx\_Data} \\
&= 1 \cdot 200\,\mu\text{s} \; + 192\,\mu\text{s} \; + 50 \cdot \frac{8Bit}{250.000Bit/s} \\
&= 1.992\,\mu\text{s} \\
t_{RtsCtsData} &= 2244\,\mu\text{s} \; + 744\,\mu\text{s} \; + 1992\,\mu\text{s} \\
&= 4.980\,\mu\text{s}
\end{aligned}
$$

Pro wettbewerbsbasierter Slot-Region werden insgesamt 6 Datenrahmen mit vorherigem RTS/CTS versendet, d.h. in der Summe ergibt sich (ohne Beachtung von evtl. Neuübertragungen wegen Kollisionen zwischen RTS/CTS-Rahmen) eine benötigte Zeit von $6 \cdot 4980\,\mu\text{s} \; = 29,88\,ms$. Da eine wettbewerbsbasierte Slot-Region 50 ms andauert, kann in diesem Szenario auch mit RTS/CTS-Reservierung im Durchschnitt jeder Datenrahmen versendet werden. Mit einem Spielraum von ca. $50\,ms - 30\,ms = 20\,ms$ sind auch Neuübertragungen aufgrund von kollidierenden RTS/CTS-Rahmen zeitlich möglich. Allgemein verdeutlicht die Rechnung aber, dass die effektiv nutzbare Übertragungsrate mit dem RTS/CTS-Mechanismus abnimmt. Denn ohne RTS/CTS-Sequenz beträgt die durchschnittliche Zeit zur Übertragung eines 50 Byte großen Rahmens:

$$
\begin{aligned}
t_{DataSolo} &= t_{ifs\_DataSolo} + t_{BackoffInterval\_DataSolo} + t_{BlindPeriodRx} + t_{Tx\_DataSolo} \\
&= 2 \cdot 200\,\mu\text{s} \; + 6,5 \cdot 200\,\mu\text{s} \; + 192\,\mu\text{s} \; + 50 \cdot \frac{8Bit}{250.000Bit/s} \\
&= 3.492\,\mu\text{s}
\end{aligned}
$$

D.h. der Overhead durch den RTS/CTS-Mechanismus beträgt ca. $1.500\,\mu\text{s}$, was im Vergleich zu der Übertragung ohne den Mechanismus 43% Overhead entspricht. Anders formuliert, gehen in diesem Szenario bei 50 Byte großen Datenrahmen ca. 30% der Übertragungsrate für den RTS/CTS-Mechanismus verloren. In Abbildung 5.13 ist diese drastische Unterschied nicht erkennbar, weil die vorhandene Übertragungsrate höher ist als die benötigte.

Der Vergleich zwischen beiden Abbildungen 5.13 zeigt einen weiteren Nachteil von RTS/CTS-Sequenzen auf: Wenn keine RTS/CTS-Rahmen verwendet werden, ist die

Menge der gesendeten Daten nahezu konstant und bewegt sich um 2,4 kByte/s. Bei Verwendung von RTS/CTS-Rahmen ist die Senderate variabler und bewegt sich zwischen 2,0 kByte/s und 2,5 kByte/s, wobei sie als Mittelwert weiterhin 2,4 kByte/s hat. Dieses Verhalten liegt an der Verzögerung, die durch Kollisionen zwischen zwei RTS-Rahmen auftreten kann. Wenn zwei RTS-Rahmen bei dem Empfänger kollidieren, erhalten die Sender kein CTS und beginnen nach einem Timeout einen neuen Arbitrierungsversuch. Dadurch verzögert sich natürlich der Datenrahmen, der mit dem RTS-Rahmen angekündigt werden sollte. Es kommt zu großen Schwankungen im Zwischenankunftsintervall, die nicht mehr nur von dem gewählten Backoff-Intervall abhängen, sondern auch von möglichen Kollisionen von RTS-Rahmen, die für eine nötige Wiederholung des Arbitrierungsvorgangs sorgen. Diese Verzögerungsschwankungen haben nun Auswirkungen auf die Übertragungsrate, da diese die Anzahl an übertragenen Rahmen bzw. Bytes in einem Zeitfenster von 1 s betrachtet.

Mit der letzten Simulations-Reihe wurde gezeigt, dass der RTS/CTS-Mechanismus von *MacZ* funktionstüchtig ist und die Wahrscheinlichkeit einer Kollision von Datenrahmen verringert. Kollisionen können nicht ganz verhindert werden, da zum Beispiel durch Topologieänderungen Knoten in Sendereichweite kommen können, die weder einen RTS-Rahmen noch einen CTS-Rahmen gehört haben. Die Ergebnisse (bzw. genauer gesagt, die variable Senderate/-verzögerung) weisen ebenfalls darauf hin, dass der RTS/CTS-Mechanismus für Echtzeitszenarien, wie zum Beispiel die Übertragung von Sprache, weniger geeignet ist, da die Verzögerungsschwankungen sehr hoch sein können. Allerdings ist die Übertragung ohne RTS/CTS-Rahmen auch keine Lösung für Echtzeitszenarien, da die Verlustrate bei einer Hidden-Station-Topologie viel zu hoch ist. Die beste Lösung für Echtzeitszenarien ist ein reservierungsbasierter Versand, weil bei disjunkten Reservierungen und statischer Netzwerktopologie keine Kollisionen auftreten. Durch *gute* Reservierungen kann man die Verzögerungen und Schwankungen sogar auf ein Minimum reduzieren. Die Reservierungen müssen sowohl einen Hop um den Sender gültig sein, als auch einen Hop um den Empfänger. Eine wettbewerbsbasierte Übertragung *mit* Prioritäten grenzt zwar zusammen mit dem RTS/CTS-Mechanismus das Problem der Kollisionen von Datenrahmen ebenfalls ein, verhindert aber nur bedingt Verzögerungsschwankungen, weil die Prioritäten erst bei Beginn einer Arbitrierungsphase Auswirkungen haben und diese erst beginnen kann, wenn das Medium frei ist bzw. die durch RTS/CTS angekündigte Belegung des Mediums vorüber ist.

