Fachhochschule Bonn-Rhein-Sieg

Fachbereich Elektrotechnik, Maschinenbau und Technikjournalismus (FB 03)

Aufbau und Untersuchung einer 10 Gbit/s Glasfaserübertragungsstrecke zur Übertragung digitalisierter Analogsignale mit hohen Bandbreiten

Diplomarbeit

von Christian Peter Manko,
geboren am 05.03.1983 in Santa Ana, Philippinen

Römerstr. 245
53117 Bonn
Email: christian.manko@web.de
Tel.: (01 77) 401 73 29

Studiengang: Elektrotechnik/Kommunikationstechnik
Matrikelnummer: 9004847

Erstbetreuer: Prof. Dr. Andreas Bunzemeier

Zweitbetreuer: Prof. Dr. Marco Winzker

Betreuer am Max-Planck-Institut für Radioastronomie: Dr. Reinhard Keller

erstellt im Sommersemester 2007

eingereicht am 05.09.2007

Meinen Eltern und meinen Geschwistern

Eidesstattliche Erklärung

Ich versichere, dass ich die vorliegende Diplomarbeit selbstständig und ohne andere als die angegebenen Hilfsmittel erstellt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Textstellen und Abbildungen sind als solche kenntlich gemacht.

Bonn, den 05.09.2007

_____

Unterschrift

Inhaltsverzeichnis

Inhaltsverzeichnis ..... **iii**

Vorwort ..... **vi**

1 Einleitung **1**

1.1 Das Radioteleskop Effelsberg 3

2 Theoretische Grundlagen **4**

2.1 Ethernet 4

2.1.1 Das OSI-Referenzmodell 5

2.1.2 IEEE 802.3ae 8

2.1.3 Beziehung zwischen IEEE 802.3ae und dem OSI Modell 10

2.1.4 Medium Access Control (MAC) 11

2.1.5 Reconciliation Sublayer (RS) und 10 Gigabit Media Independent Interface (XGMII) 14

2.1.6 Physical Coding Sublayer (PCS) 14

2.1.7 Physical Medium Attachment (PMA) 15

2.1.8 Physical Medium Dependent (PMD) 16

2.1.9 10 Gigabit Attachment Unit Interface (XAUI) und XGMII Extender Sublayer (XGXS) 16

2.1.10 10 Gigabit Small Form Factor Pluggable Modules (XFP) und High Speed 10 Gb/s serial Electrical Interface (XFI) 18

2.2 Netzwerkhardware 20

2.2.1 Aufbau und Funktion einer Netzwerkkarte 20

2.2.2 Myricom 10 Gigabit Ethernet Netzwerkkarte 24

2.2.3 UDP und TCP Checksum Offload 28

2.2.4 Interrupt Coalescing 28

2.2.5 Zero-Copy 29

2.2.6 TCP Segmentation Offload 29

2.2.7 Receive-Side Scaling 30

2.2.8 Large Receive Offload 30

2.2.9 Multicast Filtering 30
2.2.10 Write Combining 31
2.3 Transmission Control Protocol 32
2.3.1 Das TCP/IP Referenzmodell 32
2.3.2 Services 33
2.3.3 TCP-Header 36
2.4 User Datagram Protocol (UDP) 39
2.4.1 UDP-Header 40
2.5 Internet Protocol (IP) 41
2.5.1 IP-Header 42
2.5.2 Internet Control Message Protocol (ICMP) 45
3 Die 10-Gbit/s-Übertragungsstrecke 46
3.1 Auswahl der Hardware und Versuchsaufbau 46
3.2 Netperf 50
3.2.1 Design von Netperf 50
3.2.2 Testverfahren 51
3.2.3 Einstellungen 52
3.3 Ergebnisse 53
3.3.1 Erste Versuche mit Netperf 56
3.3.2 Vergleich MTU 9000 mit MTU 1500 59
3.3.3 Versuche mit verschiedenen WCFiFo- und Timestamps-Optionen 61
3.3.4 Versuche mit verschiedenen Allocation Order Werten 63
3.3.5 Versuche mit verschiedenen Interrupt Coalescing Zeiten 64
3.3.6 Die CPU Auslastung bei verschiedenen Versuchen 67
3.3.7 Das optimale Ergebnis 69
3.3.8 UDP-Datenrate in Abhängigkeit der Sende- und Empfangs-puffer 71
3.3.9 UDP-Datenrate in Abhängigkeit der Nachrichtengröße 74
3.4 Diskussion 75
3.4.1 Der Weg zur optimalen Konfiguration 75

3.4.2 Die Datenrate in Abhängigkeit der Puffer- und Nachrichtengröße 79
3.4.3 Vergleich der gemessenen Werte mit den Ergebnissen der Firma Myricom 80
4 Das Backend 82
4.1 Einführung in Very Long BaseLine Interferometry (VLBI) 82
4.2 Diskussion über ein mögliches Backend 84
5 Das FiLa 10G Board 86
5.1 Einführung zum Digital Baseband Converter (DBBC) 86
5.2 Einführung zum digitalen Empfänger 87
5.3 Die Idee des FiLa 10G Boards 88
6 Zusammenfassung und Ausblick 91
Literaturverzeichnis 92
Anhang 95
A.1 Messergebnisse 95

Vorwort

Diese Diplomarbeit ist der Abschluss meines Studiums an der Fachhochschule Bonn-Rhein-Sieg. Ich studiere seit September 2003 Elektrotechnik mit dem Schwerpunkt Kommunikationstechnik. Ich hatte große Freude am Studium. Die Ausarbeitung erfolgte von Juni bis Anfang September 2007 am Max-Planck-Institut für Radioastronomie.

Ich möchte mich an dieser Stelle bei Herrn Dr. Reinhard Keller für die Betreuung am Max-Planck-Institut und bei Herrn Prof. Dr. Bunzemeier für die Betreuung an der Fachhochschule bedanken.

Des Weiteren möchte ich mich bei Herrn Robert Neiser und bei allen Personen bedanken, die zum Gelingen dieser Arbeit beigetragen haben.

Bonn, den 05.09.2007

1 Einleitung

In der modernen Radioastronomie werden für die Übertragung von ZF-Signalen immer größere Bandbreiten benötigt. Die derzeitige Technik mit Koaxialkabeln erlaubt eine Bandbreite von bis zu 1 GHz. Damit stößt diese Art der Übertragung am Radioteleskop Effelsberg an ihre Grenzen, so dass neue Technologien verwendet werden müssen, um den Forderungen nach Bandbreite bis in den GHz-Bereich erfüllen zu können. „In modernen radioastronomischen Empfangssystemen wird das zu übertragene Analogsignal daher direkt im Empfänger digitalisiert und mittels Glasfaserkabel zur Weiterverarbeitung in den Backend-Raum übertragen“ [19]. Am Max-Planck-Institut für Radioastronomie wurde bereits die erste Version des digitalen Empfängers entwickelt. Dieser wird zurzeit auf elektromagnetische Verträglichkeit untersucht [33, 31].

In der zweiten Ausbaustufe dieses Projektes soll eine Zusatzkarte mit dem Namen FiLa[1] 10G Board entwickelt werden, die im Wesentlichen nur aus Schnittstellen besteht, unter anderem eine 10-Gbit-Ethernet-Schnittstelle. Sie soll in erster Linie die digitalisierten Daten des Empfängers am Teleskop über eine optische Übertragungsstrecke in den Kontrollraum schicken und sie dort einem Backend System zur Verfügung stellen.

Dieses Backend kann ein Digital Baseband Converter (DBBC) sein, welches eine Entwicklung von verschiedenen Instituten der Radioastronomie unter Leitung von Gino Tuccari vom Instituto di Radioastronomia in Noto auf Sizilien ist. Er ist auch der Projektleiter des FiLa 10G Board Projektes. Dieses Backend kann die Daten nach VLBI[2] Standard oder nach anderen Methoden verarbeiten.

Das DBBC ist Teil des digitalen Empfängers und würde sich direkt am Radioteleskop befinden. Da digitale Hochfrequenztechnik eine geringe elektromagnetische Verträglichkeit besitzt, sollen möglichst viele Teile der Hardware zur Datenverarbeitung vom Empfänger entfernt untergebracht werden. Das Senden digitalisierter Rohdaten vom Radioteleskop in den Backend ist nicht nur für die EMV von Vorteil, sondern auch im Sinne von effizienter Datennutzung für die Forschung in der Radioastronomie. Je früher eine numerische Repräsentation

[1] FiLa steht für „First Last“ und kennzeichnet eine Schnittstellenkarte, die sich am Anfang oder am Ende eines Datenverarbeitungssystems befinden kann. Dieses FiLa 10G Board soll auch innerhalb des Systems angebracht werden können.

[2] VLBI steht für *Very Long Baseband Interferometry* und beschreibt eine Methode der astronomischen Interferometrie mit Radioteleskopen.

der empfangenen Informationen hergestellt werden kann, umso besser ist dies für die Forschung.

Am FiLa 10G Board arbeiten im Moment drei Personen, Gino Tuccari, Michael Wunderlich und Christian Peter Manko. Diese Diplomarbeit befasst sich mit der 10-Gbit-Ethernet-Übertragungsstrecke, ihren Protokollen und ihren Schnittstellen. Eine Testübertragungsstrecke wurde aufgebaut und dahingegen untersucht, ob es möglich ist hohe Datenraten nahe der Theoretischen zu erreichen und welche Einstellungen für eine optimale Ausnutzung der Bandbreite vorgenommen werden müssen. Dabei sind die inneren Prozesse von großer Bedeutung. Auf dem FiLa 10G Board wird sich ein Xilinx Field-Programmable Gate Array (FPGA) befinden mit einem Medium Access Control (MAC) Core[3]. Die Erfahrungen aus der Testübertragungsstrecke sollen später bei der Programmierung des FPGAs zur optimalen Ausnutzung der Bandbreite helfen.

Es soll außerdem die Möglichkeiten der Empfängerseite untersucht werden. Die große Datenmenge muss an ein dafür vorgesehenes Backend weitergeleitet werden. Die Art des Backends wird von der Art der Daten bestimmt. Dort müssen die Daten gespeichert oder verarbeitet werden. Es soll auch die Möglichkeit bestehen, die verarbeiteten Daten im Backend mit einem weiteren FiLa 10G Board an ein anderes System zu schicken.

Im letzten Teil dieser Diplomarbeit wird sich eine Einführung in das FiLa 10G Board und in die Systeme, in die es implementiert werden soll, befinden. Es soll drei Schnittstellen besitzen, einen *High Speed Input/Output* (HSI/HSO) Bus, ein *VLBI Standard Interface* (VSI) und eine 10-Gbit-Ethernet-Schnittstelle. Der Verfasser wurde damit beauftragt sich mit den Schnittstellen, der XAUI und der XFI, dem XAUI/XFI Schnittstellen Wandler und dem XFP Transceiver vertraut zu machen.

[3] Ein MAC Core ist ein von Xilinx zum Verkauf programmiertes Makro, welches in das FPGA geladen werden kann.

1.1 Das Radioteleskop Effelsberg

Das Radioteleskop Effelsberg ist das Hauptbeobachtungsinstrument des Max-Planck-Instituts für Radioastronomie. Es wurde in Effelsberg bei Bad Münstereifel errichtet und am 1. August 1972 in Betrieb genommen. Man wählte diesen Standort wegen der Abgeschiedenheit von möglichen Störquellen. Es befindet sich in einem Tal, was den Einfluss von Störsignalen weiter minimiert. Mit einem Reflektordurchmesser von 100 m ist es das zweitgrößte vollbewegliche Radioteleskop der Welt.

Das Radioteleskop besitzt zwei Empfangsysteme. Der Primärfokus befindet sich 30 m über der Mitte des Parabolspiegels. Es ist an vier verstrebten Stützbeinen befestigt. Die Zentrale des Primärfokus ist die Fokuskabine. In der Bodenmitte dieser Kabine befindet sich eine kreisrunde Öffnung, in welche die Empfängerbox hineingeschoben wird. Der Sekundärfokus befindet sich im Zentrum des 100-Meter-Spiegels (siehe Bild 1.1).

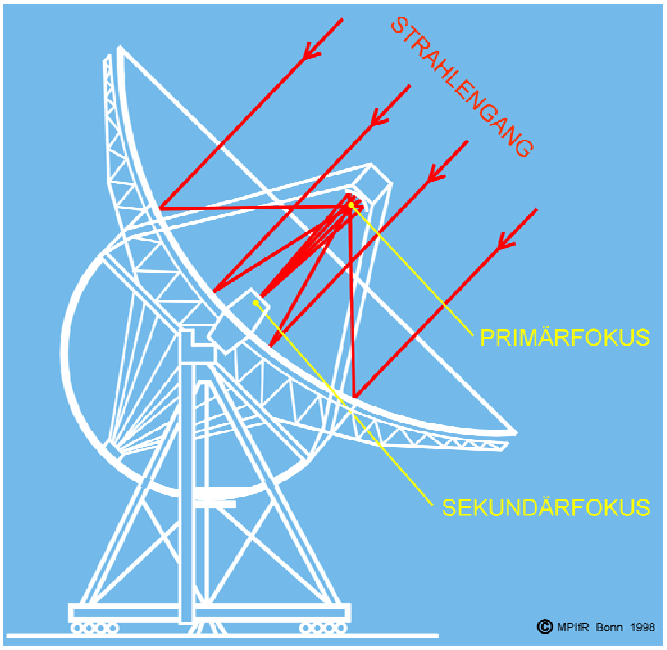

Bild 1.1: Primär- und Sekundärfokus am Radioteleskop Effelsberg [21]

Hinter dem Brennpunkt des Primärfokus befindet sich ein 6,5 m breiter Spiegel, mit dem die Radiowellen zum Sekundärfokus umgelenkt werden. Der Vorteil des Sekundärfokus ist der simultane Betrieb mehrerer Empfänger. Ein gleichzeitiger Betrieb beider Fokusse ist nicht möglich.

2 Theoretische Grundlagen

2.1 Ethernet

Der Erfinder des Ethernets ist Robert M. Metcalfe. Er arbeitete als Ingenieur der Elektrotechnik in den 70er Jahren bei XEROX in deren Forschungszentrum in Palo Alto (Palo Alto Reserach Center, PARC). Dort arbeitete er an dem Projekt „Office of the Future“. Das erste Ethernet ist eine Adaption eines auf Hawaii entwickelten, paketorientierten Kommunikationsnetzes auf Funk-Basis namens Aloahnet. 1973 baute Metcalfe ein Testnetz in Palo Alto auf. Als Medium, an das sich jede Station anschließen konnte, wurden Koaxialkabel genutzt. Die Übertragungsgeschwindigkeit von 2,94Mbps leitete sich aus der Taktfrequenz einer frühen Version eines Personal Computers ab, dem „Alto“ von XEROX. Metcalfe nannte diese Technologie Ethernet, um mit dem Wort „ether“ (engl. für Äther[4]) das physikalische Medium zu bezeichnen.

Metcalfe stellte die These auf, dass der Wert von Netzwerken exponentiell mit der Anzahl der Nutzer steigt. In den 70er Jahren sagte Intel Mitbegründer Gordon Moore voraus, dass alle 18 Monate die Leistung der Mikroprozessoren sich verdoppeln wird, während der Preis um die Hälfte sinkt. Diese beiden Theorien bewahrheiteten sich. Während die Geschwindigkeit des Ethernets fortwährend stieg, von 10 Mbps, zu 100 Mbps, dann zu 1 Gbps und schließlich zu 10 Gbps, sank der Preis der Ethernet Technologie. Gleichzeitig gewannen Netzwerke immer mehr an Bedeutung, so dass Ethernet sich in Local Area Networks (LAN) durchgesetzt und andere LAN-Technologie wie Token Ring verdrängt hat. Mit der Einführung von Gigabit Ethernet, dachte man auch verstärkt an die Nutzung von Ethernet in Metropolitan Area Networks (MAN). Durch die Entwicklung von 10-Gbit-Ethernet will man mit der Technologie weiter in den MAN Bereich vordringen, um es dann letztendlich auch in Wide Area Networks zu nutzen.

1980 veröffentlichte ein Ausschuss des Institute of Electrical and Electronics Engineers-(IEEE-)802 einen geringfügig geänderten Ethernet-Standard. Die Bezeichnung ist 802.3 und es deckt alle CSMA/CD-Netzwerke ab. CSMA/CD steht für: Carrier Sense Multiple Access/Collision Detection. Dieses spezielle Verfahren regelt die Sendeberechtigungen der einzelnen Stationen. Das ursprüngliche

[4] Es wurde im 19 Jh. angenommen, dass Äther das Medium zur Übertragung von Funkwellen ist.

Ethernet verwendete ein Medium, welches von allen Teilnehmern genutzt wurde. Da es sein kann, dass zwei Stationen zur gleichen Zeit auf das Medium zugreifen, musste ein Zugriffsverfahren entwickelt werden. Bei Ethernet ist das die Aufgabe des Medium-Access-Control (MAC)-Protokolls. 10-Gbit-Ethernet braucht dieses Zugriffsverfahren nicht, da es nur im Vollduplex Modus arbeitet. 10-Gbit-Ethernet wurde mit der Ergänzung 802.3ae standardisiert.

2.1.1 Das OSI-Referenzmodell

Das Open Systems Interconnection (OSI) Referenzmodell ist ein Architekturmodell für offene Kommunikationssysteme. Offene Kommunikationssysteme besitzen die Fähigkeit mit anderen Systemen zu kommunizieren und zu kooperieren. „Kooperieren bedeutet dabei die Kommunikation zwischen Systemen zu dem Zweck, eine gemeinsame Aufgabe zu erledigen bzw. ein gemeinsames Anfangsverständnis durch Übertragung von Informationen fortzuschreiben“ [27]. Das Ziel des OSI Referenzmodells ist es, eine herstellerunabhängige Kommunikation zwischen verschiedenen Teilnehmern zu ermöglichen. Dazu werden standardisierte Protokolle genutzt.

Das ursprüngliche Modell, welches von der International Organization for Standardization (ISO) entwickelt wurde, besteht aus 7 Schichten. Es ist eine Aufteilung der für eine Kommunikation notwendigen Funktionen. Jede Schicht hat ihre eigenen Protokolle, die für einen geregelten Informationsaustausch zwischen Schichten der gleichen Ebene von verschiedenen Endterminals sorgt. Dabei baut eine Schicht auf der anderen auf, d. h. eine Schicht N stellt ihre Kommunikationsmöglichkeiten als Dienst der übergeordneten Schicht N+1 zur Verfügung und nutzt selber die Dienste der Schicht N−1. Die Informationen der Schicht N+1 werden in sogenannte Service Data Units (SDU) gepackt und an die Schicht N weitergegeben. Dort werden eigene Steuerinformationen, die Protocol Control Information (PCI) hinzugefügt. PCI und SDU werden zusammen als Protocol Data Unit (PDU) bezeichnet. Diese PDU wird als SDU an die Schicht N−1 weitergeleitet.

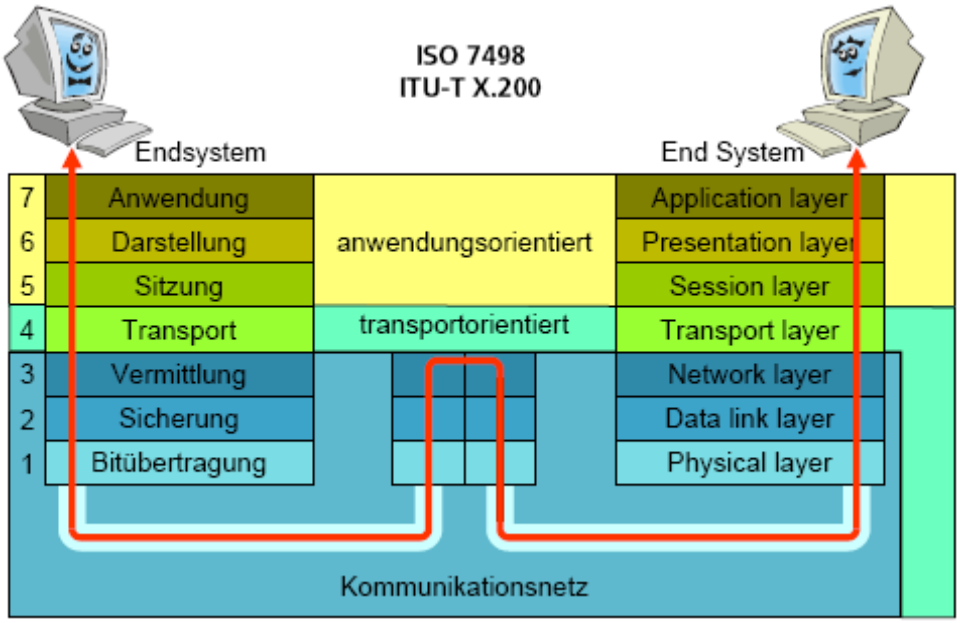

Bild 2.1: Das OSI-Referenz Modell [34]

Die ersten 4 Schichten werden als die Transportschichten bezeichnet. Die Schichten 5-7 sind die „anwendungsorientierten Schichten“. Die Schichten 4-7 werden durch Software implementiert. Die Schicht 3 ist in der Regel eine Softwarelösung, kann aber auch bei Hochgeschwindigkeitsnetzen eine Hardwarelösung sein. Die Schichten 1-3 sind an das Kommunikationsnetz orientiert.

