Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Friedrich Groß

Mikrocontroller basierte Messung von Paketlaufzeiten in Time-Triggered-Ethernet Netzwerken

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Friedrich Groß

Mikrocontroller basierte Messung von Paketlaufzeiten in Time-Triggered-Ethernet Netzwerken

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

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

Betreuender Prüfer: Prof. Dr. Franz Korf
Zweitgutachter: Prof. Dr. Hans H. Heitmann

Abgegeben am 24. August 2011

Friedrich Groß

Thema der Bachelorarbeit
Mikrocontroller basierte Messung von Paketlaufzeiten in Time-Triggered-Ethernet Netzwerken

Stichworte
Echtzeit-Ethernet, Time-Triggered-Ethernet, Messung, TTEthernet

Kurzzusammenfassung
Diese Arbeit befasst sich mit der Entwicklung eines Messgeräts, mit dem Paketlaufzeiten im Time-Triggered-Ethernet gemessen werden können. Eine weitere Messart ermöglicht das Messen des Jitters eines Time-Triggered-Senders. Das Messgerät kann sich auf die im Netzwerk ausgehandelte Zeit synchronisieren, sodass die Messergebnisse in der Zeitbasis des Netzwerks dargestellt werden können. Solche Messungen sind notwendig, um festzustellen ob Echtzeitanforderungen von Hardware und Netzwerk erfüllt werden können.

Title of the paper
Microcontroller-based measurement of packet delays in Time-Triggered-Ethernet networks

Keywords
Real-time Ethernet, Time-Triggered-Ethernet, Measurement, TTEthernet

Abstract
This thesis deals with the development of a measuring instrument, that can measure the packet delays in Time-Triggered-Ethernet networks. Another measuring method allows to measure a jitter of a Time-triggered transmitter. The measuring instrument can be synchronized to the network negotiated time, so that the results can be displayed in the time base of the network. Such measurements are necessary to determine whether real-time requirements can be met by deployed hardware and network.

Inhaltsverzeichnis

Kapitel 1

Einführung

1.1 Motivation

In einem Automobil werden “Bus-Systeme” für die Kommunikation zwischen den einzelnen Steuergeräten, Sensoren und Aktoren verwendet. Die steigende Anzahl an “Extras” in einem Automobil führt zu einer steigenden Anzahl an Steuergeräten, was wiederum zu einer komplexeren Kommunikation und einem erhöhten Bandbreitenbedarf führt. Insbesondere “Extras” wie Rückfahrkamera, Multimediadienste in den Kopfstützen, Verkehrsschildererkennung oder Spurerkennung steigern den Bandbreitenbedarf in einem Automobil. Ein Teil dieser Kommunikation muss in Echtzeit durchgeführt werden, was insbesondere von sicherheitsrelevanten Extras verlangt wird.
Im Jahr 2000 wurde von einigen Automobil- und Elektronikherstellern das FlexRay-Konsortium gegründet, wodurch ein neues Bussystem namens FlexRay entwickelt wurde (vgl. FlexRay Consortium). FlexRay ist echtzeitfähig und liefert eine Bandbreite von $10\frac{Mbit}{s}$. 2007 wurde das erste Fahrzeug ausgeliefert, welches Flexray als weiteres Bus-System verwendet (vgl. Böke, 2008). Da diese Bandbreite für Bild- und Multimediaübertragungen nicht reicht und die Echtzeitfunktionalität nicht überall benötigt wird, werden weitere Bus-Systeme, die für ihren Einsatzzweck optimiert sind, in einem Fahrzeug verwendet. Beispielsweise wird der kostengünstige *Local Interconnect Network-Bus (LIN)* (vgl. LIN-Administration) für abgesetzte Sensoren und Aktoren z. B. in Türen oder Sitzen eingesetzt. Ein weiteres Beispiel ist der *Media Oriented System Transport-Bus (MOST)* (vgl. MOST Cooperation), der für die Übertragung von Multimediadaten im Fahrzeug verwendet.
Der Einsatz von verschiedenen Bus-Systemen steigert die Komplexität der Steuergeräte und erfordert teilweise verschiedene Schnittstellen und Gateways in einem Steuergerät. „Eigentlich ist das Bordnetz im Gesamtfahrzeug bereits heute nicht mehr vernünftig zu beherrschen“ und „Die Komplexität steigt weiter“ (Badstübner, 2008, BMW Group Forschung und Technik). Ein einheitliches, gleichzeitig schnelles und echtzeitfähiges Bussystem würde das Bandbreiten-

problem lösen und die Komplexität verringern.

Das Ethernetprotokoll ist ein, in anderen Gebieten, weitverbreitetes Kommunikationsprotokoll, mit dem auf unterschiedlichen physikalischen Medien, Daten mit unterschiedlicher Bandbreite übertragen werden können. Die maximale Bandbreite liegt momentan bei $10\frac{Gbit}{s}$, welches um den Faktor 1000 höher ist als beim Flexray-Bus. Dieses Bus-System liefert zwar eine hohe Bandbreite, bietet aber in der Grundform keine Echtzeitfunktionalität.
Allerdings wurden in den letzten Jahren Echtzeiterweiterungen für das Ethernetprotokoll vorgestellt. Eine Echtzeiterweiterung davon ist das Time-Triggered Ethernet, welches von der TU Wien spezifiziert und von dem Unternehmen TTTech entwickelt wurde (vgl. TTTech Computertechnik AG). Vorarbeiten haben gezeigt, dass Time-Triggered-Ethernet eine Erfolg versprechende Technologie für die Kommunikation in Fahrzeugen ist (vgl. Steinbach u. a., 2010). Dieses Bussystem wird derzeit noch nicht im Automobil verwendet, es befindet sich praktisch in der “Evaluierungsphase” (vgl. SAE - AS-2D Time Triggered Systems and Architecture Committee, 2009).
Das Time-Triggered-Ethernetprotokoll beschreibt drei Nachrichtenklassen, eine davon ist der Time-Triggered-Traffic. Diese Nachrichten werden zeitgesteuert vom Absender versendet und mit konstanter Latenz durch das Netzwerk zum Empfänger geleitet. Dieses deterministische Verhalten ermöglicht es, zeitkritische Echtzeitkommunikation über ein Netzwerk zu leiten. Um dies realisieren zu können, müssen die herkömmlichen Switches durch spezielle TTEthernet-Switches ausgetauscht werden. Eine Zeitsynchronisation auf eine globale Uhr ist zwischen den Parteien ebenfalls notwendig.

Um bei der Entwicklung von TTEthernet-Hardware das Zeitverhalten dieser Geräte, und somit auch die Qualität bestimmen zu können, werden Messgeräte benötigt, die diese Aufgabe übernehmen.
Ein Messgerät, das die physikalische Austrittszeit einer zeitgesteuerten Nachricht misst, und das Messergebnis in der im Netzwerk synchronisierten Zeit anzeigt, könnte helfen Aussagen, über die Einhaltung von Zeitanforderung und somit über die Zuverlässigkeit eines Geräts, zu treffen.
Auch das Messen von Paketlaufzeiten durch ein Netzwerk, mit hoher Genauigkeit und in der Zeitbasis der synchronisierten Zeit könnte helfen, Aussagen über die Einhaltung von Echtzeitanforderungen zu treffen.
Diese Arbeit beschäftigt sich mit der Entwicklung eines solchen Messgeräts.

1.2 Struktur der Arbeit

In Kapitel 2 werden fachliche Grundlagen beschrieben, die zum besseren Verständnis dieser Arbeit betragen sollen. Die Grundlagen befassen sich mit Inhalten, die über das Grundstudium

eines Informatikstudiengangs hinausragen.

In Kapitel 3 werden Messungen, die vorgenommen werden sollen, definiert und Anforderungen an diese Messungen gestellt. Anschließend werden Anforderungen an die im Messaufbau verwendete Hardware gestellt.

In Kapitel 4 werden Hardwarekomponenten auf Ihre Verwendbarkeit analysiert. Es werden die im Kapitel 3 aufgestellten Anforderungen mit den Analyseergebnissen verglichen, um die Verwendbarkeit festzustellen.

In Kapitel 5 werden die Architektur und das damit verbundene Konzept beschrieben. Anschließend wird die Realisierung des Messgeräts beschrieben.
In Kapitel 6 werden zwei Beispielmessungen und deren Ergebnisse präsentiert, die mit dem realisierten Messgerät vorgenommen wurden.

In Kapitel 7 werden die Ergebnisse dieser Arbeit zusammengefasst. Anschließend wird ein Ausblick auf zukünftige Arbeiten gegeben.

Kapitel 2

Grundlagen

In diesem Kapitel werden fachliche Grundlagen beschrieben, die zum besseren Verständnis dieser Arbeit betragen sollen.
Die Grundlagen befassen sich mit Inhalten, die über das Grundstudium eines Informatikstudiengangs hinausragen.
Zunächst werden ein paar Fachbegriffe beschrieben und anschließen folgen die Grundlagen zu Ethernet und Time-Triggered-Ethernet.

Echtzeit Unter Echtzeit versteht man, dass ein System auf ein Ereignis innerhalb einer bestimmten Zeit garantiert reagieren muss. Abhängig von den Anforderungen wird dann festgelegt wie schnell reagiert werden soll. Reagiert ein System nicht innerhalb der festgelegten Zeit, so ist die Reaktion nicht nur zu spät, sondern auch falsch. (vgl. Tanenbaum, 2009, S. 208-209)

Latenz Der Begriff Latenz, der in dieser Arbeit häufig verwendet wird, ist eine wichtige Metrik in einem Netzwerk. In der Literatur wird der Begriff oft durch Verzögerungszeit oder Signallaufzeit definiert. Im Kontext dieser Arbeit beinhaltet die Latenz auch die Verarbeitungszeit der Geräte, die zwischen Sender und Empfänger liegen.

Jitter Der Begriff Jitter wird in dieser Arbeit verwendet, um die Schwankung der Latenz zu beschreiben.

2.1 Ethernet IEEE 802.3

Dieses Kapitel ist eine Zusammenfassung aus (Tanenbaum, 2003, S. 309 - 312).
Ethernet ist eine Netzwerktechnik, die den Datenaustausch zwischen Computern ermöglicht. Es ist im OSI-Modell in Schicht 1 und 2 angesiedelt. In Schicht 1 (Bitübertragungsschicht) definiert Ethernet, wie Daten in Bitströmen versendet werden sollen. In Schicht 2 (Sicherungsschicht) definiert Ethernet einen Rahmen, der die Adressierung eines Gerätes regelt und eine

Fehlererkennung für falsch übertragene Daten beinhaltet. Der Rahmen wird bei Ethernet Frame genannt. In Abbildung 2.1 ist das Frameformat IEEE 802.3 zu sehen.

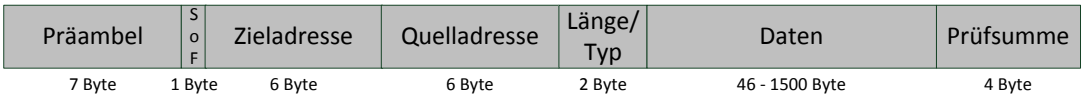

Abbildung 2.1: Aufbau Ethernetframe IEEE 802.3

Im Folgenden werden die Felder des Frames beschrieben:

Präambel Ein Frame beginnt mit einer 7 Byte großen Präambel, wobei jedes Byte das Bitmuster "10101010“ enthält. Dieses Feld dient zur Synchronisation der Bitabstände von Sender und Empfänger.

SoF Das Start of Frame beendet die Synchronisation und signalisiert dem Empfänger den Beginn eines Frames. Es unterscheidet sich zur Präambel im letzen Bit, welches "1“ ist.

Zieladresse Adressen werden bei Ethernet "Mac-Adresse“ genannt. Jede Netzwerkkarte hat eine weltweit eindeutige Adresse. Dieses 6 Byte große Feld dient dazu, das Ziel dieses Pakets zu adressieren.

Quelladresse Analog zur Zieladresse, nur dass der Absender hier seine eigene Adresse einträgt.

Länge / **Typ** In dieses Feld wird die Länge des Pakets oder der Nachrichtentyp eingetragen. Werte, die höher als 1500 sind, werden als Nachrichtentyp interpretiert. Werte, die kleiner sind, werden als Länge des Datenfeldes interpretiert.

Daten In dieses Feld werden Daten, die vermittelt werden sollen, eingetragen. Die minimale Größe beträgt 46 Byte, die maximale Größe beträgt 1500 Byte. Sollen weniger Daten als 46 Byte übertragen werden, so muss dieses Feld aufgefüllt werden, bis es 46 Byte groß ist.

Prüfsumme In diesem Feld steht eine CRC-Prüfsumme der Daten. Diese dient zur Erkennung von Übertragungsfehlern.

2.2 Time-Triggered-Ethernet

Eine detaillierte, gut verständliche Beschreibung des Time-Triggered-Ethernet kann in Arbeit (Bartols, 2010) gefunden werden. In diesem Kapitel werden die Informationen aus dieser Arbeit zusammengefasst dargestellt und nur, die für diese Bachelorarbeit wichtigen Informationen, detailliert beschrieben. Eine vollständige Spezifikation kann in (Steiner, 2008) gefunden werden.

Time-Triggered-Ethernet (im Folgenden TTEthernet) ist eine Echtzeiterweiterung für das Ethernet. Es ermöglicht u.a. das zeitgesteuerte Kommunizieren innerhalb eines Netzwerks, wobei für die zeitgesteuerten Nachrichten eine deterministische Paketlaufzeit durch das Netzwerk garantiert wird. Um dieses Verhalten zu ermöglichen, werden spezielle Switches zwischen Netzwerkteilnehmern benötigt. In anderen Bereichen des Netzwerks können normale Switches verwendet werden.
Das TTEthernet-Protokoll unterstützt drei Nachrichtenklassen, die im Folgenden beschrieben sind:

Time-Triggered-Traffic (TT) wird für den zeitgesteuerten Datenaustausch verwendet. TT-Frames haben kurze Paketlaufzeit mit einem niedrigen Jitter, was für ein deterministisches Verhalten sorgt. TT-Frames werden von allen Netzwerkgeräten mit höchster Priorität behandelt. Diese Nachrichtenklasse wird für zeitkritische Echtzeitkommunikation verwendet.

Rate-Constrained-Traffic (RC) wird für weniger zeitkritische Echtzeitkommunikation verwendet. Die Echtzeitfähigkeit wird durch eine vorher festgelegte garantierte Bandbreite realisiert. Der Nachrichtenaustausch erfolgt eventbasiert und muss wie bei den TT-Nachrichten durch ein spezielles Switch geleitet werden. Diese Nachrichtenklasse entspricht dem AFDX-Protokoll-Standard (vgl. AIM GmbH).

Best-Effort-Traffic (BE) entspricht dem Standard-Ethernet Verkehr. Nachrichten dieser Klasse werden mit niedrigster Priorität behandelt, sodass eine Übertragung nicht garantiert werden kann.

Bei RC- und TT-Nachrichten wird im Ethernetframe die Zieladresse anders interpretiert (siehe Abbildung 2.2). Dort wird ein 32-Bit großer TT-Marker und eine 12-Bit große TT-ID eingetragen. In das Feld TT-Marker wird ein konstanter Wert eingetragen, der zeitkritische Nachrichten

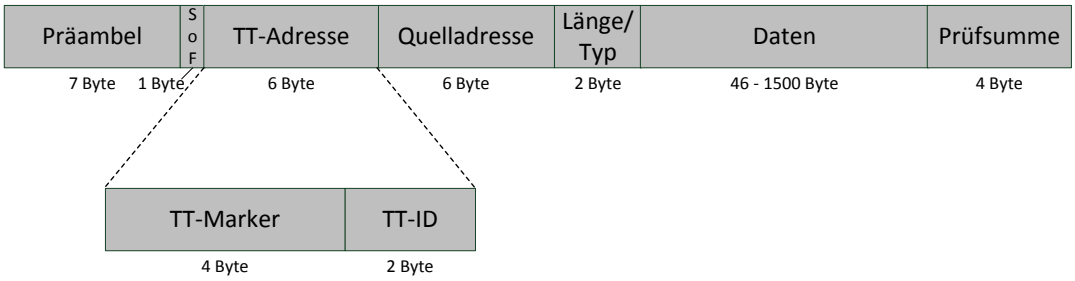

Abbildung 2.2: Time-Triggered-Ethernet Frame (vgl. Bartols, 2010)

kennzeichnet. Im Feld TT-ID wird eine Identifikationsnummer einer Nachricht eingetragen. So ist es demnach möglich $2^{12} = 4096$ Nachrichten zu definieren. Die IDs werden an TT- und

RC-Frames vergeben. Dadurch, dass das Feld für die Zieladresse anders verwendet wird, geht den TT- und RC-Frames die Information des Ziels verloren. Dieser Informationsverlust wird durch die Switches ausgeglichen. Dazu wird in den Switches mittels einer Routingtabelle zu jeder TT-ID der Ausgangsport konfiguriert. Der Switch „weiß“ also, welche zeitkritischen Nachrichten an welchen Ausgangsport gesendet werden müssen, ohne dass eine Ziel-MAC-Adresse im Frame eingetragen ist.
In der Switch wird außerdem ein Schedule für Nachrichten der Klasse TT festgelegt. In der Switch wird außerdem auch konfiguriert, zu welchem Zeitpunkt Nachrichten der Klasse TT von welchem Absender an welchen Empfänger gesendet werden. Dies wird durch ein Schedule realisiert. Zu diesen Zeitpunkten hält der Switch den Sendepuffer des Empfängers frei, sodass diese Nachricht ohne ungeplante Verzögerung weitergeleitet werden kann. Dazu wird in der Switch ein Zeitfenster konfiguriert, in der die Nachricht vom Absender in der Switch eintreffen darf. Trifft die Nachricht außerhalb des Zeitfensters ein, so wird die Nachricht verworfen. Um einen Zeitpunkt auszumachen, ist in allen TT-Netzwerkteilnehmern eine Zykluszeit definiert. Ein TT-Sendevorgang wird dann innerhalb der Zykluszeit festgelegt. Damit bei allen Netzwerkteilnehmern die Zykluszeit zum gleichen Zeitpunkt ihren Nullpunkt hat und die Zykluszeiten mit der gleichen Geschwindigkeit laufen, ist eine Zeitsynchronisation der TT-Netzwerkgeräte notwendig. Im Folgenden wird der Ablauf der Synchronisation beschrieben.

Die Zeitsynchronisation wird durch eine Zwei-Wege-Synchronisation realisiert, in der den Netzwerkteilnehmern unterschiedliche Rollen zugeteilt werden, die im Folgenden beschrieben werden:

Synchronisation-Master (SM) leiten die Synchronisation ein, in dem PCF-Frames (weiter unten beschrieben) versendet werden. Das PCF-Frame beinhaltet die aktuelle Zeit des SM. Diese Rolle wird mehreren Teilnehmern zugeteilt, um das Netzwerk fehlertolerant zu halten.

Compression-Master (CM) sammeln eintreffende PCF-Frames, errechnen aus den empfangenen Zeiten eine neue Durchschnittszeit, synchronisieren sich auf diese Zeit und leiten diese an alle Teilnehmer weiter. Diese Rolle wird üblicherweise zentralen Switches zugeteilt.

Synchronisation-Client (SC) synchronisieren sich auf ein eintreffendes PCF-Frame und leiten diese ggf. an andere Teilnehmer weiter. Vor dem Weiterleiten wird auf die im PCF-Frame eingetragene Zeit die eigene Verzögerung aufaddiert. Diese Rolle wird Endteilnehmern und Switches die nicht CM sind zugeteilt.

Ein PCF-Frame (Protocol Control Frames) ist ein Frame minimaler Größe. Der Datenteil beinhaltet mehrere Synchronisationsdaten von dem für diese Bachelorarbeit nur das Feld mit der

eingetragenen Zeit wichtig ist. Es wird die Zeit des physikalischen Austritts des Geräts eingetragen. PCF-Frames haben den Ethernet-Typ 0x891d.
In Abbildung 2.3 wird anhand eines Beispielnetzwerks die Zwei-Wege-Synchronisation gezeigt.