6. Zusammenfassung und Ausblick

In dieser Bachelorarbeit wurde *MacZ*, ein MAC-Layer für Ad-Hoc-Netzwerke mit Dienstgüteunterstützung, mit dem mikroprotokollbasierten Ansatz restrukturiert und anschließend erfolgreich evaluiert. Durch die Restrukturierung sollte die Struktur von *MacZ* übersichtlicher werden und der Aufbau modularer, was den unabhängigen Austausch einzelner Komponenten ermöglicht. Mit dem mikroprotokollbasierten Ansatz wurden problemorientierte Komponenten gefunden, die jeweils eine einzige verteilte Protokollfunktionalität realisieren. Neben einigen Mikroprotokollen konnte eine weitere Komponente identifiziert werden, die die Eigenschaften der Verteiltheit nicht erfüllt, aber nach objektorientierten Grundsätzen gekapselt wurde.

Die Restrukturierung beschränkte sich zum größten Teil auf den Service-Layer von *MacZ*, der Schnittstelle zwischen MAC-Schicht und Netzwerk-Schicht.

- Der Basic-Layer, der für eine Synchronisation zwischen den Knoten und die Einteilung des Mediums in Macro-/Micro-Slots zuständig ist, wurde bereits in [KF05] auf Basis von Mikroprotokollen strukturiert, so dass in dieser Arbeit nur wenig verändert wurde. Die Veränderungen betrafen ausschließlich die Identifikation eines neuen Mikroprotokolls für das De-(Kodieren) von Black Bursts.

- Im Service-Layer wurden größere Änderungen durchgeführt. Es konnten vier symmetrische Mikroprotokolle und eine weitere Komponente identifiziert werden, die in SDL-Packages gekapselt wurden. Eines der Mikroprotokolle verkörpert die Funktionalität einer Übertragung bei wettbewerbsbasiertem Zugriff auf das Medium, ein weiteres ist für die Übertragung bei wettbewerbsfreiem Zugriff zuständig. Die anderen beiden Mikroprotokolle haben die Aufgaben, das Medium in Slot-Regionen auf Basis von Micro-Slots einzuteilen und zwischen Service-Layer auf der einen Seite und Basic-Layer bzw. einem weiteren Multiplexer auf der anderen Seite zu (de-)multiplexen. Dabei wurde der Multiplexer erweitert und ist nun zustandsbehaftet, damit eine gezielte Weiterleitung an die reservierungsbasierte oder wettbewerbsbasierte Sende-Komponente möglich ist. Die Komponente, welche die Definition eines Mikroprotokolls nicht erfüllt, ist für die Zwischenspeicherung von Paketen zuständig.

Für die Dokumentation der bereits identifizierten und neu identifizierten Mikroprotokolle wurde die Dokumentationsfunktion von *ConTraST* erprobt und eingesetzt. Das Werkzeug extrahiert Annotationen an der SDL-Spezifikation und erstellt hieraus

automatisch eine LaTeX-Datei, die das relevante Verhalten der Komponente nach außen zeigt. Obwohl sich die Dokumentationsfunktion von *ConTraST* noch in der Erprobungsphase befindet und nicht bei allen SDL-Konstrukten die LaTeX-Datei ohne Fehler erzeugt, sind das Potential und die Vorteile der toolgestützten Dokumentation deutlich erkennbar, denn bei Etablierung einer einheitlichen Dokumentation lassen sich Mikroprotokolle auf einer höheren Abstraktionsebene und ohne Blick in die konkrete SDL-Spezifikation auswählen und verwenden.

Zur Evaluation von *MacZ* wurden frühere Simulationen wiederholt und die Ergebnisse verglichen. Es konnte gezeigt werden, dass sich das funktionale Verhalten und die Performanz durch die Restrukturierung nicht verändert haben und kleinere Abweichungen von den früheren Ergebnissen auf Änderungen in den Parametern von *MacZ* und dem Ausbreitungsmodell zurückzuführen sind. Zum Testen von Übertragungen mit vorheriger RTS/CTS-Sequenz wurde ein neues Szenario erstellt und simuliert. Dadurch wurde gezeigt, dass auch die RTS/CTS-Funktionalität wie erwartet arbeitet.

In einer derzeitigen Arbeit wird ein Reservierungsprotokoll entworfen, welches auf Basis der reservierbaren Micro-Slots dynamisch während der Laufzeit Reservierungen durchführt. Für die simulierten Szenarien wurden die Reservierungen offline vorab vorgenommen, was in einem Ad-Hoc-Netzwerk in der Regel nicht möglich ist. Damit verbunden soll in zukünftigen Arbeiten eine Möglichkeit gefunden werden, die Einteilung eines Macro-Slots in Slot-Regionen dynamisch vorzunehmen. Dadurch wird es möglich sein, auf Änderungen der Anforderungen zu reagieren und zum Beispiel mehr reservierbare Micro-Slots bereitzustellen, falls die verfügbaren Slots reserviert sind.

Des Weiteren soll durch zukünftige Arbeiten optional das Anfordern einer Empfangsbestätigung (ACK) möglich sein. Dadurch sollen Rahmenverluste bereits auf MAC-Ebene erkannt werden und es kann eine schnellstmögliche Retransmission erfolgen. Empfangsbestätigungen auf MAC-Ebene sind auch in anderen Protokollen zu finden, wie zum Beispiel WLAN [IEEE99, IEEE05].