Beschreibung der einzelnen Schichten:

Schicht 7 Anwendungsschicht (Application Layer)

Die Anwendungsschicht ist die höchste Schicht im OSI Referenz Modell. In dieser Schicht befinden sich die Programme, die der Benutzer eines Endsystems bedient. Von hier werden die Daten versendet oder empfangen, die der Anwender nutzt. Applikationen sind z. B. www-Browser, Email-programme oder Voice-over-IP Programme.

Schicht 6 Darstellungsschicht (Presentation Layer)

Die Darstellungsschicht setzt die systemabhängige Darstellung der Daten in eine unabhängige Form um, so können die gleichen Anwendungen auf verschiedenen Endsystemen miteinander kommunizieren (z. B. ASCII Code). Datenkompression und Verschlüsselung sind auch Aufgaben der Darstellungsschicht.

Schicht 5 Kommunikationssteuerungsschicht (Session Layer)

Die Schicht 5 sorgt für die ordentliche und organisierte Eröffnung, Durchführung und Schließung einer Kommunikation oder eines Datenaustau-

sches. Sie stellt Dienste zur Synchronisation der Verbindung zur Verfügung

Schicht 4 Transportschicht (Transport Layer)

In dieser Schicht wird der Datenfluss zwischen zwei Endsystemen hergestellt. Es bietet den anwendungsorientierten Schichten einen einheitlichen Zugriff, so können diese ihre Daten unabhängig von den Eigenschaften des Kommunikationsnetzes versenden. Zu den Aufgaben dieser Schicht zählen auch die Segmentierung des Datenstroms in die für die physikalische Übertragung zu Grunde liegende Größe und die Stauvermeidung. Bei verbindungsorientierten Transportprotokollen werden hier auch die Pakete nummeriert, Pakete bestätigt und bei Bedarf eine Anforderung für ein erneutes Senden eines Pakets gestellt. Die wichtigsten Protokolle sind hier das Transmission Control Protocol (TCP) und das User Datagram Protocol (UDP).

Schicht 3 Vermittlungsschicht (Network Layer)

Die Aufgabe dieser Schicht ist das Schalten und Steuern von Verbindungen und das Übertragen von Pakete über das gesamte Kommunikationsnetz hinweg. Ein wichtiges Element dieser Schicht ist die Leitweglenkung (Routing), d. h. dass mindestens ein Weg über die Knoten des Kommunikationsnetzes gefunden werden muss. Dabei werden die Pakete von den Knoten weitergeleitet und je nach Protokoll ein Zähler um 1 heruntergesetzt. Der Weg von A nach B muss nicht der Gleiche sein wie der Weg von B nach A. Das wichtigste Protokoll ist hier das Internet Protocol (IP).

Schicht 2 Sicherungsschicht (Data Link Layer)

In der Schicht 2 wird aus der ungesicherten Übertragung der Schicht 1 eine weitgehend fehlerfreie Übertragung. Die Hauptfunktionen sind Fehlererkennung und Korrektur sowie Flusskontrolle. Dies wird durch Prüfsummen realisiert, die beim Sender und beim Empfänger mit der gleichen Formel ausgerechnet werden. Die Flusskontrolle kann die Sendegeschwindigkeit der Sender steuern. Weitere Aufgaben sind die Aufteilung der Daten der Schicht 3 in Blöcke und deren Nummerierung. Beispiele sind die Medium-Access-Control (MAC)-Protokolle der LANs.

Schicht 1 Physikalische Schicht oder Bitübertragungsschicht (Physical Layer)

Die Physikalische Schicht ist die niedrigste Schicht. Hier werden die Daten der oberen Schichten ungesichert über das physikalische Medium übertragen. Die Festlegungen für Schicht 1 umfassen vor allem die mechanischen (Steckverbinder, Kabel, etc.), elektrischen (Pegel, Spannung, Pulsform etc.) beziehungsweise optischen Eigenschaften des Übertra-

gungsmediums (Kabel, Glasfaser, Funk etc.). Die Daten werden leitungscodiert und nach Aktivierung der physikalischen Übertragungsstrecke versendet. Danach wird die Übertragungsstrecke wieder deaktiviert.

Diese Diplomarbeit beschäftigt sich hauptsächlich mit den Schichten 1-4. Hier sind alle relevanten Protokolle angesiedelt. Für weitere Ausführungen zu dem OSI Referenz Modell siehe [27, 25, 29].

2.1.2 IEEE 802.3ae

Der 10-Gbit-Ethernet-Standard wurde mit der IEEE 802.3ae im August 2002 veröffentlicht. Die IEEE 802.3ae Task Force hatte fünf wesentliche Kriterien, die der 10-Gbit-Ethernet-Standard erfüllen musste, um als ein Standard übernommen zu werden [1]:

- 10-Gbit-Ethernet muss ein breites Marktpotenzial besitzen und dabei eine breite Palette von Anwendungen unterstützen, die von vielen verschiedenen Anbietern und Nutzern genutzt werden.
- Es muss sowohl zu anderen existierenden 802.3 Protokoll-Standards, als auch mit den Open Systems Interconnection (OSI) und dem Simple Network Management Protocol (SNMP) Spezifikationen kompatibel sein.
- Es muss wesentlich anders von den anderen 802.3 Standards sein, um es somit als eine einzigartige Lösung für ein Problem als eine alternative Lösung zu machen.
- Die technische Ausführbarkeit muss nachgewiesen werden bevor es final ratifiziert wird.
- Die Nutzung dieser Technologie muss wirtschaftlich realisierbar für die Verbraucher sein und somit für angemessene Kosten für die gewünschte Steigerung der Leistung sorgen, inklusive der Installations- und Verwaltungskosten.

10-Gbit-Ethernet erfüllt diese Anforderungen, was auch dafür gesorgt hat, dass diese Technologie vom Max-Planck-Institut für Radioastronomie ausgewählt wurde. 5 Jahre nach Veröffentlichung des Standards gibt es viele Anbieter für die 10-Gbit-Ethernet-Technologie. Unter anderem aus diesem Grund sank der Preis für einen 10-Gbit-Ethernet-Port. Für das Max-Planck-Institut war für die 10-Gbit-Übertragungsstrecke wichtig, dass handelsübliche Bauteile verwendet werden, die von mehreren Firmen angeboten werden, um auf diese Weise Geld zu sparen. Weiterhin ist Ethernet ein sehr verbreiteter Standard, was bedeutete, dass eventuell auf Erfahrungen zurückgegriffen werden konnte.

Mit der Ausweitung von Ethernet auf WAN ergibt sich ein entscheidender Vorteil gegenüber anderen Standards. Ausgedehnte Netzwerk-Topologien lassen sich damit in einer homogenen, auf IP und Ethernet basierenden Technologie realisieren. Das heißt, dass beispielsweise im heutigen Internet ein PC mit einem weit entfernten Server kommunizieren kann, ohne dass Protokolle auf der Übertragungsschicht umgesetzt werden müssen.

Der IEEE 802.3ae-2002 Standard ist in gewisser Hinsicht verschieden von den früheren Ethernet Standards. Als physikalisches Medium wird Glasfaser verwendet und es ist nur ein Vollduplex-Betrieb möglich. Damit wird das Zugriffsprotokoll CSMA/CD nicht mehr benötigt. In diesem Standard wurden insgesamt sieben verschiedene physikalische Schnittstellen definiert, um eine hohe Kompatibilität mit existierenden LAN- und WAN-Verkabelungstypen zu erreichen. Tabelle 2.1 zeigt die verschiedenen Schnittstellen.

Tabelle 2.1: 10-GbE-PHY-Standards

	Modus	WIS	Wellenlänge / nm	Medium	Kodierung	Brutto-Bitrate / Gbit/s
LAN-PHY						
10GBase-SR	serial	nein	850	MMF	64B/66B	10,3
10GBase-LR	serial	nein	1310	SMF	64B/66B	10,3
10GBase-ER	serial	nein	1550	SMF	64B/66B	10,3
10GBase-LX4	WDM[5]	nein	1310	MMF	8B/10B	4x3,125
WAN-PHY						
10GBase-SW	serial	ja	850	MMF	64B/66B	9,953
10GBase-LW	serial	ja	1310	SMF	64B/66B	9,953
10GBase-EW	serial	ja	1550	SMF	64B/66B	9,953

WIS=WAN Interface Sublayer; MMF=Multimode Fiber; SMF=Single Mode Fiber

Wie aus der Tabelle ersichtlich ist, definieren die Buchstaben nach der Klassenbezeichnung 10GBase die verschiedenen Optionen [25]:

- 1. Buchstabe: Wellenlänge (S = short, 850 nm; L = long, 1310 nm; E = extra long, 1550 nm)

[5] WDM steht für *wavelength division multiplexing*. Es ist ein optisches Frequenzmultiplexverfahren, das bei der Übertragung von Daten über Glasfaserkabel verwendet wird.

- 2. Buchstabe: Codierung und Rahmen (X = LAN, 8B/10B; R = LAN, 64B/66B; W = WAN, 64B/66B mit SONET/SDH-Rahmen)

Das WAN Interface Sublayer (WIS) koppelt das Ethernet Protokoll mit SONET/SDH. Die Übertragungsrate ist hierbei nur 9,58464 GBit/s, daher muss die Datenrate angepasst werden.

2.1.3 Beziehung zwischen IEEE 802.3ae und dem OSI Modell

Der 10-Gigabit-Ethernet-Standard sollte zu vielen verschiedenen, existierenden Standards kompatibel sein. Ein wichtiger Punkt war die Kompatibilität zu Standards aus dem OSI-Umfeld. Abbildung 2.2 zeigt die verschiedenen Unterschichten im Bezug zu dem OSI Referenz Model.

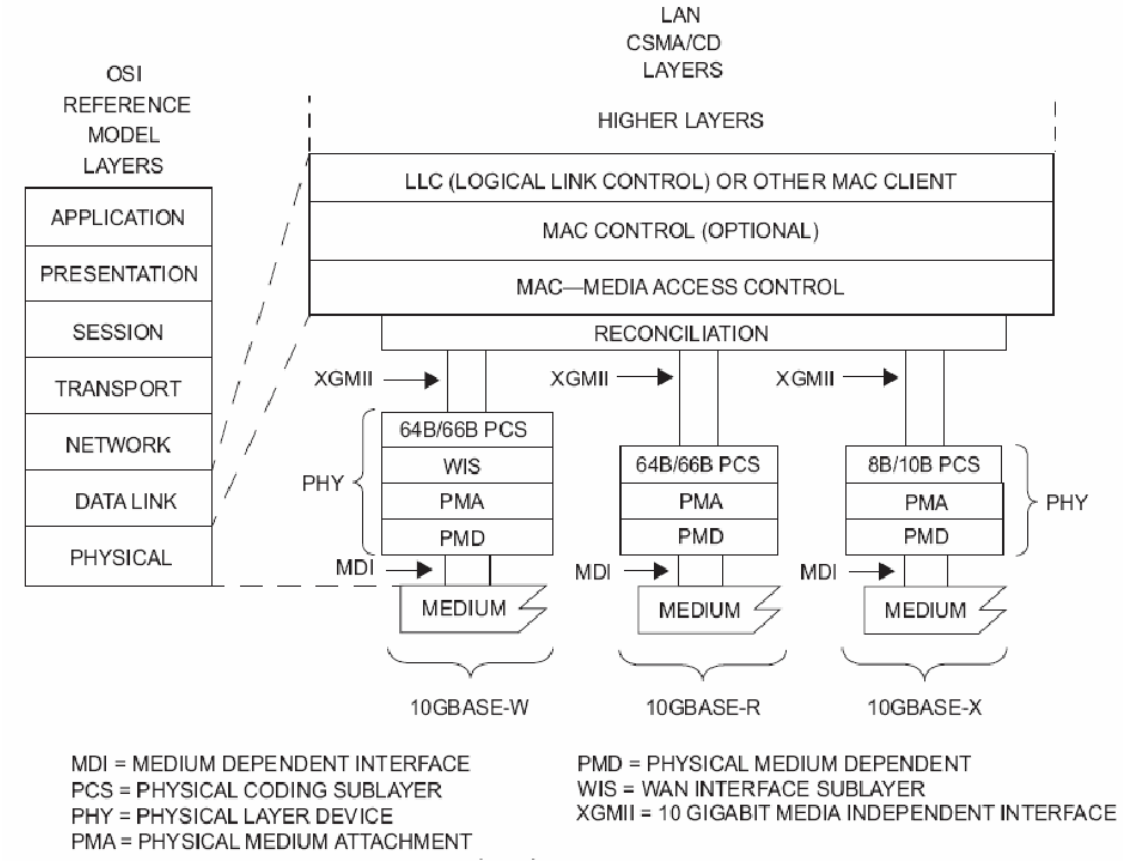

Bild 2.2: IEEE 802.3ae im OSI Referenz Model [13, Seite 2]

Auf dem Bild 2.1 kann man sehen, dass die OSI-Schichten weiter unterteilt sind, damit erreicht man eine große Flexibilität. „IEEE 802 deckt den Bereich der physikalischen Schicht (Schicht 1) und der Sicherungsschicht (Link Layer, Schicht 2) ab.“ [25, Seite 121] Die Sicherungsschicht ist abgesehen von der MAC Control in eine obere Hälfte und eine untere Hälfte unterteilt. Die obere Hälfte ist die Logical Link Control. Diese ist nicht in der IEEE 802.3 definiert, aber sie ist für alle verschiedenen physikalischen Medien der IEEE 802 wie Ethernet,

Token Ring oder WLAN gleich. Die untere Hälfte besteht aus der MAC. Die MAC und alle darunterliegenden Unterschichten und Schnittestellen sind für 10-Gigabit-Ethernet in der IEEE 802.3ae definiert.

2.1.4 Medium Access Control (MAC)

Die MAC Unterschicht beschreibt eine Medium unabhängige Einrichtung, die auf einer Medium abhängigen Schicht aufbaut, der physikalischen Schicht, und unter der LLC Unterschicht (oder einem anderem MAC Client) liegt. Die MAC Schicht ist auf eine große Anzahl von verschiedenen physikalischen Schnittstellen im LAN, MAN und mit 10-Gigabit-Ethernet auch WAN anwendbar. Eine Hauptaufgabe der MAC ist die Durchführung des Zugriffsverfahrens CSMA/CD. Das ist auch der wesentliche Unterschied des 10 GbE MACs zu den übrigen Verfahren. Für die Übertragung von 10 Gbit sind nur Punkt-zu-Punkt Verbindungen vorgesehen, d. h. dass es nur im Vollduplex Modus arbeitet. Der 10 GbE MAC stellt diese Möglichkeit zur Verfügung.

Die anderen Aufgaben eines MACs übernehmen auch die 10 GbE MACs. Das sind:

- Daten Verkapselung (Versand und Empfang)
    - Rahmung (Einstellung der Rahmengrenzen, Rahmensynchronisation)
    - Adressierung (Bearbeitung von Quellen- und Senken-Adressen)
    - Fehlererkennung (die Erkennung von Übertragungsfehler des physikalischen Mediums)
- Zugriff auf tieferliegende Schichten

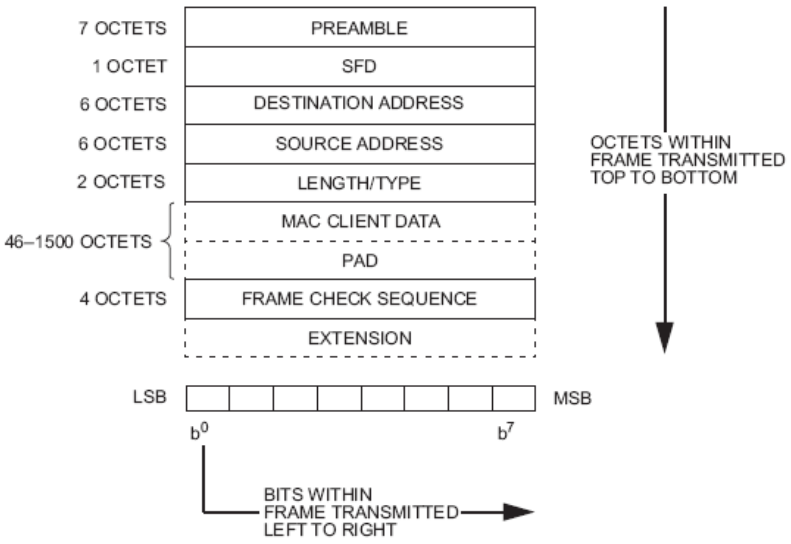

Bild 2.3: Standard MAC Rahmen [16, Seite 49]

Die 10 GbE Rahmenstruktur entspricht dabei vollständig jener der anderen Ethernet Rahmen. Bild 2.3 zeigt die neun Felder des Standard MAC Rahmens. In der IEEE 802.3-2005 sind zwei verschiedene Rahmenformate spezifiziert. Außer dem Standard Rahmen gibt es noch den Tagged-MAC-Rahmen (siehe Bild 2.4). Hierbei werden nach der Quell-MAC-Adresse 4 Bytes mit dem QTAG-Präfix eingeschoben. Dieser QTAG ist in der IEEE 802.1Q definiert. Es ermöglicht bis zu 4096 virtuelle lokale Netzwerke (VLANs) auf einem physikalischen Medium.

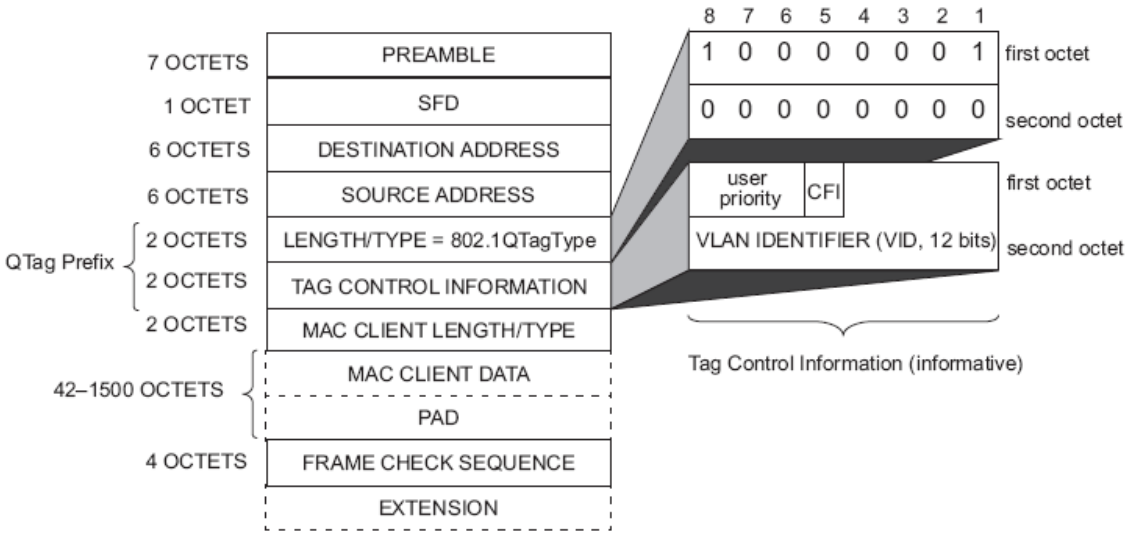

Bild 2.4: Tagged-MAC-Rahmen [16, Seite 53]

Die *Preamble* (Präambel) ist eine 7 Byte lange alternierende Bitfolge (10101010 1010…). Diese Bitsequenz wird zur Stabilisierung und Synchronisation benutzt. Wie auf dem Bild 2.3 zu sehen ist, wird die Bitfolge von links nach rechts übertragen. Benutzt wird eine Manchester Codierung mit einer steigenden Flanke für eine 1.

Anschließend folgt der *Start Frame Delimiter* (SFD). Der SFD ist nur 1 Byte lang und hat folgenden Wert: 10101011. Dieser Wert kennzeichnet den Start eines Ethernet Rahmens.