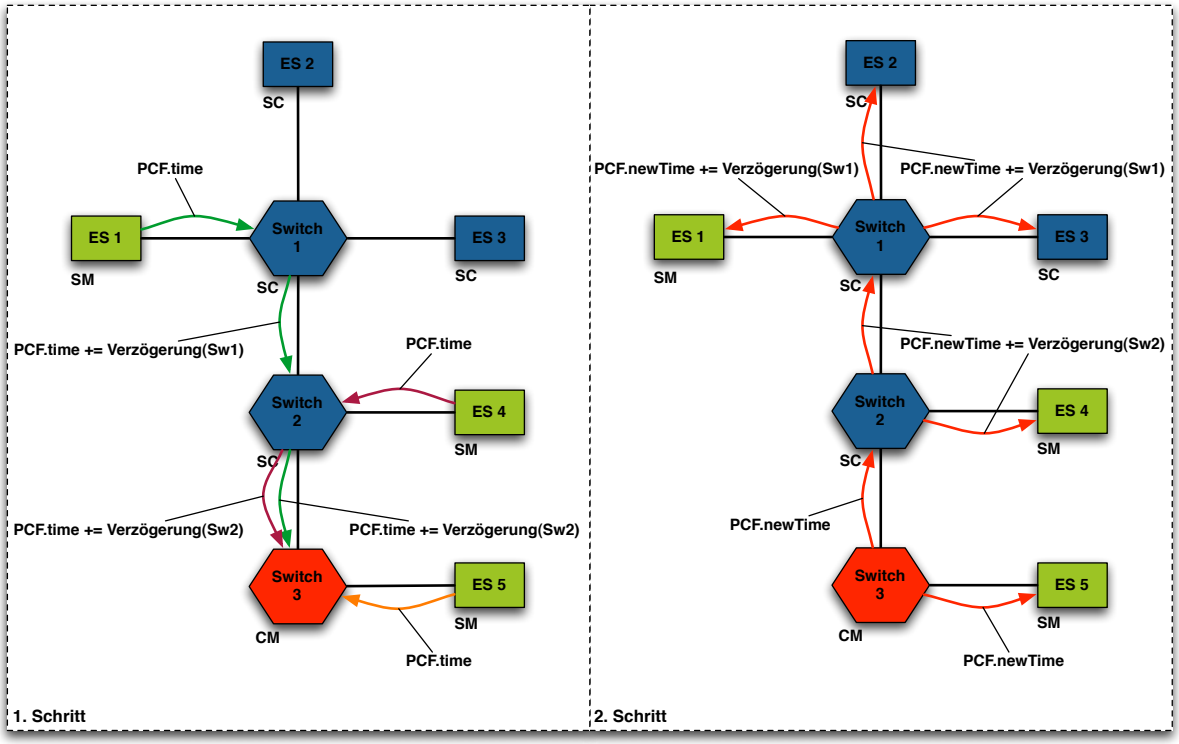

Abbildung 2.3: Beispiel: Synchronisation im TTEthernet (Quelle: Bartols, 2010)

In Schritt 1 versenden die Sync-Master (im Bild grün) ihre aktuelle Zeit. Diese werden von den Sync-Client-Switches zum Compression-Master geleitet. Während des Weiterleitens wird in den PCF-Frames die Verzögerung der Switch aufaddiert. Der Compression-Master erhält alle PCF-Frames und rechnet eine Durchschittszeit aus. In Schritt 2 sendet der Compression-Master die errechnete Durchschnittszeit an alle Netzwerkteilnehmer. Die Sync-Client-Switches leiten die PCF-Frames an die Sync-Master und Sync-Clients weiter und erhöhen dabei wieder die Zeit um die eigene Verzögerung. Alle Netzwerkteilnehmer können sich nun auf die ausgehandelte Zeit synchronisieren. Ein Synchronisationsvorgang wird üblicherweise zu Begin eines Zyklus durchgeführt, kann aber auch zu jedem anderen Zeitpunkt im Zyklus durchgeführt werden.

Kapitel 3

Anforderungen

Dieses Kapitel beginnt damit Messungen, die vorgenommen werden sollen, zu definieren und Anforderungen an diese Messungen zu stellen. Anschließend werden Anforderungen an die im Messaufbau verwendete Hardware gestellt.
Die Anforderungen haben verschiedene Prioritäten. Anforderungen mit der Priorität 1 sind wichtige Anforderungen und die mit der Priorität 2 sind sogenannte “nice to have” - Anforderungen.

3.1 Messanforderungen

In diesem Kapitel werden Anforderungen zu den Messungen aufgestellt. Es sollen mehrere Arten von Messungen möglich sein. In Kapitel 3.1.1 werden Anforderungen an eine Messung gestellt, bei der die Qualität eines Gerätes bestimmen soll. Dabei sendet ein Gerät zeitgesteuerte Nachrichten im konfigurierten Schedule, die vom Messgerät analysiert werden. Im Kapitel 3.1.2 werden Anforderungen an eine weitere Messung gestellt, bei der die Latenz eines NUT (Network under Test) gemessen werden soll. Also die Zeit, die ein Datenpaket durch ein Netzwerk braucht, um von einem Absender zu einem Empfänger zu gelangen.

3.1.1 Qualitätsbestimmung eines Senders

Mit dieser Messung soll die Abweichung zwischen dem eingestellten Zeitpunkt des Versendens im Schedule und den tatsächlichen Versenden der Nachricht gemessen werden. Der für diese Messung notwendige Messaufbau ist in Abbildung 3.1 dargestellt. Anhand der Messergebnisse kann bei den Time-Triggered-Switches der genaue Eintreffzeitpunkt und das Eintreffzeitfenster eingestellt werden. Dadurch können zu groß eingestellte Zeitfenster in der Switch vermieden werden. Diese führen dazu, dass während des Zeitfensters kein anderer Datenverkehr zum Empfänger gelangen kann. Da diese Einstellungen in der im Netzwerk synchronisierten Zeit eingestellt werden, muss dem Messgerät die Netzwerkzeit bekannt sein.

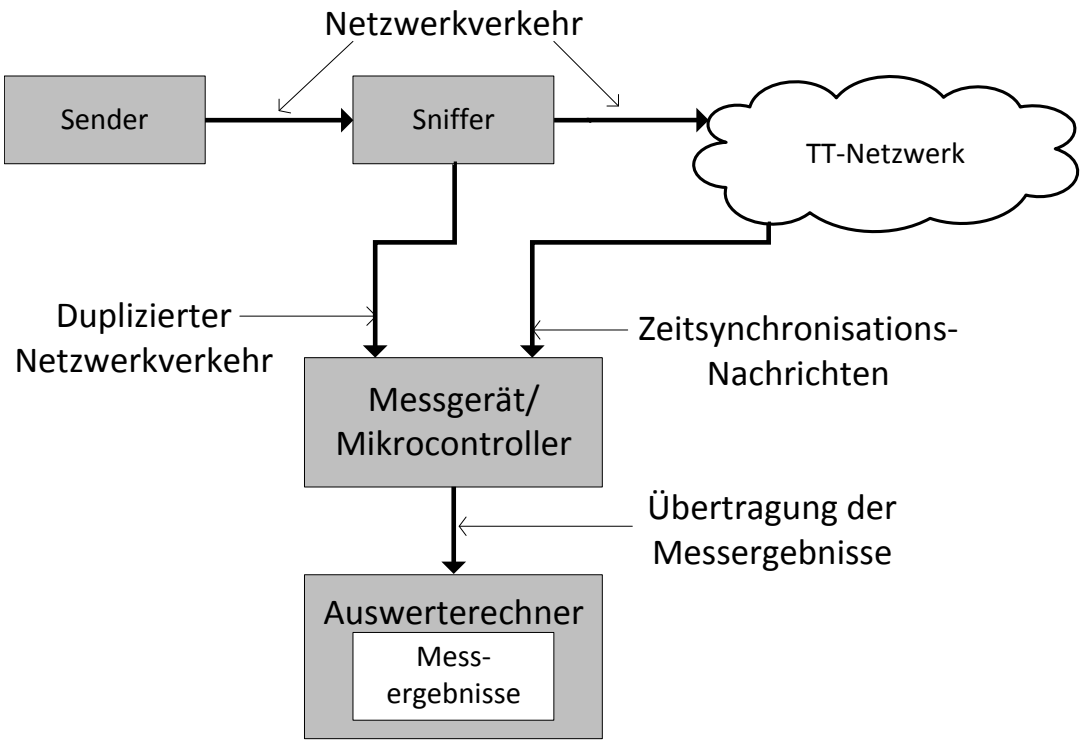

Abbildung 3.1: Schematischer Messaufbau zur Qualitätsbestimmung eines TT-Senders

Dazu muss die Systemzeit des Messgeräts und des Netzwerks mit der gleichen Geschwindigkeit laufen. Dies erfordert eine Zeitsynchronisation des Mikrocontrollers mit dem Netzwerk. Momentan werden in dem Labor des CoRE-Teams (CoRE RG) Zeitfenster von $\pm$10µs vom erwarteten Eintreffzeitpunkt eingestellt. Eine Festlegung des Zeitfensters in 0,1 µs Schritten wäre dabei wünschenswert. Das führt dazu, dass das Messgerät auf 0,1 µs genau die Zeiten erfassen muss. Mindestanforderung soll jedoch sein, das Zeitfenster in 1µs Schritten einzustellen, womit der minimale Messgenauigkeit mit 1µs festgelegt wird.

Für diese Messung sind nur TT-Nachrichten relevant. Dazu muss das Messgerät TT-Nachrichten für die Auswertung filtern. Es soll auch eine Filterung auf bestimmte CT-IDs möglich sein. Für die Messung muss die Nachricht einmal pro Zyklus versendet werden. Nach dem Empfangen eines Pakets soll der Mikrocontroller die gemessenen Zeiten und Pakete zu einem Auswerterechner übertragen. Der Auswerterechner soll die Daten auswerten und übersichtlich darstellen. Dazu muss eine Mindestbandbreite bestimmt werden, mit der eine Schnittstelle die Messdaten zu dem Auswerterechner überträgt. In einem Time-Triggered Netzwerk werden Zykluszeiten (Dauer einer Periode im Netzwerk) im Bereich von 2ms bis

6ms gewählt. Bei der Berechnung der Bandbreite muss davon ausgegangen werden, dass die Nachricht eine maximale Größe von 1514 Bytes haben kann. Bei der Übertragung mit dem Abstand von 2ms ergibt sich folgende Bandbreite:

$$\frac{(1514*8)Bit}{0{,}002s} = 6.056.000Bit/s = 6,1MBit/s$$

Da auch noch die gemessenen Zeiten und evtl. noch andere Metadaten übertragen werden müssen, wird hier die Anforderung auf eine minimale Bandbreite von 7 MBit/s gesetzt.

Um die Datenpakete dem Netzwerk abgreifen zu können, ohne das Zeitverhalten der Datenpakete zu beeinflussen, muss im Messaufbau ein für das Netzwerk transparenter Sniffer eingesetzt werden. Anforderungen zum Sniffer werden im Kapitel 3.4 aufgestellt.
In der Tabelle 3.1 werden die im vorangegangen Text beschriebenen Anforderungen zusammengefasst.

Nr.	Anforderung	Prio.
1	Messung des Absendezeitpunkts eines Gerätes in der Zeitbasis des Netzwerks	1
2	Messung des Jitters des Absendezeitpunkts in der Zeitbasis des Netzwerks	1
3	Zeitsynchronisation mit der Netzwerkzeit	1
4	Messgenauigkeit von 0,1µs	2
5	Messgenauigkeit von 1µs	1
6	Filterfunktion auf Nachrichten mit bestimmter CT-ID	1
7	Schnittstelle zu einem Auswerterechner mit Bandbreite von min. 7 Mbit/s	1
8	Verwendung von einem transparenten Sniffer, der Datenpakete dupliziert	1

Tabelle 3.1: Anforderungen zur Messung bei der Qualitätsbestimmung eines Senders

3.1.2 Messen von Paketlaufzeiten in einem Netzwerk

Mit dieser Messung sollen Paketlaufzeiten und deren Jitter in einem Netzwerk ermittelt werden. Dabei sollen der Zeitpunkt des Absendens eines Gerätes und der Zeitpunkt des Empfangs eines anderen Gerätes gemessen werden. Der für diese Messung notwendige Messaufbau ist in Abbildung 3.2 dargestellt. Die Differenz der gemessenen Zeiten ist die Durchlaufzeit des Paketes in einem Netzwerk. Anhand der Messergebnisse kann dann verifiziert werden, ob eine zeitkritische Nachricht rechtzeitig beim Empfänger ankommt und somit die Echtzeitbedingungen erfüllt werden. Hierbei soll gewählt werden, ob in der synchronisierten Netzwerkzeit oder in der eigenen Systemzeit gemessen wird. Auch bei dieser Messung soll das Setzen eines Filters möglich sein, der die Messung auf bestimmte CT-IDs einschränkt. Wie bei der im vorangegangenen Kapitel beschriebenen Messung ist es hier notwendig, die Datenpakete zu duplizieren. Das Duplizieren der Nachrichten soll auch hier transparent sein. Das bedeutet, dass die Sniffer das Zeitverhalten der zu messenden Nachrichten nicht beeinflussen darf.

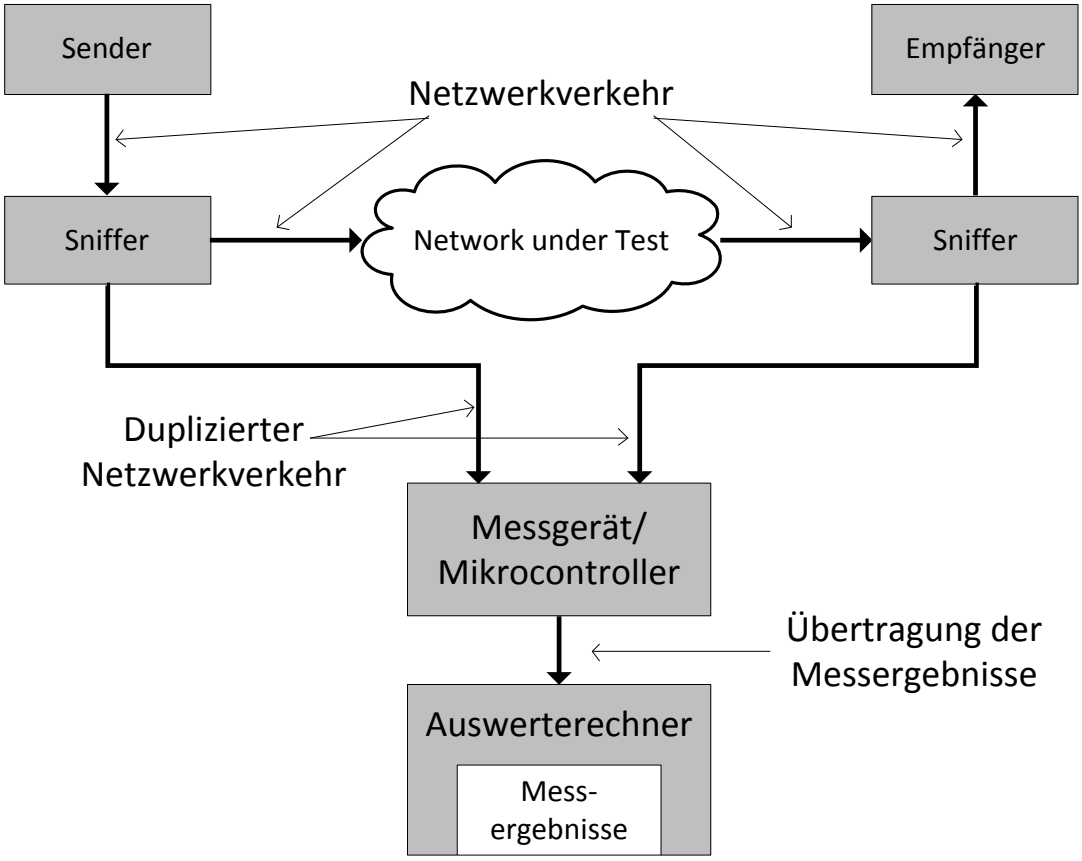

Abbildung 3.2: Schematischer Messaufbau zur Latenzmessung eines Netzwerks

Anforderungen zum Sniffer werden im Kapitel 3.4 aufgestellt.

Beide Zeiten (Sende- und Empfangszeit) sollen auf nur einem Mikrocontroller aufgezeichnet werden. Das soll dazu führen, dass die Zeiten vom gleichen System und somit von derselben Uhr aufgezeichnet werden.
Da bei dem Messaufbau zur *Qualitätsbestimmung eines Senders* schon eine hohe Messgenauigkeit gefordert ist, wird hier keine höhere Anforderung zur Messgenauigkeit aufgestellt. Es bleibt bei der Anforderung von mindestens 1µs und wünschenswerten 0,1µs. Eine Schnittstelle zu einem Auswerterechner wird bei diesem Messaufbau ebenfalls benötigt. Bei dieser Messung werden Pakete an zwei Stellen abgegriffen. Es soll dabei möglich sein die Paketinhalte beider Messpunkte an den Auswerterechner zu übertragen. Der Inhalt der Pakete muss mit übertragen werden, um diese auswerten zu können. Beispielsweise sind bei Zeitsynchronisationsnachrichten der im Datenfeld eingetragene Delay für eine Auswertung interessant. Dadurch verdoppelt sich die Anforderung für die Mindestbandbreite gegenüber der Messung

zur *Qualitätsbestimmung eines Senders*. Aus dem Grund wird die Anforderung Nr. 7 von 7Mbit/s auf 14Mbit/s geändert.
Um den kompletten Datenverkehr auswerten zu können, muss der gesamte Datenverkehr zum Auswerterechner übertragen werden. Dazu ist eine Bandbreite von 2 * 100 Mbit/s notwendig. Zu dieser Bandbreite müsste noch die Übertragung der Messzeiten hinzugerechnet werden. Jedoch erscheint eine 200 Mbit/s Schnittstelle an dieser Stelle für einen Mikrocontroller unrealistisch und so wird diese Anforderung als eine „nice to have“ -Anforderung aufgenommen.

In der Tabelle 3.2 werden die im vorangegangen Text beschriebenen Anforderungen redundanzfrei zum vorherigen Kapitel zusammengefasst.

Nr.	Anforderung	Prio.
7	(Geändert) Schnittstelle zu einem Auswerterechner mit Bandbreite von min. 14 Mbit/s	1
9	Messung von Paketlaufzeiten in einem Netzwerk	1
10	Messung des Jitters von Paketlaufzeiten in einem Netzwerk	1
11	Zeitaufnahme beider Messpunkte von einer Uhr	1
12	Schnittstelle zu einem Auswerterechner mit Bandbreite von min. 200 Mbit/s	2

Tabelle 3.2: Anforderungen Messung von Latenzen eines Netzwerks

Anhand der Anforderung die bis jetzt gestellt wurden, ist deutlich geworden welche Art von Hardware für die Messaufbauten benötigt wird. In den folgenden Kapiteln werden Anforderungen an diese Hardware aufgestellt.

3.2 Mikrocontroller

Im Folgenden werden Anforderungen an den verwendeten Mikrocontroller aufgestellt

- Der Mikrocontroller muss aufgrund der geforderten Messaufbauten zwei NIC (Network Interfaces) besitzen. Diese müssen den 100 Mbit oder den 1 Gbit Bandbreite unterstützen, weil diese vom TTEthernet-Standard gefordert wird.