Zur Zeit wird auch das Signalisieren von Konflikten bei Vermischen zweier Netze in SDL spezifiziert. Dazu soll ein Knoten nach erkannter Synchronisationsphase eines Fremdnetzes einen Jamming-Frame senden, der von anderen Knoten weitergeleitet wird [GK08], damit anschließend eine gemeinsame Synchronisation beider Netze stattfinden kann.

In Zukunft wird *MacZ* auch für die *Imote2*-Plattform von Crossbow-Technology zur Verfügung stehen [Croa]. Der *Imote2* ist ein Sensorknoten mit deutlich mehr Ressourcen als der *MicaZ*, für welchen *MacZ* in der Vergangenheit primär entwickelt wurde. Die plattformunabhängige Spezifikation von *MacZ* in SDL erlaubt das einfache Portieren auf die *Imote2*-Plattform. Plattformspezifische Parameter, wie zum Beispiel die Blind-Periode beim Wechsel des Transceivers vom Sende- in den Empfangsmodus, wurden bei der Spezifikation von *MacZ* in Konfigurationsparameter ausgelagert. Allerdings ist der *Imote2* auch mit dem CC2420-Transceiver von Texas Instruments ausgestattet [Tex], so dass viele Parameter nicht geändert werden müssen.

A. Mikroprotokoll-Dokumentation

Im Folgenden ist exemplarisch die Mikroprotokoll-Dokumentation zur wettbewerbsfreien Übertragung herausgegriffen. Die Dokumentation wurde automatisch aus der SDL-Spezifikation mit *ConTraST* generiert, wobei kleinere Änderungen manuell durchgeführt wurden, da sich die Dokumentationsfunktion von *ConTraST* zur Zeit in der Erprobungsphase befindet und nicht bei allen SDL-Konstrukten fehlerfrei arbeitet.

Name: ResTxRx

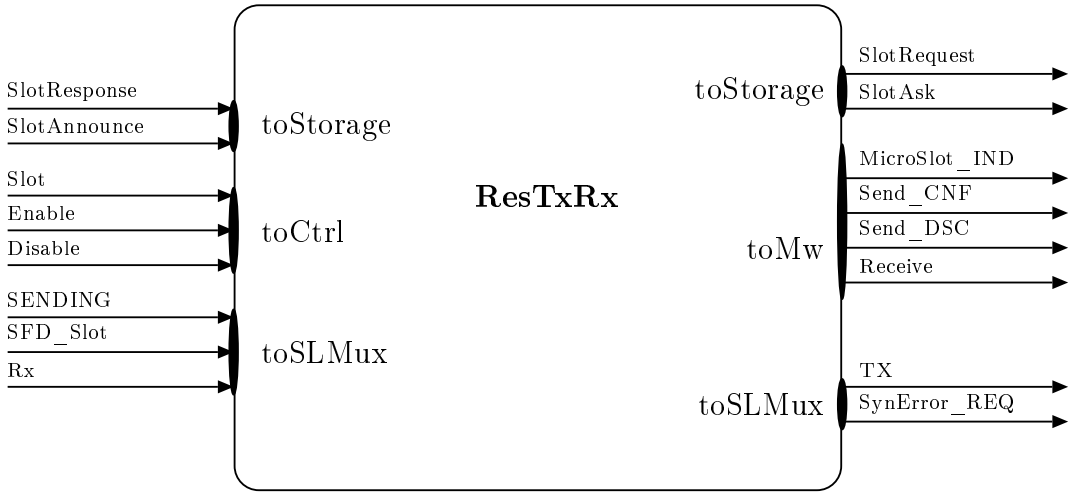
Version: 1.0

Author: Philipp Becker, Dennis Christmann

Intent

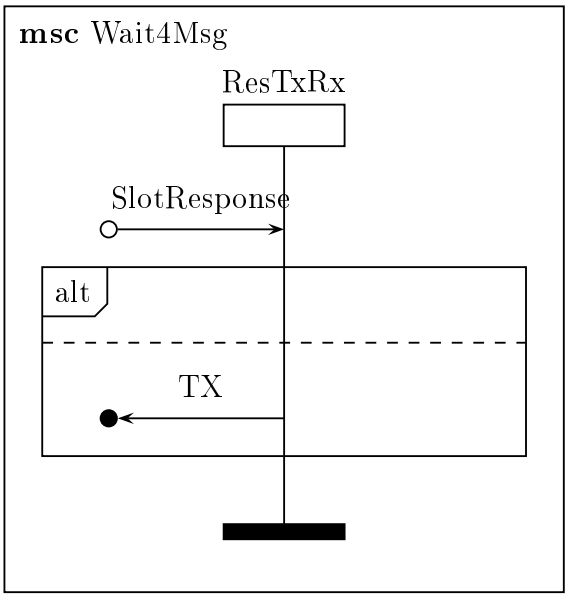
This symmetrical micro protocol provides the sending and receiving of frames in a slotted, contention-free virtual slot region. You must announce the messages with their scheduled slot and the micro protocol will send the frame in this slot. I.e. you need a storage component as glue, which announces new send requests and returns frames for a given slot. On the peer process instance(s) the frame can be received by this micro protocol. You can also disable the micro protocol, e.g. if you also have contention-based virtual slot regions.