Am Anfang des Ethernet Rahmens befindet sich die Quell- und Ziel-MAC-Adresse. Diese Adressen sind 48 Bit breit und vom Hersteller weltweit eindeutig in die Schnittstellenkarte einprogrammiert. Sie wird üblicherweise in Hexadezimalform dargestellt. Das erste Bit, also das LSB, kennzeichnet, ob es eine individuelle oder eine Gruppen-Adresse ist. Hat das Bit den Wert 1, ist es eine Gruppen-Adresse und adressiert keinen, einen oder mehr Stationen im verbunden LAN. Das zweite Bit unterscheidet zwischen *globally administered*-(0)- oder *locally administered*-(1-Adressen. Die restlichen 22 Bit der ersten 3 Bytes einer MAC-Adresse geben den Hersteller der Netzwerkkarte an. Diese Adressen

werden von der IEEE vergeben. Die letzten 3 Bytes einer MAC-Adresse werden von der Firma ausgewählt und verwaltet. Die Zuordnung der IP-Adressen zu den MAC-Adressen geschieht durch das *Address Resolution Protocol* (ARP). Umgekehrt wird das *Reverse Address Resolution Protocol* (RARP) genutzt.

Das *Length/Type* ist 2 Bytes breit und hat zwei Bedeutungen abhängig vom numerischen Wert dieses Feldes. „Ist der Wert des Feldes kleiner als 0600_{HEX}, dann handelt es sich um ein Längenfeld, ist der Wert gleich oder größer, dann handelt es sich um den Typ“ [25, Seite 135]. Die Typen werden von der IEEE bestimmt und verwaltet. Die MTU von 1518 Bytes für Ethernet wurde zu einer Zeit bestimmt, da noch keine hohen Datenraten von 1 Gbit/s oder höher vorhanden waren. Um die CPU-Ausnutzung und die Datenrate zu erhöhen wurden sogenannte *Jumbo Frames* eingeführt. Diese neuen Rahmen übertragen Daten im Ethernet Rahmen von 9000 Bytes. Die *Jumbo Frames* wurden nicht von der IEEE in den Standard 802.3 aufgenommen, da die Interoperabilität mit anderen IEEE 802 Protokollen wie Token Ring oder Wireless LAN nicht vorhanden ist.

Nach dem *Length/Type* Feld folgen die zu übertragenden Daten. Die Maximale Länge nach IEEE 802.3-2005 beträgt 1500 Bytes. Sie errechnet sich aus dem Maximalen Standard Ethernet Rahmen minus Ziel-/Quell-MAC-Adresse, *Length/Type* Feld und der Prüfsumme. Für eine sichere Kollisionserkennung muss eine minimale Rahmenlänge vorhanden sein. Diese beträgt bei allen Ethernet Rahmen 64 Bytes. Wenn die minimale Länge nicht durch die Daten allein gewährleistet ist, wird das Datenfeld mit Extrabits erweitert. Das ist das *PAD*-Feld.

Eine zyklische Redundanzprüfung (engl. *Cyclic Redundancy Check*, CRC) wird zur Überprüfung des Ethernet-Rahmens im *Frame Check Sequence*-Feld benutzt. Dieser 4 Byte breite CRC Wert wird aus einer Funktion der Ziel-/Quell-MAC-Adresse, dem *Length/Type*-Feld, dem Daten und *PAD*-Feld generiert. Die Präambel, der *SFD*, die FCS und das *Extension*-Feld werden nicht im Algorithmus benutzt. Die FCS sendet das höchstwertigste Bit zuerst. Alle anderen Felder beginnen mit dem niederwertigsten Bit.

Das *Extension*-Feld „dient in Hochgeschwindigkeitsversionen zur Verlängerung des Rahmens, um die Zeitbedingungen für CSMA/CD wieder herzustellen“ [25, Seite 136].

2.1.5 Reconciliation Sublayer (RS) und 10 Gigabit Media Independent Interface (XGMII)

Das XGMII ist eine Medium unabhängige Schnittstelle, die die MAC-Unterschicht mit der Physikalischen Schicht verbindet. Optional kann eine *10 Gigabit Attachment Unit Interface* benutzt werden. Diese kann die Betriebsdistanz von XGMII mit einer geringeren Anzahl von Pins verlängern (siehe 2.1.9). Der *Reconciliation Sublayer* wandelt die serielle Bitübertragung der MAC-Unterschicht in eine für die 10-Gigabit Physikalische Schicht geeignete parallele Übertragungsform. Das XGMII übergibt die Daten in Sende- und Empfangsrichtung jeweils über 32 Bit breite Busse. Mit den Steuer- und Taktsignalen ergeben sich insgesamt 74 Leitungen. Die Daten werden in 4 Datenbahnen parallel übertragen.

Tabelle 2.2: XGMII-Leitungsbelegung [13]

Signal	Richtung	Beschreibung
XGMII_TX_CLK	Output	Takt nach PHY
XGMII_TXD[31:0]	Output	Übertragungsdaten nach PHY, DDR-Signalisierung
XGMII_TXC[3:0]	Output	Steuerleitungen nach PHY
XGMII_RX_CLK	Input	Aus den Empfangsdaten regenerierter Takt von PHY
XGMII_RXD[31:0]	Input	Empfangsdaten von PHY, DDR-Signalisierung
XGMII_RXC[3:0]	Input	Steuerleitungen von PHY

DDR = Double Data Rate

2.1.6 Physical Coding Sublayer (PCS)

Die *Physical Coding Sublayer* (PCS) ist die oberste Unterschicht der Physikalischen Schicht nach IEEE 802.3-2005. XGMII ist die Service-Schnittstelle der PCS (siehe Bild 2.2). An der anderen Seite befindet sich nach 10GBASE Standard die *Physical Medium Attachment* (PMA) oder die *WAN Interface Sublayer* (WIS). Das WIS ist eine optionale PHY-Unterschicht, die nur in WAN-PHYs eingefügt wird. Die Hauptaufgabe der WIS ist es eine 10GBASE-W PHY zu erstellen, die in der Datenrate und dem Format sowohl kompatibel mit dem SONET STS-192c Übertragungsformat, als auch mit den *Synchronous Digital Hierarchy* (SDH) VC-4-64c Container ist. So kann der eigentliche Ethernet Datenstrom direkt auf STS-192c oder VC-4-64c Ströme auf physikalischer Ebene abgebildet werden.

Für 10 Gigabit Ethernet wurden zwei verschiedene Codierungsverfahren ausgewählt. Bei 10GBASE-LX4 kommt die 8B/10B-Kodierung zum Einsatz. Diese überträgt je Byte zusätzlich zwei redundante Bits mit. Hierbei sind vier parallele Datenströme vorhanden, die jeweils eine Nutzdatenrate von 2,5 GBit/s besitzen. Die Bruttodatenrate beträgt entsprechend 3,125 GBit/s. Das sind insgesamt 12,5 GBit/s. Bei der 64B/66B-Kodierung werden pro 64 Bits, die übertragen werden, ein Overhead von 2 Bits angehängt. 10GBASE-R und 10GBASE-W nutzen diese Art der Kodierung. Die 2 Bytes werden zur Takt-Synchronisation und zur Fehlererkennung genutzt.

Die PCS bietet alle Dienste, die von der XGMII benötigt werden, inklusive folgenden Funktionen [13]:

- De-/Kodierung der Daten-Bytes von der XGMII. Entweder 8B/10B oder 64B/66B
- Übertragung der kodierten Daten von und zum PMA in 16 bit Übertragungen.
- Wenn es mit einer WIS verbunden ist, Löschen und Hinzufügen von Bytes, um die Differenz der Datenraten zu kompensieren.

2.1.7 Physical Medium Attachment (PMA)

Das *Physical Medium Attachment* (PMA) ist eine Medienunabhängige Unterschicht in der Physikalischen Schicht, deren Hauptaufgabe die Verbindung der Medienabhängigen Unterschicht nach Wahl der *Physical Medium Dependent* (PMD) mit der dienstanfordernden Unterschicht, die PCS oder WIS, ist. Folgend werden die Funktionen der PMA jeweils in Senderichtung, d. h. von der PCS oder WIS zur PMD, und in Empfangsrichtung, d. h. von der PMD zur WIS oder PCS, beschrieben [13]:

Senderichtung:

- Bereitstellung des Sendetaktes für das dienstanfordernde Gerät
- Anordnung des 16-bit Datenstroms in einen seriellen Bitstrom
- Übertragung der seriellen Daten zur PMD

Empfangsrichtung:

- Gewinnung des Bittaktes aus den seriellen Daten der PMD
- Bereitstellung des Empfangstaktes für das dienstanfordernde Gerät
- Übertragung der parallelen Daten zu dem dienstanfordernden Gerät
- Bereitstellung von Anschluss Status Informationen

2.1.8 Physical Medium Dependent (PMD)

Das *Physical Medium Dependent* (PMD) ist die unterste Unterschicht der physikalischen Schicht. Sie ist Medienabhängig und verbindet die Medienunabhängigen, dienstanfordernden Unterschichten, wie die PMA, mit dem *Medium Dependent Interface* (MDI) (siehe Bild 2.2). Hier sind die optischen und elektrischen Eigenschaften der Transceiver festgelegt. Die Datenbits aus den oberen Schichten werden in die für die Übertragung auf dem entsprechenden Medium (Glasfaser oder Kupfer) umgewandelt, d. h. bei Glasfaser findet eine Umwandlung von einem elektrischen Signal in ein optisches Signal oder umgekehrt statt. Für jeden 10GBase Standard sind die optischen Spezifikationen in der IEEE 802.3-2005 vorgegeben. Neben der Umwandlungs- und der Weiterleitungsfunktion von Daten für den Versand und Empfang von den oberen Schichten zum Übertragungsmedium und zurück, muss die PMD auch eine Funktion zur Signalerkennung zur Verfügung stellen.

Über Glasfaser werden bei 10-Gigabit-Ethernet die drei „optischen Fenster" bei 850, 1310 und 1550 Nanometer benutzt. Tabelle 2.3 zeigt die verschiedenen Standards auf der PMD-Unterschicht mit dem Übertragungsmedium und der Reichweite.

Tabelle 2.3: 10GBASE PMDs mit Reichweite [13]

Name	Beschreibung	Medium	Reichweite
10GBASE-SR	850 nm serielle LAN PHY	MMF	300m
10GBASE-LR	1310 nm serielle LAN PHY	SMF	15km
10GBASE-ER	1550 nm serielle LAN PHY	SMF	40km
10GBASE-SW	850 nm serielle WAN PHY	MMF	300m
10GBASE-LW	1310 nm serielle WAN PHY	SMF	15km
10GBASE-EW	1550 nm serielle WAN PHY	SMF	40km
10GBASE-LX4	1310 nm WWDM LAN PHY	MMF	15km
10GBASE-CX4	LAN PHY über Kupferkabel	Kupfer	15m

2.1.9 10 Gigabit Attachment Unit Interface (XAUI) und XGMII Extender Sublayer (XGXS)

Das *10 Gigabit Attachment Unit Interface* (XAUI) ist eine optionale Hochgeschwindigkeitsschnittstelle. Sie kann zur Verlängerung der XGMII oder auch anstatt der XMGII benutzt werden. Die XAUI und die *XGMII Extender Sublayer* (XGXS) liegen zwischen der *Reconciliation Sublayer* und der *Physical Coding*

Sublayer (siehe Bild 2.5). Die Absicht hinter dem *XGMII Extender* ist die Verlängerung der XGMII und die Reduzierung der Pin-Anzahl. Dabei ist es auch möglich eine physikalische Trennung von MAC-Unterschicht und PHY-Unterschicht vorzunehmen.

Die XAUI ist ein vierspuriges serielles Interface. Jede Spur überträgt differentiell die Daten, d. h. das eine Spur zwei Datenleitungen besitzt, was insgesamt für Sende- und Empfangsrichtung eine Anzahl von 16 Datenleitungen bedeutet. Im vergleich dazu benutzt XGMII 74 Datenleitungen. Eine Datenbahn muss mit einer Datenrate von 2,5 GBit/s betrieben werden, damit 10 GBit/s übertragen werden können. Die XAUI verwendet dabei die 8B/10B-Kodierung. Folglich hat eine Datenleitung eine Bruttodatenrate von 3,125 GBit/s.

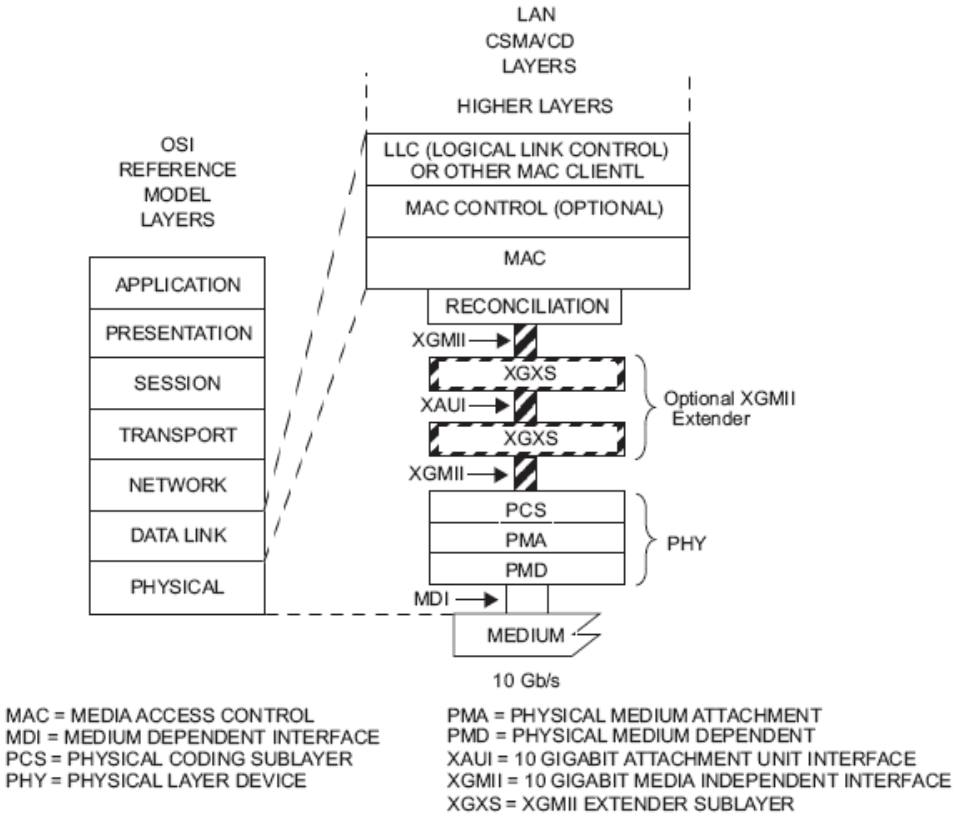

Bild 2.5: Beziehung von XAUI und XGXS zu dem OSI Referenz Modell und IEEE 802.3 [13, Seite 177]

Die Sende-XGXS nimmt die XGMII-Datenströme aufgeteilt auf vier Datenbahnen aus seinem Eingang, bildet die XGMII Daten- und Kontrollwörter in XAUI Kodegruppen ab und kodiert diese vor der Übertragung. Auf der Empfangs-XGXS werden die Kodegruppen dekodiert, die vier Datenbahnen demultiplext, Taktraten-Unterschiede kompensiert und die XAUI-Kodegruppen wieder zurück in XGMII-Daten- und Kontrollwörter abgebildet. Die XAUI ist bidirektional, d. h. das jede XGXS Sende- und Empfangsfunktionen haben kann.

Die XAUI ist ein selbstgetakteter Bus. Die Ziel-XGXS entnimmt aus jeder der vier Datenbahnen den Takt, entzerrt den Versatz der vier Takte und wandelt diese in einen Takt für die XGMII um. Neben den Kontrollwörter werden auch Leerbefehle in 8B/10B Kodesequenzen konvertiert. Diese „leeren" Kodesequenzen werden zur Taktratenkompensation von der Ziel-XGXS genutzt, bevor die Kodesequenzen wieder in den XGMII-Kode umgewandelt werden. Die XAUI hat somit eine gute Stabilität gegen Laufzeitunterschiede. Durch die die differentielle Übertragung können hohe Datenraten, hohe elektromagnetische Verträglichkeit und Robustheit gegen Übertragungsstörungen erreicht werden.

Die XGXS nutzt den gleichen Kode und Kodierungsregeln wie im Fall des 10GBASE-X Unterstandard. Wird die optionale XAUI mit einem solchen Standard genutzt, erfüllt das XGXS an der *Reconciliation Sublayer* die gleichen Funktionen wie zwischen der PCS und der PMA, die von der Physikalischen Schicht verlangt werden. Eine XGXS wird in der Physikalischen Schicht nicht mehr benötigt.

2.1.10 10 Gigabit Small Form Factor Pluggable Modules (XFP) und High Speed 10 Gb/s serial Electrical Interface (XFI)

Das *High Speed 10Gb/s serial Electrical Interface* (XFI) ist eine elektronische Schnittstelle zum Anschluss von *10 Gigabit Small Form Factor Pluggable Modules* (XFP). Die XFP Spezifikationen wurden wie die XFI Schnittstelle von der XFP Multi-Source Agreement Group entwickelt [2]. Diese Spezifikation definiert die elektrischen, verwaltungstechnischen und mechanischen Schnittstellen eines XFP Moduls. Das Modul ist ein *hot pluggable*, serieller, von den Daten unabhängiger optischer oder elektronischer Transceiver mit kleiner Basisfläche. *Hot pluggable* bedeutet, dass das Modul während dem Betrieb der Netzwerkkarte ohne Schaden ein- oder ausgesteckt werden kann. Es ist für den Einsatz in Telekommunikations-(SONET OC-192 und G.709 „OTU-2") sowie Datenanwendungen (10-Gigabit-Ethernet und 10-Gigabit-Glasfaser) gedacht. Die normalen Datenrate reichen von 9,95 Gb/s, 10,31 Gb/s, 10,52 Gb/s, 10,70 Gb/s und den aufkommenden 11,09 Gb/s. Es werden alle Datenkodierungen für oben genannte Technologien unterstützt. Es können Single-Mode Fasern oder Multi-Mode Fasern mit optischen Schnittstellen bei 850 nm, 1310 nm und 1550 nm benutzt werden.

Bild 2.6: XFP Modul [17]

Die XFI Signalisierung basiert auf einer Hochgeschwindigkeits-Logik mit geringer Spannung und 100 Ω differentieller Impedanz. Das Modul ist kapazitiv gekoppelt. Für eine gute EMV und niedrige Leistungsaufnahme beträgt das nominelle differentielle Signallevel 500 mV Spitze-Spitze. Bild 2.7 zeigt die XFI im Bezug zum MAC/Ethernet Kontroller. Sie befindet sich zwischen dem XFP Transceiver und einem Hochgeschwindigkeits-Multiplexer-Demultiplexer. Die XFI Schnittstelle wurde zur Unterstützung von SONET OC-192, IEEE.Std-802.3ae, 10GFC und G.709 (OTU-2) Anwendungen entwickelt. Der Mux/Demux wird für die Erfüllung der Anwendungs-anforderungen für Jitter Generation und Übergabe benötigt, wenn dieser mit einem konformen XFP Modul verbunden ist. Bei 10 Gigabit Ethernet oder 10 Gigabit Fibre Channel sind geringere Anforderungen für den Jitter als bei Telekommunikations-anforderungen gesetzt. Tabelle 2.4 zeigt die von der XFI unterstützen Datenraten.

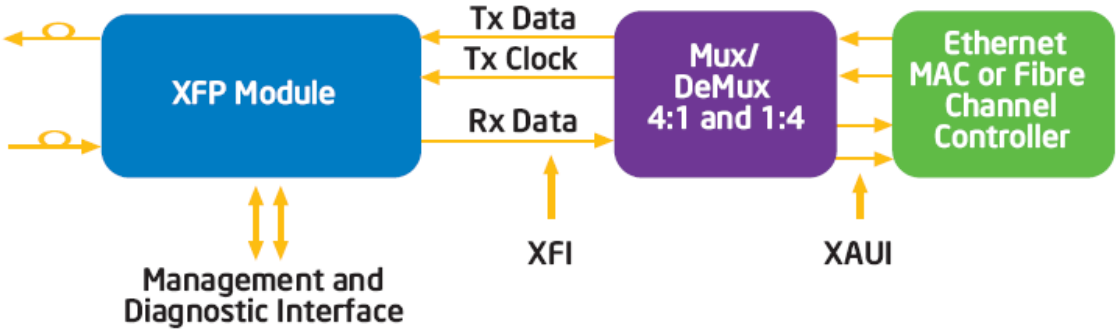

Bild 2.7: XFP Transceiver und XFI Block Diagramm [17]

Tabelle 2.4: XFI unterstützte Datenraten [2, Seite 17]

Standard	Beschreibung	Datenrate / Gbd
OC-192/SDH-64	SONET	9,95
IEEE std-802.3ae	10 Gb/s Ethernet LAN PHY	10,31
INCITS/T11 Project 1413-D	10GFC	10,52
ITU G.709 (OUT-2)	OC-192 Over FEC	10,70
Emerging	10 Gb/s Ethernet Over G.709	11,09

2.2 Netzwerkhardware

Für die Untersuchung der 10-Gigabit-Glasfaserübertragungsstrecke wurde ein Punkt-zu-Punkt Netzwerk aus zwei handelsüblichen PCs aufgebaut. Auch die spätere Verwendung des FiLa 10G Boards ist nur für eine Proprietäre-Verbindung gedacht. Um die Übertragungsstrecke zu verstehen, muss man sich nicht nur mit den verwendeten Protokollen und Schnittstellen vertraut machen, sondern auch mit der verwendeten Hardware. Das entscheidende Bauteil in einem PC für die Übertragung mittels eines Netzwerkes ist die Netzwerkkarte, die zwei Systeme miteinander verbindet. Als die Datenraten der Netzschnittstellen die Datenraten der Busschnittstellen noch nicht übertroffen haben, waren die restlichen Bauteile in einem PC, der Bus auf dem Motherboard, die CPU, der Arbeitsspeicher, die Busschnittstelle für den Anschluss einer Festplatte und die Festplatte selbst für das Erreichen der Höchstgeschwindigkeit der Netzschnittstelle nicht von Bedeutung. Mit der Entwicklung von 1-Gigabit-Ethernet und 10-Gigabit-Ethernet hat sich jedoch der Engpass gerade auf diese Bauteile gelegt.