- Der verwendete Mikrocontroller muss eine Einheit besitzen, die den Ankunftszeitpunkt von Paketen stempelt. Aus der Qualität des Stempels ergibt sich die Messgenauigket. Daher soll die Genauigkeit des Stempels die Anforderung Nr. 4 und Nr. 5 erfüllen.
  Der Deserialisierungszeitpunkt in dem gestempelt wird, sollte konstant paketgrößenunabhängig sein. Wird z. B. zum Zeitpunkt der Deserialisierung der Ziel-MAC-Adresse gestempelt, so muss dies bei allen Paketgrößen der Fall sein. Diese Anforderung wird gestellt, weil bei einer verwandten Arbeit (vgl. Bartols, 2010) festgestellt wurde, dass einige Netzwerkkarten unterschiedliche Caches für unterschiedlich große Datenpakete besitzen. Die Caches für kleine Pakete waren schnell, die für große Pakete waren langsamer. Das führte dazu, dass das Abrufverhalten der Nachricht in Abhängigkeit zur

Paketgröße nicht linear verlief. Es ist zwar unwahrscheinlich, dass Ethernetport von Mikrocontroller unterschiedliche Caches besitzen, dennoch soll hier sichergestellt werden, dass die Zeitstempeleinheit paketgrößenunabhängig arbeitet.

- Der Mikrocontroller muss eine weitere Kommunikationsschnittstelle besitzen. Diese soll dazu genutzt werden alle gemessenen Nachrichten mit deren Inhalt zu einem Auswerterechner zu übertragen. Die Bandbreite, die diese Schnittstelle liefern muss, wurde in dem vorangegangenen Kapitel besprochen.

Aus dem vorangegangenen Text ergeben sich Anforderungen, die in Tabelle 3.3 dargestellt sind.

Nr.	Anforderung	Prio.
13	Unterstützung der 100 Mbit Bandbreite	1
14	Unterstützung der 1 Gbit Bandbreite	2
15	Zeitstempeleinheit, die den Ankunftszeitpunkt eines Datenpakets stempelt	1
16	Jitter des Stempelzeitpunkts soll kleiner als 1µs sein	1
17	Paketgrößenunabhängiger Stempelzeitpunkt	1
18	Eine weitere Schnittstelle zur Übertragung der Messdaten muss vorhanden sein	1

Tabelle 3.3: Anforderungen Mikrocontroller

3.3 Anforderungen an den Auswerterechner

Der Auswerterechner muss eine Schnittstelle besitzen, die mit der Schnittstelle des Mikrocontrollers kompatibel ist. Über diese Schnittstelle werden Messergebnisse und die Paketinhalte übertragen.
In dem Auswerterechner muss ein Programm die Messdaten in Empfang nehmen und diese in einem mit der Anwendung Wireshark (Combs, 2010) kompatiblem Dateiformat abspeichern. In Tabelle 3.4 werden die Anforderungen für den Auswerterechner zusammengefasst.

Nr.	Anforderung	Prio.
19	Mikrocontrollerkompatible Schnittstelle zur Übertragung der Messergebnisse	1
20	Darstellung der Messergebnisse in Wireshark	1
21	Die Messergebnisse soll die Zeitbasis des Mikrocontrollers haben	2

Tabelle 3.4: Anforderungen Auswerterechner

3.4 Sniffer für Netzwerkverkehr

In diesem Kapitel werden Anforderungen an die im Messaufbau verwendeten Geräte zum Sniffen von Nachrichten aufgestellt. Für die Messungen im Time-Triggered Ethernet ist es wichtig, dass Datenpakete durch ein solches Gerät nicht verzögert werden. Das bedeutet, dass diese Geräte keine oder nur eine geringe Latenz aufweisen dürfen. Zeitkritische Nachrichten müssen während des Messvorgangs im Schedule vorgegebenen Zeitfenster bei einer Switch eintreffen. Der Switch verhält sich dabei so, dass die Nachricht immer zum gleichen Zeitpunkt weitergeleitet wird, egal zu welchem Zeitpunkt im Zeitfenster diese eingetroffen ist. Es ist schwierig hier genaue Grenzen für die Latenz des Sniffers zu setzten, weil diese Anforderung von Zeitfenster in der Switch und dem Jitter des Absenders abhängt. Angenommen das Zeitfenster in der Switch wäre $\pm$10µs groß und der Sender jittert mit $\pm$5µs um den im Schedule vorgesehenen Absendezeitpunkt. Dann dürfte die Latenz des Sniffers max. 5µs betragen. Allerdings ist das Jittern des Senders genau das, was gemessen werden soll und somit unbekannt. Würde der Sender mit $\pm$10µs jittern, so wäre keine Latenz erlaubt.
Zum jetzigen Zeitpunkt wird darauf verzichtet genaue Grenzen für die Latenz und den Jitter des Sniffers zu setzten. Das sniffen von Netzwerkverkehr könnte mit einem Test-Access-Point (TAP Funktionsweise siehe Kapitel 4.2 auf Seite 27) durchgeführt werden. Einige Hersteller werben damit, dass ihre TAPs eine “zero latenzy” haben. Sollte sich dies im Analyseteil dieser Bachelorarbeit bewahrheiten, so wäre dass setzten von genauen Grenzen für Latenz und Jitter hinfällig.

Aus dem vorangegangenen Text ergeben sich Anforderungen, die in Tabelle 3.5 dargestellt sind.

Nr.	Anforderung	Prio.
22	Sniffer mit geringer/keiner Latenz	1
23	Sniffer mit geringem/keinem Jitter	1

Tabelle 3.5: Anforderungen Sniffer

3.5 Zusammenfassung

In diesem Kapitel wurden Anforderungen an die Messungen und die Hardware aufgestellt. Dadurch sind auch die Messaufbauten, die für die beiden Messungen notwendig sind, konkretisiert worden.

Die Tabelle 3.6 fasst alle aufgestellten Anforderungen zusammen.

Nr	Anforderung	Prio.
Anforderungen: Qualitätsbestimmung eines Senders		
1	Messung des Absendezeitpunkts eines Gerätes in der Zeitbasis des Netzwerks	1
2	Messung des Jitters des Absendezeitpunkts in der Zeitbasis des Netzwerks	1
3	Zeitsynchronisation mit der Netzwerkzeit	1
4	Messgenauigkeit von 0,1µs	2
5	Messgenauigkeit von 1µs	1
6	Filterfunktion auf Nachrichten mit bestimmter CT-ID	1
7	Schnittstelle zu einem Auswerterechner mit Bandbreite von min. 14 Mbit/s	1
8	Verwendung von einem transparenten Sniffer, der Datenpakete dupliziert	1
Anforderungen: Messung von Latenzen im Netzwerk		
9	Messung von Paketlaufzeiten in einem Netzwerk	1
10	Messung des Jitters von Paketlaufzeiten in einem Netzwerk	1
11	Zeitaufnahme beider Messpunkte von einer Uhr	1
12	Schnittstelle zu einem Auswerterechner mit Bandbreite von min. 200 Mbit/s	2
Anforderungen: Mikrocontroller		
13	Unterstützung der 100 Mbit Bandbreite	1
14	Unterstützung der 1 Gbit Bandbreite	2
15	Zeitstempeleinheit, die den Ankunftszeitpunkt eines Datenpakets stempelt	1
16	Jitter des Stempelzeitpunkts soll kleiner als 1µs sein	1
17	Paketgrößenunabhängiger Stempelzeitpunkt	1
18	Eine weitere Schnittstelle zur Übertragung der Messdaten muss vorhanden sein	1
Anforderungen: Auswerterechner		
19	Mikrocontrollerkompatible Schnittstelle zur Übertragung der Messergebnisse	1
20	Darstellung der Messergebnisse in Wireshark	1
21	Zeitbasis des Mikrocontrollers in den Messergebnissen	2
Anforderungen: Sniffer		
22	Sniffer mit geringer/keiner Latenz	1
23	Sniffer mit geringem/keinem Jitter	1

Tabelle 3.6: Anforderungstabelle

Kapitel 4

Analyse

In diesem Kapitel werden Hardwarekomponenten auf Ihre Verwendbarkeit analysiert. Anschließend werden die im Kapitel 3 aufgestellten Anforderungen mit den Analyseergebnissen verglichen, um die Verwendbarkeit festzustellen.

4.1 Mikrocontroller

In diesem Kapitel wird der Mikrocontroller *NXHX500-ETM* des Herstellers Hilscher (Hilscher Gesellschaft für Systemautomation mbH) analysiert.
Dieser Mikrocontroller rechnet mit einer 32 Bit ARM9-CPU, die mit 200 MHz getaktet ist. Das restliche System wird mit 100 MHz getaktet (vgl. Hilscher, 2008, S. 6). Der Mikrocontroller bietet diverse Schnittstellen an. Im Folgenden werden Schnittstellen beschrieben, die für die Messungen in Frage kommenden könnten.

Ethernet Der Mikrocontroller bietet zwei Ethernetschnittstellen, die den 100 BASE.TX Standard unterstützen. Als Anschluss dienen RJ45-Schnittstellen. Besonderheit hier ist, dass beide Ethernetports einen eigenen Mikrokern besitzen, die unabhängig von der ARM-CPU arbeiten. Das führt dazu, dass Pakete empfangen und im Speicher abgelegt werden, ohne die ARM-CPU zu belasten (vgl. Hilscher, 2008, S. 7). Die ARM-CPU kann dann die empfangenen Datenpakete aus dem Speicher ausgelesen. Gleiches gilt für den Sendevorgang. Die ARM-CPU legt ein Datenpaket an einen bestimmten Speicherort ab und signalisiert dann dem Mikrokern das Paket zu senden. Ab diesem Zeitpunkt wird die ARM-CPU mit keiner Rechenleistung belastet. Zudem besitzen beide Schnittstellen eine Zeitstempeleinheit, die die Ankunftszeit der Pakete aufzeichnet. Diese Stempeleinheit wird im Kapitel 4.1.1 analysiert.

Dual-Port-Memory Der Mikrocontroller verfügt über eine Dualport-Memory-Schnittstelle (DPM), mit der ein externes Gerät auf beliebigen Speicher parallel zugreifen kann. Es lassen sich bis zu acht verschieden Speicherbereiche an beliebigen Speicherorten

festlegen, die auf einen 64 kByte großen Dualport-Memory Bereich abgebildet werden können. Von diesem Bereich werden 512 Byte statisch auf Konfigurationsregister abgebildet (vgl. Hilscher, 2008, S. 42). Eine konkrete Bandbreitenangabe konnte der Hersteller leider nicht machen. Im Datenblatt wird jedoch erklärt, dass die Zugriffszeit von dem Speicherbereich, auf den zugegriffen wird, abhängig ist (vgl. Hilscher, 2008, S. 56). Aus dem Grund wird die Dualport-Memory Bandbreite für den Zugriff auf den internen SRAM, in den die Ethernetports Datenpakete ablegen, in Kapitel 4.1.2 analysiert.

USB Der Mikrocontroller verfügt über eine USB 1.1 Schnittstelle, die den Low-Speed Modus mit der Bandbreite 1,5 Mbit/s und den Full-Speed Modus mit der Bandbreite 12 Mbit/s unterstützt (vgl. Hilscher, 2008, S. 73). Die USB-Schnittstelle wird als Möglichkeit zur Übertragung der Messdaten gesehen und wird im Kapitel 4.1.2 diskutiert.

Multi Media Card Der Mikrocontroller verfügt über einen MMC-Steckplatz (Multi Media Card). Auf die MMC-Karte kann der Programmcode gespeichert werden, so, dass der Mikrocontroller davon booten kann. Die MMC-Karte kann allerdings auch zum Ablegen und Auslesen von Daten genutzt werden. So ist es denkbar, dass eine MMC-Karte zum Abspeichern von Messdaten genutzt werden kann. Diese Möglichkeit wird in Kapitel 4.1.2 diskutiert.

Synchronisation

In der Arbeitsgruppe CoRE (*Communication over Realtime Ethernet*) wurde ein Softwaremodul entwickelt, das die Systemzeit mit der Netzwerkzeit synchronisiert. Mit der Netzwerkzeit ist hier die Zeit gemeint, auf die sich alle Time-Triggered-Komponenten im Netzwerk synchronisieren. Dieses Softwaremodul bietet eine Schnittstelle, in der die Zykluszeit des Netzwerks und Filter für Time-Triggered-Nachrichten eingestellt werden können (vgl. Müller, 2011).

Zusammenfassung

In diesem Kapitel wurde der Mikrocontroller mit seinen Eigenschaften vorgestellt. Der Mikrocontroller bietet viele Kommunikationsschnittstellen, die sich für die Übertragung der Messergebnisse eignen könnten. Diese Schnittstellen werden in Kapitel 4.1.2 diskutiert und analysiert. Ein großer Vorteil ist, dass für diesen Mikrocontroller ein fertiges Zeitsynchronisationsmodul verwendet werden kann. Der Mikrocontroller bietet eine Zeitstempeleinheit, mit der die Ankunft eines Datenpakets mit der synchronisierten Systemzeit aufgezeichnet werden kann. Diese Einheit wird im Kapitel 4.1.1 analysiert.

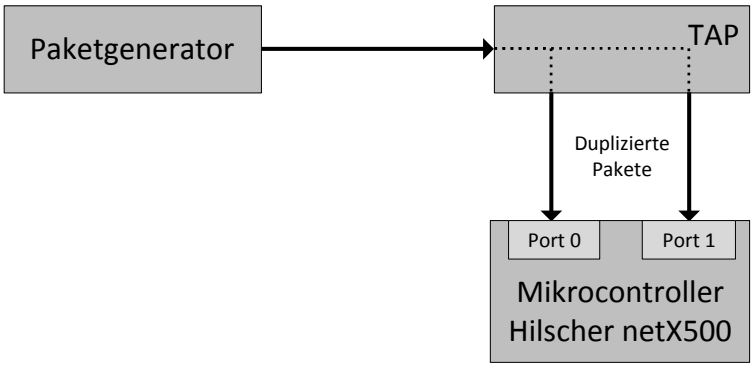

Abbildung 4.1: Messaufbau: Analyse des Zeitstemepels auf Jitter

4.1.1 Zeitstempel

Die Dokumentation des Mikrocontrollers beschreibt eine Zeitstempeleinheit welches den Eintreffzeitpunkt eines Pakets stempelt (vgl. Hilscher, 2009, S. 119). In diesem Kapitel wird die Qualität der Zeitstempeleinheit analysiert. Es wird geprüft, ob die Datenpakete immer zum gleichen Deserialisierungszeitpunkt gestempelt werden und ob unterschiedliche Paketgrößen den Zeitstempel beeinflussen.

Messung des Jitters

In Abbildung 4.1 wird der für die Analyse verwendete Messaufbau dargestellt. Es wurde ein Paketgenerator verwendet, der zyklisch Datenpakete mit unterschiedlichen Paketgrößen absendet.
Es wurden unterschiedliche Paketgrößen jeweils 200 mal abgesendet. Dabei wurden Paketgrößen verwendet, die im RFC-2544 zur Leistungsmessung von Netzwerken vorgeschlagen werden (vgl. RFC2544, 1999).
Der Generator ist an einer TAP (Test Access Point) angeschlossen, der hier zum duplizieren der Pakete verwendet wird. Die beiden Ethernetports des Mikrocontrollers sind ebenfalls an der TAP angeschlossen, sodass beide die gleichen Pakete des Generators empfangen.
Um auszuschließen, dass die beiden Ausgänge der TAP eine unterschiedliche Latenz haben, wurde die Messung ein zweites Mal durchgeführt, wobei die Anschlüsse an der TAP vertauscht wurden.

Das Ergebnis der Messung ist, dass beide Zeitstempel über alle Messungen die gleichen Werte haben. Das Tauschen der Netzwerkleitungen an der TAP hat das Ergebnis nicht beeinflusst. Da beide Ethernetports bei gleicher Ankunft von Paketen, die gleiche Zeit Stempeln, kann hier davon ausgegangen werden, dass die Zeitstempeleinheit immer zum gleichen Deserialisierungzeitpunkt die Ankunftszeit aufzeichnet. Die Systemzeit, mit der die Zeitstempeleinheit

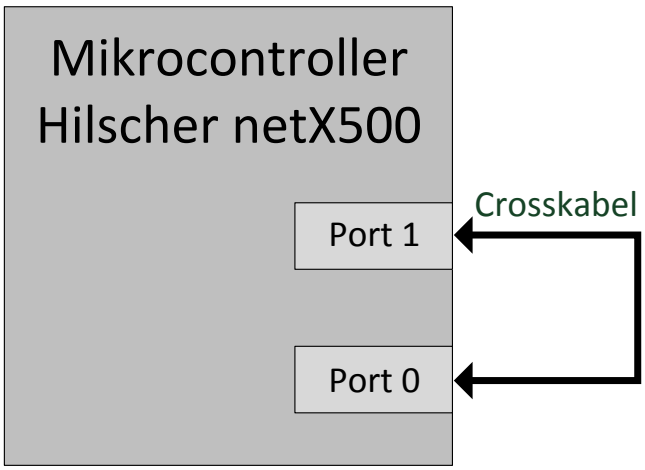

Abbildung 4.2: Messaufbau: Analyse Zeitstempeleinheit

aufzeichnet, wird mit 100MHz getaktet. Das entspricht einer Zeitgranularität von 10ns. Somit liegt der Jitter für das Stempeln bei < 10ns.

Mit dieser Messung wurde der Jitter der Zeitstempeleinheit gemessen, jedoch wurde hiermit nicht festgestellt, ob die Stempeleinheit paketgrößenunabhängig arbeitet, weil bei dieser Messung zu einem Zeitpunkt an beiden Port gleichgroße Pakete gemessen wurden. Auch ist die Verzögerung des Stempelvorgangs nicht bekannt. Also die Zeit zwischen physikalischen Eintreffzeitpunkt und dem Stempelvorgang. Die Verzögerung und die Paketgrößenabhängigkeit werden im Folgenden besprochen.

Feststellung der Paketgrößenabhängigkeit

In Abbildung 4.2 wird, der für diese Messung verwendete Messaufbau, dargestellt. Es wurden beide Ethernetports mit einem Crosskabel verbunden. Anschließend wurden Datenpakete von Port0 nach Port1 und umgekehrt gesendet und folgende Zeiten aufgezeichnet:

- $t_{wurdeGesendetIRQ}$: Zeitpunkt der ersten Programmzeile in der ISR, die aufgerufen wird, nachdem ein Paket vollständig versendet wurde.
- $t_{timeStamp}$: Zeitstempel eines eingegangenen Pakets.

Es wird die Differenz der beiden Zeiten definiert:

$$\Delta t_{IrqTs} = t_{wurdeGesendetIRQ} - t_{timeStamp}$$

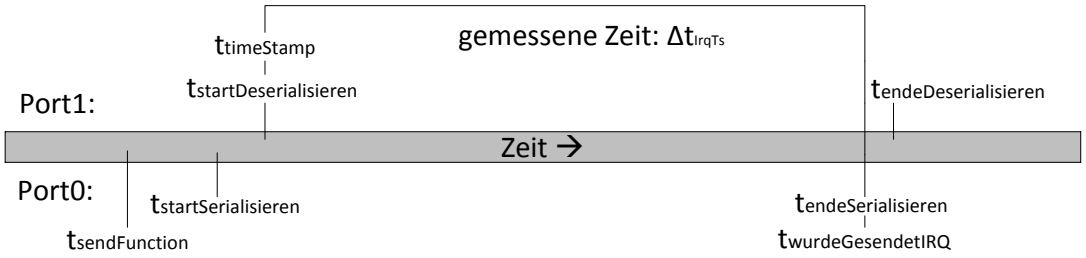

Abbildung 4.3: Schematischer Zeitverlauf einer Messung zur Analyse des Zeitstempels

Abbildung 4.3 zeigt den schematischen, erwarteten Zeitverlauf dieser Messung. Aus dieser Abbildung lässt sich entnehmen, dass die gemessenen Zeiten Δt_{IrqTs} nahe der Serialisierungszeit liegen sollte.
Die Messung wurde nach dem RFC-2544 Standard für Leistungsmessung in Netzwerken durchgeführt (vgl. RFC2544, 1999). In diesem Standard werden Paketgrößen 64, 128, 256, 512, 1024, 1280, 1518 Byte empfohlen.
Es wurde pro Paketgröße 200 Mal gemessen, um Jitter ausfindig zu machen. Die Tabelle 4.1 zeigt die gemessenen Zeiten Δt_{IrqTs} mit der Häufung, wie oft ein Wert von 200 Messungen gemessen wurde. Das Messergebnis zeigt, dass der Jitter maximal 100 ns beträgt. Dieser ist bei der Paketgröße 256 Byte zu sehen (22210 ns - 22110 ns = 100 ns).