Interface signature



Interface behaviour

Scenario 1

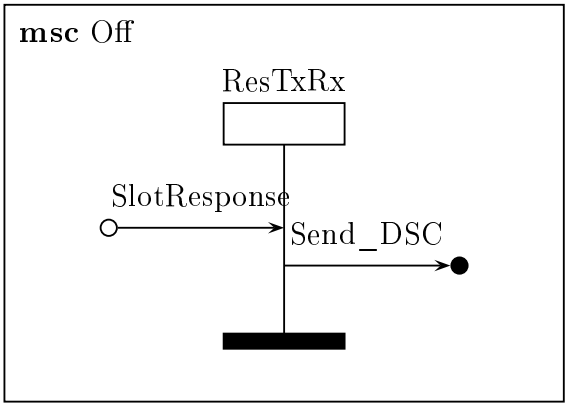


Description:

The signal *SlotResponse* provides a storage element, which was requested before. Now, there are two alternatives:

- If the requested element was not found in the storage, no further operations are performed.
- If the requested element was found, the signal *TX* sends the frame (after checking if the frame is valid) towards the transceiver.

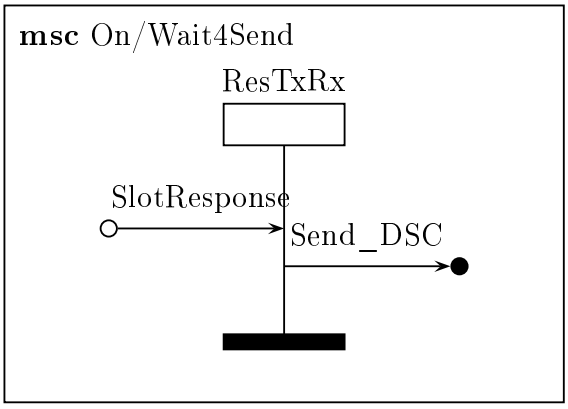
Scenario 2



Description:

The signal *SlotResponse* provides a storage element, which was requested in the state „on“. This should not happen in the current state („off“), because requested storage elements must be sent before the process goes to the state „off“. The signal *Send_DSC* indicates the failure of the transmission of the frame given in the storage element.

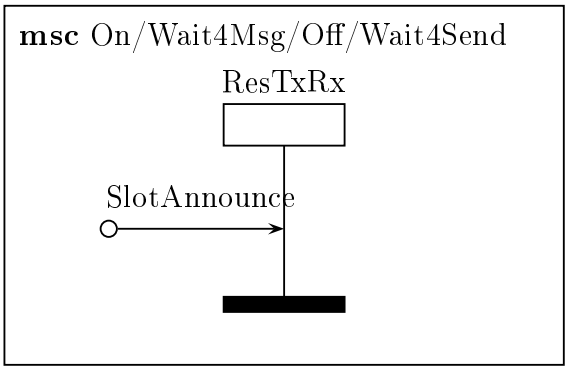
Scenario 3



Description:

The signal *SlotResponse* provides a storage element, but no storage element was requested in the current state. So this is an error. The signal *Send_DSC* indicates the failed transmission of the frame provided in the storage element.

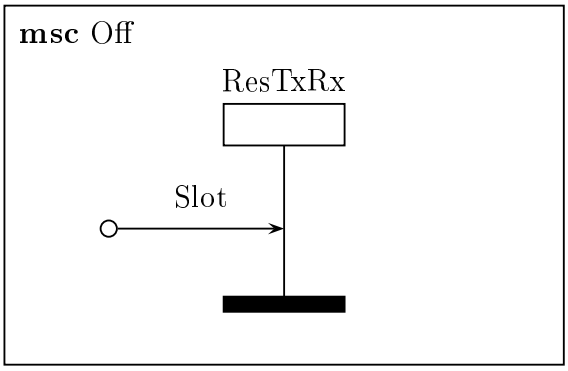
Scenario 4



Description:

The signal *SlotAnnounce* announces the arrival of a new packet at the storage (slot number is given as parameter). If the deadline of the announced slot will expire first, the process saves the announced slot number as next job.

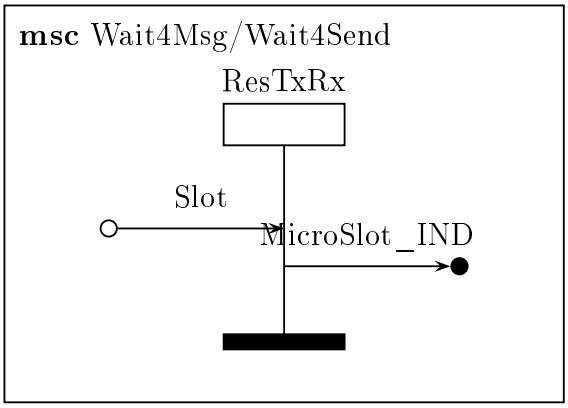
Scenario 5



Description:

The signal *Slot* informs the micro protocol about the current slot number. This should not happen in this state, because the component is disabled. No further operations are performed.

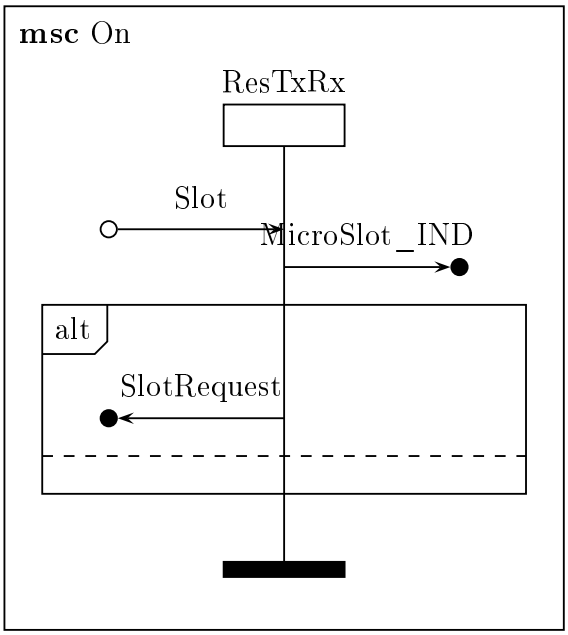
Scenario 6



Description:

The signal *Slot* informs the micro protocol about the current reservation slot number. The signal *MicroSlot_IND* forwards the current slot number to the network layer.