In diesem Kapitel soll die Funktion einer Netzwerkkarte umrissen werden und die wesentlichen Technologien für eine hohe Netzwerk-Performance vorgestellt werden.

2.2.1 Aufbau und Funktion einer Netzwerkkarte

Eine Netzwerkkarte ist eine Platine oder eine andere Hardware-Komponente, die Daten mit anderen Computern austauscht. Aktuelle Netzwerkadapter für das meist benutzte Lokale Netzwerk, das Ethernet, werden auf dem Motherboard direkt implementiert. Dabei ist der MAC/Ethernet-Kontroller entweder direkt im Chipsatz des PCs (Northbridge oder Southbridge) oder es befindet sich ein eigenständiger Kontroller in Form eines Mikroprozessors auf dem Motherboard. Der Begriff Netzwerkkarte kommt aus der alten Verwendung eines Netzwerkadapters, wo dieser als eine Erweiterungskarte in das Bussystem eines PCs (z. B. ISA, PCI oder PCI Express) eingesteckt wurde. Solche Erweiterungskarten gibt es immer noch, jedoch sind es meist spezielle Netzwerkkarten mit z. B. mehreren Ethernet-Anschlüssen, Karten für den Anschluss an andere LAN Techniken wie Token Ring oder neuere Technologien wie 10-Gigabit-Ethernet.

Der Netzwerkadapter besitzt mindestens zwei Schnittstellen. Eine zur Verbindung mit der Busstruktur des PCs und somit an den Arbeitsspeicher, der CPU und anderen Bauteilen und die andere ist die physikalische Schnittstelle zum Kommunikations-netzwerk. Das ist in meisten Fällen ein LAN oder MAN und

mit der Entwicklung von höheren Übertragungsraten kann es auch ein WAN sein.

Eine Netzwerkkarte ist ein nach dem OSI-Referenz-Modell Schicht 2 und Schicht 1 Gerät, d. h. dass die Netzwerkkarte die Aufgaben der Sicherungsschicht und der Bitübertragungsschicht übernimmt. Alle anderen Dienste der oberen Schichten werden von der CPU oder von dem Chipsatz auf dem Motherboard erledigt. Bei Netzwerkkarten mit neueren Technologien werden jedoch einige Dienste der Vermittlungsschicht und der Transportschicht, die zur Übertragung von Daten über ein Netzwerk genutzt werden, von der CPU oder dem Chipsatz auf die Netzwerkkarte übertragen. Damit erreicht man eine geringere CPU-Auslastung, wobei dann die eingesparten Ressourcen der CPU für andere Aufgaben verwendet werden kann, und bei geeigneter Einstellung können hohe Datenraten nahe der Maximalen für den entsprechenden Standard des Netzwerkes erreicht werden.

Bild 2.8 zeigt den Aufbau einer klassischen Netzwerkkarte. Eine entsprechende Adapterkarte zeigt Bild 2.9. Über den Busanschluss werden die Daten und Befehle von der CPU über seinen Arbeitsspeicher an den Ethernet Chip der Netzwerkkarte geschickt. Die Daten werden im RAM zwischengespeichert. Die Aufgabe des Ethernet Chips ist nun die Aufbereitung der Daten in eine Form, die für die Übertragung über das angeschlossene Netzwerk geeignet ist. Danach können die Daten über einen Anschluss über das physikalische Medium zu einem anderem Computer übertragen werden. Auf der anderen Seite werden die empfangenen Daten in eine für diesen Computer verständliche Form übersetzt. Hier werden ebenso über den Busanschluss die Daten von der Netzwerkkarte in den Arbeitsspeicher des Computers geschrieben.

Es gibt zwei verschiedene Techniken, wie die Daten von der Peripherie in den Hauptspeicher geladen werden, und zwei verschieden Techniken, wie der CPU signalisiert wird, dass Daten übertragen werden sollen. Eine Netzwerkkarte sowie andere Peripherie-Geräte wie ein CD-ROM Laufwerk, kann eine oder mehrere dieser Techniken nutzen [22].

Signalisierungstechniken:

- Beim Abfragebetrieb (engl. *Polling*) wird der Status bestimmter Register im Hauptstatusregister ständig überprüft. Ist das der Fall, sollen entweder Daten von der Peripherie in den Hauptspeicher oder die CPU gelesen oder geschrieben werden. Nachteile dieses Verfahrens ist zum einen die Belegung von CPU-Ressourcen für das ständige Abfragen. Diese Ressourcen könnten für andere, wichtigere Anwendungen genutzt werden. Zum zweiten muss die Abfrageroutine in das Programm der CPU integriert werden, was zu mehr Code und geringerer Performance führt.

- Eine andere Methode ist die des Hardware-Interrupts. Dabei teilt die Hardware selbst der CPU mit, dass Daten ausgetauscht werden können. Es wird ein Signal IRQ aktiviert, um dem Prozessor anzuzeigen, dass ein vollständiges Signal empfangen worden ist und an den Prozessor übergeben werden kann. Die Interrupts werden von einem eigenen Interrupt-Kontroller verwaltet, welcher dann eine Interrupt-Anforderung an den Prozessor stellt. Da die CPU direkt weiß, dass Daten zur Übertragung bereit stehen, entfällt das ständige Prüfen. Durch den geringeren Aufwand erhöht sich die Ausführungsgeschwindigkeit und es können höhere Übertragungsraten erzielt werden.

Speicherzugriff:

- Der Speicherzugriff auf externe Geräte wird zum einen über einen I/O-Adressraum durchgeführt. „Die CPU kann Daten von einem Peripheriegerät in ein internes Register und von diesem Register schließlich in den Hauptspeicher übertragen.“ [22, Seite 576] Die Übertragung funktioniert auch in die andere Richtung. Der I/O-Adressbereich wird dazu verwendet, Register in Peripherieeinheiten anzusprechen. Der Zugriff von der CPU erfolgt dabei wie beim Zugriff auf den Hauptspeicher. Anhand von Steuersignalen erkennt die Logik jedoch, dass auf den I/O-Adressbereich anstatt auf den Speicher zugegriffen werden soll. Statt der Speichersteuerung wird die I/O-Steuerung aktiviert. Die CPU übernimmt hierbei die komplette Datenübertragung, was dazu führen kann, dass beim Austausch von großen Datenblöcken die CPU lange mit dieser Aufgabe beschäftigt ist.
- Beim direkten Speicherzugriff (engl. *Direct Memory Access*, DMA) enthält der Computer neben der CPU eine weitere Schaltungseinheit, den DMA-Kontroller, die selbstständig auf den Hauptspeicher oder die Peripherie zugreift. Die Datenübertragung erfolgt hierbei nicht über den Prozessor, sondern über einen eigenständigen Datenbus zwischen Hauptspeicher und Peripherie. Der DMA-Kontroller übernimmt die Aufgaben der Adressierung des Speichers und die Steuerung des Datenbusses. Die Peripherieeinheit kann somit direkt auf den Hauptspeicher zugreifen.

Neben der Aufbereitung der Daten für die Übertragung über die Busschnittstelle oder über die Netzwerkschnittstelle und die Übertragung der Daten über das physikalische Medium hat die Netzwerkkarte die Aufgabe der Flusssteuerung (engl. *Flow Control*). Steht auf Empfangsseite nicht genug *Receive Buffer* zur Verfügung, werden bei ausgeschalteter Flusssteuerung die ankommenden Pakete verworfen. Ist sie eingeschaltet, sendet die Empfangsseite ein Pause-Kommando an die Sendeseite. Der Sender hört mit der Übertragung der Pakete auf bis der Empfänger wieder genügend Speicher hat, um die ankommenden

Pakete zu verarbeiten, und er einen Befehl zur Aufhebung des Pause-Kommandos gesendet hat.

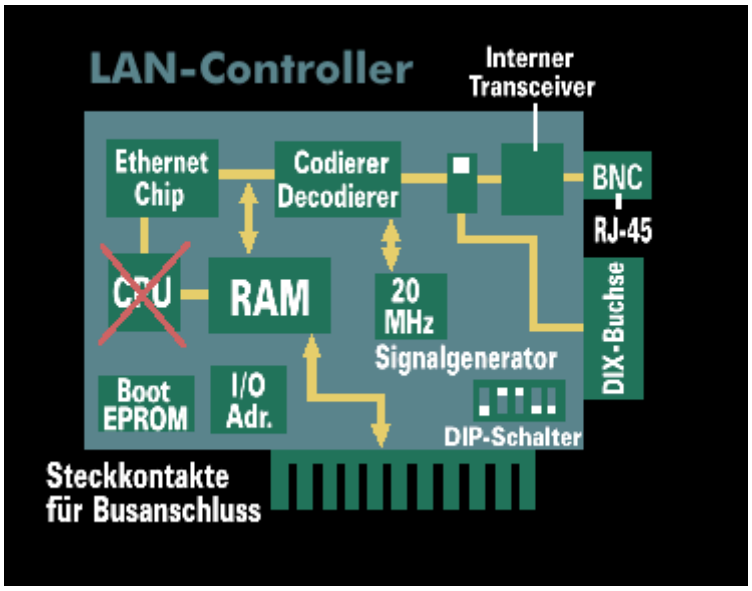

Bild 2.8: Klassischer Aufbau einer Netzwerkkarte [24]

Die Netzwerkkarten auf dem Bild 2.8 und dem Bild 2.9 haben drei verschiedene Anschlussbuchsen je nach Ethernet Version. Sie sind mit einem 15poligen D-Sub-Stecker, einem BNC[6] Stecker und einer RJ-45 Buchse ausgestattet. Aktuelle Netzwerkadapter nutzen die RJ-45 Buchse.

Jede Netzwerkkarte hat eine weltweit eindeutige MAC Adresse. Diese MAC Adressen werden von der IEEE verwaltet und vergeben. Diese Adressen sind meist fest vorgegeben und können nicht verändert werden (siehe 2.1.4).

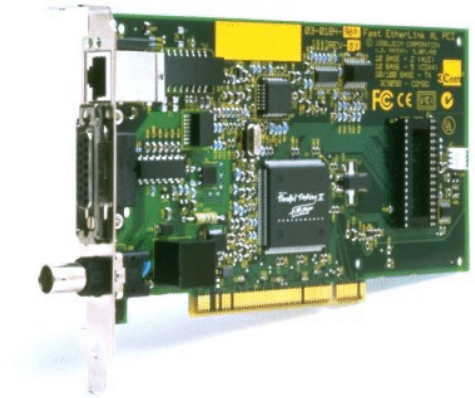

Bild 2.9: Adapterkarte für Ethernet, Fast Etherlink von 3COM [24]

[6] BNC steht für *Bayonet Nut Connector*.

2.2.2 Myricom 10 Gigabit Ethernet Netzwerkkarte

Für den Aufbau einer 10-Gigabit-Testübertragungsstrecke wurde von Myricom die Myri-10G PCI Express Netzwerkkarte mit einem 10GBase-R Port erworben. Es ist eine Vollduplex 10 Gigabit Ethernet Netzwerkkarte mit einer PCI Express Schnittstelle und einer Baudrate von 10,3125 GBaud. Abhängig vom XFP Transceiver sind der physikalische Anschluss und das Übertragungsmedium ein

- 10GBase-SR Port mit einem Laser mit der Wellenlänge von 850 nm auf einer 26-300 m langen MMF
- 10GBase-LR Port mit einem Laser mit der Wellenlänge von 1310 nm auf einer bis zu 10 km langen SMF
- 10GBase-ER Port mit einem Laser mit der Wellenlänge von 1550 nm auf einer bis zu 40 km langen SMF

Die Netzwerkkarte kann mit dem Ethernet und dem Myrinet Protokoll operieren. Das Myrinet Protokoll ist eins von Myricom selber entwickeltes Protokoll der Schicht 2 und 1. Es bietet für PC Cluster einige Vorteile gegenüber Ethernet. Myrinet wird jedoch keine Anwendung für die Übertragungsstrecke im digitalen Empfänger des Max-Planck-Instituts finden, daher wird Myrinet in dieser Diplomarbeit nicht näher beschrieben.

Bild 2.10: Myri-10G PCI Express Netwerkkarte mit einem 10GBase-R Port [23]

Die Myri-10G besitzt eine 8-spurige PCI Express Schnittstelle. Die Schnittstelle besitzt eine theoretische Übertragungsrate von 2 GBytes/s, damit ist sie Schnell genug, um die 1,25 GBytes/s des 10 Gigabit Ethernets zu übertragen.

Die Netzwerkkarte basiert auf einem Myricom eigenen 32-bit VLSI Chip mit dem Namen Lanai Z8E. Der RISC und andere Prozessoren innerhalb des Lanai Z8E arbeiten mit einer Taktfrequenz von 313 MHz. Der Chip erfüllt die hohen Datenratenanforderungen durch interne Paket-Puffer zusätzlich zum externen SRAM auf der Netzwerkkarte. Der SRAM ist 2 MB groß (256K x 8B). Er operiert mit der gleichen Taktfrequenz wie der Lanai Z8E und hat eine Speicher-Bandbreite von 2.400 MB/s. Eine Byteparität wird erzeugt und auf allen Spei-

chern innerhalb und außerhalb des NIC-Prozessors überprüft. Der SRAM wird hauptsächlich für die Ausführung der Firmware und für die Pufferung von Paket Overheads genutzt. Die Firmware, die für die PCI-Geräte-Initialisierung und für Etherboot[7] Treiber benötigt wird, wird auf einem 512 KB großen EEPROM auf der Netzwerkkarte gespeichert.

Bild 2.11 zeigt das Schema eines Lanai ZE Prozessors, welches von Myricom entwickelt wurde. Der Lanai Z8E benutzt XAUI als Schnittstelle zu dem XFP Transceiver (siehe 2.1.9). Über das JTAG (Joint Test Action Group) Interface kann der Prozessor auf verschiedene Funktionen getestet und nach Fehlern überprüft werden. Weiterhin zeigt das Bild den Fluss der Daten und des Overheads beim Empfang von Paketen.

Für den Zugriff auf die Netzwerkkarte wird der Gerätetreiber benötigt. Jede Kommunikation zwischen dem Gerätetreiber, der in das Betriebsystem installiert werden muss, geschieht in Form von Anfragen und Antworten. Die Anfragen werden über einen PIO-Mode an die Netzwerkkarte geschickt. Sobald die Befehle bearbeitet worden sind, schickt die Netzwerkkarte einen Statusbefehl über die Komplettierung der Anfrage an den Hauptspeicher, der dann vom Gerätetreiber verarbeitet wird.

Es gibt zwei Arten von Anfragen. Die Erste konfiguriert die Netzwerkkarte mit Befehlen, die dem Gerätetreiber zur Verfügung stehen. Diese Befehle werden bei der Treiber-Initialisierung ausgeführt. Der Treiber kann nur einen dieser Anfragen zur gleichen Zeit bearbeiten. Weitere Konfigurationsbefehle werden blockiert, bis die vorherige Anfrage fertig behandelt wurde.

Die zweite Art sind Anfragen zum Senden und Empfangen von Paketen. Diese werden asynchron von der Netzwerkkarte und von dem Gerätetreiber bearbeitet, um die Leistung zu erhöhen. Sie benutzen Ringe von Deskriptoren. Das ist ein physikalisch zusammenhängendes Feld von Deskriptoren im Speicher der Netzwerkkarte. Deskriptoren enthalten für den Versand Informationen über den Ort und die Größe der Pakete zusammen mit Anweisungen über Operationen, die die Netzwerkkarte an den Paketen durchführen muss. Der Deskriptor für den Empfang von Paketen zeigt den Anfang des Empfangspuffers im Hauptspeicher an.

Für den Empfang stellt der Gerätetreiber für die Myricom-Netzwerkkarte im Hauptspeicher Empfangspuffer zur Verfügung. Der rote durchgezogene Pfeil zeigt den Weg der Nutzdaten eines empfangenen Pakets an. Diese werden über

[7] Etherboot ist ein freies Software Packet für die Erstellung von Boot-ROMs zum Hochfahren von Linux oder anderen Bertriebsystemen über ein Netzwerk, welches Internet Protokolle nutzt.

interne Puffer des Lanai Z8E an die PCI Express Schnittstelle weitergegeben. Über diese Schnittstelle gelangen die Daten mit dem Overhead über einen direkten Speicherzugriff (DMA) in den dafür vorgesehenen Bereich im Hauptspeicher. Der türkise, gestrichelte Pfeil zeigt den Weg des Overheads des empfangenen Pakets an. Die Header-Daten werden in den SRAM kopiert. Die Myricom 10-Gigabit-Netzwerkkarte unterstützt einige Offload-Funktionen wie *checksum offload* (siehe 2.2.3), wobei der Overhead des Pakets verarbeitet und verändert wird. Daher befindet sich die CPU physikalisch in der Nähe des SRAM Interfaces.

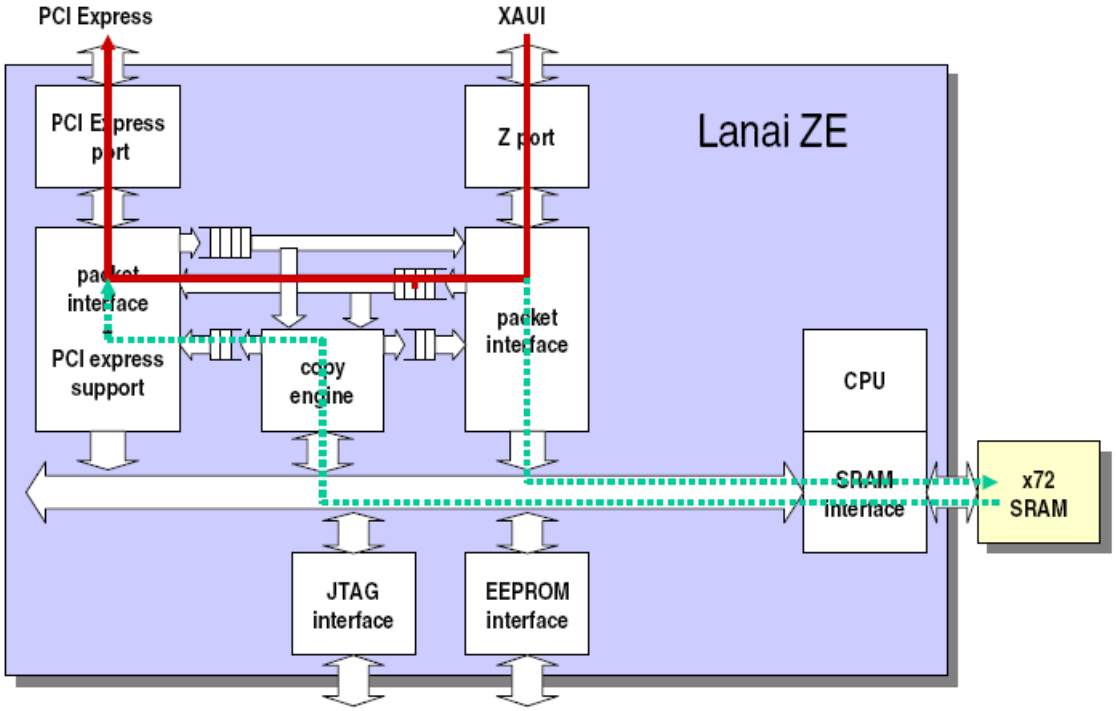

Bild 2.11: Schema eines Myricom Lanai ZE Chips [7]

Nach der Bearbeitung der empfangenen Pakete muss der Gerätetreiber der Netzwerkkarte weiteren Empfangspuffer im Hauptspeicher zuweisen. Es gibt zwei verschiedene Arten von Empfangspuffer.

Der kleine Empfangspuffer beinhaltet immer ein komplettes Paket. Die Größe muss immer ein Vielfaches von 4 sein und darf maximal 4 KB betragen. Jeder Puffer muss komplett in einer 4 KB großen *page* sein. Eine *page* ist ein physikalisch zusammenhängender Teil von Speicherzellen im Hauptspeicher. Wenn ein Paket kleiner als die maximale Größe des kleinen Empfangspuffers ist, speichert die Netzwerkkarte die Pakete in diesem Puffer.