Da im vorangegangenen Abschnitt festgestellt wurde, dass die Zeitstempeleinheit nicht jittert, kann hier davon ausgegangen werden, dass nicht die gemessene Zeit $t_{timeStamp}$, sondern die Zeit $t_{wurdeGesendetIRQ}$: jittert.

Paketgröße							
64 Byte		128 Byte		256 Byte		512 Byte	
Δt_{IrqTs}[ns]	Häufung	Δt_{IrqTs}[ns]	Häufung	Δt_{IrqTs}[ns]	Häufung	Δt_{IrqTs}[ns]	Häufung
6690	11	11870	26	22110	33	42590	48
6700	81	11880	46	22120	53	42600	76
6710	30	11890	51	22130	35	42610	42
6720	25	11900	38	22140	30	42620	19
6730	18	11910	20	22150	26	42630	12
6740	35	11920	18	22160	20	42640	3
		11930	1	22170	0		
				22180	1		
				22190	1		
				22200	0		
				22210	1		

Paketgröße					
1024 Byte		1280 Byte		1518 Byte	
Δt_{IrqTs}[ns]	Häufung	Δt_{IrqTs}[ns]	Häufung	Δt_{IrqTs}[ns]	Häufung
83550	30	104030	23	123070	40
83560	103	104040	73	123080	70
83570	24	104050	28	123090	28
83580	3	104060	18	123100	8
83590	17	104070	22	123110	27
83600	23	104080	35	123120	25
		104090	0	123130	1
		104100	1	123140	0
				123150	0
				123160	1

Tabelle 4.1: Messergebnisse: Analyse Paketgrößenunabhängigkeit

Das Minimum der Zeit Δt_{IrqTs} je Paketgröße ist in Bezug zur Paketgröße im Graphen in Abbildung 4.4 dargestellt. An dem Graphen ist zu erkennen, dass die Zeit Δt_{IrqTs} linear mit der Paketgröße steigt. Die Steigung der Geraden beträgt: $\frac{123070ns-6690ns}{1518Byte-64Byte} = 80,04\frac{ns}{Byte}$
Sollsteigung in einem 100Mbit/s Netzwerk ist: $80\frac{ns}{Byte}$
Relative Abweichung: $\frac{80ns/Byte}{80,04ns/Byte} => 0,05\%$
Die Abweichung vom zum rechnerischen Wert ist sehr gering, womit nachgewiesen ist, dass die Zeitstempeleinheit paketgrößenunabhängig zum gleichen Deserialisierungszeitpunkt stempelt.

Mit den bisherigen Analysen konnte festgestellt werden, dass die Zeitstempeleinheit einen Jitter von < 10ns hat und paketgrößenunabhängig stempelt. Dies bedeutet, dass die Verzö-

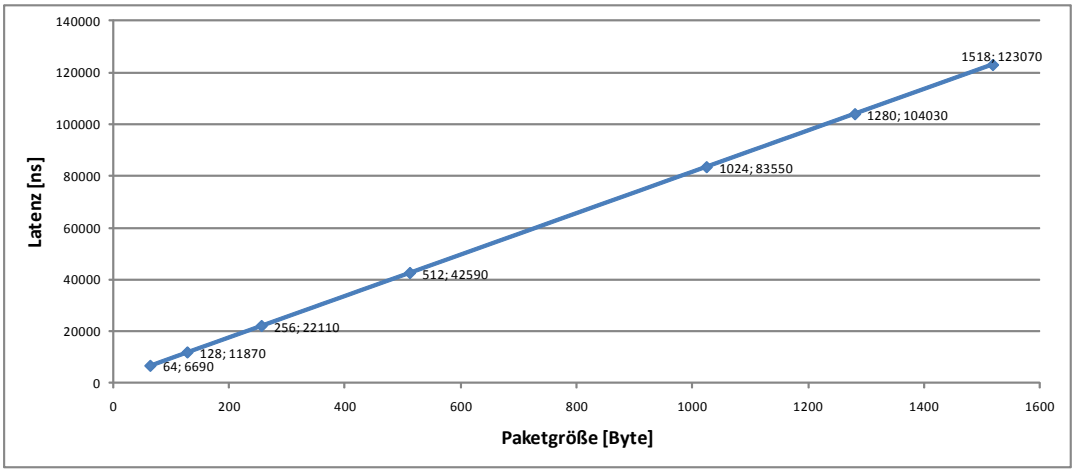

Abbildung 4.4: Diagramm: Paketgrößenunabhängigkeit; Steigung der Latenz in Bezug zur Paketgröße

gerung zwischen dem physikalischen Eintreffen der Datenpakete und dem Stempeln der Zeit immer konstant ist.
Allerdings konnte die Verzögerung bis gestempelt wird nicht ermittelt werden. Anhand der Messergebnisse lässt sich ablesen, dass die Gemessenen Δt_{IrqTs} Zeiten nahe der Serialisierungszeit liegt. Daraus lässt sich schließen, dass diese Verzögerung gering ist.

Anmerkung zur Messung

Diese Messung wurde mit eingeschaltetem Daten- und Instructioncache durchgeführt. Das Ausschalten des Instruction-Cache führte entgegen den Erwartungen zu einem höheren Jitter von 400 ns. Eine mögliche Erklärung, warum der Jitter bei eingeschaltetem Instruction-Cache kleiner ist, ist die Größe des 16kByte-Caches (vgl. Hilscher, 2009, S. 31). Während der Messung wurde nur wenig Programmcode ausgeführt. Es ist also möglich, dass alle Instruktionen im Cache lagen, und somit nach dem erstmaligen Ausführen mit nahezu konstanter Zeit auf die Instruktionen im Cache zugegriffen werden konnte.
Das Ausschalten des Daten-Cache verändert das Ergebnis nur gering. Mit ausgeschaltetem Daten-Cache entstanden gelegentlich Ausreisser in den Messungen, die zur Verdopplung des Jitter führten. Mit eingeschaltetem Daten-Cache erhielt man gar keine Ausreißer in der Messung.

4.1.2 Datenübertragung zum Auswerterechner

Der *NXHX500-ETM*-Mikrocontroller bietet viele Schnittstellen, die für die Übertragung der Messdaten zum Auswerterechner in Frage kommen könnten. In diesem Kapitel werden diese Schnittstellen analysiert.

Dual-Port-Memory Die Dualport-Memory Schnittstelle erlaubt es zur Laufzeit verschiedene Speicherbereiche des Mikrocontrollers auszulesen und zu beschreiben. Es lassen sich bis zu acht verschiedene Speicherbereiche festlegen. Alle Speicherbereiche dürfen in der Summe 64kByte groß sein. 512 Byte des 64kByte Bereichs werden statisch auf ein Konfigurationsregister abgebildet (vgl. Hilscher, 2008, S. 42). Es lassen sich also maximal 63,5kByte auf Nutzdaten abbilden. Einem Auswerterechner erscheint dieser Speicher als ein linearer 64kByte Speicherbereich auf dem PCI-Bus.

Den beiden Ethernetports sind jeweils ein 32kByte großer Speicherbereich zugewiesen, in denen die Datenpakete abgelegt werden. Die Datenpakete werden in 1560Byte Schritten gespeichert. Die ersten 1560Byte werden für die Speicherung der Interrupt Vektortabelle freigehalten und werden nicht von den Ethernetports verwendet. Der verwendete Speicherbereich für einen Ethernetport kann nun folgendermaßen errechnet werden:

$$\lfloor \frac{32768Byte}{1560Byte} \rfloor * 1560Byte - 1560Byte = 31200Byte$$

Für beide Ethernetports ergibt sich somit ein Speicherbereich von 62400Byte. Dieser Speicherbereich kann also in den 63,5kByte großen Dual-Port-Memory-Bereich abgebildet werden. Es können dann 2624Byte für Metadaten, die bei der Übertragung notwendig werden könnten, verwendet werden.
Die Bandbreite dieser Schnittstelle hängt vom Speicher ab auf den zugegriffen wird. In der Dokumentation sind für die einzelnen Speicher keine Bandbreiten angegeben. Die Dual-Port-Memory-Schnittstelle würde sich also für die Übertragung der Messergebnisse eignen, wenn die Bandbreite für den Zugriff auf den internen SRAM der Anforderung Nr. 5 aus dem Kapitel 3 genügen würde. Dies wird am Ende dieses Kapitels analysiert.

USB Der Mikrocontroller bietet eine USB-Schnittstelle, mit der im Full-Speed Modus 12 Mbit/s übertragen werden könnten. Die Bandbreite würde die Anforderung Nr. 7 aus Kapitel 3 nicht erfüllen. Dazu müsste der Mikrocontroller aktiv die Daten versenden. Die USB-Schnittstelle hat keinen eigenen Mikrokern, wie einige andere Schnittstellen und müsste somit die ARM-CPU beim Senden über USB beanspruchen. Mit dieser Schnittstelle können die aufgestellten Anforderungen nicht erfüllt werden.

Multi Media Card Der Mikrocontroller besitzt eine Steckplatz für eine Multimedia Karte (MMC). Eine Alternative wäre es, die Messergebnisse auf eine MMC zu speichern und nach den Messungen auf den Auswerterechner zu spielen. Allerdings hängt hier die Übertragungsrate stark von der verwendeten MMC Karte ab. Dazu kommt, dass der Speicherplatz begrenzt ist und somit die Messdauer eingeschränkt wäre. Wie bei der USB-Schnittstelle wäre die CPU des Mikrocontrollers aktiv an der Speicherung beteiligt, was die Rechenleistung beansprucht. Diese Nachteile führen dazu, dass diese Alternative in diese Bachelorarbeit nicht analysiert wird.

Im folgenden Kapitel wird die Bandbreite der Dual-Port-Memory Schnittstelle analysiert.

Dual Port Memory

In diesem Kapitel wird die Bandbreite der Dual-Port-Memory Schnittstelle analysiert. Um eine Verbindung mit dem Auswerterechner herstellen zu können, wird die PCI-Karte *NXPCA-PCI* des Mikrocontrollerherstellers verwendet. Der für die Messung aufgebaute Messaufbau ist in Abbildung 4.5 dargestellt. Für die Bandbreitenmessung wurden zwei von den acht zur Verfügung stehenden Dualport-Mappingbereiche verwendet. Der erste Speicherbereich wurde auf einen linearen 32kByte großen Speicherbereich im internen SRAM abgebildet, den auch die Ethernetports zur Speicherung der Datenpakete nutzen (vgl. Hilscher, 2008, S. 97). Der zweite Speicherbereich wurde auf die Register abgebildet, bei denen die Systemzeit des Mikrocontrollers abgelesen werden kann. Vom Auswerterechner aus wurde nun die Systemzeit zwei Mal direkt hintereinander ausgelesen, um die Dauer für das Auslesen der Systemzeit zu kennen.
Anschließend wurde Daten verschiedener Größen (siehe Tabelle 4.2) aus dem ersten abgebildeten Speicherbereich gelesen. Direkt vor und nach dem Lesen wurde die Systemzeit gelesen. Jede Messung wurde 50 Mal durchgeführt. Anschließend wurde von den gemessen Systemzeiten die Differenz gebildet und davon die Dauer für das Auslesen der Systemzeit abgezogen. Somit erhält man die Zeit für das Auslesen der einzelnen Speicherbereich, woraus sich die Bandbreite errechnen lässt. In Tabelle 4.2 werden die Messergebnisse dargestellt.

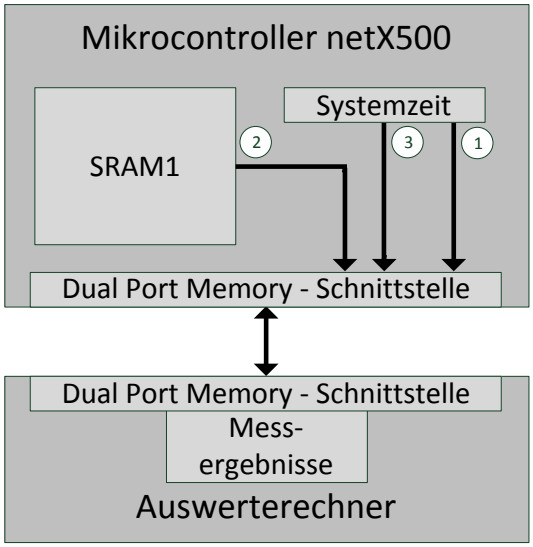

Abbildung 4.5: Messaufbau: DPM-Schnittstelle Bandbreitenmessung

Gelesene Bytes	Median Zeitdifferenz	Errechnete Bandbreite
60 Byte	19.680 ns	24.390.244 Bits/s
100 Byte	34.180 ns	23.405.500 Bits/s
200 Byte	65.570 ns	24.401.403 Bits/s
300 Byte	98.175 ns	24.446.142 Bits/s
400 Byte	130.990 ns	24.429.346 Bits/s
500 Byte	163.775 ns	24.423.752 Bits/s
600 Byte	196.425 ns	24.436.808 Bits/s
700 Byte	229.285 ns	24.423.752 Bits/s
800 Byte	261.920 ns	24.434.942 Bits/s
900 Byte	294.645 ns	24.436.186 Bits/s
1000 Byte	327.400 ns	24.434.942 Bits/s
1100 Byte	360.110 ns	24.436.978 Bits/s
1200 Byte	392.730 ns	24.444.275 Bits/s
1300 Byte	425.590 ns	24.436.664 Bits/s
1400 Byte	458.225 ns	24.442.141 Bits/s
1500 Byte	491.040 ns	24.437.928 Bits/s
1600 Byte	523.705 ns	24.441.241 Bits/s
1700 Byte	556.460 ns	24.440.211 Bits/s
1800 Byte	589.260 ns	24.437.430 Bits/s
1900 Byte	621.820 ns	24.444.373 Bits/s
2000 Byte	654.695 ns	24.438.861 Bits/s
4000 Byte	1.309.195 ns	24.442.501 Bits/s
6000 Byte	1.964.115 ns	24.438.488 Bits/s

Tabelle 4.2: Messergebnisse: Dualport-Memory Bandbreite

Die Ergebnisse zeigen, dass die Bandbreite zwischen 23,4 Mbit/s und 24,4 Mbit/s liegt und somit die Anforderung Nr. 7 aus Kapitel 3 erfüllt wird. Die Dual-Port-Memory Schnittstelle ist somit für die Messungen, die im Anforderungskapitel beschreiben sind, geeignet. Dazu ist zusagen, dass die Bandbreite unter dem Betriebssystem *Windows XP* durchgeführt wurde, welches kein Echtzeitbetriebssystem ist und somit auch kein konstantes Verhalten garantiert werden kann, was auch den kleinen Ausreißer in den Messergebnissen bei 100 Byte erklären könnte.

4.2 Netzwerk Sniffer

Wie im Kapitel 3.4 besprochen, werden für den Messaufbau zwei Geräte benötigt, die den Netzwerkverkehr sniffen. Diese Geräte müssen für das Netzwerk transparent sein und somit möglichst keine Latenz und keinen Jitter aufweisen.
In diesem Kapitel werden Alternativen diskutiert, die das Sniffen von Netzwerkverkehr ermöglichen. Anschließend wird eine dieser Alternativen analysiert.

4.2.1 Alternativen

Hub

Eine Hub ist ein Netzwerkgerät, welches Datenpakete an alle Netzwerkteilnehmer weiterleitet. Es arbeitet auf dem Layer 1 im OSI-Modell und es ist theoretisch möglich, dass eine Hub nahezu latenzfrei arbeiten könnte, weil Pakete bei Ankunft direkt weitergeleitet werden können, ohne abzuwarten bis sie komplett oder ein Teil davon eingetroffen ist. Jedoch Besitzen Hubs den Nachteil, dass über diese nur eine Half-Duplex Kommunikation stattfinden kann. Das Time-Triggered-Ethernet fordert jedoch eine Full-Duplex Kommunikation, somit kann ein Hub nicht für die Messungen verwendet werden.

Switch mit Monitorport

Ein Switch ist ein Netzwerkgerät, das Datenpakete an Netzwerkteilnehmer weiterleitet, an die diese adressiert sind. Pakete werden also nicht wie bei einer Hub an alle Teilnehmer weitergeleitet. Einige Switches besitzen einen konfigurierbaren Monitorport, der sich so konfigurieren lässt, dass Datenpakete, die an oder von einem Port kommen, an diesen Monitorport dupliziert weitergeleitet werden. So könnte man sich vorstellen, Netzwerkports auf diese Art zu sniffen. Ein Switch unterstützt auch eine Full-Duplex Kommunikation. Um Pakete an den richtigen Teilnehmer weiterleiten zu können, muss eine Switch auf Layer 2 im OSI-Modell arbeiten, um die Zieladresse des Pakets auslesen zu können. Dies bedeutet, dass mit dem Weiterleiten der

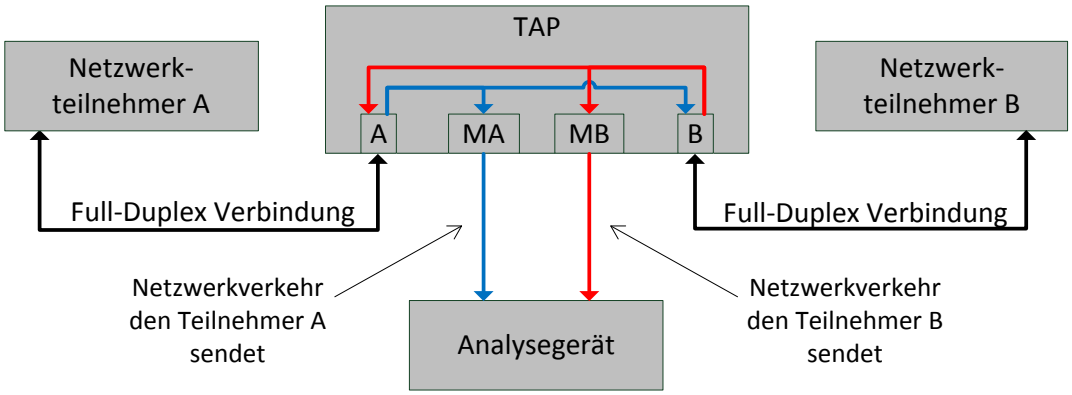

Abbildung 4.6: Schematische Darstellung der Funktion eines TAPs

Nachricht mindestens so lange gewartet werden muss, bis diese Zieladresse vollständig gelesen wurde. Vor der Mac-Adresse liegt die Präambel die 8 Byte lang ist. Die Mac-Adresse ist 6 Byte lang. Die Übertragungsdauer bei einem 100 Mbit Netzwerk liegt bei $80ns$ pro Byte. Aus der folgender Rechnung $(8Byte + 6Byte) * 80ns => 1120ns$ erhält man eine Latenz von 1120 ns. Dieser optimale Fall würde auch nur bei einem Switch eintreten, das die Cut-through Arbeitsweise implementiert hat.
Ein Switch führt also nicht dazu, dass die Full-Dubplexfähigkeit in einem Netzwerk verloren geht, weist jedoch rechnerisch eine Latenz von 1120ns auf. Da keine klaren Grenzen für Latenz und Jitter gezogen wurden, wird hier der Einsatz einer Switch an diese Stelle nicht komplett ausgeschlossen.