Scenario 7

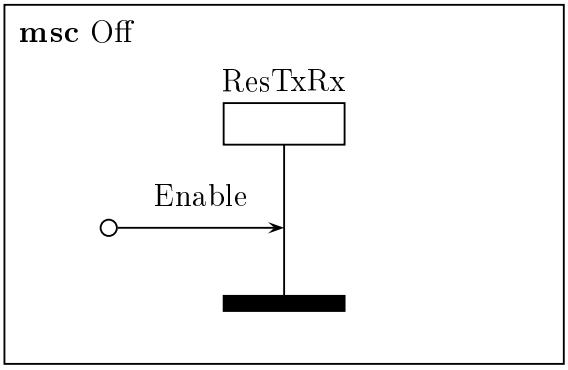


Description:

The signal *Slot* informs the micro protocol about the current slot number. The signal *MicroSlot_IND* forwards the current slot number to the network layer. Now, there are two alternatives:

- If a job is available for the current slot, the signal *SlotRequest* requests the storage element with the provided slot number from the storage.
- If no job is available for the current slot, no further operations are performed.

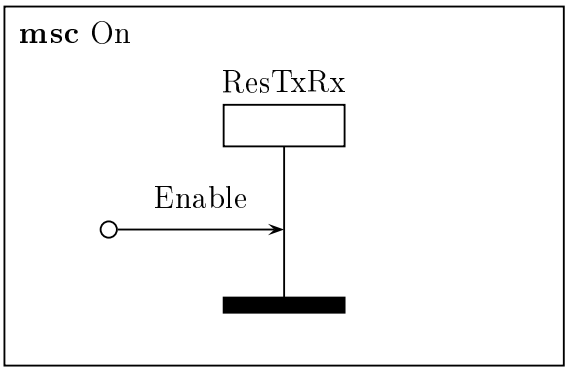
Scenario 8



Description:

The signal *Enable* triggers a state transition from „off“ to „on“, i.e. this component is now enabled. No further operations are performed.

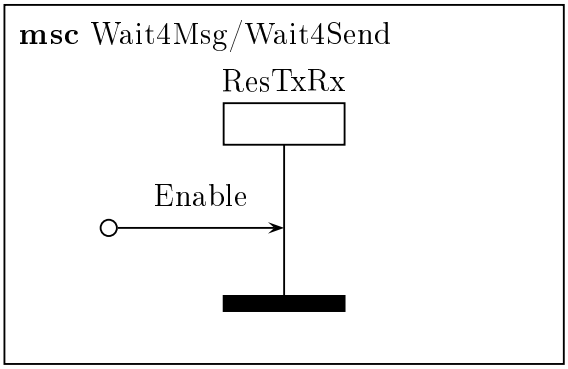
Scenario 9



Description:

The signal *Enable* instructs the process to go to the „on“ state. But the process is already enabled. So it is an error. No further operations are performed.

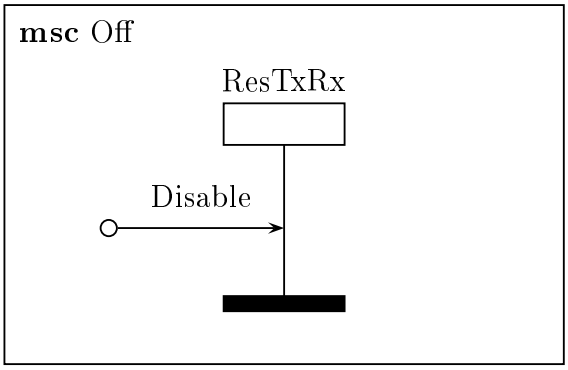
Scenario 10



Description:

The signal *Enable* instructs the process to go to „on“ state, but the process is already enabled. So this is an error. No further operations are performed.

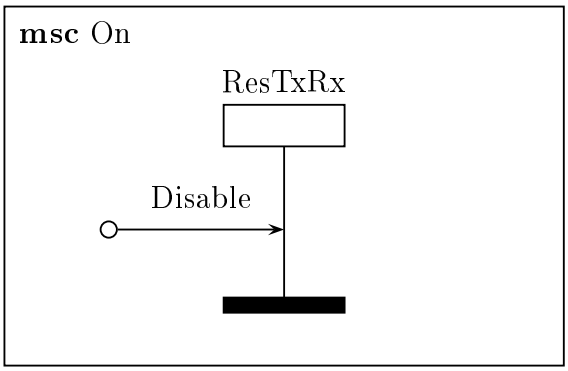
Scenario 11



Description:

The signal *Disable* instructs the process to go to the „off“ state. but the process is already in „off“. So this is an error. No further operations are performed.

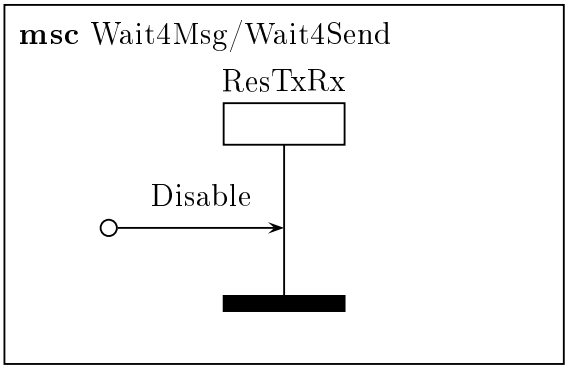
Scenario 12



Description:

The signal *Disable* instructs the process to go to the „off“ state. No further operations are performed.