Die Größe des großen Empfangspuffers muss mindestens 4 KB und eine Potenz von 2 sein. Wenn das empfangene Paket größer als der kleine Empfangspuffer umfasst, wird das Paket im großen Empfangspuffer gespeichert. Das Paket

kann hier auf mehrere Puffer aufgeteilt werden, sollte es größer als der große Empfangspuffer sein. Der große Empfangspuffer ist für Pakete mit Jumbo-Rahmen gedacht, während der kleine Empfangspuffer für eine Standard-MTU mit 1520 Bytes ausgelegt ist. Über den Gerätetreiber kann jedoch diese Größe verändert werden.

Für den Versand von Paketen ist die Prozedur ähnlich wie beim Empfang von Paketen. Der Netzwerkkarte wird ein Sendepuffer im Hauptspeicher des Computers zur Verfügung gestellt. Die Pakete, die versendet werden sollen, werden in diesem Bereich gespeichert. Der Gerätetreiber erstellt dann die Deskriptoren für den Versand und kopiert diese in den Speicher der Netzwerkkarte. Der Lanai Z8E untersucht diese Deskriptoren, nimmt die Pakete und verschickt sie.

Ein Paket besteht aus einer Serie von physikalisch zusammenhängenden Hauptspeicherbereichen. Bevor die Pakete verschickt werden, teilt der Gerätetreiber diese Bereiche in Serien von Segmenten und benachrichtigt die Netzwerkkarte von diesen Segmenten. Ein physikalisch zusammenhängender Speicherbereich ist standardmäßig maximal 4 KB groß.

Der Treiber und die Firmware unterstützen *zero-copy* auf der Senderseite für alle Betriebsysteme und je nach Betriebsystem nutzt es eine Reihe von *stateless offloads*. *Stateless offloads* bezeichnet TCP/IP Funktionen, die nicht vom Betriebsystem durchgeführt werden, sondern auf die Netzwerkkarte oder den Treiber „abgeladen“ werden. Bei *stateless offloads* werden der Protokoll Stack Zustände der TCP Verbindungen nicht an die *offload engine* abgegeben. Werden *stateful offloads* wie *TCP Offload Engine* (TOE) genutzt, müssen die Zustände vom Gerätetreiber oder der Netzwerkkarte während den Anforderungen für Datenübertragungen aufrechterhalten werden. TOE erlaubt dem Treiber oder der Netzwerkhardware Teile oder die ganzen Aufgaben des TCP/IP Protokoll zu übernehmen. Es birgt keine erheblichen Leistungssteigerungen gegenüber der Bearbeitung des TCP/IP-Protokoll-Stacks durch das Betriebssystem. Weiterhin werden bei hoher Auslastung, d. h. die gleichzeitige Bearbeitung von vielen Verbindungen, hohe Anforderungen an den Speicher der Netzwerkkarte und dessen Prozessor gestellt, die es meist nicht erfüllen kann.

Die Myricom-10-Gigabit-Ethernet-Netzwerkkarte und dessen Software haben keine *stateful offloads* implementiert. An *stateless offloads* unterstützt es unter anderem folgende:

- Interrupt Coalescing
- UDP und TCP Checksum Offload, Versand und Empfang
- TCP Segmentation Offload (TSO), auch genannt Large Send Offload
- Receive-Side Scaling (RSS)

- Large Receive Offload (LRO)
- Multicast Filtering

2.2.3 UDP und TCP Checksum Offload

Für Pakete, die nicht mit dem TCP Segmentation Offload (TSO) Verfahren behandelt wurden, unterstützt die Netzwerkkarte TCP und UDP *Checksum* (engl. für Prüfsumme) *Offload*. Die Netzwerkkarte kann die TCP und UDP Prüfsummen berechnen und diese korrekt in die Prüfsummen Felder der TCP und UDP Header (siehe 2.3.3 und 2.4.1) platzieren. Dieses Feature ist optional und es wird der Netzwerkkarte über ein Flag von dem Gerätetreiber vermittelt, ob es die Prüfsummenberechnung für einzelne Pakete durchführen soll. Das Betriebssystem soll weiterhin die IP Prüfsummen generieren und es soll pseudo-header Prüfsummen berechnen. Der IP Header muss die korrekte Prüfsumme beinhalten, während der TCP oder UDP Header die pseudo-header Prüfsummen aufweisen muss. Es macht keinen Unterschied für die Berechnung der TCP und UDP Prüfsummen durch die Netzwerkkarte, ob IPv4 oder IPv6 (siehe Kapitel 2.5) benutzt wird, da die IP Prüfsumme vom Betriebsystem generiert werden.

Für TSO Pakete müssen ebenso korrekte IP und pseudo-header Prüfsummen in den entsprechenden Feldern erstellt werden, da die Netzwerkkarte diese als Basis für die finale Berechnung der Prüfsummen der individuellen TSO-unterteilten Pakete generiert.

2.2.4 Interrupt Coalescing

Interrupt Coalescing bedeutet das Zusammenfügen von Interrupts. Es sammelt mehrere Interrupts und löst nach Ablauf eines Interrupt-Zählers oder nach einer bestimmten vorgegebenen Zeit einen einzigen Interrupt aus. Damit soll verhindert werden, dass zu viele Interrupts im Host System ausgelöst werden. Viele Interrupts bedeutet mehr CPU-Zyklen und somit eine höhere Auslastung. Im Allgemeinen muss ein Kompromiss zwischen Latenz und CPU-Leistung gefunden werden. Die Myricom-10-Gigabit-Karte generiert einen Interrupt in einem fixen Intervall. Standardmäßig beträgt dieser 75 µs. Er ist jedoch variabel und die Funktion kann auch ausgeschaltet werden.

Sollten wenige Pakete übertragen werden, erhöht Interrupt Coalescing die Latenz für die Bearbeitung des Pakets. Ein einzelnes Paket, welches an der Netwerkkarte angekommen ist und in den Empfangspuffer übertragen wurde, muss erst das Interrupt-Coalescing-Intervall abwarten bis es weiterverarbeitet wird.

Bei einer hohen Ladung arbeitet das System effizienter, da es weniger CPU-Zyklen für die Bearbeitung der Pakete braucht. Die Host-CPU kann bei einem Interrupt, der ausgelöst wurde, mehrere Pakete bearbeiten.

2.2.5 Zero-Copy

In einem einfachen Aufruf eines Gerätetreibers wie für eine Netzwerkkarte für einen Sendebefehl eines Paketes werden die zwei Standardbefehle „read“ und „write“ ausgeführt. Diese zwei einfachen Befehle verursachen intern einen großen Overhead, indem die zu versendende Datei im Hauptspeicher mehrmals kopiert wird. Die Datei muss vom Kernel-Puffer in den User-Puffer der Anwendung und schließlich wieder in der Kernel-Adressbereich kopiert werden. Der letzte Kopiervorgang verschiebt die Datei in einen vom Netzwerk-Socket speziell genutzten Puffer. Dieses Kopiervorgänge nehmen CPU Ressourcen in Anspruch, die anstatt für komplexere Operationen genutzt werden können.

Zero-Copy beschreibt eine Eigenschaft in Computer Operationen, die diese internen Kopiervorgänge bei einem einfachen Systemaufruf verringert. Linux hat einen Systemaufruf implementiert, der nur noch einen internen Kopiervorgang vom Kernel-Puffer in einen anderen Bereich des Kernelreservierten Speichers, der vom Netzwerk-Socket genutzt wird. Dieser Vorgang wird durch Hardwarefunktionen, wie sie in der Myricom-10-Gigabit-Karte vorhanden sind, überflüssig. Anstatt des Kopierens der Dateien werden Deskriptoren erstellt und der Netzwerkkarte übergeben. Diese Deskriptoren enthalten die Informationen über den Speicherort der Dateien. Nun werden nur noch direkte Speicherzugriffe durchgeführt von I/O-Geräten.

2.2.6 TCP Segmentation Offload

TCP-Segmentation-Offload (TSO) ist eine weitere Technik, um die CPU-Auslastung zu verringern. Bevor große Dateien über ein Computernetzwerk verschickt werden können, müssen diese in kleinere Segmente eingeteilt werden. Dies wird normalerweise vom TCP-Protokoll-Stack des Host-Computers durchgeführt. Das „Abladen“ (engl. *Offloading*) dieser Last an die Netzwerkkarte wird *TCP-Segementation-Offload* genannt.

Die Myricom-10-Gigabit-Netzwerkkarte unterstützt diese Funktion. Vom Betriebsystem erstellte TCP-Pakete, deren Nutzlast größer als die *Maximum Segment Size* (MSS) ist, werden von der Netzwerkkarte in kleinere TCP-Pakete eingeteilt, deren Nutzlast maximal die Größe der MSS besitzt. Die TCP-Header der TSO-segmentierten Pakete tragen die gleichen Kontrolbits wie das Originalpaket. Die Ausnahme sind die PUSH- und FIN-Flags. Diese werden abgesehen vom letzten Segment entfernt.

2.2.7 Receive-Side Scaling

Computer mit mehreren Prozessoren können den Vorteil durch erhöhte CPU-Leistung in Hochgeschwindigkeitsnetzen mit der Architektur des *Network Driver Interface Specification* (NDIS) 5.1 nicht nutzen. In dieser Version werden Empfangs Indikationen von einer einzelnen Netzwerkkarte immer noch von einem Prozessor des Mehrprozessor Systems verarbeitet. Der Interrupt, den ein oder mehrere Pakete beim Empfang auslösen, wird von dem Prozessor behandelt, der der Netzwerkkarte zugewiesen ist. Löst dieser einen Interrupt aus, werden alle weiteren von dem Netzwerkadapter blockiert, bis der Erste fertig bearbeitet wurde.

In der NDIS-Version 6.0 ist die Technik *Receive Side Scaling* (RSS) implementiert, die dieses Problem dahingegen behebt, indem sie die Netzwerklast auf mehrere CPUs verteilt. RSS wurde von der Microsoft Scalable Networking Initiative entwickelt. RSS behält bei der Bearbeitung mehrerer Pakete simultan die Empfangsreihenfolge ein und erlaubt, wenn es der Computer und der Netzwerkadapter unterstützt, mehrere Interrupts.

2.2.8 Large Receive Offload

Large Receive Offload (LRO) assistiert dem Host bei der Bearbeitung von ankommenden Paketen. Es vereinigt während dem Empfang mehrere kleiner Pakete zu einem großen Paket. Diese Funktion ist auf der Hardware der Netzwerkkarte implementiert. Durch die Bearbeitung eines großen Pakets und somit relativ zur Nutzlast weniger Overhead muss die CPU weniger Instruktionen ausführen und spart damit Ressourcen, die für andere Anwendungen genutzt werden können.

Diese Funktion wird durch die Myricom-10-Gigabit-Ethernet-Karte von dem Lanai Z8E und dessen Gerätetreiber durchgeführt. Wie in Kapitel 2.2.2 beschrieben, wird der Paket-Overhead von ankommenden Paketen separat im SRAM der Netzwerkkarte gespeichert. Mithilfe des Overheads können zusammengehörige Pakete identifiziert werden. Diese Pakete werden dann zu einem einzigen übergroßen Paket zusammengefügt. Die LRO-Engine muss außerdem einige Prüfungen durchführen, um sicherzugehen, dass die Pakete zu einem LRO-Rahmen vereinigt werden können. Der Gerätetreiber verbindet die Nutzlasten und erstellt für das LRO-Paket einen einzigen Header.

2.2.9 Multicast Filtering

Multicast-Filtering ist die Eigenschaft Multicast-Ethernet-Rahmen nach ihren Ziel-MAC-Adressen zu filtern. Dabei vergleicht die Netzwerkkarte die Ziel-Adresse mit einer Multicast-Filter-Tabelle. Sollte sich eine Übereinstimmung

finden, überträgt die Netzwerkkarte die empfangenen Pakete in den Hauptspeicher. Ansonsten wird das Paket fallengelassen. Diese Eigenschaft kann durch den Gerätetreiber jederzeit während des Betriebes ein- oder ausgeschaltet werden.

2.2.10 Write Combining

Write Combining ist eine spezielle Speichermethode des Zwischenspeichers. Hierbei werden mehrere Daten, die über den PCI-Express-Bus weiter geschickt werden sollen, in die gleiche Zwischenspeicherzeile geschrieben. Durch diese spezielle Anordnung der Daten können diese in einer einzigen großen Bustransaktion übertragen werden. Diese Methode ist effizienter als die Übertragung von einzelnen Bytes oder Datenwörtern. Dadurch wird die Schreibleistung auf den Hauptspeicher gesteigert.

2.3 Transmission Control Protocol

Das *Transmission Control Protocol* (TCP) ist ein zuverlässiges, verbindungsorientiertes Bytestrom Protokoll der Schicht 4. Seine Hauptaufgabe ist die Bereitstellung einer gesicherten logischen Ende-zu-Ende-Verbindung zum Transport von Daten über ein Netzwerk. Es wurde ursprünglich in der RFC 793 definiert. Fehler und Inkonsistenzen wurden in späteren RFCs behoben und TCP wurde auch um einige Anforderungen ergänzt.

2.3.1 Das TCP/IP Referenzmodell

Das TCP/IP Referenzmodell besteht im Gegensatz zum OSI Referenzmodell nur aus vier Schichten. Jede Schicht hat ihre eigenen Protokolle, die jeweils für einen Aspekt der Kommunikation zuständig ist. Für die Verbindung eines Hosts mit einem anderen wird eine Kombination aus verschiedenen Protokollen auf verschiedenen Schichten verwendet.

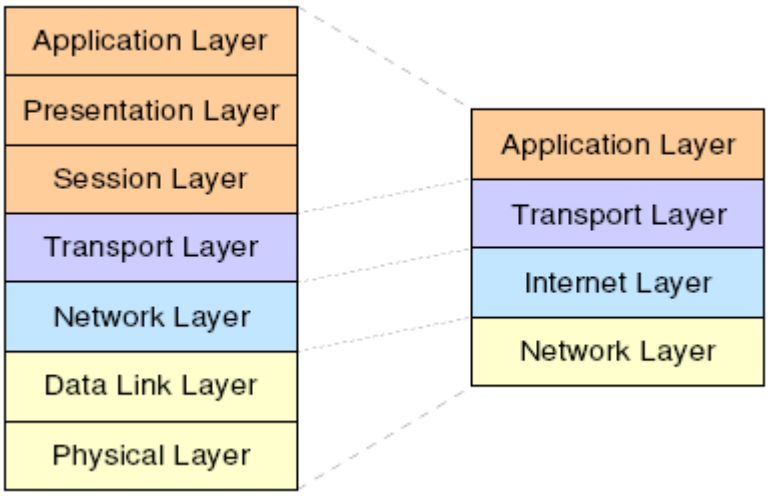

Bild 2.12: Vergleich des OSI Referenzmodells mit dem TCP/IP Referenzmodell [11, Seite 12]

Applikationsschicht (*application layer*): In der Applikationsschicht des TCP/IP-Referenzmodells sind alle anwendungsorientierten Schichten des OSI Referenzmodells zusammengefasst. Hier werden alle Details der unterschiedlichen Anwendungen behandelt. Zu den vielen Protokollen gehören Telnet, das *File Transfer Protocol* (FTP), das *Simple Mail Transfer Protocol* (SMTP) oder das *Hypertext Transfer Protocol* (HTTP).

Transportschicht (*transport layer*): In der Transportschicht wird wie im OSI Referenzmodell der Anwendung eine logische Ende-zu-Ende-Verbindung zur Verfügung gestellt. Es existieren zwei wesentliche Protokolle. Das *Transmission Control Protocol* (TCP) ist ein zuverlässiges verbindungsorientiertes Bytestrom

Protokoll. Das *User Datagram Protocol* (UDP) ist ein unzuverlässiges verbindungsloses Protokoll.

Internetschicht (*internet layer*): In der Internetschicht des TCP/IP-Modells gibt es nur das *Internet Protocol* (IP). Dieses Protokoll überträgt die Pakete über das Netzwerk. In dieser Schicht findet das Routing (Wegfindung) der Pakete statt. Das *Internet Control Message Protocol* (ICMP) dient zur Übertragung von Diagnose- und Fehlerinformationen des *Internet Protocols*. Es ist mit dem *Internet Group Management Protocol* (IGMP) ein fester Bestandteil dieser Schicht.

Netzwerkschicht (*network layer*): In dieser Schicht sind die Hardwarespezifikationen für die Netzwerkkarte und die Gerätetreiber definiert. Es befasst sich außerdem mit allen Details für den Anschluss an das physikalische Medium. Die Daten der oberen Schichten werden hier in eine für die Übertragung geeignete Form gebracht und dann übertragen. Standards in dieser Schicht sind unter anderem Ethernet oder Token Ring.

2.3.2 Services

Die Zuverlässigkeit wird durch verschiedene Mechanismen, die im TCP implementiert sind, gewährleistet. Es wird als *Positive Acknowledgement with Re-Transmission* (PAR) bezeichnet. Wenn TCP Daten von einem Client empfängt, quittiert es diese, indem es eine Bestätigung (ACK) zurücksendet. Die Bestätigung wird nicht sofort verschickt. Der Vorgang wird um eine bestimmte Zeit verzögert, um zu prüfen, ob Daten mit der Bestätigung an den Client geschickt werden können. Ist dies der Fall werden die Daten zusammen mit der Bestätigung für das vorangegangene Paket verschickt. Dadurch wird das Versenden eines mindestens 20 Bytes großen Overheads erspart. Eine Bestätigungsnachricht des TCP besteht nur aus dem 20 Bytes großen Overhead. Sendet TCP ein Paket, löst es einen Timer aus. Wenn in dieser festgelegten Zeit keine Bestätigungsnachricht von der anderen Seite eintrifft, dann wird die Nachricht noch einmal gesendet. Die Übertragung der Daten wird solange wiederholt bis eine Bestätigung eintrifft oder eine Obergrenze für den Timer erreicht wurde.

TCP teilt die Anwendungsdaten in Segmente, wobei TCP eine für die Übertragung sinnvolle Größe festlegt. Die *Maximum Segment Size* (MSS) ist die maximale Größe für die Nutzlast eines TCP Paketes. Die MSS wird beim Aufbau einer Verbindung angegeben. Dabei hat jede Seite die Möglichkeit seine eigene MSS der anderen Seite mitzuteilen. Wenn eine Seite keine MSS angibt, wird diese Standardmäßig auf 536 Bytes gesetzt. Mit dem TCP-Header und dem IP-Header von jeweils 20 Bytes ergibt das insgesamt ein IP-Datagramm von 576 Bytes. Über diesen Wert muss ein Host keine Datagramme annehmen. IP-Datagramme können größere Werte annehmen, jedoch werde diese von der Sicherungsschicht dem Standard entsprechend fragmentiert. Allgemein gilt, je

größer eine MSS, desto besser, jedoch solange wie keine Fragmentierug auftritt. Je größer das Segment ist, desto mehr Daten können mit einem Overhead verschickt werden, was eine geringere Bearbeitung von Paketen durch die Host-CPU bedeutet. Ethernet hat eine *Maximum Transfer Unit* (MTU) von 1500 Bytes (siehe 2.1.2), d. h. dass eine MSS von 1460 Bytes das Optimum für eine TCP Verbindung über Ethernet ist. Bei der Benutzung von *Jumbo Frames* sind es entsprechend 8960 Bytes.

TCP errechnet für jedes Segment eine Prüfsumme, die im Header des zu übertragenden Pakets platziert wird. Die Prüfsumme wird für den Header und die Daten berechnet. Beim Empfang eines Pakets wird dieselbe Berechnung für das TCP-Segment durchgeführt. Sollten die Werte nicht übereinstimmen, war die Übertragung fehlerhaft. Der Empfänger verwirft das Paket und sendet keine Bestätigung. Da der Sender eine Bestätigung erwartet, aber nicht erhält, wird er das Segment erneut schicken.

TCP ist ein verbindungsorientiertes Transportprotokoll, d. h. dass zuerst eine Verbindung aufgebaut werden muss, bevor Daten ausgetauscht werden können. Um diese logische Ende-zu-Ende-Verbindung aufzubauen, wird eine bestimmte Prozedur angewandt, das *Three-Way-Handshake* genannt wird. Für den Verbindungsaufbau sendet die anfordernde Seite, der Client, an den Server ein SYN-Segment. Ein SYN-Segment ist ein TCP-Paket, welches nur aus dem Header besteht mit gesetzter SYN-Flag (siehe 2.3.3). Weiterhin muss das SYN-Segment die Portnummer des Servers, zu dem der Client die Verbindung aufbauen möchte, und die ursprüngliche Sequenznummer angeben. Der Server antwortet dem Client, indem er eine Bestätigungsnachricht zurückschickt. In dieser Nachricht ist ebenfalls die SYN-Flag gesetzt. Zur Bestätigung muss das ACK-Flag gesetzt sein und die ursprüngliche Sequenznummer des Clients plus eins im Bestätigungsnummernfeld der Bestätigungsnachricht des Servers eingegeben sein. Der Server sendet ebenfalls seine ursprüngliche Sequenznummer mit. Der Client muss diese SYN-Nachricht des Servers bestätigen, indem er ein ACK mit der ursprünglichen Sequenznummer des Servers plus eins zurückschickt. Nach drei ausgetauschten Nachrichten ist die logische Ende-zu-Ende-Verbindung aufgebaut. In einer TCP Verbindung kommunizieren genau zwei Punkte miteinander.