TAP

Ein TAP (Test Access Point) ist ein Gerät zum Sniffen von Netzwerkverkehr. Da TAPs kaum verbreitet sind und in der Literatur nur selten erwähnt werden, soll Abbildung 4.6 die Funktionsweise veranschaulichen.
Teilnehmer A und B können im Full-Duplex Modus kommunizieren. An den Anschluss MA (Monitor A) / MB (Monitor B) werden die Daten, die an Anschluss A / B gesendet werden dupliziert. TABs gibt es für Kupfer- und Glasfaserkabel, die unterschiedliche Bandbreiten unterstützen. Einige Hersteller werben damit, dass ihre TABs eine „zero latency“ haben.

Im Kapitel 4.2.2 werden kommerzielle TAPs vorgestellt. Eine kommerzielle TAP, die für die Bachelorarbeit zur Verfügung steht, wird analysiert. Im Kapitel 4.2.3 wird eine im Rahmen der Bachelorarbeit selbst gebaute passive TAP vorgestellt und analysiert.

4.2.2 Kommerzielle TAP

In diesem Kapitel werden kommerzielle TAPs analysiert. Eine Recherche hat ergeben, dass es fünf TAP Hersteller gibt. Keiner dieser Hersteller hat auf seiner Internetseite, auf Produkt- oder Datenblätter Angaben über Latenz oder Jitter in Form einer Zeitangabe gemacht. Bei einigen Herstellern konnte die Angabe „zero latency“ gefunden werden. Der Mangel dieser Auskunft liegt wohl daran, dass der häufigste Anwendungsfall das Sniffen vom zeitunkritischen Best-Effort Traffic ist.
Um doch evtl. ein Zeitangabe zu erhalten, wurden bei allen Herstellern per Mail Latenz und Jitter angefragt. Nicht alle konnten die Frage beantworten. Das Ergebnis dazu und der Preis der TAPs sind in Tabelle 4.3 zu sehen.

Hersteller	Bandbreite	Preis	Latenz	Jitter
Netoptics	100 Mbit	400 €	zero	k. A.
Netoptics	1000 Mbit	1100 €	zero	k. A.
Network Instruments	100 Mbit	330 €	281 - 342ns	61ns
Network Instruments	1000 Mbit	880 €	252 - 292ns	40ns
Dual Comm	100 Mbit	90 $	120.000ns	k. A.
Dual Comm	1000 Mbit	150 $	120.000ns	k. A.
Barracuda Networks	100 Mbit	129 $	k. A.	k. A.
Network Critical	100 Mbit	395 €	zero	k. A.
Network Critical	1000 Mbit	995 €	zero	k. A.

Tabelle 4.3: TAP Herstellerangabe Preis, Latenz und Jitter

Ein TAP von Netoptics, welches eine Bandbreite von 10/100 Mbit unterstützt, steht für diese Bachelorarbeit zur Verfügung. Diese wird im Folgenden analysiert.
Der Analysevorgang entspricht dem in Kapitel 4.1.1 beschriebenen Vorgang der Analyse des Zeitstempels, mit dem Unterschied, dass im Messaufbau zwischen den beiden Ethernetports ein TAP dazwischen geschaltet wird. Der Messaufbau ist in Abbildung 4.7 dargestellt.
Es werden folgende Latenzen gemessen:

- Δt_{ab}: Latenz zwischen Port A und Port B
- Δt_{ama}: Latenz zwischen Port A und Port MA (Monitor A)
- Δt_{bmb}: Latenz zwischen Port B und Port MB (Monitor B)

Es wurden wie in Kapitel 4.1.1 200 Messungen für die sieben verschiedenen Paketgrößen, die im RFC-2544 zur Leistungsmessung von Netzwerken empfohlen werden, durchgeführt (vgl. RFC2544, 1999). Δt_{ab}, Δt_{ama} und Δt_{bmb} liefern alle sehr ähnliche Messergebnisse, was den Schluss zulässt, dass erstmal Port A und B sich gleich verhalten und die Datenpakete die TAP am Ausgangsport und Monitorport gleichzeitig verlassen. Da die Ergebnisse sehr ähnlich sind, werden in Tabelle 4.4 nur die Messergebnisse von Δt_{ab} dargestellt. Vergleicht man die

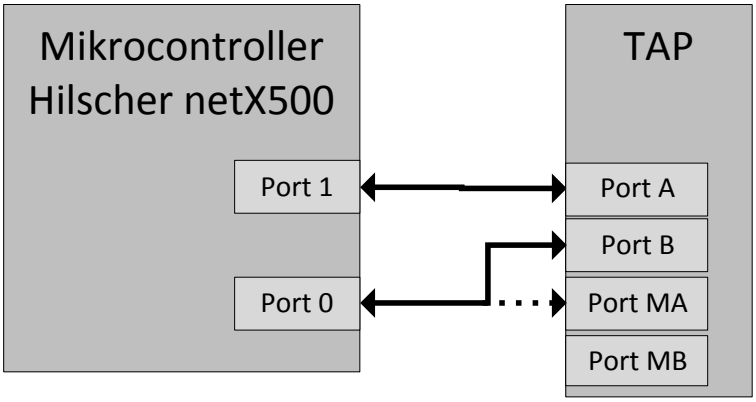

Abbildung 4.7: Messaufbau: Messung Latenz einer TAP

Messergebnisse mit den Ergebnissen im Kapitel 4.1.1, in dem der Zeitstempel durch eine Direktverbindung von Port0 und Port1 analysiert wurde, so sieht man, dass das Minimum der Messergebnisse bei allen Paketgrößen gleich dem Minimum der Messergebnisse in Kapitel 4.1.1 ist. Der Jitter von max. 100ns ist ebenfalls nicht angestiegen. Die Verteilung der Messergebnisse ist ebenfalls sehr ähnlich. Diese Messung verhält sich also ähnlich der Messung bei dem Port0 und Port1 direkt mit einem Kabel verbunden wurde. Dies lässt den Schluss zu, dass diese TAP eine sehr geringe Latenz hat. Mit einer Messgranularität von 10 ns konnte keine Latenz festgestellt werden. Somit muss laut dieser Messung die Latenz < 10 ns sein.

Paketgröße							
64 Byte		128 Byte		256 Byte		512 Byte	
Δt_{IrqTs}[ns]	Häufung	Δt_{IrqTs}[ns]	Häufung	Δt_{IrqTs}[ns]	Häufung	Δt_{IrqTs}[ns]	Häufung
6690	9	11870	32	22110	27	42590	58
6700	65	11880	52	22120	60	42600	66
6710	45	11890	46	22130	41	42610	34
6720	34	11900	36	22140	21	42620	22
6730	24	11910	16	22150	26	42630	15
6740	23	11920	18	22160	22	42640	4
				22170	1	42650	1
				22190	2		

Paketgröße					
1024 Byte		1280 Byte		1518 Byte	
Δt_{IrqTs}[ns]	Häufung	Δt_{IrqTs}[ns]	Häufung	Δt_{IrqTs}[ns]	Häufung
83550	30	104030	22	123070	45
83560	104	104080	53	123080	80
83570	20	104090	33	123090	27
83580	10	104100	24	123100	5
83590	14	104110	27	123110	16
83600	22	104120	41	123120	26
				123160	1

Tabelle 4.4: Messergebnisse: Analyse Latenz Netoptics TAP

Anmerkung zur Messung

Die eben genannte Messung wurde mehrmals durchgeführt. Dabei ist aufgefallen, dass wenn zuerst der Mikrocontroller eingeschaltet und programmiert ist und dann die Netzwerkkabel zur Messung eingesteckt werden, andere Messergebnisse erzielt werden. Der Differenz der Messergebnisse verhält sich folgendermaßen:
Wenn die Netzwerkkabel nach dem Einschalten des Mikrocontrollers eingesteckt werden, dann verschiebt sich die Verteilung der Latenz um bis zu 40ns. Es ist jedoch nicht so, dass die Verschiebung um 40ns schwankt, sondern dass bei einer Messung die Verschiebung genau 20ns beträgt und bei der nächsten Messung, bei der das Netzwerkkabel auch nach dem Einschalten eingesteckt wurde, z. B. 40ns beträgt. Sind alle Netzwerkkabel vor dem Einschalten eingesteckt, so erhält man, bezüglich Minimum und Maximum, immer die gleichen Messergebnisse.

Um dieses Verhalten genauer analysieren zu können, wurde der Messaufbau aus Kapitel 4.1.1 auf Seite 19 erneut aufgebaut. Der Messaufbau dazu ist in Abbildung 4.1 auf Seite 19 zu sehen. Die Messung zur Analyse des Jitters wurde nun mehrmals durchgeführt. Das Ergebnis

dieser Messung war, dass wenn die Netzwerkkabel nach dem Einschalten eingesteckt wurden, die Zeitstempel von Port0 und Port1 nicht mehr gleich waren, sondern wie oben beschrieben um bis zu 40ns verschoben. Wurden die Netzwerkkabel vor dem Einschalten eingesteckt, so waren je Paket die Zeitstempel von Port0 und Port1 exakt gleich. Eine Recherche im Internet ergab, dass PHY-Einheiten die eintreffenden Daten parallelisiert an die MAC-Einheit weitergeben (vgl. Texas Instruments, 2009). Eine PHY-Einheit „sammelt“ eine bestimmte Anzahl von seriellen Bits und gibt diese, wenn die vorgesehene Anzahl eingetroffen ist, parallel an die MAC-Einheit weiter. Sind die Netzwerkkabel vor dem Einschalten des Mikrocontroller eingesteckt, so werden beide Netzwerkschnittstellen gleichzeitig initialisiert und bauen anschließen gleichzeitig einen Link zum Netzwerk auf. Beide fangen zeitgleich an die Daten zu deserialisieren und geben die “gesammelten” Bits zeitgleich an die jeweiligen MAC-Einheiten weiter. Die MAC-Einheiten, in der die Zeitstempelfunktionalität eingebaut ist, stempeln nun zeitgleich den Empfangszeitpunkt der Pakete.
Werden die Netzwerkkabel nach dem Einschalten eingesteckt, so ist es unmöglich zwei Netzwerkkabel auf 10ns genau in die Netzwerkports zu stecken, dadurch werden die Links zu unterschiedlichen Zeitpunkten aufgebaut, die PHY-Einheiten haben zu unterschiedlichen Zeiten ihre Bit voll “gesammelt” und geben diese zu unterschiedlichen Zeiten an die MAC-Einheit weiter. Dies führt wiederum dazu, dass die MAC-Einheit die Pakete zu unterschiedlichen Zeitpunkten stempelt, obwohl die Datenpakete physikalisch gleichzeitig eingetroffen sind. Eine Dokumentation, der im Mikrocontroller eingebauten PHY-Einheiten liegt leider nicht vor. Jedoch kann aus den Messergebnissen gesehen werden, dass die PHY-Einheit 4 Bits parallelisiert an die MAC-Einheit weitergibt, weil bei keiner Messung eine höhere Verschiebung als 40ns gemessen wurde.

Dies würde die konstante Verschiebung erklären und führt zur folgenden Einschränkung für alle Messungen die mit diesem Mikrocontroller durchgeführt werden:

Vor dem Einschalten der Mikrocontrollers müssen alle Netzwerkkabel angeschlossen sein, andernfalls muss ein Messfehler von 40ns eingerechnet werden.

4.2.3 Passive Eigenbau TAP

In diesem Kapitel geht es um den Bau und die Analyse einer Eigenbau-TAP. Man muss jedoch beachten, dass es sich um eine passive TAP handelt, die die Signale nicht aggregiert. Was beteuted, dass die Signalleistung auf zwei Leitungen aufgeteilt wird, ohne die Signale aufzufrischen. Der Grund, warum in dieser Bachelorarbeit diese TAP trotzdem gebaut wurde, ist der enorme Preisunterschied. Die im vorangegangen Kapitel analysierte TAP von Netoptics kostet knapp 400€. Hier werden zwei TAPs auf einem Patch-Feld für 20€ realisiert. Das Patchfeld wurde, nach einem im Internet gefundenen Verdrahtungsplan, siehe Abbildung 4.8 , verdrahtet und analysiert.

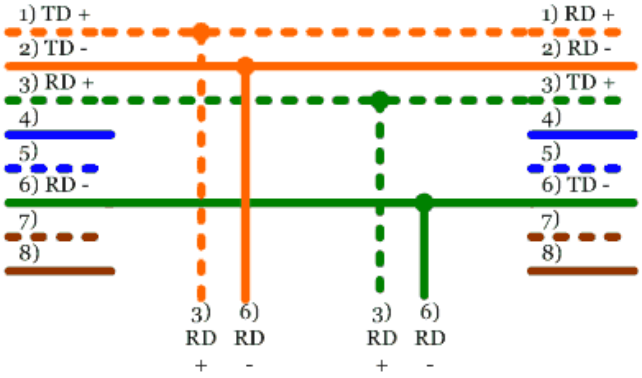

Abbildung 4.8: Verdrahtungsplan Eigenbau-TAP (Quelle: Gómez, 2004)

Die Eigenbau-TAP wurde, wie die kommerzielle TAP im vorangegangen Kapitel analysiert. Die Messergebnisse sind, was das Minimum, Maximum und Jitter betrifft gleich der Messergebnisse der Analyse der kommerziellen TAP von Netoptics in Kapitel 4.2.2. Was bedeutet, dass die Eigenbau-TAP ebenfalls keine Latenz und keinen Jitter hat. Dies ist nicht verwunderlich, weil hier ja auch nur eine Leitung von Port0 nach Port1 geht, die zwischendurch durch ein Patchfeld geleitet wird, ohne, dass elektronische Bauteile dazwischen sind. Jedoch ist zu beachten, dass im Patch-Feld die Leitungen nicht geschirmt sind, was zu einer schlechten elektromagnetischen Verträglichkeit führen könnte.

In einer weiteren Messung wurden Fullsize-Frames (1514 Byte) 30 Sekunden lang mit voller Bandbreite (100 Mbit) durch die TAP geschickt. Es sind alle Pakete ohne Checksummenfehler durch die TAP gegangen. Es ist zu beachten, dass alle Messungen mit Netzwerkleitungen durchgeführt wurden, die nicht länger als 2 Meter waren. Bei längeren Netzwerkleitungen kann man annehmen, dass der Spannungsabfall auf den Leitungen zu groß wird, und die Signalleistung nicht mehr ausreicht, um Daten zuverlässig zu übertragen. Auch kann man sich vorstellen, dass bei höheren Bandbreiten, wie etwa 1 Gbit/s die fehlende Abschirmung in dem Patchfeld zu Verlust von Daten führen kann.
Da aber in den Anforderungen für den Messaufbau beschriebene Bandbreite 100Mbit beträgt und die Messung mit kurzen Netzwerkleitungen durchgeführt werden kann, genügt die Eigenbau-TAP den Anforderungen und kann somit innerhalb dieser Bachelorarbeit verwendet werden.

4.3 Ergebnis der Analyse

In diesem Kapitel wird überprüft, ob die aufgestellten Anforderungen erfüllt werden können. Anschließen werden die Ergebnisse der Analyse zusammengefasst.

4.3.1 Überprüfung der Anforderungen

In diesem Kapitel wird anhand der Analyseergebnisse überprüft, ob die in Kapitel 3 auf Seite 9 aufgestellten Anforderungen erfüllt werden.

Anforderungen: Qualitätsbestimmung eines Senders

Nr	Anforderung	Prio.
1	Messung des Absendezeitpunkts eines Gerätes in der Zeitbasis des Netzwerks	1

Nach den Analyseergebnissen lässt sich sagen, dass diese Anforderung noch nicht erfüllt werden kann. Wie im Kapitel 4.1.1 auf Seite 19 festgestellt wurde, konnte der genaue Zeitpunkt des Stempelns während der Deserialisierung, mit den gegebenen Mitteln nicht gemessen werden konnte. Die Analyseergebnisse lassen annehmen, dass dieser Zeitpunkt sehr früh ist, jedoch konnte dieser nicht genau bestimmt werden. Sollte der Zeitpunkt in Zukunft bekannt sein, so kann diese Anforderung erfüllt werden.

Nr	Anforderung	Prio.
2	Messung des Jitters des Absendezeitpunkts in der Zeitbasis des Netzwerks	1

Nach den Analyseergebnissen kann angenommen werden, dass diese Messung realisierbar ist, da der Zeitpunkt des Stempelns während der Deserialisierung immer gleich ist und nicht jittert. Somit kann der Jitter des Absendezeitpunkts gemessen werden.

Nr	Anforderung	Prio.
3	Zeitsynchronisation mit der Netzwerkzeit	1

Diese Anforderung kann, durch das in einer verwandten Bachelorarbeit (Müller, 2011) entwickelte Zeitsynchronisationsmodul, erfüllt werden.

Nr	Anforderung	Prio.
4	Messgenauigkeit von 0,1µs	2

Nach den Analyseergebnissen, ist eine Messgenauigkeit dieser Größenordnung realisierbar. Die Messgenauigkeit ist von der Genauigkeit des Zeitstempels und von den verwendeten Sniffer abhängig. Da die Zeitstempeleinheit auf < 10ns genau stempelt und die analysierten TAPs keine Latenz und keinen Jitter aufweisen, ist eine Messgenauigkeit bezüglich der Messung aus Anforderung Nr. 2 von $\pm$10ns zu erwarten. Dies muss aber nach der Realisierung verifiziert werden.

Nr	Anforderung	Prio.
5	Messgenauigkeit von 1µs	1

Vergleiche Anforderung Nr. 4.

Nr	Anforderung	Prio.
6	Filterfunktion auf Nachrichten mit bestimmter CT-ID	1

Eine in dieser Bachelorarbeit verwendete Basissoftware-Architektur für den Mikrocontroller bringt diese Funktionalität mit sich (Müller, 2011). Somit ist diese Anforderung erfüllt.

Nr	Anforderung	Prio.
7	Schnittstelle zu einem Auswerterechner mit Bandbreite von min. 14 Mbit/s	1

Wie im Kapitel 4.1.2 auf Seite 24 analysiert, bietet eine Dual-Port-Memory Schnittstelle eine Bandbreite von mind. 23,4 Mbit/s bei Zugriff auf den internen SRAM. Somit ist diese Anforderung erfüllt.

Nr	Anforderung	Prio.
8	Verwendung von einem transparenten Sniffer, der Datenpakete dupliziert	1

Ein TAP kann in dieser Messung als transparenter Sniffer verwendet werden. Die Transparenz einer TAP von Netoptics wurde im Kapitel 4.2.2 auf Seite 29 und einer Eigenbau-TAP im Kapitel 4.2.3 auf Seite 32 nachgewiesen.

Anforderungen: Messung von Latenzen im Netzwerk

Wie im Kapitel 3.1.2 auf Seite 11 beschrieben, gelten die Anforderungen 3 bis 8 aus der *„Qualitätsbestimmung eins Senders“* entsprechend für die Anforderungen zur *„Messung von Paketlaufzeiten“*. Diese Anforderungen wurden, wie im vorangegangen Text beschrieben, erfüllt.

Nr	Anforderung	Prio.
9	Messung von Paketlaufzeiten in einem Netzwerk	1

Nach den Analyseergebnissen kann solch eine Messung realisiert werden, da der analysierte Mikrocontroller zwei Ethernetports besitzt. An beiden Ethernetports kann die Ankunftszeit der Datenpakete mit einer Zeitstempeleinheit, die nicht jittert, aufgezeichnet werden, woraus die Differenz und somit die Paketlaufzeit ermittelt werden kann.

Nr	Anforderung	Prio.
10	Messung des Jitters von Paketlaufzeiten in einem Netzwerk	1

Vergleiche Anforderung nur 9. Hier muss die Schwankung der Zeitdifferenzen aufgezeichnet werden, woraus sich der Jitter ergibt.