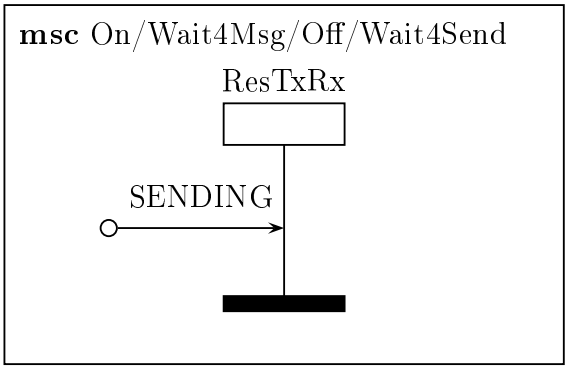
Scenario 13



Description:

The signal *Disable* instructs the process to go to „off“ state, but it has just requested a storage element. This seems like an error of the reservation. No further operations are performed.

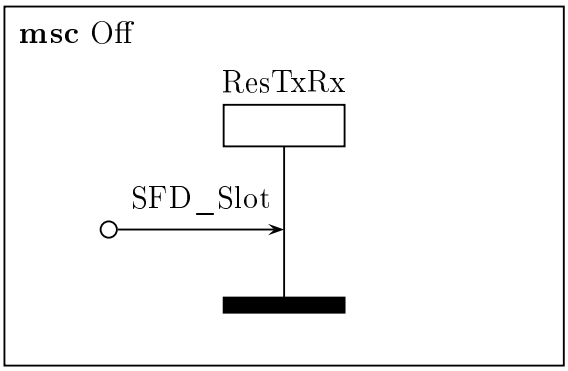
Scenario 14



Description:

The signal *SENDING* is just consumed in every state. It is not needed, because frames are sent without CCA in contention-free periods. No further operations are performed.

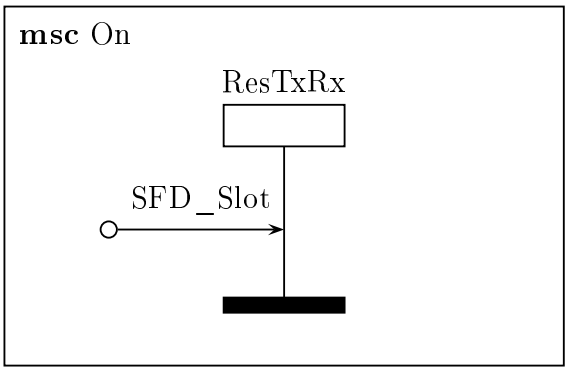
Scenario 15



Description:

The signal *SFD_Slot* confirms the sending of a frame, but all transmissions should be finished before the process changed to the „off“ state. So this is an error. No further operations are performed.

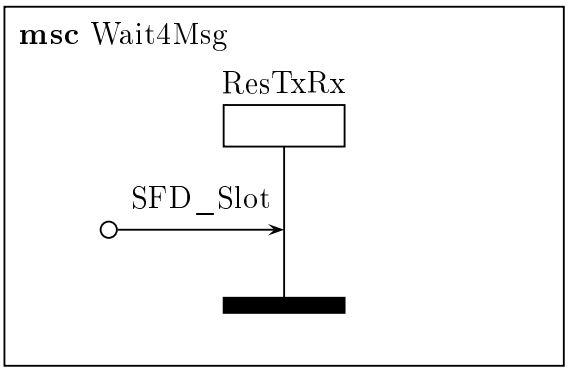
Scenario 16



Description:

The signal *SFD_Slot* confirms that the sending of a frame is complete. This should not happen in this state (on), because nothing was sent. No further operations are performed.

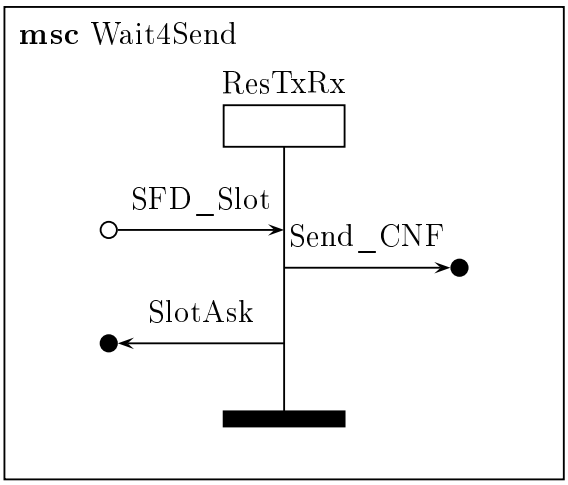
Scenario 17



Description:

The signal *SFD_Slot* confirms that the sending of a frame is complete. This should not happen here, because nothing was sent in this state. No further operations are performed.