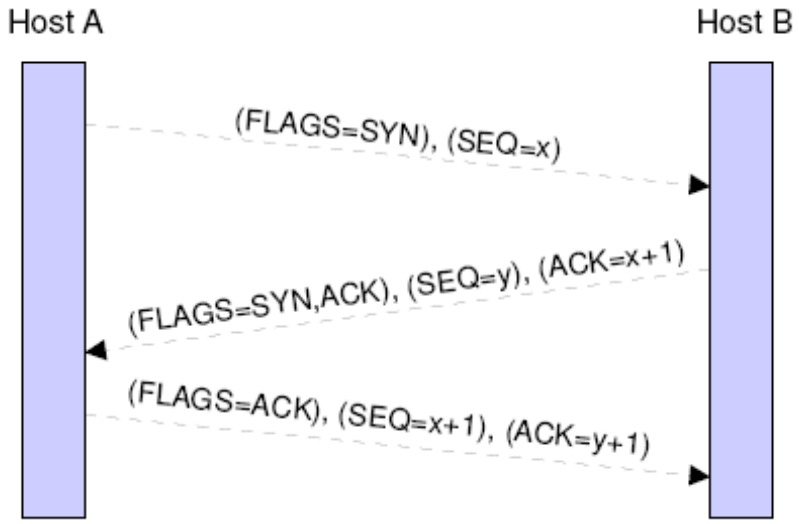

Bild 2.13: Three-Way-Handshake beim Verbindungsaufbau [11, Seite 34]

TCP versendet seine Pakete über IP-Datagramme. Diese Datagramme können in einem Netzwerk über verschiedene Wege geroutet werden, so dass TCP-Pakete in der falschen Reihenfolge eintreffen können. Mithilfe der Sequenznummer kann TCP diese jedoch wieder richtig ordnen. Auf der Empfängerseite werden die Daten bei Bedarf umgestellt. IP-Datagramme können auch doppelt auftreten, daher besitzt TCP einen Mechanismus, welches doppelte Daten verwirft. Außerdem wird eine Flusssteuerung von TCP zur Verfügung gestellt. Diese sorgt dafür, dass der Sender mit dem Senden der Pakete aufhört, so lange der Empfänger nicht genügend Puffer bereit hat. „Damit wird vermieden, dass ein schneller Host sämtliche Puffer eines langsamen Hosts belegt“ [29, Seite 298].

Die Daten, die über eine TCP-Verbindung geschickt werden, haben für TCP keine Struktur. Es gibt keine Kennzeichnung eines Datensatzendes. TCP sieht einen 8-Bit-Bytestrom von der Applikation seines Hosts und schickt diese segmentiert an die andere Seite. Die Gegenseite fügt die Segmente zusammen und sieht ebenfalls nur einen Bytestrom.

2.3.3 TCP-Header

Die TCP Pakete, die über das Netzwerk geschickt werden, werden mit einem TCP Header versehen. Dieser ist mindestens 20 Bytes groß. Dieser Header besitzt ein Feld für Optionen, der den Header vergrößern kann.

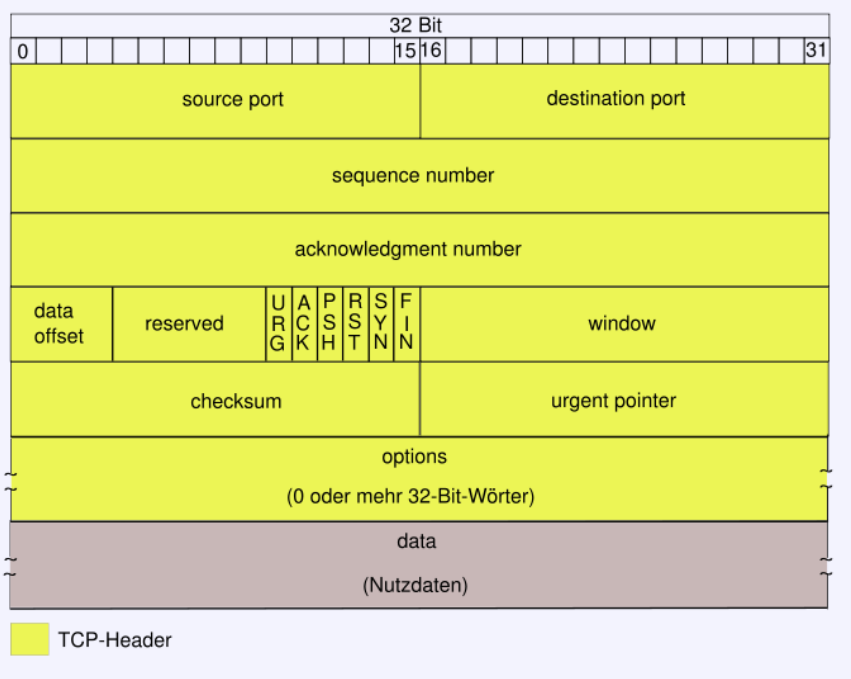

Bild 2.14: TCP Header [37]

Jedem TCP-Segment wird der *source port* (Quell-Portnummer) und der *destination port* (Ziel-Quellnummer) angehängt. Diese identifizieren die Applikationen auf beiden Seiten der Verbindung. Zusammen mit den Quell- und Ziel-IP-Adressen wird die TCP-Verbindung im Internet eindeutig erkannt.

Die *sequence number* (Sequenznummer) gibt die Position jedes einzelnen Bytes im Datenstrom an, der über die TCP-Verbindung übertragen wird. TCP vergibt für jedes Byte eine Sequenznummer. Diese ist ein 32 Bit Wert ohne Vorzeichen, der auf 0 zurückgesetzt wird, sobald er 2^{32}-1 erreicht hat. Mit der Sequenznummer wird die richtige Reihenfolge der TCP-Pakete bestimmt, falls diese nicht in der richtigen Ordnung bei dem Empfänger ankommen. Die 32-Bit breite *acknowledgment number* (Bestätigungsnummer) zeigt den Wert des nächsten Bytes an, den es vom Sender erwartet. Das ist der Wert der Sequenznummer des letzten korrekt empfangenen Datenbytes plus eins. Die ACK-Flag muss aktiviert sein, damit es eine gültige Bestätigungsnachricht ist. Im Sequenznummernfeld wird auch die ursprüngliche Sequenznummer übertragen. Zusammen mit dem gesetzten SYN-Flag dient es zum Verbindungsaufbau. Diese Sequenznummer wird von beiden Seiten unabhängig von einander generiert und je-

weils an die andere Seite geschickt und quittiert. Da jede Seite einer Verbindung eine eigene Sequenznummer für die Daten führt, stellt TCP einen Vollduplex-Service für die Applikationsschicht bereit.

Das *data offset*-Feld oder auch die Header-Länge gibt die Länge des TCP-Headers in 32 Bit Wörter an. Der TCP-Header ist standardmäßig 20 Bytes lang. Also hat das *data offset*-Feld normalerweise einen Wert von 5. Die Länge ist durch das *options*-Feld variabel.

TCP besitzt 6 Flags, die einzeln oder gleichzeitig eingeschaltet sein können. Sie führen bestimmte Aktionen im TCP Protokoll aus:

- **URG:** Hiermit wird das *urgent pointer*-Feld aktiviert. Ist der Wert 0 wird, das Feld ignoriert.
- **ACK:** Hiermit wird das *acknowledgment number*-Feld aktiviert. Es wird dann ein empfangenes TCP-Paket bestätigt, indem es das Feld bestätigt. Ist der Wert 0 wird, das Feld ignoriert.
- **PSH:** Ist dieses Bit gesetzt, sollen die Daten soll schnell wie möglich an die adressierte Anwendung weitergegeben werden.
- **RST:** Hiermit wird eine Verbindung zurückgesetzt. Das geschieht im Allgemeinen, wenn ein TCP-Paket eintrifft, das für die Verbindung nicht richtig erscheint. Resets werden auch ausgeführt, wenn eine Verbindungsanforderung an einen nicht existenten Port geht oder ein Host abgestürzt ist und eine TCP Verbindung somit nicht ordentlich beendet wurde.
- **SYN:** Dieses Flag soll die Sequenznummern vom Sender und vom Empfänger synchronisieren, damit eine Verbindung aufgebaut werden kann.
- **FIN:** Hiermit wird von einer Seite die Verbindung geschlossen. Das bedeutet, dass der Sender dieses Flags keine weiteren Daten mehr zu senden hat. Dabei kann aber die andere Seite weiter Daten schicken. Die Versendung eines Segmentes mit aktivierten FIN-Flag muss bestätigt werden, d. h. dass eine ordentliche Trennung einer Verbindung im Gegensatz zum Verbindungsaufbau aus 4 Nachrichten besteht.

Im *window*-Feld wird die Fenstergröße angegeben. Damit kann ein Host dem anderen Host mitteilen, wie viele Bytes er empfangen kann. Das Fenster beginnt mit dem ersten Byte, welches durch die Bestätigungsnummer der letzten empfangenen Nachricht angegeben ist. TCP realisiert damit eine Flusssteuerung. Ein Sender darf die Anzahl Bytes schicken, die durch die Fenstergröße der letzten empfangenen Nachricht angegeben wurde. Die Fenstergröße kann in jeder Nachricht, die von beiden Seiten versendet wird, verändert werden. Kann ein Host vorerst keine weiteren Daten empfangen, schickt es mit einem

Segment eine Fenstergröße mit dem Wert 0. Damit weiß die andere Seite, dass sie solange keine Pakete schicken soll, bis sie wieder die Erlaubnis erteilt bekommt. Das *window*-Feld ist ein 16 Bit breites Feld, wodurch die Fenstergröße maximal 65535 Bytes groß ist.

Im *checksum*-Feld steht eine Prüfsumme, die über den TCP-Header, die TCP-Daten und einem 12-Byte-Pseudo-Header berechnet wird. Die 16 Bit Wörter aus dem TCP-Segment mit Header und dem Pseudo-Header werden im 1er-Komplement addiert. Deren Summe wird in das *checksum*-Feld eingetragen. Der Wert des Prüfsummenfelds wird bei der Berechnung auf 0 gesetzt. Da das TCP Segment wegen der variablen Dateigröße eine ungerade Anzahl von Bytes haben kann, wird bei Bedarf ein 0 Byte angehängt. Der Empfänger führt den gleichen Algorithmus über das TCP-Segment mit dem Pseudo-Header aus. Es kann dann mit der Prüfsumme, die sich im Header des empfangenen Pakets befindet, überprüfen, ob das Paket ordnungsgemäß übertragen wurde. Der Pseudo-Header enthält die Quell- und die Ziel-Adresse, eine reserviertes Feld mit 8 Nullen, einem Protokollnummernfeld, wo das übergeordnete Protokoll angegeben wird und die Länge des Pakets. Der Pseudo-Header wird bei der Berechnung vor den Header gesetzt. Es dient nur zur Berechnung der Prüfsumme und wird nicht mitgeschickt. TCP kann mit dem Pseudo-Header bestimmen, ob das Paket richtig vermittelt wurde und ob es Differenzen bei der Protokollnummer und der Segmentlänge gibt.

Das *urgent pointer*-Feld ist nur aktiv, wenn das URG-Flag gesetzt ist. Der Wert ist ein positiver Offset-Wert, d. h. dass dieser Wert zu der Sequenznummer addiert wird. Somit weiß TCP, dass dieses Byte das letzte Byte ist, welches dringend bearbeitet werden sollte. Mit diesem Modus können dringende Daten übertragen werden.

Das *options*-Feld ermöglicht es Funktionen bereitzustellen, die durch den normalen Header nicht vorgesehen sind. Das gängigste Optionsfeld ist die *Maximum Segment Size* (MSS). Diese kann beim Verbindungsaufbau durch beide Seiten angegeben werden.

Eine weitere Option ist die timestamp-Option (Zeitstempel). Hierbei kann ein Host einen Zeitstempelwert in einem Segment speichern und übertragen. Der Empfänger spiegelt diesen Wert in der Bestätigung wieder, damit kann die *Round Trip Time* (RTT) besser bestimmt werden.

2.4 User Datagram Protocol (UDP)

Das *User Datagram Protocol* (UDP) ist ein minimales, unzuverlässiges und verbindungsloses Protokoll der Schicht 4. Spezifiziert wurde es in der RFC 768. Dieses Protokoll verfügt nur über minimale Anforderungen an Mechanismen, die ein Netzwerkprotokoll besitzen muss. Außer der Multiplex-/Demultiplex-Funktion für die IP-Datagramme und den Anwendungen, die auf die UDP- und IP-Dienste zugreifen, und einer Prüfsummenberechnung, besitzt UDP keine weiteren Funktionen. Es greift direkt auf die Daten der Anwendungen zu, die mit den Portnummer eindeutig identifiziert werden können, fügt diesen einen 8 Bytes großen Overhead an und gibt sie direkt an die Internetschicht mit dem IP-Protokoll weiter. Dort wird das UDP-Paket in ein IP-Datagramm verkapselt. Jeder Sendeaufruf einer Anwendung löst genau ein UDP-Datagramm aus, was wiederum zu genau einem IP-Datagramm führt. TCP, ein Bytestrom-Service, besitzt hingegen Mechanismen, die Datenströme teilen oder zusammenfügen können. Auf der Empfangsseite werden mithilfe der Portnummern die Daten aus dem UDP-Datagramm direkt an die Anwendungen weitergeleitet. Die Applikation ist für die Größe des UDP-Datagramms und somit auch für die Größe des IP-Datagramms verantwortlich. Theoretisch können IP-Datagramme eine Größe von 65535 Bytes erreichen, da das Längenfeld des IP-Headers eine 16 Bit Zahl ist. Ist das IP-Datagramm größer als die MTU des Netzwerkes, wird es fragmentiert.

UDP ist ein unzuverlässiges Protokoll. Es besitzt keinerlei Funktionen, die die sichere Übertragung eines Pakets von einem Ende einer Datenübertragungsstrecke zu einem anderen Ende garantiert. Ein Host sendet die UDP-Pakete, ohne zu wissen, ob sie bei dem Empfänger ankommen. Weiterhin kann nicht geprüft werden, ob die Datagramme in der richtigen Reihenfolge empfangen werden. Protokolle der höheren Schichten müssen sich um die korrekte Zustellung und die Sortierung der Pakete kümmern.

Für die Zustellung von Paketen muss keine logische Ende-zu-Ende-Verbindung aufgebaut werden. Im Gegensatz zu TCP, bei dem ein *Three-Way-Handshake* für den Verbindungsaufbau genutzt wird, kann UDP die Daten einfach versenden. Somit kann der Datenaustausch schneller begonnen werden, da keine Verzögerung für den Verbindungsaufbau auftritt.

Da UDP keine Verbindung aufbaut, speichert es auch keine Zustände in den Endsystemen. TCP nutzt unter anderem Zustände für die Sende- und Empfangspuffer, für die Parameter zur Überlastungsvermeidung und der Sequenz- und Bestätigungs-nummern. Diese sind vor allem für die zuverlässige Zustellung der Pakete und der Kontrolle der Überlastungsvermeidung erforderlich.

UDP nutzt diese Mechanismen nicht, was weniger Rechenleistung in Anspruch nimmt. Diese kann für andere Anwendungen genutzt werden oder es können, vergleichsweise zu dem TCP-Protokoll, mehr Anwendungen und Prozesse gestartet werden, die das UDP-Protokoll verwenden. Die Bearbeitung des kleineren Headers von 8 Bytes spart weiterhin Rechenleistung.

Die Kontrolle zur Überlastungsvermeidung stellt beim Sender die Geschwindigkeit für das Versenden der Pakete ein, sobald eine oder mehrere Verbindungen überlastet sind. Diese Drosselung hat negative Auswirkungen bei Echtzeitanwendungen zur Folge, die eine Mindest-Datenrate benötigen. Die Geschwindigkeit hingegen, mit der UDP Daten versendet, ist nur von der Rate, mit der die Anwendungen die Daten generieren, von der Kapazität des Hostsystems und von dem physikalischen Anschluss an das Netzwerk abhängig. Auf der anderen Seite erreichen UDP-Datagramme nicht ihr Ziel, wenn die Verbindung überlastet ist, oder werden verworfen, falls nicht genügend Puffer zur Verfügung steht [vgl. 20].

2.4.1 UDP-Header

Ein UDP-Paket besteht aus den UDP-Daten und aus dem 8 Bytes großen UDP-Header.

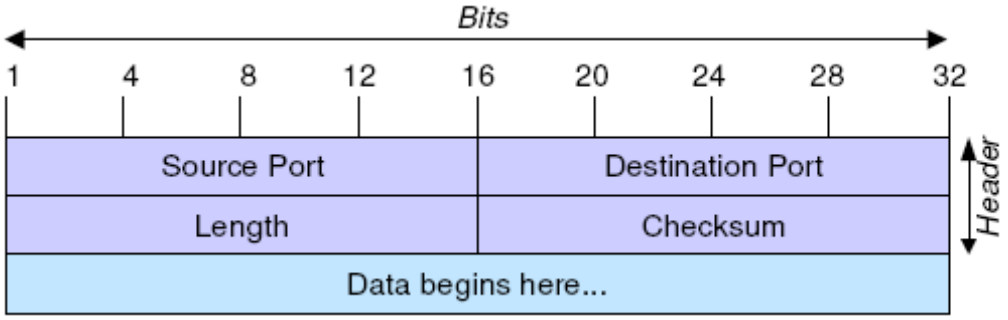

Bild 2.15: UDP-Header [11, Seite 40]

UDP benutzt genauso wie TCP Portnummern, um die Nutzdaten an die richtigen Prozesse in den Endsystemen weiterzuleiten. Das *Length*-Feld beinhaltet die komplette Länge des UDP-Pakets (UDP-Header und UDP-Daten). Im *Checksum*-Feld wird eine Prüfsumme über das komplette UDP-Paket und einem Pseudo-Header eingetragen. Der Algorithmus zur Berechnung der Prüfsumme ist der gleiche wie bei TCP (siehe 2.3.3). Die Prüfsumme ist bei UDP optional. Wird keine berechnet, wird eine 0 in dieses Feld eingetragen.

2.5 Internet Protocol (IP)

Das *Internet Protocol* stellt die Basisdienste für die Übermittlung von Daten in der TCP/IP-Protokollsuite bereit und ist in der RFC 791 spezifiziert. Es bemüht sich den besten Weg für die TCP- oder UDP-Pakete zu deren Zieladresse zu finden. Dabei kann IP nicht garantieren, dass die Pakete ihr Ziel erreichen. IP ist ein unzuverlässiges verbindungsloses Protokoll. Es verfügt über keine Mechanismen für die korrekte Zustellung von Datagrammen. Es beinhaltet einen einfachen Fehlerbehandlungs-mechanismus, der, wenn das Paket wegen nicht ausreichender Pufferkapazität verworfen werden muss, eine *Internet Control Message Protocol*-(ICMP)-Nachricht an den Sender zurückschickt. Dieses Protokoll nutzt auch das IP, um Informationen zu versenden. Deren Auswertung müssen aber Protokolle höherer Schichten übernehmen, wie z. B. TCP.

IP baut keine logische Ende-zu-Ende-Verbindung vor dem Datenaustausch auf. Der Empfang von IP-Datagrammen wird dem Sender nicht bestätigt und jedes einzelne Paket wird unabhängig voneinander behandelt. Mehrere Pakete, die von einem Host A an einen Host B gesendet werden, müssen nicht die gleiche Route durch das Netzwerk haben. Daher kann es auch sein, dass Pakete in der falschen Reihenfolge beim Empfänger ankommen. IP besitzt keine Funktion, die Pakete zu ordnen. Dies muss ebenfalls von Protokollen höherer Schichten bewältigt werden.

„Die Funktionen von IP umfassen:

- Die Definition von Datengrammen, welche die Basiseinheiten für die Übermittlung von Daten im Internet bilden.
- Definition des Adressierungsschemas.
- Übermittlung der Daten von der Transportebene zur Netwerkschicht.
- Routing von Datengrammen durch das Netz.
- Fragmentierung und Zusammensetzen von Datengrammen“ [11, Seite 17].