Nr	Anforderung	Prio.
11	Zeitaufnahme beider Messpunkte von einer Uhr	1

Diese Anforderung kann erfüllt werden, da die Zeitstempeleinheiten beider Ports die Systemzeit des Mikrocontrollers und somit die selbe Zeit-Quelle nutzen.

Nr	Anforderung	Prio.
12	Schnittstelle zu einem Auswerterechner mit Bandbreite von min. 200 Mbit/s	2

Diese Anforderung kann nicht erfüllt werden, da der Mikrocontroller keine Schnittstelle mit dieser Bandbreite bietet. Allerdings hat diese Anforderung mit der Priorität 2 und ist somit nur eine „nice to have“ Anforderung.

Anforderungen: Mikrocontroller

Nr	Anforderung	Prio.
13	Unterstützung der 100 Mbit Bandbreite	1

Diese Anforderung wird erfüllt, da der analysierte Mikrocontroller Ethernetports besitzt, die mit dieser Bandbreite Daten übertragen können.

Nr	Anforderung	Prio.
14	Unterstützung der 1 Gbit Bandbreite	2

Diese Anforderung wird nicht erfüllt, da der analysierte Mikrocontroller keine Ethernetports besitzt, die mit dieser Bandbreite Daten übertragen können. Es handelt sich hierbei jedoch um eine Anforderung mit der Priorität 2.

Nr	Anforderung	Prio.
15	Zeitstempeleinheit, die den Ankunftszeitpunkt eines Datenpakets stempelt	1

Diese Anforderung wird erfüllt, da der analysierte Mikrocontroller solch eine Zeitstempeleinheit besitzt.

Nr	Anforderung	Prio.
16	Jitter des Stempelzeitpunkts soll kleiner als 1µs sein	1

Diese Anforderung wird erfüllt, da im Kapitel 4.1.1 auf Seite 19 nachgewiesen wurde, dass der Jitter der Zeitstempeleinheit < 10ns ist.

Nr	Anforderung	Prio.
17	Paketgrößenunabhängiger Stempelzeitpunkt	1

Diese Anforderung wird erfüllt. Im Kapitel 4.1.1 auf Seite 19 wurde nachgewiesen, dass die Zeitstempeleinheit paketgrößenunabhängig arbeitet.

Nr	Anforderung	Prio.
18	Eine weitere Schnittstelle zur Übertragung der Messdaten muss vorhanden sein	1

Diese Anforderung ist erfüllt. Der Mikrocontroller bietet eine Dual-Port-Memory Schnittstelle, womit die Messdaten übertragen werden können.

Anforderungen: Auswerterechner

Nr	Anforderung	Prio.
19	Mikrocontrollerkompatible Schnittstelle zur Übertragung der Messergebnisse	1

Diese Anforderung kann erfüllt werden. Dieser Bachelorarbeit steht eine Dual-Port-Memory PCI-Karte des Mikrocontrollerherstellers zur Verfügung, womit diese Schnittstelle realisiert werden kann.

Nr	Anforderung	Prio.
20	Darstellung der Messergebnisse in Wireshark	1

Diese Anforderung kann erfüllt werden. Zur Speicherung der Messergebnisse kann das Dateiformat Libpcap verwendet werden, welches von Wireshark zur Darstellung von Messergebnissen unterstützt wird.

Nr	Anforderung	Prio.
21	Zeitbasis des Mikrocontrollers in den Messergebnissen	2

Diese Anforderung kann erfüllt werden, da in der Anwendung Wireshark eingestellt werden kann, dass die eingetragenen Zeiten in einer Libpcap-Datei als Nanosekunden interpretiert werden sollen. Dies gilt jedoch nur für die Anwendung Wireshark. Es kann nicht garantiert werden, dass andere Anwendungen, die dieses Dateiformat unterstützen, die eingetragene Zeit auch als Nanosekunden interpretieren können.

Anforderungen: Sniffer

Nr	Anforderung	Prio.
22	Sniffer mit geringer/keiner Latenz	1
23	Sniffer mit geringem/keinem Jitter	1

Diese Anforderungen enthalten den relativen Begriff „gering“, was nicht messbar ist. Diese Anforderungen wurden nicht konkretisiert, in der Hoffnung, dass die Analyse der TAPs, die von einigen TAP-Herstellern geworbene „zero Latency“ verifiziert. Dies konnte verifiziert werden und somit gelten diese Anforderungen als erfüllt.

Ergebnisse der Anforderungsüberprüfung

Nach der Analyse kann gesagt werden, dass 20 von 23 Anforderungen erfüllt werden können. Eine von den nicht erfüllten Anforderung hat die Priorität 1. Somit ist klar, dass der Absendezeitpunkt in der Zeitbasis des Netzwerks nicht gemessen werden kann. Diese Anforderung kann nicht erfüllt werden, weil die Zeit zwischen dem physikalischen Eintreffen einer Nachricht und der Zeit, in der gestempelt wird, nicht gemessen werden konnte. Es konnte jedoch nachgewiesen werden, dass dieser Stempelzeitpunkt konstant ist. Sollte sich in Zukunft eine Möglichkeit ergeben diese Zeit zu messen, beispielsweise mit einem Oszilloskop, der das Messen von 100Mbit-Ethernet unterstütz, so könnte diese Anforderung erfüllt werden. Solch ein Oszilloskop stand dieser Bachelorarbeit nicht zur Verfügung. In der Messung *Qualitätsbestimmung eines Senders* kann also vorerst nur der Jitter eines Senders gemessen werden, welches auch schon eine wertvolle Information ist. Aus dem Grund wird diese Messung trotzdem implementiert.
Die anderen beiden nicht erfüllten Anforderungen, *Unterstützung des 1 Gbit Standards* und *Schnittstelle zu einem Auswerterechner mit Bandbreite von min. 200 Mbit/s* mit der Priorität 2, werden im Kapitel *Ausblick* dieser Arbeit diskutiert.

4.3.2 Zusammenfassung der Analyse

In diesem Kapitel wurde Hardware auf Ihre Verwendbarkeit für die geforderten Messungen analysiert. Das Ergebnis ist, dass der Mikrocontroller netX500 des Herstellers *Hilscher* verwendet werden kann. Zur Zeitaufzeichnung werden die Zeitstempeleinheiten in den beiden Ethernetports verwendet. Zur Übertragung der Messergebnisse zum Auswerterechner wird die Dual-Port-Memory Schnittstelle verwendet. Zum Duplizieren des Netzwerkverkehrs kann eine TAP des Herstellers *Netoptics* oder unter bestimmten Bedingungen eine Eigenbau-TAP verwendet werden.
Im folgenden Kapitel *Konzeption und Realisierung* kann nun auf die Verwendung dieser Hardware gesetzt werden.

Kapitel 5

Konzeption und Realisierung

Zu Beginn dieses Kapitels wird die Softwarearchitektur anhand einer Grafik beschrieben. Daraufhin werden die in der Architektur definierten Softwaremodule konzipiert. Anschließend wird die Realisierung der Module beschrieben.

5.1 Architektur

Die Architektur verteilt sich auf zwei Rechnern. Den Mikrocontroller, auf dem die Messungen erfolgen und dem Auswerterechner, auf dem die Messergebnisse nach der Messung angezeigt werden. Beim Mikrocontroller wird auf eine Basisarchitektur aufgebaut, die in einer Bachelorarbeit im CoRE-Team (CoRE RG) entwickelt wurde (Müller, 2011).
Die Basisarchitektur beinhaltet einen Startupcode für den Mikrocontroller, ein Linkerskript, Hardwareinitialisierung und ein Zeitsynchronisationsmodul, welches dafür sorgt, dass die Systemzeit des Mikrocontrollers sich mit der Netzwerkzeit synchronisiert. In Abbildung 5.1 ist die Architektur mit dem Datenverlauf schematisch dargestellt. Die Fett umrandeten Module müssen in dieser Bachelorarbeit entwickelt werden.
Der Mikrocontroller dient dabei zur Entgegennahme der Datenpakete, Stempeln der Ankunftszeit und weitergabe der Datenpakete mit den Zeitstempeln an den Auswerterechner. Der Auswerterechner nimmt die Daten an und schreibt die Informationen in ein Dateiformat, welches mit der Anwendung Wireshark geöffnet werden kann. In Wireshark können die Messergebnisse dann betrachtet werden. Im folgenden Kapitel werden die Module aus der Abbildung 5.1 beschrieben und wenn notwendig konzipiert.

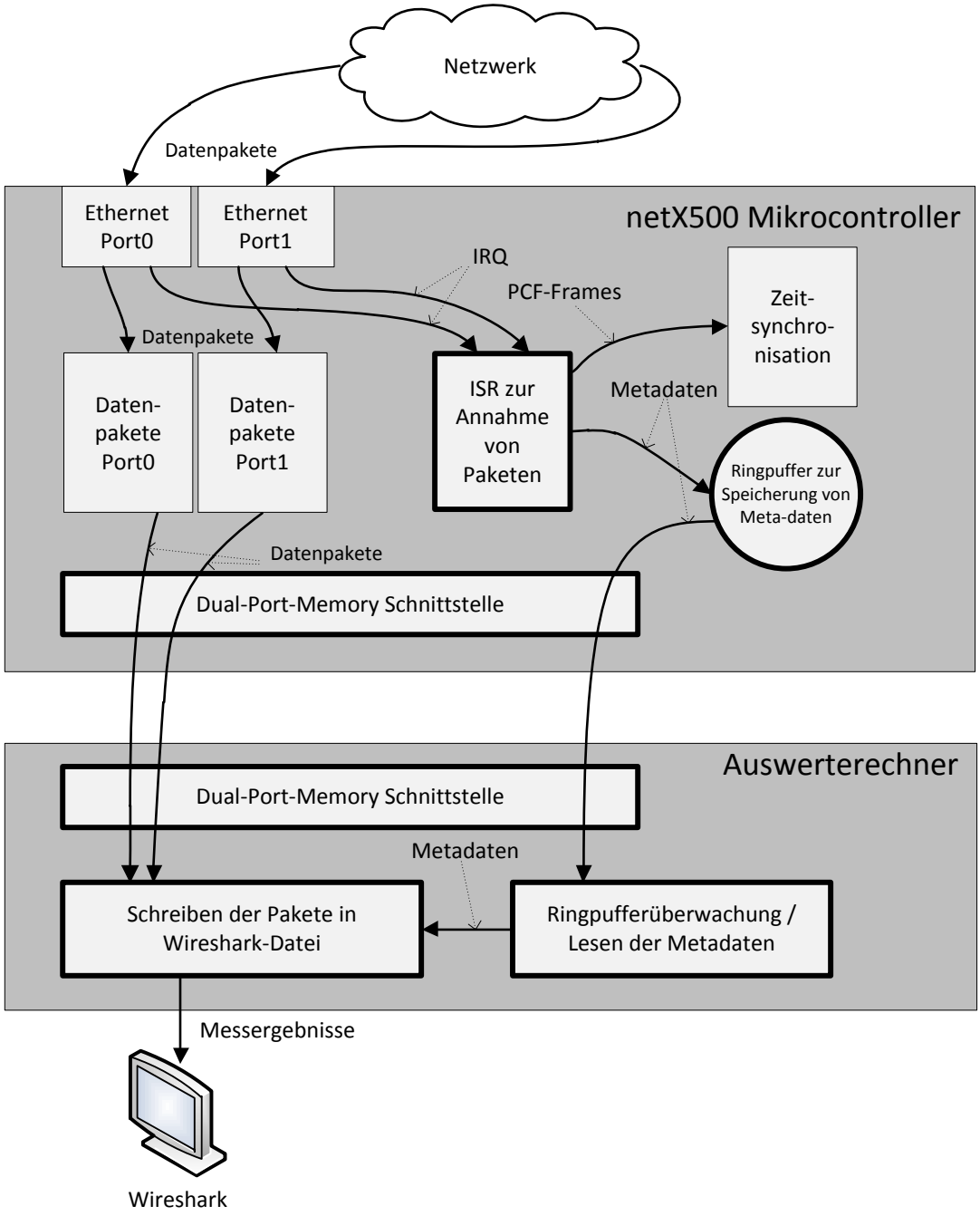

Abbildung 5.1: Schematische Architektur mit Datenverlauf

5.2 Konzeption der Module

Ethernet Port0 und Port1

Ethernet Port0 und Port1 sind Schnittstellen zum dem Netzwerk, bei dem die zu messenden Datenpakete eintreffen. Die Ports deserialisieren die Daten aus dem Netzwerk, setzen einen Zeitstempel und legen die Daten in den jeweils zugewiesenen Speicher. Anschließen wird ein Interrupt ausgelöst, um einer angebundenen Interrupt-Service-Routine mitzuteilen, dass ein Paket eingetroffen ist. Eine Speicherverwaltung regelt das Belegen und Freigeben der Speicherbereiche für die Datenpakete.
Dieses Modul ist bereits vorhanden und muss nicht entwickelt werden.

ISR zur Annahme von Datenpaketen

Dieses Modul besteht aus zwei Interrupt-Service-Routinen (ISR). Jedem Ethernetport ist eine dieser ISRs zugewiesen. Diese werden aufgerufen, sobald ein Paket vollständig eingetroffen ist. Dieses Modul soll die Zeitsynchronisation einleiten (Anforderung Nr. 3) und Daten für das Modul *Ringpuffer für Metadaten* sammeln und diesem Modul übergeben. Eine Filterfunktion soll dafür sorgen, dass nur messrelevante Datenpakete weitergegeben werden (siehe Anforderung Nr. 6). Das Modul zum Teil vorhanden und muss um folgende Punkte erweitert werden:

- Zusammenstellen der Metadaten: Speicherort des Datenpakets, Größe des Datenpakets und Ankunftsport
- Übergabe der Metadaten an das Modul *Ringpuffer zur Speicherung der Metadaten*
- Prüfen ob der Auswerterechner Pakete vollständig ausgelesen hat und diesen Speicherbereich freigeben.

Ringpuffer für Metadaten

Dieses Modul ist ein Ringpuffer, auf dem zwei Module parallel zugreifen können. Um den parallelen Zugriff zu ermöglichen, muss ein Schutzmechanismus vorhanden sein, der verhindert, das auf ein Element des Ringpuffers zwei Module gleichzeitig lesen oder schreiben. Dieser Speicher muss Dual-Port-Memmory-fähig sein, damit der Auswerterechner hier die Daten auslesen kann. Dieses Modul muss vollständig entwickelt werden.

Zeitsynchronisation

Dieses Modul ist für die Zeitsynchronisation zuständig. Dazu nimmt es PCF-Frames und deren Eingangszeit entgegen. Und passt die Systemzeit anhand des im PCF-Frames eingetragenen Delays und der Eingangszeit. Dieses Modul schafft es die Systemzeit auf $\pm$500ns pro Zyklus, unabhängig von der Zykluslänge, an die Netzwerkzeit anzupassen (vgl. Müller, 2011). Es muss

untersucht werden, ob dies in den Messfehler des Messgeräts eingerechnet werden muss. Dieses Modul ist in der Basisarchitektur enthalten und muss nicht entwickelt werden.

Datenpaketespeicher Port0 und Port1

Diese Module sind einfacher Speicher, der von den Ports belegt und verwaltet wird. Die Ports legen hier Ihre Datenpakete und weitere Metadaten ab. Dieser Speicher muss Dual-Port-Memmory-fähig sein, damit der Auswerterechner diese Daten auslesen kann. Dieses Modul ist in der Basisarchitektur initialisiert.

Dual-Port-Memory Schnittstelle Mikrocontroller

Dieses Modul dient als Schnittstelle zum Auswerterechner. Es soll dem Auswerterechner ermöglichen auf Speicherbereiche des Mikrocontrollers lesend und schreibend zuzugreifen. Der Speicherbereich der Module *Datenpaketespeicher Port0 und Port1* und *Ringpuffer für Metadaten* soll zugreifbar gemacht werden. Dieses Modul muss ist auf dem Mikrocontroller vorhanden und muss konfiguriert werden.

Dual-Port-Memory Schnittstelle Auswerterechner

Dieses Modul dient als Schnittstelle zum Mikrocontroller. Diese Schnittstelle wird durch eine spezielle PCI-Karte realisiert, über die der Mikrocontroller mit einem Kabel verbunden wird. Nach der Installation eines Treibers kann eine auf dem Auswerterechner programmierte Anwendung über einen linearen Speicherbereich im PCI-Bus auf den Speicher des Mikrocontrollers zugreifen.

Ringpufferüberwachung / lesen von Metadaten

Dieses Modul greift über die DPM-Schnittstelle auf das Modul *Ringpuffer für Metadaten* zu und überwacht diesen nach neu hinzugefügten Elementen. Wurde ein neues Element in diesem Modul gefunden, so werden die darin liegenden Metadaten ausgelesen. Diese Metadaten werden an das Modul *Schreibe Daten in Wireshark-Datei* weitergegeben. Nachdem die Meldung erhalten wird, dass die Datei geschrieben wurde, muss dieses Modul dem Ringpuffer signalisieren, dass der Speicherbereich des Datenpakets im Mikrocontroller freigegeben werden kann.
Dieses Modul muss vollständig entwickelt werden.

Schreibe Daten in Wireshark-Datei

Dieses Modul erhält Metadaten vom Modul *Ringpufferüberwachung / lesen von Metadaten*. Aus den Metadaten wird der Speicherort, Ankunftsport und Paketgröße ausgelesen. Anschließend sollen anhand dieser Informationen die Datenpakete mit deren Zeitstempel über die

DPM-Schnittstelle aus dem Modul *Datenpaketespeicher Port0 und Port1* ausgelesen werden. Die ausgelesenen Datenpakete sollen im Libpcap-Dateiformat abgespeichert werden. Für jeden Port wird eine Libpcap-Datei benötigt, da dieses Format nicht vorsieht, verschiedene Netzwerkschnittstellen in einer Datei zu unterscheiden.
Dieses Modul muss vollständig entwickelt werden.

5.2.1 Zusammenfassung

In diesem Kapitel wurden die Module aus dem Architekturbild 5.1 auf Seite 40 beschrieben und ggf. konzipiert. Einige dieser Module sind bereits vorhanden und müssen nicht entwickelt werden. Im nächsten Kapitel wird die Realisierung der Module beschrieben.

5.3 Realisierung der Softwaremodule

In diesem Kapitel wird die Realisierung der konzipierten Module beschrieben. Nicht alle, im vorangegangenen Kapitel beschriebenen Module müssen programmiert werden, da einige vom Basisprojekt (Müller, 2011) gegeben sind. Zunächst wird die genaue Aufteilung des Speichers des Moduls *Datenpaketespeicher Port0 und Port1* angeschaut, um festzustellen, an welcher Stelle welche Metadaten entnommen werden müssen. Die Speicheraufteilung ist durch die Module *Port0 und Port1* definiert, die von der Basisarchitektur gegeben sind. Anschließend wird die Erweiterung des Moduls *ISR zur Annahme der Datenpakete* beschrieben. An dieser Stelle ist bekannt, wo, welche Metadaten entnommen werden können, die dann dem Modul *Ringpuffer für Metadaten* übergeben werden können, welches anschließend beschrieben wird. Anhand dieser Informationen wird dann die DPM-Schnittstelle zum Auswerterechner festgelegt.