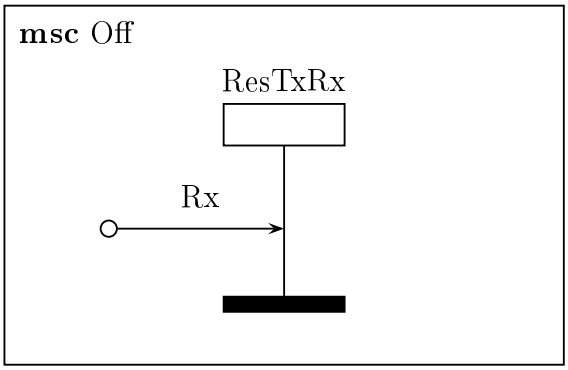
Scenario 18



Description:

The signal *SFD_Slot* confirms that the physical sending of a frame is complete. The signal *Send_CNF* indicates to the network layer that the sending of a frame was successful and the signal *SlotAsk* requests the next job from the storage.

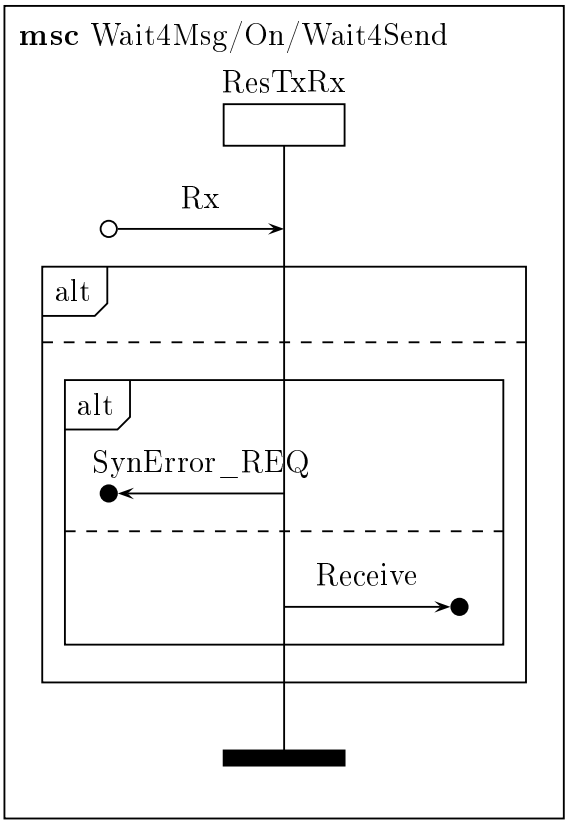
Scenario 19



Description:

The signal *Rx* represents the reception of a new frame. But this process should not receive a frame in the current state (off). No further operations are performed.

Scenario 20



Description:

The signal *Rx* represents the receiving of a new frame. Now, there are two alternatives:

- If checksum isn't correct, no further operations are performed.
- If checksum is correct, there are two alternatives:
  - If type of frame is RTS or CTS, the signal *SynError_REQ* indicates a synchronization error, which is sent to the basic layer.
  - Otherwise, the signal *Receive* sends the frame to the network layer.

Imported and exported definitions

Used packages

- MsgProcessing (procedure definitions shared with PrioTxRx)
- SLSignals (signal definition)
- SDLFixed (data type definition)
- SLDataTypes (data type definition)

- MacZSharedConfig (signal definition, data type definition)
- MacZMWInterface (signal definition)
- SenfConfig (procedure definition)
- Logging

Required definitions

- Procedure getConfigIKey
- Data type Octet_StringFixed
- Signal SlotResponse(Boolean,::SLDataTypes::Newtype::StorageElement)
- Signal SlotAnnounce(Integer)
- Signal SlotRequest(Integer)
- Signal SlotAsk(Integer)
- Signal MicroSlot_IND(Integer)
- Signal Send_CNF(::SLDataTypes::Newtype::StorageElement)
- Signal Send_DSC(::SLDataTypes::Newtype::StorageElement)
- Signal Receive
- Signal SigAlertCNF
- Signal Slot(Integer)
- Signal Enable
- Signal Disable
- Signal SENDING(Boolean)
- Signal SFD_Slot(Boolean)
- Signal Rx(::SDLFixed::Newtype::Octet_StringFixed,Boolean)
- Signal TX(::SLDataTypes::Newtype::StorageElement)
- Signal SynError_REQ

Provided definitions

- Process type ResTxRx

Checklist

– *none* –

B. CD

Auf dieser CD befindet sich folgender Inhalt:

- Projektordner von Telelogic Tau mit der SDL-Spezifikation von *MacZ* **vor** der Restrukturierung
- Projektordner von Telelogic Tau mit der SDL-Spezifikation von *MacZ* **nach** der Restrukturierung
- SDL-Spezifikation der identifizierten Mikroprotokolle
- mit *ConTraST* generierte Dokumentation der identifizierten Mikroprotokolle

Literaturverzeichnis

[AG] AG Vernetzte Systeme. http://vs.informatik.uni-kl.de.

[BDSZ94] Bharghavan, Vaduvur, Alan J. Demers, Scott Shenker und Lixia Zhang: *MACAW: A Media Access Protocol for Wireless LAN's.* In: *SIGCOMM*, Seiten 212–225, 1994.

[Bec06] Becker, Philipp: *Entwicklung des MacZ Service Layers.* Diplomarbeit, Fachbereich Informatik, Technische Universität Kaiserslautern, 2006.

[BGK07] Becker, Philipp, Reinhard Gotzhein und Thomas Kuhn: *MacZ - A Quality-of-Service MAC Layer for Ad-hoc Networks.* Proceedings of 7th Conference on Hybrid Intelligent Systems (HIS), Kaiserslautern, Germany, 2007.

[BGK08] Becekr, Phillipp, Reinhard Gotzhein und Thomas Kuhn: *Model-driven Performance Simulation of Self-organizing Systems with PartsSim.* Praxis der Informationsverarbeitung und Kommunikation, Seiten 45–50, 1/2008.

[Croa] Crossbow Technology Inc: *Datenblatt Imote 2.* http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/Imote2_Datasheet.pdf.