2.5.1 IP-Header

Der IP-Header ist mindestens 20 Bytes groß. Es besitzt ein *Options*-Feld, welches den Header vergrößern kann. Im Header stehen alle nötigen Informationen, um das Paket an den richtigen Zielhost im Netzwerk zu übermitteln und bei einem fragmentierten Paket das Originalpaket wieder zusammenstellen zu können.

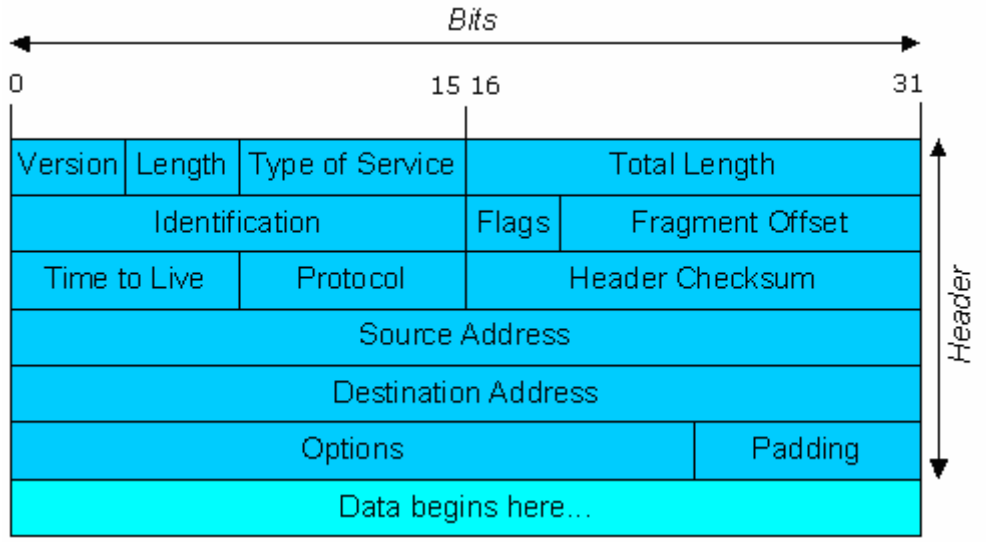

Bild 2.16: IP-Header [11, Seite 17]

Im *Version*-Feld steht die Version des *Internet Protocols*. Die ältere und noch gängigste Version ist IPv4. Dieser IP-Header und die Erklärungen in diesem Kapitel beziehen sich auf IPv4. Es wurde bereits eine neuere Version des IPs eingeführt. Der Grund dafür ist, dass die 2^{32} Adressen der IPv4-Adressen bald ausgeschöpft sind. Es werden immer mehr Endgeräte an das weltweite Netz angeschlossen, von denen jedes ein eindeutige Adresse benötigt. Daher wurde IPv6 eingeführt. IPv6-Adressen sind 128 Bit lang.

Das *Length*-Feld beinhaltet die Länge des Headers. Dieser ist wegen des *Options*-Felds variabel. Es werden 32-Bit-Wörter gezählt, d. h. der Standardwert ist 5 bei 20 Bytes Headergröße. Das *Length*-Feld ist 4 Bit breit und somit ist der Header auf 60 Bytes begrenzt.

Das 8 Bit breite *Type of Service*-Feld umfasst ein 3 Bit breites Präzedenzfeld. Es gibt die Priorität an, wenngleich dieses Feld in der Praxis ignoriert wird. Darauf folgen 4 Flags. Sie geben Aufschluss über besondere Behandlungskriterien der Nachrichten. Sie stehen für: Verzögerung minimieren, Durchsatz maximieren, Zuverlässigkeit maximieren und Kosten minimieren. Das letzte Bit muss immer 0 sein.

Im *Total Length*-Feld steht die komplette Länge des Pakets, d. h. es umfasst die Daten und den Header. Mit dem *Length*-Feld kann bestimmt werden, wo die Nutzdaten beginnen. Das *Total Length*-Feld ist 16 Bit breit. Das IP-Datagramm kann somit eine Länge von 655535 Bytes haben. Die Größe des Pakets, welches über das physikalische Medium transportiert wird, ist aber von der MTU abhängig. Daher fragmentieren IP oder IP-Router IP-Datagramme. Das *Total Length*-Feld ist obligatorisch, da manche Netzwerktypen eine Mindestrahmengröße besitzen. Bei Bedarf müssen die Rahmen auf die Mindestgröße erweitert werden. Die Mindestgröße eines Ethernet-Rahmens beträgt 46 Bytes.

Mithilfe des *Identification*-Felds können Fragmente von einem IP-Datagramm wieder eindeutig zusammen gesetzt werden. Jedes Paket, das von einem Sender verschickt wird, hat eine andere Identifikationsnummer. Muss es fragmentiert werden, erhalten alle Fragmente die gleiche Nummer. Im *Flags*-Feld stehen zwei Flags für die Konfiguration von IP-Fragmenten. Das Feld ist 3 Bit breit, wobei das erste reserviert ist und immer den Wert 0 besitzt. Das *„Don't Fragment"*-(DF)-Flag kennzeichnet, dass das IP-Datagramm nicht fragmentiert werden darf, auch wenn es dann verworfen werden muss. In diesem Fall wird dem Absender eine ICMP-Fehlermeldung geschickt. Das *„More Fragments"*-(MF)-Flag zeigt an, dass diesem Fragment noch andere Fragmente folgen. Dieses Bit ist in jedem IP-Header eines Fragments bis auf das Letzte gesetzt.

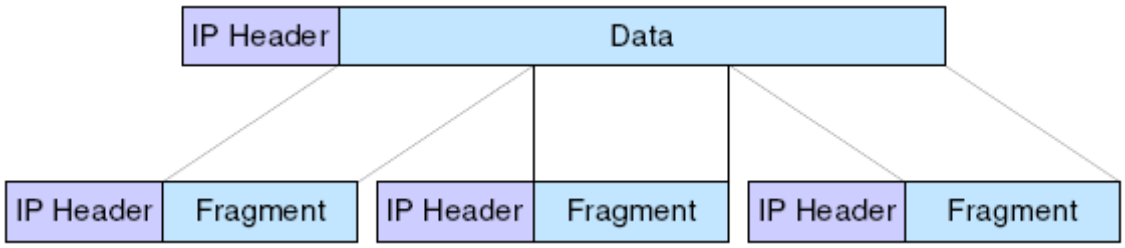

Bild 2.17: Fragmentierung eines IP-Pakets [11, Seite 30]

Wie in Bild 2.17 zu sehen ist, wird jedem Fragment, welches entweder durch den Absender oder durch einen zwischen Quelle und Ziel liegenden Router erzeugt wurde, ein eigener IP-Header angefügt. In diesen wird in das *Identification*-Feld die Nummer des Original IP-Datagramms kopiert, die MF-Flag gesetzt und der Wert für das *Fragment Offset*-Feld berechnet. Dieser Wert gibt die Position des Fragments in dem Original IP-Datagramm an. Es ist ein Offset-Wert in 8-Byte-Einheiten vom Anfang des ursprünglichen Datagramms. „Alle Fragmente, außer dem letzten, müssen ein Vielfaches von 8 Byte sein." [11, Seite 19] Das *Fragment Offset*-Feld ist 13 Bit breit und somit können maximal 8192 Fragmente erstellt werden. Das *Total Length*-Feld in den Headern der Fragmente wird ebenso geändert. Mit den Information der *Identification-, Flags-, Fragment Offset-*

und *Total Length*-Felder kann der Zielhost die IP-Fragmente wieder zum ursprünglichen IP-Datagramm zusammensetzen.

Das *Time to Live* (TTL)-Feld begrenzt die Lebensdauer eines Pakets im Netzwerk. Das Paket soll nicht endlos im Netzwerk übermittelt werden, falls es Probleme bei der Auffindung des Ziels gibt. Das Feld ist 8 Bit breit und kann somit einen Maximalwert von 256 annehmen. Dieses Feld wird vom Absender gesetzt und bei jedem Router, den es zwischen Quelle und Ziel durchläuft, dekrementiert. Erreicht das Feld den Wert 0, wird es verworfen und eine ICMP-Fehermeldung wird an den Absender geschickt.

Im *Protocol*-Feld wird das Protokoll der Transportschicht angegeben. Damit kann das IP seine Nutzdaten an das richtige höhere Protokoll weiterleiten. Die Protokollnummern sind im gesamten Internet einheitlich und werden von der *Internet Assigned Numbers Authority* (IANA) definiert.

Im *Header Checksum*-Feld wird die Prüfsumme für den IP-Header eingetragen. Die Prüfsumme wird nur über den IP-Header berechnet. TCP und UDP berechnen die Prüfsumme über das gesamte Paket. Der Algorithmus zur Berechnung der Prüfsumme ist der gleiche wie bei TCP und UDP. Die Prüfsumme muss an jeden Netzknoten neuberechnet werden, da sich dort der Header wegen des TTL-Felds verändert.

Die *Source* und *Destination Address* sind die 32-Bit-Internet-Adressen des Senders und des Empfängers. Jeder Host und jeder Netzknoten muss eine weltweit eindeutige IP-Adresse besitzen. Die Vergabe der IP-Adressen wird zentral organisiert, um Adresskonflikte zu vermeiden. Diese Aufgabe hat die Internet Network Information Center, genannt InterNIC, übernommen.

Das letzte Feld, das *Options*-Feld, soll optionale Informationen übertragen. Es ist von variabler Länge, muss aber bei Bedarf durch Füllbytes auf ein Vielfaches von 32 Bit erweitert werden. Die Optionen werden nicht von allen Hosts und Routern unterstützt. Es gibt folgende Optionen:

- Eine *Security* Option gibt den Geheimhaltungsgrad des IP-Datagramms an. Es wird hauptsächlich von militärischen Anwendern genutzt.
- Mit der *Record-Route*-Option wird der Weg des IP-Datagramms durch das Netzwerk gespeichert. Dabei wird die IP-Adresse aller Netzknoten im *Options*-Feld gespeichert. Da der Header nur 60 Bytes groß werden kann, ist die Speicherung der Route begrenzt.
- Mit der Zeitstempel-Funktion wird zu der IP-Adresse noch die Uhrzeit angegeben, an der das Datagramm den Netzknoten passiert hat.

- Das *Loose-Source-Routing* gibt eine Liste von IP-Adressen an, die vom IP-Datagramm durchlaufen werden sollen. Er muss diese Netzknoten passieren, kann aber dazwischen andere Wege nehmen.
- Beim *Strict-Source-Routing* muss genau der Weg benutzt werden, der in der Liste angegeben ist.

2.5.2 Internet Control Message Protocol (ICMP)

Das *Internet Control Message Protocol* ist ein fester Bestandteil des Internet Protocols. Es dient zur Übertragung von Fehler- und Diagnoseinformationen. ICMP ist in RFC 792 spezifiziert. Bild 2.18 zeigt eine komplette ICMP-Nachricht. Der Header ist immer 32 Bits lang. Das *Type* und *Code* Feld geben den Nachrichtentyp an. Das *Checksum* Feld enthält die Prüfsumme der kompletten Nachricht. Der Algorithmus ist der gleiche wie bei der Berechnung der IP-Prüfsumme.

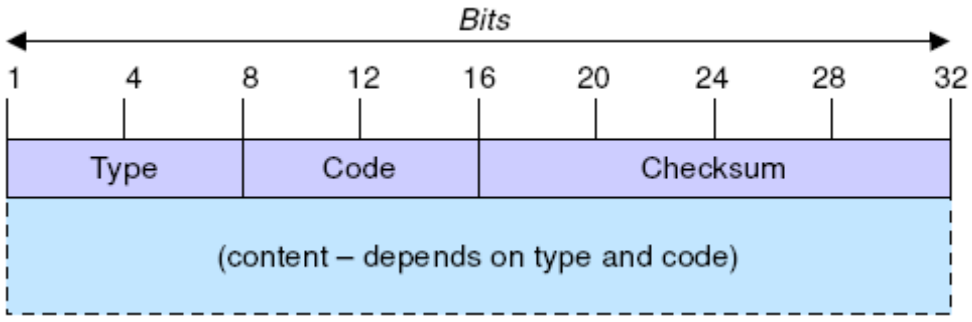

Bild 2.18: ICMP-Nachricht [11, Seite 31]

IP nutzt ICMP, um Fehler- und Diagnoseinformationen zu versenden, während ICMP IP nutzt, um die Nachricht zu übertragen. Dabei wird eine ICMP-Nachricht in ein IP-Datagramm verkapselt. IP oder Protokolle höherer Schichten können diese Meldungen bearbeiten.

Eine ICMP-Nachricht ist immer eine Antwort auf ein IP-Datagramm, das auf ein Problem gestoßen ist oder eine ICMP-Anfrage enthält. Es kann nicht auf ICMP-, Broadcast- oder Multicast-Nachrichten antworten [29, vgl. Seite 103 ff.].

3 Die 10-Gbit/s-Übertragungsstrecke

Das FiLa 10G Board soll große Mengen digitalisierter Daten aus dem digitalen Empfänger des Max-Planck-Instituts für Radioastronomie an ein dafür vorgesehenes Backend senden. In erster Linie sollen dabei die Daten aus einem Analog/Digitalwandler der ersten Generation, der mit einer Taktrate von 2^{30} Hz arbeitet und eine Auflösung von 8 Bit besitzt, verarbeitet werden. Das bedeutet eine Datenmenge von 8 GBit pro Sekunde. Der Standard IEEE 802.3ae für 10-Gigabit-Ethernet wurde im August 2002 veröffentlicht. Seitdem sind die Entwicklung der Technologie und die Markteinführung nur langsam vorangeschritten. Gründe dafür sind die hohen Kosten pro 1-Gigabit Bandbreite bei der Hardware. Weiterhin gibt es auch wenige wissenschaftliche Arbeiten zu dieser Technologie. In diesem Kapitel werden der Aufbau und die Untersuchung einer 10-Gigabit-Ethernet Punkt-zu-Punkt-Übertragungsstrecke beschrieben.

Der Zweck der Untersuchungen war das Sammeln von Erfahrungen mit der Technologie, um deren Potenzial bestmöglich ausnutzen zu können. Mit dem A/D-Wandler der ersten Generation wird bereits ein Datenstrom von 8 GBit/s erzeugt. Die zweite Generation wird eine Taktrate von 2,2 GHz und eine Auflösung von 10 Bit besitzen, was maximal einen Datenstrom von 22 GBit/s bedeutet. Es soll untersucht werden, welche reellen Datenraten mit der 10-Gigabit-Ethernet-Technologie erreicht werden können, um zu ermitteln, wie viele Daten von den A/D-Wandlern an ein Backend übertragen werden können. Die Erfahrungen, die mit der Testübertragungsstrecke gesammelt werden, sollen bei der Entwicklung und Einstellung des AD-Board der zweiten Generation und des FiLa 10G Boards helfen.

3.1 Auswahl der Hardware und Versuchsaufbau

Für die Testübertragungsstrecke wurden die Bauteile für zwei Endsysteme eingekauft und zusammengebaut. Die wesentlichen Bauteile für Übertragungsstrecke sind die Netzwerkkarten und deren optische Transceiver. Es wurde die Myri-10G-Netzwerkkarte der Firma Myricom gewählt. Myricom ist ein mittelständisches Unternehmen aus Kalifornien, USA, welches sich auf die Vernetzung von Hochleistungsrechnern zu einem Supercomputer spezifiziert hat. Seit 2006 führen sie 10-Gigabit-Ethernet-Netzwerkkarten, die über eine PCI-Express Schnittstelle verfügt(siehe 2.2.2). Die Myri-10G-Netzwerkkarte wurde hauptsächlich wegen ihres guten Preis/Leistungs-Verhältnisses ausgesucht. Außerdem bietet die Firma einen gute Produktunterstützung.

Der optische Transceiver muss separat erworben werden. Die Myri-10G-Netzwerkkarte besitzt einen R-Port (siehe 2.1.2) für einen XFP-Transceiver. Somit muss ein LAN-XFP-Transceiver mit 64/66B-Kodierung verwendet werden. Es gibt je nach gewünschter Länge der Übertragungsstrecke drei verschiedene Transceiver. Die physikalischen Schnittstellen wurden von dem Standard IEEE 802.3ae definiert.

Tabelle 3.1: XFP-Transceiver

Standard	Wellenlänge	Medium	Reichweite	Kosten
10GBASE-SR	850 nm	MMF	300 m	ca. 500-1400 €
10GBASE-LR	1310 nm	SMF	15 km	ca. 1900-2500 €
10GBASE-ER	1550 nm	SMF	40 km	ab 6000 €

Tabelle 3.1 zeigt die drei möglichen Transceiver mit den Entfernungen und den Kosten. Für die Anwendung des Max-Planck-Instituts gilt es, eine Entfernung von 350 m zu überbrücken. Das entspricht der Strecke eines Lichtwellenleiters am Empfänger des Radioteleskops in Effelsberg bis zum Kontrollraum. Aus Kostengründen wurden für die Testübertragungsstrecke zwei XFP-Transceiver nach dem 10GBase-SR-Standard erworben. Ein 350 m langes Glasfaserkabel kostet ca. 650 € und muss speziell angefertigt werden. In erster Linie sollte überprüft werden, ob die theoretischen Datenraten erreicht werden können. Daher reichten in einem ersten Test diese XFP-Transceiver aus. Später soll ein 350 m langes Glasfaserkabel genutzt werden. Ziel ist es, zu erfahren, wie viel Datenrate man mit einem Lichtwellenleiter (LWL), die 16,6% über der empfohlenen maximalen Reichweite liegt, noch erreicht werden kann.

Für die Testübertragungsstrecke im Labor des Max-Planck-Instituts für Radioastronomie wurde ein Glasfaserkabel von 10 m benutzt. Es gibt drei verschiedene Arten Laser optimierter Multimode-Fasern. Es wurde eine OM3-Faser verwendet.

Tabelle 3.2: Laseroptimierte MMF [8]

	OM3 Faser	OM2 Faser	OM1 Faser
Kern-Durchmesser	50 µm	50 µm	62,5 µm
Faser-Durchmesser	100 µm	100 µm	100 µm
Optimierte Datenrate bei 850 nm	10 GBit/s über 300 m	10 GBit/s über 100 m	
	1 GBit/s über 1000 m	1 GBit/s über 600 m	1 GBit/s über 300 m

Bei der Wahl des XFP-Transceivers waren ebenfalls die Kosten maßgebend. Es wurde ein Intel TXN18107 10 Gbps XFP-Transceiver ausgewählt [17]. Während der Großteil der 10GBase-SR-Transceiver über 1100€ gekostet haben, hat das Produkt von Intel einen Preis von ca. 500€. Der Test mit diesen Transceivern über eine Länge von 350 m wurde noch nicht durchgeführt. Sollte nicht die gewünschte Datenrate erreicht werden, werden diese Transceiver an Herrn Gino Tuccari vom *Instituto di Radioastronomia* auf Sizilien übergeben. Das Radioteleskop dort hat nur einen Durchmesser von 32 m. Dementsprechend ist die Entfernung von Empfänger bis zum Kontrollraum geringer. Daher können hier die erworbenen Transceiver benutzt werden.

Die Firma Myricom testete mit verschiedenen Motherboards, Chipsätzen und Prozessoren ihre 10-Gigabit-Ethernet-Karte. Mit dem Tyan Thunder K8WE S2895 wurden zum Zeitpunkt des Einkaufs die besten Ergebnisse erzielt. Im Max-Planck-Institut sollten zwei ähnliche Systeme aufgebaut werden, um die Datenraten miteinander vergleichen zu können (siehe www.myricom.com). Daher wurden zwei dieser Motherboards erworben (siehe www.tyan.de). Diese bieten jeweils zwei CPU Sockets für AMD Opteron Prozessoren (siehe www.amd.de), 8 Steckplätze für DDR-Arbeitsspeicher und zwei PCI-Express-Schnittstellen. Die Technologie für die Verbindung der PCI-Express-Schnittstelle über den Chipsatz und die CPU an den Arbeitsspeicher weist ausreichend Bandbreite aus, um die 10 GBit Daten, die von der Netzwerkschnittstelle empfangen oder versendet werden, bearbeiten zu können.

Die restlichen wichtigen Bauteile wie Arbeitsspeicher und Prozessor wurden nach Empfehlungen der Hersteller Myricom und Tyan und sowie dem besten Preis/Leistungsverhältnis ausgewählt. Zur Ermittlung des besten Preis/Leistungsverhältnisses wurden Preissuchmaschinen im Internet benutzt. Dabei wurden folgende wesentliche Bauteile ausgesucht:

- **Prozessor:** 2x AMD Single-Core Opteron 252 Sockel-940 mit 2,6 GHz Taktrate
- **Arbeitsspeicher:** 8x Corsair 1 GB reg. ECC PC3200 DDR SDRAM

Der AMD Prozessor, die 8 GB Corsair-Arbeitsspeicher und das Tyan Motherboard liefern genug Leistung, damit die Netzwerkleistung allein von der Netzwerkkarte und deren physikalischer Schnittstelle begrenzt ist.