5.3.1 Datenpaketespeicher Port0 und Port1

Dieses Modul ist bereits vorhanden und initialisiert, jedoch soll in diesem Abschnitt die Aufteilung des Speichers bekannt gemacht werden. Diese Informationen werden in den anderen Modulen benötigt.
Jedem Ethernetport ist ein 64 kByte großes internes SRAM-Modul zugewiesen. Der erste SRAM-Speicher hat den Adressbereich $0x00000000$ bis $0x00007FFF$, der Zweite den Adressbereich $0x00008000$ bis $0x0000FFFF$ (vgl. Hilscher, 2008, S. 17). Das Basisprojekt ist so konfiguriert, dass Port0 die Speicherung der Pakete bei Adresse $0x00000618$ beginnt, und Port1 bei Adresse $0x00008618$. Die Pakete werden in 1560_{10} Byte-Abständen gespeichert, egal wie groß diese sind. Das Paket mit der höchsten Adresse hat bei Port0 die Adresse $0x000079E0$ und bei Port1 die Adresse $0x0000F9E0$. Port0 und Port1 haben somit jeweils 20 Speicherplätze für Datenpakete. Auf die Anfangsadressen der 1560_{10} Byte großen Speicherbereiche kann folgende Struktur abgebildet werden, woraus der Inhalt der Speicher-Blöcke deutlich wird:

```
1 typedef struct ETHERNET_FRAME_Ttag
```

```
2  {
3    unsigned char   tDstAddr[6];    /* Destination MAC address */
4    unsigned char   tSrcAddr[6];    /* Source MAC address      */
5    unsigned short  usType;         /* Frame length/type       */
6    unsigned char   abData[1504];   /* Frame data + CRC        */
7    unsigned char   abRes[18];      /* reserved                */
8    unsigned long   ulTimestampNs;  /* receive timestamp       */
9    unsigned long   ulTimestampS;   /* receive timestamp       */
10 } __attribute__((packed)) ETHERNET_FRAME_T;
```

Anhand der vorangegangenen Informationen ist nun bekannt, wie die Datenpakete abgespeichert werden und somit wie und wo sie rausgelesen werden können. Auch ist die Anzahl der Datenpakete bekannt, woraus sich die Größe des Ringpuffers ergibt.

5.3.2 ISR zur Annahme der Datenpakete

Dieses Modul ist bereits vorhanden und muss erweitert werden.
Das vorhandene Modul leitet ankommende PCF-Frames an das Modul *Zeitsynchronisation* weiter, welche die Systemzeit mit der Netzwerkzeit synchronisiert.
Das Konfigurationsmodul des Basisprojekts bietet verschiedene Filterfunktionen, die bereits in diesem Modul verwendet werden. Es kann allgemein nach zeitkritischen Nachrichten, nach bestimmten Zeitkritischen Nachrichten (CT-ID) und speziell nach PCF-Frames gefiltert werden. Diese Filter können für die Messungen verwendet werden. Das Modul muss um folgende Punkte erweitert werden:

- Zusammenstellen der Metadaten: Speicherort des Datenpakets, Größe des Datenpakets und Ankunftsport
- Übergabe der Metadaten an das Modul *Ringpuffer zur Speicherung der Metadaten*
- Prüfen ob der Auswerterechner Pakete vollständig ausgelesen hat und diesen Speicherbereich freigeben.

5.3.3 Ringpuffer für Metadaten

Dieses Modul ist ein Ringpuffer, welches Metadaten für die Messungen aufnimmt. Es wird eine Ringpuffergröße von 40 Elementen festgelegt, weil die Ethernetports je 20 Datenpakete aufnehmen kann und somit maximal 40 Datenpakete im Mikrocontroller zur Übertragung vorgehalten werden können.
Da der Mikrocontroller und der Auswerterechner parallel auf den Ringpuffer lesend und schreibend zugreifen, muss ein Mechanismus erschaffen werden, der verhindert, dass beide auf ein Element gleichzeitig zugreifen. Ein Element vom Ringpuffer soll folgende Informationen enthalten:

- Eingangsport
- Handler der für die Speicherfreigabe des Pakets benötigt wird
- Startadresse des Datenpakets im SRAM
- Paketgröße
- Ein Flag zum Schutz des konkurrierenden Zugriffs

Alle Informationen sollen als 32-Bit-Variablen abgespeichert werden, um atomar auf die Informationen zugreifen zu können. Das Flag soll folgende Zustände annehmen können:

- FREI - Das Ringelement ist nicht belegt.
- BELEGT - Das Ringelement ist belegt.
- RELEASE - Das Ringelement wurde vom Auswerterechner ausgelesen und der Speicher kann freigegeben werden.

Dieser Ringpuffer muss eine Ablegeposition haben, auf der die ISR die Daten ableget. Der Ringpuffer muss zwei Abholpositionen haben. Eine Abholposition für den Auswerterechner, an der Metadaten mit dem Flag BELEGT ausgelesen werden, eine Abholposition für die ISR, die an dieser Position prüft ob Speicherbereiche freigegeben werden können. Folgender Ablauf soll die Funktion des Flags verdeutlichen:

- Wenn der Ringpuffer leer ist, sind alle Ringelemente als FREI markiert. Die Ablegeposition und beide Abholpositionen zeigen auf dasselbe Ringelement.
- Der Auswerterechner pollt auf dem Flag auf seiner Leseposition so lange, bis es BELEGT ist.
- Der Mikrocontroller erhält ein Paket. Die ISR sammelt die Metadaten und speichert diese in die Ablegeposition. Sobald alles gespeichert wurde, wird das Flag atomar auf BELEGT gesetzt. Die Ablegeposition springt um ein Ringelement weiter.
- Der Auswerterechner hört auf zu Pollen, weil er jetzt atomar BELEGT gelesen hat. Der Auswerterechner ließt zuerst die Metadaten und anschließend das Datenpaket aus. Nach dem das Datenpaket vollständig ausgelesen wurde, setzt der Auswerterechner das Flag auf seiner Leseposition auf REALEASE. Die Leseposition vom Auswerterechner springt um ein Ringelement weiter.
- Beim nächsten ISR-Durchlauf, also wenn wieder ein Paket eintrifft, prüft der Mikrocontroller in seiner Abholposition, ob das Flag den Wert REALEASE hat. Falls ja, wird eine Funktion aufgerufen, die dafür sorgt, dass der Speicherplatz vom Datenpaket freigegeben wird. Nach dem Es freigegeben wurde, wird das Flag atomar auf FREI gesetzt. Die Abholposition vom Mikrocontroller springt um ein Ringelement weiter.

5.3.4 DPM-Schnittstelle zum Auswerterechner

In diesem Kapitel wird die Dual-Port-Memory Schnittstelle beschrieben. Im Mikrocontroller lassen sich bis zu 8 verschiedene unterschiedlich große Speicherbereiche in den 64 kByte großen DPM-Bereich abbilden. Die obersten 512 Byte werden statisch auf DPM-Konfigurationsregister abgebildet, sodass nur 63,5 kByte zur Verfügung stehen. Laut Dokumentation müssen die Startadressen der DPM-Speicherbereich durch 256_{10} teilbar sein, was hexadezimal bedeutet, dass die beiden niederwertigen Stellen in der Adresse Null sein müssen. In Abbildung 5.2 wird die konzipierte und realisierte Speicheraufteilung dargestellt.

5.4 Realisierung der Softwaremodule auf dem Auswerterechner

In diesem Kapitel wird die Realisierung der Softwaremodule beschrieben, die auf dem Auswerterechner ausgeführt werden.

5.4.1 Ringpufferüberwachung / Lesen von Metadaten

Dieses Modul soll über die DPM-Schnittstelle Metadaten aus dem Ringpuffer lesen. Folgende Schritte sind dazu notwendig:

1. Pollen auf der Leseposition des Auswerterechners auf dem Flag des Ringpufferelements, bis das Flag BELEGT ist.
2. Sobald das Flag BELEGT ist, sollen die Metadaten aus dem Ringelement ausgelesen werden.
3. Übergabe der Metadaten an das Modul *Schreiben der Pakete in Wireshark-Datei*
4. Sobald dieses Modul meldet, dass die Paketdaten ausgelesen wurden, wird das Flag in dem Ringelement auf RELEASE gesetzt, was dem Mikrocontroller signalisiert, dass der Speicherplatz für dieses Datenpaket freigegeben werden kann.
5. Die Leseposition springt um ein Ringelement weiter.
6. Zurück zu Schritt 1.

5.4.2 Schreiben der Pakete in eine Libpcap-Datei

Dieses Modul erhält die Metadaten von dem Modul *Ringpufferüberwachung / Lesen von Metadaten*, ließt anhand dieser Daten die Datenpakete aus der DPM-Schnittstelle und schreibt diese in eine Libpcap-Datei. Das Libpcap-Dateiformat wird von vielen Netzwerkanalyseprogrammen zu Speicherung von Messdaten verwendet. Da dieses Dateiformat als Standard für die Speicherung von Netzwerkdaten gilt und Wireshark dieses Format unterstützt, wird dieses

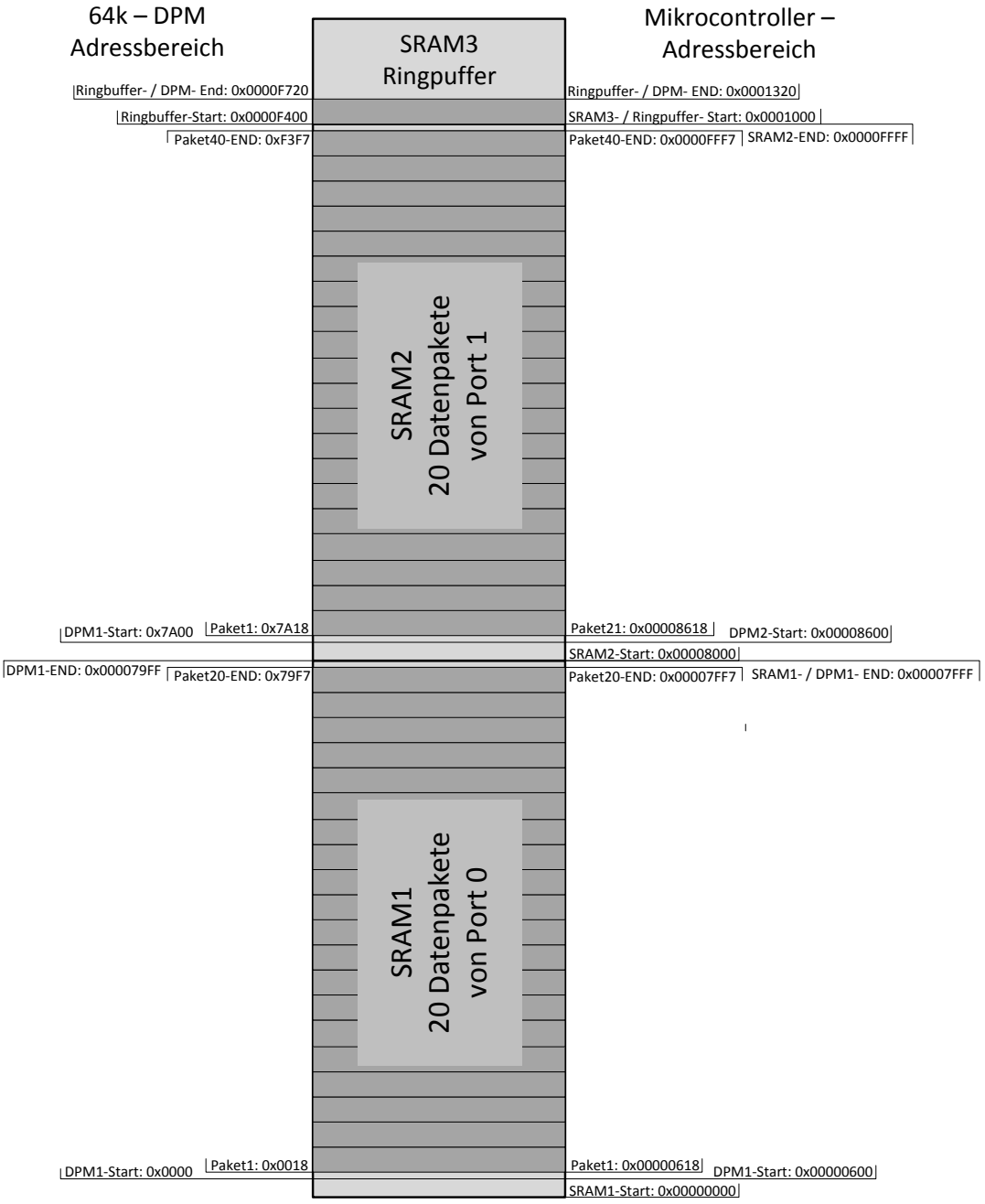

Abbildung 5.2: Speicheraufteilung DPM

Format von diesem Modul zur Speicherung der Messergebnisse verwendet.
Abbildung 5.3 soll den schematischen Aufbau der Datei darstellen.

Abbildung 5.3: Libpcap-Dateiformat schematischer Aufbau (vgl. Harris, 2011)

Die Datei beginnt mit einem Libpcap-Dateiheader, anschließend wird zu jedem Datenpaket der Paket-Header und der Datenteil des Pakets gespeichert.
Der Libpcap-Dateiheader ist folgendermaßen aufgebaut:

```
1 typedef struct pcap_hdr_s {
2     uint32 magic_number;   /* magic number */
3     uint16 version_major;  /* major version number */
4     uint16 version_minor;  /* minor version number */
5     int32  thiszone;       /* GMT to local correction */
6     uint32 sigfigs;        /* accuracy of timestamps */
7     uint32 snaplen;        /* max length of captured packets, in octets */
8     uint32 network;        /* data link type */
9 } pcap_hdr_t;
```

(vgl. Harris, 2011)

magic_number Dient zur Erkennung des Dateiformats und der Bytereihenfolge. Das Programm, welches die Datei erstellt, schreibt in seiner Bytereihenfolge 0xa1b2c3d4 in diesen Speicherplatz. Das Programm, welches die Datei ließt, ließt dann entweder 0xa1b2c3d4 oder 0xd4c3b2a1 (vertauscht) und weiß dann, ob alle anderen zu lesenden Daten vertauscht oder nicht vertauscht eingelesen werden müssen.

version_major Der obere Teil der Versionsnummer des Dateiformats. Bei der aktuellen Version 2.4 würde hier der Wert 2 stehen.

version_minor Der unter Teil der Versionsnummer des Dateiformats. Bei der aktuellen Version 2.4 würde hier der Wert 4 stehen.

thiszone Die Korrektur der Zeit in Sekunden zwischen der Zeitzone GMT (UTC) und der lokalen Zeitzone, die im *Paket Header* der gemessenen Pakete angegeben ist. Wenn der Zeitstempel in der GMT-Zeitzone interpretiert werden soll, so ist dieser Wert 0. Wenn der Zeitstempel in der GMT+1-Zeitzone interpretiert werden soll, so ist dieser Wert -3600. In der Praxis wird dieser Wert immer auf 0 gesetzt.

sigfigs Theoretisch kann hier die Genauigkeit des Zeitstempels eingetragen werden. In der Praxis setzten alle Tools diesen Wert auf 0.

snaplen Maximale Länge eines Datenpakets. In der Praxis wird der Wert 65535 (0xFFFF) eingetragen.

network Hier wird der Link-Layer Headertyp eingestellt. Das Libpcap-Dateiformat kann Daten von unterschiedlicher Hardware abspeichern. Beispielsweise kann der USB-, Bluetooth-, DVB-, und andere Art von Datenverkehr aufgezeichnet werden. Für Ethernet ist hier der Wert 1 einzutragen.

Nachdem der *Global-Header* geschrieben wurde, können die Datenpakete in die Datei geschrieben werden. Zuerst wird der *Paket-Header* geschrieben gefolgt von *Paket-Data*. Der *Paket-Header* ist wie folgt aufgebaut:

```
1 typedef struct pcaprec_hdr_s {
2    uint32 ts_sec;         /* timestamp seconds */
3    uint32 ts_usec;        /* timestamp microseconds */
4    uint32 incl_len;       /* number of octets of packet saved in file */
5    uint32 orig_len;       /* actual length of packet */
6 } pcaprec_hdr_t;
```

ts_sec Datum und Uhrzeit, wann das Paket aufgenommen wurde. Dieser Wert wird in Sekunden die seit dem 01.01.1970 um 00:00h vergangen sind abgespeichert.

ts_usec Mikrosekunde, in der das Paket aufgenommen wurde. Mit neueren Wireshark-Versionen lässt sich dieses Feld auch als Nanosekunde interpretieren.

incl_len Anzahl der Bytes der Paketdaten, die tatsächlich erfasst und in dieser Datei gespeichert wurden. Dieser Wert darf nicht höher als der Wert *snaplen* im Dateiheader sein.

orig_len Anzahl der Bytes der Paketdaten, die das Datenpaket evtl. vorgibt zu haben.

In *Paket-Data* wird das Datenpaket in folgendem Format abgespeichert, was dem Standardformat des Ethernet-Frames entspricht:

```
1 typedef struct Ethernet_Frame_t {
2   uchar   tDstAddr[6];       /* Destination MAC address */
3   uchar   tSrcAddr[6];       /* Source MAC address      */
4   ushort  usType;            /* Frame length/type       */
5   uchar   abData[incl_len];  /* Frame data; incl_len aus Paket-Header*/
6 } ethernet_frame_t;
```

Nun ist das Dateiformat bekannt und es kann definiert werden, wie die Daten aus dem Mikrocontroller auf dieses Dateiformat abgebildet werden können. Zunächst wird jedoch definiert, wie der Dateiheader beschrieben werden soll:

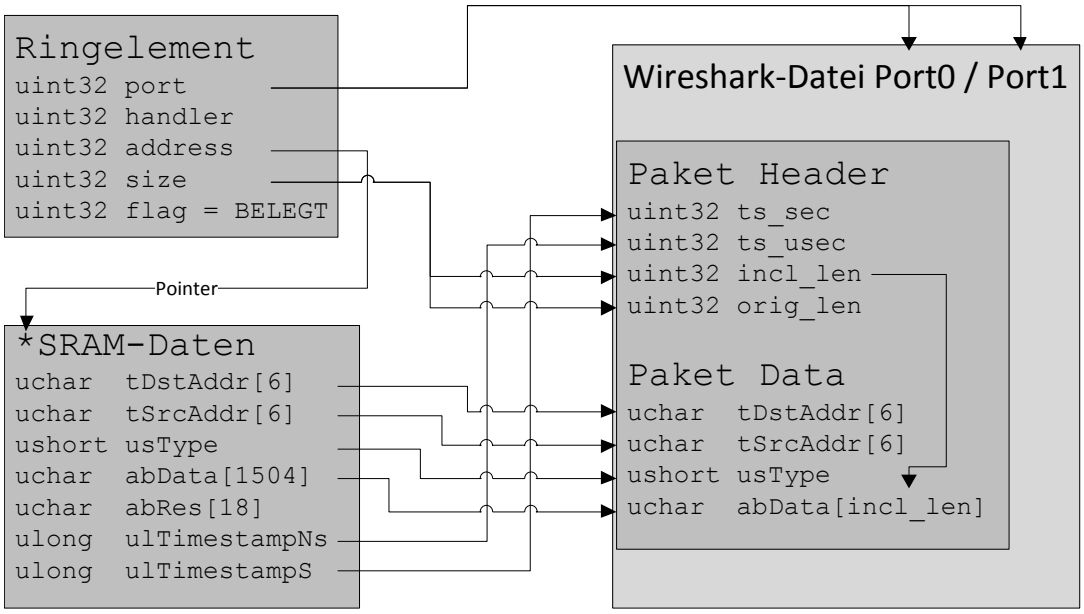

Abbildung 5.4: Mapping von Mikrocontrollerdaten zu Libpcap-Datei

```
1 pcap_hdr_s header;
2 header.magic_number = 0xa1b2c3d4;   /* Dateiformat und Bytereihenfole */
3 header.version_major = 2;           /* 2 aus Version 2.4 */
4 header.version_minor = 4;           /* 4 aus Version 2.4 */
5 header.thiszone = -3600;            /* Sekundendifferenz zu GMT */
6 header.sigfigs = 0;                 /* Genauikgeit */
7 header.snaplen = 65535;             /* Max. Länge eines Datenpakets */
8 header.network = 1;                 /* Link-Layer Header: Ethernet */
```

Abbildung 5.4 zeigt das Mapping der Daten aus dem Ringpuffer und aus dem SRAM auf das Libpcap-Dateiformat:

Kapitel 6

Beispielmessungen

In diesem Kapitel werden zwei Beispielmessungen präsentiert, die mit dem realisierten Messgerät vorgenommen wurden. Sie dienen als Beispiel um Szenarien zeigen zu können, für die sich das entwickelte Messgerät eignen könnte.