[Crob] Crossbow Technology Inc: *Datenblatt MicaZ.* http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf.

[EHS97] Ellsberger, Jan, Dieter Hogrefe und Amardeo Sarma: *SDL - Formal Object-oriented Language for Communication Systems.* Prentice Hall, 1997.

[FGGS04] Fliege, Ingmar, Alexander Geraldy, Reinhard Gotzhein und Philipp Schaible: *A Flexible Micro Protocol Framework.* In: *Proceedings of 4th SAM (SDL and MSC) Workshop, Ottawa, Canada*, Seiten 224–236, 2004.

[FGW06] Fliege, Ingmar, Rüdiger Grammes und Christian Weber: *ConTraST - A Configurable SDL Transpiler and Runtime Environment.* In: *Proceedings of 5th SAM (SDL And MSC) Workshop, Kaiserslautern*, Seiten 216–228, 2006.

[Fli07] Fliege, Ingmar: *Documentation of micro protocols.* Technischer Bericht 358/07, Technische Universität Kaiserslautern, 2007.

[GHJV95] Gamma, Erich, Richard Helm, Ralph Johnson und John Vlissides: *Design Patterns - Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[GK08] Gotzhein, Reinhard und Thomas Kuhn: *Tick Synchronization for Multi-hop Medium Slotting in Wireless Ad Hoc Networks using Black Bursts*. IEEE SECON 2008 (accepted paper), San Francisco Bay Area California, USA, June 16-20 2008.

[Got07] Gotzhein, Reinhard: *Model-driven with SDL - Improving the Quality of Networked Systems Development (Invited Paper)*. In: *Proceedings of the 7th International Conference on New Technologies of Distributed Systems (NOTERE 2007), Marrakesh, Morocco*, Seiten 31–46, June 4-8 2007.

[HK04] Hellenschmidt, Michael und Thomas Kirste: *A Generic Topology for Ambient Intelligence*. In: *EUSAI*, Seiten 112–123, 2004.

[IEEE99] IEEE Std. 802.11: *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. IEEE Computer Society, 1999.

[IEEE03] IEEE Std. 802.15.4: *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)*. IEEE Computer Society, 2003.

[IEEE05] IEEE Std. 802.11e: *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements*. IEEE Computer Society, 2005.

[ITU99] ITU-T Recommendation Z.100 (11/99): *Specification and description language (SDL)*. International Telecommunication Union (ITU), 1999.

[Joh96] Johnson, David: *Programming by Design*. Prentice-Hall International, Inc., 1996.

[Kar90] Karn, Phil: *MACA - A New Channel Access Method for Packet Radio*. In: *Proceedings of the ARRL/CRRL Amateur Radio 9th Computer Networking Conference*, 1990.

[KF05] Kuhn, Thomas und Ingmar Fliege: *Micro Protocol Based Design of MacZ - A Highly Adaptive, Integrated QoS MAC Layer for Ambient Intelligence Systems*. Technischer Bericht 347/05, Technische Universität Kaiserslautern, 2005.

[KGGR05] Kuhn, Thomas, Alexander Geraldy, Reinhard Gotzhein und Florian Rothländer: *ns+SDL - The Network Simulator for SDL Systems*. In: A. Prinz, R. Reed, J. Reed (Herausgeber): *SDL 2005 - Model Driven (Proceedings of 12th International SDL Forum)*, Seiten 103–116. Springer, 2005.

[KR05] Kurose, James F. und Keith W. Ross: *Computer Networking - A Top-Down Approach Featuring the Internet.* Addison Wesley, 3. Auflage, 2005.

[Kuh06] Kuhn, Thomas: *MacZ - a QoS MAC Layer for Ambient Intelligence Systems.* In: Kiniry, T. Pfeifer; A. Schmidt; W. Woo; G. Doherty; F. Vernier; K. Delaney; B. Yerazunis; M. Chalmers; J. (Herausgeber): *Advances in Pervasive Computing 2006. Adjunct Proceedings of the 4th International Conference on Pervasive Computing*, Seite 244ff. Austrian Computer Society (OCG): Vienna, 2006.

[LG97] Lin, Chunhung Richard und Mario Gerla: *Asynchronous Multimedia Multihop Wireless Networks.* In: *INFOCOM*, Seiten 118–125, 1997.

[RR06] Ramachandran, Iyappan und Sumit Roy: *On the Impact of Clear Channel Assessment on MAC Performance.* In: *GLOBECOM*, 2006.

[Tan96] Tanenbaum, Andrew S.: *Computer Networks.* Prentice-Hall International, Inc., 1996.

[Tel] Telelogic Tau: *Produktseite.* `http://www.telelogic.com/products/tau/index.cfm`.

[Tex] TexasInstruments: *CC2420 Datenblatt.* `http://focus.ti.com/lit/ds/symlink/cc2420.pdf`.

[USC] USC Information Sciences Institute: *The Network Simulator - ns-2.* `http://www.isi.edu/nsnam/ns/`.

[WFG+04] Webel, Christian, Ingmar Fliege, Alexander Geraldy, Reinhard Gotzhein, Marc Krämer und Thomas Kuhn: *Cross-Layer Integration in Ad-Hoc Networks with Enhanced Best-Effort Quality-of-Service Guarantees.* In: *Proceedings of World Telecommunications Congress (WTC 2006)*, 2004.

[WGS07] Webel, Christian, Reinhard Gotzhein und Daniel Schneider: *Formalization of Network Quality-of-Service Requirements.* Technischer Bericht 356/07, Technische Universität Kaiserslautern, 2007.

[Zig] ZigBee Aliance. `http://www.zigbee.org`.