6.1 Messung eines weitergeleiteten Synchronisationspakets

Diese Messung soll als Beispielmessung dienen, für die im Kapitel *Anforderungen* geforderte *Messen von Paketlaufzeiten in einem Netzwerk*. Bei dieser Messung wurde bei einem Gerät untersucht, ob bei einem weitergeleiteten PCF-Frame (Zeitsynchronisationspaket), der im Frame eingetragene Delay richtig aufaddiert wird. Ein Gerät, welches PCF-Frames weiterleitet, muss den im Frame eingetragenen Delay folgende Zeit aufaddieren, bevor es dieses weiterleitet: Zeit zwischen

- physikalischem Eintreffen des PCF-Frames (Deserialisierung des ersten Bits)
- physikalischem Senden des PCF-Frames (Serialisierung des ersten Bits)

Also die Zeit, die nicht vergangen wäre, wenn dieses Gerät durch ein Kabel ersetzt werden würde. Das Gerät, das untersucht wurde, ist ein Time-Triggered-Ethernet Switch (TTTech) von dem Hersteller TTTech. In Abbildung 6.1 ist der Messaufbau dargestellt.
Wie am Messaufbau zu sehen, wurden die Frames direkt vor dem Eintreffen in der Switch und direkt nach dem Absenden der Switch dupliziert an den Mikrocontroller geleitet. Die PCF-Frames, die direkt von dem Synchronisationsmaster empfangen wurden, wurden nicht zur Messung, sondern auch zur eigenen Synchronisation der Systemzeit genutzt. Die Uhr im Mikrocontroller läuft also in der gleichen Geschwindigkeit, wie die Netzwerkzeit. Die verwendeten TAPs besitzen keine Latenz und der Mikrocontroller kann den Eintreffzeitpunkt auf 10ns genau aufzeichnen. Womit eine Messgenauigkeit von $\pm 10ns$ vorhanden ist.
Die Netzwerkgeräte sind auf eine Zykluszeit von 1ms eingestellt. Die Synchronisation wird

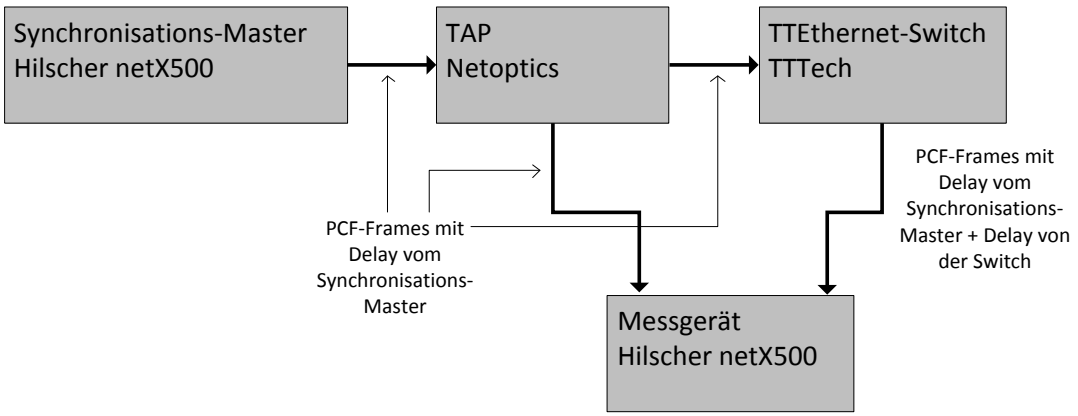

Abbildung 6.1: Messaufbau: Messung eines weitergeleiteten Synchronisationspakets

zum Zyklusbeginn eingeleitet. Um die Ergebnisse auszuwerten, wurde die Differenz der im PCF-Frame eingetragenen Delays, mit der Differenz der Zeitstempel verglichen. Die Abweichung der Differenzen ist die Zeit, die in der Switch zu wenig oder zu viel auf den Delay aufaddiert wird. Die Tabelle 6.1 zeigt 20 Messungen mit deren Ergebnissen.

PCF-Delay [ns]			Zeitstempel [ns]			Fehler	
vor Switch	nach Switch	Differenz	vor Switch	nach Switch	Differenz	Absolut	Relativ
4112	30112	26000	28462461	28489480	27019	1019 ns	3,77 %
4112	33392	29280	29462759	29493007	30248	968 ns	3,20 %
4112	31792	27680	30463053	30491741	28688	1008 ns	3,51 %
4112	30272	26160	31463334	31490441	27107	947 ns	3,49 %
4112	33392	29280	32463612	32493909	30297	1017 ns	3,36 %
4112	31792	27680	33463959	33492616	28657	977 ns	3,41 %
4112	30272	26160	34464086	34491251	27165	1005 ns	3,70 %
4112	33472	29360	35464367	35494724	30357	997 ns	3,28 %
4112	31952	27840	36464548	36493333	28785	945 ns	3,28 %
4112	30352	26240	37464755	37491980	27225	985 ns	3,62 %
4112	33552	29440	38464953	38495356	30403	963 ns	3,17 %
4112	31952	27840	39465128	39493963	28835	995 ns	3,45 %
4112	30432	26320	40465247	40492519	27272	952 ns	3,49 %
4112	33632	29520	41465397	41495900	30503	983 ns	3,22 %
4112	32032	27920	42465486	42494417	28931	1011 ns	3,49 %
4112	30512	26400	43465607	43492970	27363	963 ns	3,52 %
4112	28912	24800	44465707	44491469	25762	962 ns	3,73 %
4112	32112	28000	45465847	45494789	28942	942 ns	3,25 %
4112	30512	26400	46465934	46493305	27371	971 ns	3,55 %
4112	28912	24800	47466001	47491812	25811	1011 ns	3,92 %

Tabelle 6.1: Messergebnisse: PCF-Delay Untersuchung

Laut diesem Ergebnis rechnet der Switch knapp 1µs zu wenig auf den im PCF-Frame eingetragenen Delay auf. Aus dem Ergebnis lässt sich schließen, dass die Netzwerkgeräte, die dass PCF-Frame von dieser Switch erhalten um knapp 1µs falsch zum Synchronisationsmaster synchronisiert sind.

6.2 Messen des Jitters eines Senders

Diese Messung soll als Beispielmessung dienen, für die im Kapitel *Anforderungen* geforderte Messung *Qualitätsbestimmung eines Senders*. In dieser Messung wird Jitter eines Time-Triggered-Senders gemessen. Die Netzwerkzykluszeit beträgt 1 ms. Die Zeitsynchronisation wird zum Zyklusbeginn eingeleitet. Dazu wird ein Messaufbau (siehe Abbildung 6.2) mit folgenden drei Geräten aufgebaut:

- Hilscher netX500 Mikrocontroller, der als Synchronisationsmaster dient. Dieser sendet zum Zeitpunkt 4112 ns im Zyklus ein Zeitsynchronisationspaket (PCF-Frame) an alle beteiligten Netzwerkgeräte.

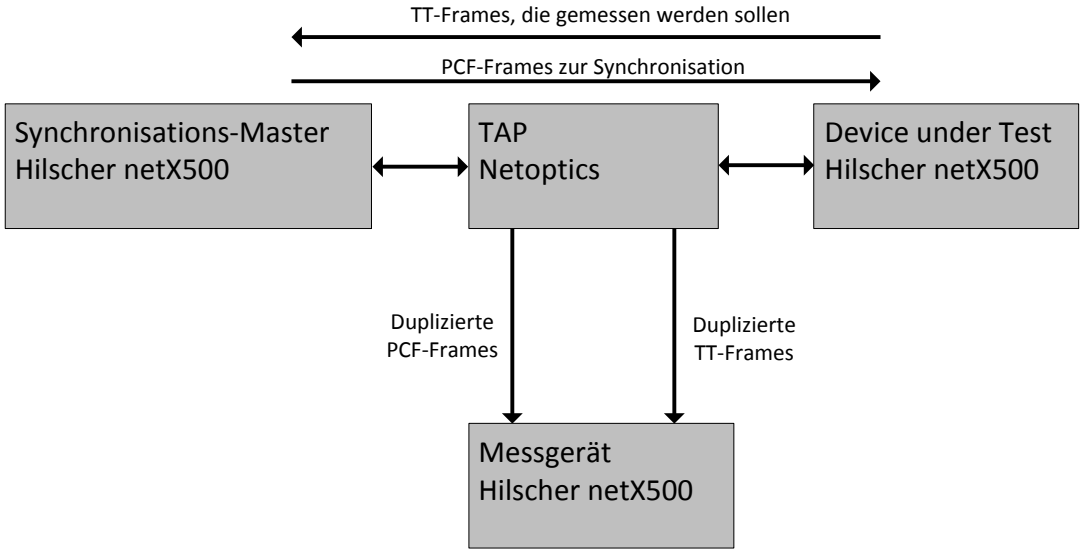

Abbildung 6.2: Messaufbau: Messen des Jitters eines Senders

- Hilscher netX500 Mikrocontroller, dessen Qualität bestimmt werden soll. Dieser sendet zum Zeitpunkt 820.000ns ein 60 Byte großes Time-Triggered Frame.
- Hilscher netX500 Mikrocontroller, der als Messgerät dient. Dieser empfängt die Time-Triggered-Frames des Mikrocontrollers, dessen Qualität bestimmt werden soll, und zeichnet den Ankunftszeitpunkt in der Zeitbasis der Netzwerkzykluszeit auf.

Anhand der Schwankung der aufgezeichneten Zeiten kann der Jitter des Time-Triggered-Senders bestimmt werden. Es wurden 40 Messungen durchgeführt, die in Tabelle 6.2 dargestellt sind.

Messungen 1 - 20 [ns]			Messungen 21 - 40 [ns]		
Ankunft	**Erwartet**	**Abweichung**	**Ankunft**	**Erwartet**	**Abweichung**
820289	820000	289	820406	820000	406
820338	820000	338	820416	820000	416
820147	820000	147	820387	820000	387
820356	820000	356	820316	820000	316
820366	820000	366	820368	820000	368
820375	820000	375	820459	820000	459
820264	820000	264	820360	820000	360
820394	820000	394	820410	820000	410
820243	820000	243	820421	820000	421
820254	820000	254	820392	820000	392
820384	820000	384	820283	820000	283
820353	820000	353	820494	820000	494
820284	820000	284	820345	820000	345
820294	820000	294	820475	820000	475
820304	820000	304	820366	820000	366
820355	820000	355	820377	820000	377
820205	820000	205	820388	820000	388
820335	820000	335	820319	820000	319
820306	820000	306	820371	820000	371
820397	820000	397	820301	820000	301
Min: 147ns		**Max: 494ns**		**Jitter: 347ns**	

Tabelle 6.2: Messergebnisse: Qualitätsbestimmung eines Senders

Die Messergebnisse zeigen, dass der Absendezeitpunkt des Time-Triggered-Senders um 347 ns (Max: 494 ns - Min 147 ns) jittert. Das Zeitfenster in der Switch beträgt $\pm$10µs. Das Zeitfenster kann also erheblich kleiner eingestellt werden, wommit Bandbreite im Netzwerk gewonnen wird.

Kapitel 7

Zusammenfassung und Ausblick

7.1 Zusammenfassung

Ziel der Arbeit war es ein Messgerät zu entwickeln, mit dem sich die Zuverlässigkeit von TTEthernet-Hardware bestimmen lässt und anhand von Messung der Paketlaufzeit eines Time-Triggered-Pakets bestimmt werden kann ob Echtzeitanforderungen in einem TTEthernet erfüllt werden können. Um dies zu realisieren, wurden zuerst Anforderungen an die Messungen und die Hardware, die für die Messungen verwendet werden sollen aufgestellt. Anschließend wurde im Analyseteil der Arbeit Hardware auf Ihre Verwendbarkeit hin analysiert. Woraufhin geprüft wurde, ob die Anforderungen erfüllt werden konnten.
Die Anforderung „*Messung des Absendezeitpunkts eines Gerätes in der Zeitbasis des Netzwerks*“ mit der Priorität 1 konnte nicht erfüllt werden, da der genaue Zeitpunkt des Stempelns nicht festgestellt wurde. Sollte dieser Zeitpunkt in Zukunft bekannt sein, so kann auch diese Anforderung erfüllt werden. Alle anderen Anforderungen mit der Priorität 1 konnten erfüllt werden.
Bei der Analyse wurde festgestellt, dass die Zeitstempeleinheit nicht jittert. Der Eintreffzeitpunkt einer Nachricht wird also immer zum gleichen Deserialisierungszeitpunkt aufgezeichnet. Dies ist entscheident für die Messgenauigkeit, die sich aus dem Jitter des Zeitstempels ergibt. Da die Systemzeit eine Granularität von 10 ns hat und von dieser Zeit der Eintreffzeitpunkt aufgezeichnet wird. Kann hier eine Messgenauigkeit von 10 ns angenommen werden. In dieser Arbeit wurde jedoch das entwickelte Messgerät nicht mit vergleichbaren Ergebnissen eines anderen Messgeräts verifiziert. Um eine Verifikation vornehmen zu können, wird ein Oszilloskop mit Ethernetprobes benötigt. Solch ein Oszilloskop stand dieser Arbeit aus Kostengründen nicht zur Verfügung. Nachdem die Analyse vorgenommen wurde, wurde die Architektur und das Konzept der Realisierung vorgestellt. Die Realisierung erfolgte auf zwei Systemen, dem Mikrokontroller und einem Auswerterechner. Auf dem Mikrocontroller wurden die Messdaten aufgezeichnet, die während des Messvorgangs vom Auswerterechner ausgelesen wurden. Es wurden zwei Beispielmessungen gezeigt. Bei der ersten Beispielmessung

wurde die Durchlaufzeit einer Zeitsynchronisationsnachricht durch ein Time-Triggered-Switch aufgezeichnet. Der Time-Triggered-Switch hatte dabei die Aufgabe während des Weiterleitens die Verzögerung in der Nachricht um die durch die Switch entstandene Verzögerung aufzuaddieren. Mit dieser Messung wurde geprüft, ob die korrekte Zeit aufaddiert wird. Bei der zweiten Beispielmessung wurde der Jitter eines Time-Triggered-Senders gemessen. Anhand der Messergebnisse kann das Zeitfenster in der Switch, an den der Time-Triggered-Sender angeschlossen ist eingestellt werden. Das Zeitfenster legt die Zeit fest in der eine bestimmte Time-Triggered-Nachricht eintreffen darf. Anhand dieser Messung ließe sich auch feststellen, ob ein Time-Triggered-Sender seine Nachrichten in der im Schedule festgelegten Zeit versendet, wenn Anforderung Nr. 1 erfüllt wäre.

7.2 Ausblick

Verifikation

In dieser Arbeit wurden die Messergebnisse aus dem Kapitel *Beispielmessungen* nicht verifiziert. Bevor endgültige Aussagen über die Messgenauigkeit dieses Messgerätes gemacht werden können, sollten die Messungen mit beispielsweise mit einem Oszilloskop, dass das Messen von Ethernet unterstützt, verifiziert werden. Solch ein Oszilloskop stand dieser Arbeit nicht zur Verfügung. Ein Beispiel für ein Oszilloskop, dass hierfür eingesetzt werden könnte, wäre das Tektronix MSO4034B (vgl. Tektronix MSO4000B Datasheet, 2010).

Zeitstempel-Delay

In dieser Arbeit konnte der Delay des Stempelvorgangs nicht gemessen werden. Mit dem eben beschriebenen Oszilloskop könnte dies ausgemessen werden. Sobald der Wert bekannt ist, kann Anforderung Nr. 1 erfüllt werden.

Gigabit-Ethernet

In dieser Arbeit wurde ein Messgerät entwickelt, dass in Netzwerken Messungen vornehmen kann, in denen mit den Bandbreiten 100Mbit und 10Mbit kommuniziert wird. Allerdings ist das Ethernet skalierbar und so existiert heute schon Time-Triggered-Ethernet Hardware, die im Gigabit-Standard kommunizieren kann. In einer Recherche wurde kein Mikrocontrollerboard gefunden, dass Gigabit-Ethernet unterstützt. Eine Möglichkeit in Gigabitbandbreiten zu messen wäre es, das Messgerät auf einem FPGA zu entwickeln. Es existieren bereits FPGAs, die Gigabit-MAC-Einheiten beinhalten (vgl. Xilinx, 2006, S. 17).

Literaturverzeichnis

[AIM GmbH] AIM GMBH: *Avionics Databus Solutions*. – URL http://www.afdx.com/. – Zugriffsdatum: 2010-02-24

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

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

[Böke 2008] BÖKE, Dr. C.: FlexRay-Entwicklungen routiniert im Griff. In: *EE Select Automotive* (2008)

[Combs 2010] COMBS, Gerald: *Wireshark*. 2010. – URL http://www.wireshark.org. – Zugriffsdatum: 2011-01-23

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

[FlexRay Consortium] FLEXRAY CONSORTIUM: *FlexRay*. – URL http://flexray.com/. – Zugriffsdatum: 2011-02-07

[Gómez 2004] GÓMEZ, Diego G.: *Network Taps*. URL: http://www.infosecwriters.com/hhworld/hh9/roc/node4.html - Abgerufen: 15.07.2011. 2004

[Harris 2011] HARRIS, Guy: *Libpcap File Format*. URL: http://wiki.wireshark.org/Development/LibpcapFileFormat - Abgerufen: 18.08.2011. 2011

[Hilscher 2008] HILSCHER: *Technical Data Reference Guide netX 500 / 100*. Hilscher Gesellschaft für Systemautomation mbH. 2008

[Hilscher 2009] HILSCHER: *The Insiders Guide to netX*. Hilscher Gesellschaft für Systemautomation mbH. 2009

[Hilscher Gesellschaft für Systemautomation mbH] HILSCHER GESELLSCHAFT FÜR SYSTEMAUTOMATION MBH: . – URL http://de.hilscher.com. – Zugriffsdatum: 2011-08-17

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

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

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

[RFC2544 1999] BRADNER, S.: *RFC2544 - Benchmarking Methodology for Network Interconnect Devices*. URL: http://www.ietf.org/rfc/rfc2544.txt. March 1999

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

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

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

[Tanenbaum 2003] TANENBAUM, Andrew S.: *Computer-Netzwerke - 4., aktualisierte Auflage*. Pearson Studium, 2003. – ISBN 3-8273-7046-9

[Tanenbaum 2009] TANENBAUM, Andrew S.: *Moderne Betriebssysteme - 3., aktualisierte Auflage*. Pearson Studium, Oktober 2009. – ISBN 978-3-8273-7342-7

[Tektronix MSO4000B Datasheet 2010] TEKTRONIX MSO4000B DATASHEET: *Mixed Signal Oscilloscopes MSO4000B, DPO4000B Series Data Sheet*. URL: http://www.farnell.com/datasheets/866462.pdf - Abgerufen 23.08.2011. 2010

[Texas Instruments 2009] TEXAS INSTRUMENTS: *Industrial Temp, Single Port 10/100 Mb/s Ethernet Physical Layer Transceiver*. URL: http://www.ti.com/lit/ds/symlink/tlk100.pdf - Abgerufen: 18.08.2011 . 2009

[TTTech] TTTECH: *TTEDevelopment Switch 100 Mbit/s 8 Ports*. URL: http://www.tttech.com/uploads/tx_products/TTTech-TTE-Development-Switch-100Mbps-Flyer.pdf - Abgerufen 18.08.2011

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

[Xilinx 2006] XILINX: *Virtex-4Q FPGA Data Sheet*. URL: `http://www.xilinx.com/support/documentation/data_sheets/ds595.pdf` - Abgerufen: 22.08.2011. 2006

Abbildungsverzeichnis

Tabellenverzeichnis

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

Hamburg, 24. August 2011 Friedrich Groß