Die restlichen Bauteile wie das Netzteil, die Grafikkarte, das 19" Gehäuse, die Festplatte und das optisches Laufwerk wurden den Restbeständen des Max-Planck-Instituts entnommen. Die 19" Gehäuse wurden für eine frühere Anwendung selbst entworfen. Das Tyan Motherboard besitzt keinen AGP-Steckplatz. AGP ist der veraltete Grafikkartenschnittstellenstandard. In den Testsystemen waren jeweils noch ein PCI-Express-Steckplatz nicht belegt, an den eine Grafikkarte angeschlossen hätte werden können. PCI-Express-Grafikkarten waren nicht Teil der Restbestände des Instituts. Es fanden sich jedoch zwei alte Grafikkarten mit einer PCI Schnittstelle aus dem Jahr 1996. Da die Grafikkarte kein wichtiger Faktor bezüglich der Netzwerkleistung ist, wurden diese Karten eingebaut. Als Betriebssystem wurde Suse Linux 10 verwendet.

Im Grunde wurden zwei Hochleistungscomputer mit handelsüblichen Bauteilen aufgebaut. Diese beiden PCs wurden mit einem Lichtwellenleiter verbunden und bilden so eine Punkt-zu-Punkt-Verbindung über ein 10-Gigabit-Ethernet. Es war wichtig handelsübliche Bauteile einzukaufen, da unstandardisierte Lösungen den Kostenrahmen überstiegen hätten.

Tabelle 3.3: Kosten eines Endsystems im Mai 2007

Bauteil	**Nettokosten in Euro**
Motherboard Tyan Thunder K8WE S2895	288,84
2 x AMD Single-Core Opteron 252 Sockel-940 mit 2,6 GHz mit Kühler	436,13
Myri-10G Netzwerkkarte mit PCI-Express Schnittstelle	662,4
Intel TXN18107 10Gbps XFP Transceiver	409,2
8 x Corsair 1 GB reg. ECC PC3200 DDR SDRAM	616,−
be quiet Blackline P5 470W ATX 2.0 Netzteil	65,55
Restliche Bauteile (Gehäuse, Festplatte, Grafikkarte, Optisches Laufwerk)	500,−
Summe	**2978,12**

3.2 Netperf

Netperf ist ein Testprogramm, welches verschiedene Aspekte der Netzwerkleistung messen kann. Das Hauptaugenmerk dieses Programms liegt in der Messung der Daten bei einer unidirektionalen Übertragung von Daten und in der Messung der Leistung von *requests* (engl. für Anfrage) und *responses* (engl. für Antwort) unter Benutzung von entweder TCP oder UDP. Netperf kann auch Tests mit anderen Protokollen und anderen Programmierschnittstellen durchführen, die hier nicht weiter beschrieben werden, da sie für diese Untersuchung irrelevant sind.

Das Programm ist ein Open Source-Programm. Es wurde von vielen Personen, die nicht alle genannt werden können, weil sie zu zahlreich und zum Teil unbekannt sind, programmiert und erweitert. Informell verwaltet und unterstützt wird es jedoch durch Rick Jones. Jones arbeitet für Hewlett-Packard; Netperf ist jedoch kein offizielles Produkt der Firma und wird dementsprechend auch nicht von ihnen unterstützt.

3.2.1 Design von Netperf

Netperf basiert auf einem einfachen Client-Server-Modell. Auf dem Client und auf dem Server muss das entsprechende Programm ausgeführt werden, damit ein Test durchgeführt werden kann. Die gepackte Datei Netperf, die man auf www.netperf.org herunterladen kann, muss auf beiden Computer entpackt werden. Auf der Client-Seite muss die ausführbare Datei *netperf* und auf der Server-Seite *netserver* gestartet werden. Man startet zuerst *netserver,* welches dann an einem bestimmten Port auf Anfragen von *netperf* wartet.

Wenn beide Programme ausgeführt werden, wird zuerst eine Kontrollverbindung aufgebaut. Über diese Verbindung teilt der Client dem Server Informationen über die Art und die Konfiguration des Tests mit. Diese Kontrollverbindung ist unabhängig von der Art des Tests, der durchgeführt wird, immer eine TCP Verbindung. Nachdem die Kontrollverbindung aufgebaut wurde, wird eine separate Datenverbindung geöffnet. Die Messungen werden mit den entsprechenden Protokollen und der entsprechenden Programmierschnittstelle ausgeführt. Wenn der Test beendet wurde, wird die Datenverbindung abgebaut und die Ergebnisse auf der Server-Seite werden über die Kontrollverbindung an den Client gesendet. Dort werden die Ergebnisse mit denen von der Anwendung *netperf* kombiniert und nachdem die Kontrollverbindung ebenfalls beendet wurde, auf dem Bildschirm ausgegeben.

Netperf kann ebenso die CPU-Auslastung messen. Es wird unabhängig von der Anzahl der CPUs in Prozent angegeben. Netperf nutzt verschiedene, Plattform-

abhängige CPU-Auslastungs-Mechanismen. Diese werden bei der Ausgabe durch einen Buchstaben gekennzeichnet. Auf den zwei Computern der Testübertragungsstrecke läuft das Betriebssystem Suse Linux. Bei den Messungen in Kapitel 3.3 nutzt Netperf interne Funktionen auf ‚/proc/stat'. Diese Funktionen messen die Zeit im Leerlauf und während Netperf Messungen durchführt. Die beiden Zeitwerte werden nach der Messung miteinander verglichen, um einen Wert für die benötigte CPU-Auslastung ausgeben zu können Der AMD Opteron unterstützt keine Funktionen wie RSS (siehe 2.2.7), daher bezieht sich die Angabe der CPU-Auslastung bei den Tests immer auf nur einen Prozessor.

3.2.2 Testverfahren

Netperf in der Version 2.4.3 verfügt über insgesamt 22 verschiedene Testverfahren. Man kann diese in zwei Kategorien unterteilen. Die erste Kategorie umfasst die Messung der Datenrate bei einem einfachen unidirektionalen Datentransfer. In der zweiten Kategorie werden sogenannte Request/Response-Tests durchgeführt. Dabei werden die Transaktionen pro Sekunde gemessen. Eine Transaktion wird als ein kompletter Austausch eines *request* und eines *response* definiert. Mit der Transaktionsrate kann die *Round-Trip-Time* ermittelt werden, in dem man die Transaktionsrate invertiert.

Die verschieden Testverfahren sind verschiedene Kombinationen aus Protokollen und Programmierschnittstellen. Für die optische Übertragungsstrecke am digitalen Empfänger des Max-Planck-Instituts interessieren nur die reine Datenübertragung und die mögliche Datenrate. Es wurden drei verschiedene Tests der ersten Kategorie durchgeführt, die auch von der Firma Myricom absolviert wurden und deren Ergebnisse auf deren Internetseite präsentiert sind. Diese Tests sind:

- **TCP-Stream-Test:** Dieser Test ist der Standardtest von Netperf. Es nutzt eine TCP-Verbindung, um Daten vom Client an den Server zu schicken. Die Zeit für den Aufbau der TCP-Verbindung wird bei der Berechnung der Datenrate nicht berücksichtigt, während die Zeit, die das letzte Paket zum Erreichen des Servers braucht, mit eingerechnet wird.
- **TCP-Sendfile-Test:** Dieser Test ist ähnlich zu dem TCP-Stream-Test. Es aber anstatt dem *send*-Befehl der *sendfile*-Befehl aufgerufen. Das führt zu einer *Zero-Copy*-Operation, welches die CPU-Auslastung verringern sollte (siehe 2.2.5).
- **UDP-Stream-Test:** Der UDP-Stream-Test funktioniert wie der TCP-Stream-Test. Es wird nur das UDP-Protokoll anstatt des TCP-Protokolls genutzt. Da UDP ein unzuverlässiges Protokoll ist und nicht garantiert werden kann, dass die UDP-Pakete ihr Ziel erreichen, werden bei der

Ausgabe der Testergebnisse zwei Datenraten angegeben. Die erste Zeile gibt die Datenrate auf der Senderseite an und die zweite Zeile die Datenrate der Empfängerseite.

Alle gemessenen Datenraten beziehen sich auf die Daten, die dem Netzwerk-Socket vom Host übergeben wurden. Das bedeutet, dass in den Resultaten keine TCP/UDP-, IP- oder Ethernet-Header mit in die Berechnungen einfließen.

3.2.3 Einstellungen

Netperf besitzt eine große Menge Einstellungsmöglichkeiten, die durch den Nutzer bei Aufruf von *netperf* und *netserver* gesetzt werden können. Die Optionen werden durch einen Buchstaben nach einem Bindestrich hinter der ausführbaren Datei eingestellt. Der komplette Befehl muss dann in die Eingabezeile der Benutzerschnittstelle getippt werden.

Es gibt zwei Arten von Optionen. Die *Global Command-line Options* beziehen sich auf annähernd alle Tests von Netperf, während die *Test-specific Command-line Options* nur für die spezielle Testart gelten. Die Optionen werden in die gleiche Zeile geschrieben, müssen jedoch von einem doppelten Bindestrich getrennt werden. Zuerst werden die *Global Command-line Options* eingetippt und nach dem doppelten Bindestrich stehen die *Test-specific Command-line Options.*

Es folgen die in dem Test verwendeten Einstellungen:

Global Command-line Options:

- **‚-c':** Diese Option fordert Netperf auf die benötigte CPU-Auslastung für den Test des Clients zu berechnen und mit den übrigen Testergebnissen anzugeben.
- **‚-C':** Diese Option führt das Gleiche wie die ‚-c'-Option aus, doch statt der des Clients wird die CPU-Auslastung des Servers berechnet.
- **‚-F':** Mit dieser Option kann eine Datei angegeben werden, mit welcher der Sendepuffer vorgeladen wird. Die Datei wird nicht komplett an das entfernte System übertragen. Der Empfänger wird ebenso nicht versuchen, die empfangenen Teile der Datei zu einer ganzen Datei zusammenzufügen. Tests die den *sendfile*-Befehl aufrufen, müssen mit dieser Option versehen werden, da sich dieses Kommando auf eine Datei bezieht.
- **‚-H':** Diese Option gibt die Adresse des Zielhosts an. Es kann die IPv4- oder die IPv6-Adresse benutzt werden. Gibt man keine Adresse an, wird standardmäßig die Loopback-Adresse gesetzt. Diese Adresse lautet 127.0.0.1, außerdem wird der Name *localhost* vergeben. Ein Datagramm, welches an diese Netzwerkadresse geschickt wird, darf nicht auf dem

physikalischen Netzwerk erscheinen. Das Datagramm wird innerhalb der Transport- und Netzwerkschicht verarbeitet und über eine Schleife zurückgeführt.

- **‚-l':** Hiermit wird die Länge eines Tests in Sekunden angegeben.
- **‚-n':** Mit dieser Option kann die Anzahl der Prozessoren des Hosts gesetzt werden. Netperf kann die Anzahl auch automatisch herausfinden, die Funktion wird jedoch bei eingeschalteter Option ignoriert.
- **‚-t':** Mit dieser Option wird der Test, der durchgeführt werden soll, angegeben.

Test-specific Command-line Options:

- **‚-m':** Diese Option stellt den Puffer ein, der mit dem *send*-Befehl erzeugt wird. Auf diese Weise kann man die Größe der Nutzdaten von einem TCP- oder UDP-Paket bestimmen. Bei TCP ist jedoch nicht garantiert, dass diese Größe eingehalten wird, da TCP über Mechanismen verfügt, die Daten aufteilen oder zusammenfügen.
- **‚-s':** Mit dieser Option wird die Größe des Sende- und des Empfangspuffers auf dem Client eingestellt. Diese Einstellung hat Einfluss auf die TCP-Fenstergröße und somit auf die Flusssteuerung und die Übertragungsrate. Die Einheit ist Bytes. Bei einem Suffix von „K", „M" oder „G" sind es entsprechend kB, MB oder GB.
- **‚-S':** Mit dieser Option wird die Größe des Sende- und des Empfangspuffers auf dem Server eingestellt.

3.3 Ergebnisse

In diesem Kapitel werden die wesentlichen Ergebnisse aufgezeigt, die sich bei der Untersuchung der 10-Gigabit-Testübertragungsstrecke ergeben haben. Alle Messungen wurden mit dem Programm Netperf durchgeführt. Die Messungen wurden mit verschiedenen Einstellungen der Netzwerkschnittstelle gestartet. Diese Konfigurationen wurden mit dem Gerätetreiber der Myri-10G-Netzwerkkarte und durch Linuxprogramme, die Änderungen an der Netzwerkschnittstelle vornehmen können, eingestellt. Die Tests wurden unter verschiedenen Linux-Kernel-Versionen ausgeführt.

Die verschiedenen Einstellungen wurden anhand Empfehlungen der *Readme* Datei, die mit den Software Treibern mitgeliefert worden war, und der Support-Internetseite der Firma Myricom ausgewählt und vorgenommen. Für jede Konfiguration wurden drei verschiedene Tests durchgeführt, deren Befehle zur

Eingabe in die Benutzerschnittstelle von Linux bis auf eine Ausnahme immer gleich war. Die Kommandos waren:

- *netperf –H 192.168.0.1 –t TCP_STREAM –c –C –l 60*

  Mit diesem Befehl wird ein 60 s langer TCP-Stream-Test vom Client an den Server mit der Adresse 192.168.0.1 gestartet. Die CPU-Auslastung von Client und Server wird ebenfalls berechnet und ausgegeben.

- *netperf –H 192.168.0.1 –t TCP_SENDFILE –c –C –l 60 –F /boot/vmlinuz*

  Mit diesem Befehl wird ein 60 s langer TCP-Sendfile-Test vom Client an den Server mit der Adresse 192.168.0.1 gestartet. Hierbei wird der Sendepuffer mit der Datei *vmlinuz*, die sich auf der Hauptfestplatte im Ordner *boot* befindet, vorgeladen. Diese Datei ist das *Bootimage* eines Linux Kernels. *vmlinuz* ist eine komprimierte Kernel-Datei, die zum Hochfahren des Betriebssystems benötigt wird. Die CPU-Auslastung von Client und Server wird ebenfalls berechnet und ausgegeben.

- *netperf –H 192.168.0.1 –t UDP_STREAM –c –C –l 60 -- -m 8972 –s 1M –S 1M*

  Die erste Befehlszeile wird für eine MTU von 9000 und die Folgende für eine MTU von 1500 verwendet.

  netperf –H 192.168.0.1 –t UDP_STREAM –c –C –l 60 -- -m 1472 –s 1M –S 1M

  Mit diesen Befehlen wird ein 60 s langer UDP-Stream-Test vom Client an den Server mit der Adresse 192.168.0.1 gestartet. Die CPU-Auslastung von Client und Server wird ebenfalls berechnet und ausgegeben. Der Sende- und Empfangspuffer des Clients und des Servers werden auf 1 MB gesetzt. Die ‚–m'-Option stellt die Nachrichtengröße ein, für das ein Datagramm erstellt und dann versendet wird. Die Nachrichtengröße ist genau um 28 Bytes kleiner als die MTU, die eingestellt wurde. Ein UDP-Nachricht braucht 8 Bytes für seinen Header und 20 Bytes für den IP-Header. Mit den 28 Bytes der Header wird genau die maximale Rahmengröße für Ethernet erfüllt.

Im Voraus wurde bei jeder Linux-Kernel-Version überprüft, ob genügend Bandbreite für den Speicherzugriff zur Verfügung steht. Die Myri-10G-Netzwerkkarte besitzt eine 8-spurige PCI-Express-Schnittstelle, die über eine theoretische Bandbreite von 2 GB/s verfügt. Mit dem Befehl *ethtool –S eth2* können Netzwerkkarten- und Treiber-spezifische Statistiken abgerufen werden. Damit lässt sich feststellen, ob die benötigte Speicherbandbreite verfügbar ist. Für den Read_DMA (Lesezugriff) beträgt der Wert 1382 MB/s und für den Write_DMA (Schreibzugriff) beträgt der Wert 1369 MB/s. Bei mehreren Überprüfungen weichen die Werte nur wenig von den oben genannten Werten ab. Es

steht somit genügend Bandbreite für den Speicherzugriff zur Verfügung, um den theoretischen maximalen Datenstrom der Netzwerkschnittstelle von 1289 MB/s (=10,3125 Gbit/s/8) verarbeiten zu können.

Die Netzwerkpuffer beider Systeme wurden vor Beginn der Messungen erhöht. Dazu muss die Datei *sysctl.conf* manipuliert werden. Mit dieser Datei können Einstellungen im TCP/IP-Protokoll-Stack und am virtuellen Speichersystem unter Linux vorgenommen werden. Die Datei wird beim Hochfahren von Linux gelesen. Sie kann aber auch während des Betriebs ausgeführt werden. Es wurden folgende Zeilen in die Datei *sysctl.conf* im Ordner */etc* auf der Hauptfestplatte hinzugefügt [6]:

net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216

Die ersten beiden Zeilen erhöhen die Obergrenze der Netzwerkpuffer. Die letzten beiden Zeilen setzen die Grenzen des Rahmens fest, in dem Linux automatisch seinen Netzwerkpuffer einstellen kann. Der linke Wert ist die Untergrenze, der rechte Wert ist die Obergrenze und die mittlere Zahl ist der Standardwert, der von Linux benutzt werden soll. Diese Zeilen beziehen sich auf TCP und Internet Protocol Version 4.

Die Myricom-Gerätetreiber liefen mit der Version 1.3.0 und, sofern nicht anderes gekennzeichnet, wurde die Firmware Version 1.4.21 genutzt. *Write Combining* (siehe 2.2.10) war für alle Tests aktiviert. Tabelle 3.4 zeigt die wichtigsten Einstellmöglichkeiten mit ihren Standardwerten, die, sollten keine anderen Werte gekennzeichnet sein, für alle Messungen übernommen wurden. Die *Allocatione Order* ist ein Maß für die Anzahl von zusammenhängendem Speicherseiten, der als Empfangspuffer zur Verfügung gestellt wird.

Tabelle 3.4: Die wichtigsten Einstellmöglichkeiten mit Standardwerten - (0 = deaktiviert, 1 = aktiviert)

Option	Standardwert
Write Combining First-In-First-Out (wcfifo)	0
Timestamps	1
Interrupt Coalescing	75 µs
Allocation Order	0
MTU	9000 Bytes

3.3.1 Erste Versuche mit Netperf

Die ersten Versuche wurden mit der Linux-Kernel-Version 2.6.16.13 durchgeführt.

Tabelle 3.5: Erster Versuch mit Linux-Kernel-Version 2.6.16.13 - Befehl: *netperf –H 192.168.0.1*

Testnummer	Datenrate / Mbit/s	Sender-CPU-Auslastung / %	Empfänger-CPU-Auslastung / %
1	1,12	0,76	0,40
2	1,13	0,40	0,35
3	1,11	0,55	0,50
4	1,12	0,51	0,41
5	1,11	0,62	0,42
Durchschnitt	1,12	0,57	0,42

Es wurden weitere Tests mit dieser Kernel-Version ausgeführt, wobei verschiedene Konfigurationen eingestellt wurden. Es waren keine Unterschiede zu den Ergebnissen in Tabelle 3.5 feststellbar. UDP-Stream-Tests zeigten Werte für die Datenrate am Sender von rund 9600 Mbit/s und für die Datenrate am Empfänger von rund 6800 Mbit/s.

Eine neue Linux-Kernel-Version erbrachte wesentlich bessere Ergebnisse. Mithilfe von [10] wurde die Version 2.6.18.5, die zuvor von der Seite www.kernel.org heruntergeladen wurde, konfiguriert, compiliert und installiert. Vor den Ergebnissen der drei Tests steht immer eine Auflistung der Konfigurationen, die bei dem Testdurchlauf eingestellt waren. Veränderte Parameter zu den Standardwerten in der Konfiguration sind Fett gedruckt.

Im ersten Testdurchlauf mit der Kernel-Version 2.6.18.5 wurden die Standardeinstellungen aus Tabelle 3.4 übernommen.

1. Testdurchlauf (Tabelle 3.6, Tabelle 3.7)
WCFiFo = 0; MTU = 9000; Interrupt coalescing = 75 µs; Allocation Order = 0; Timestamps = 1

Tabelle 3.6: 1. Testdurchlauf – TCP-Tests

Testnummer	Datenrate/ Mbit/s	Sender-CPU-Auslastung/%	Empfänger-CPU-Auslastung/%
TCP-STREAM			
1	9416,46	50,05	41,53
2	9545,04	50,02	42,44
3	9413,21	50,04	41,99
4	9349,45	57,76	41,75
5	9188,95	62,79	40,71
Durchschnitt	**9382,62**	**54,13**	**41,68**
TCP-SENDFILE			
1	9910,25	11,47	43,80
2	9910,19	10,89	43,82
3	9910,28	10,74	43,43
4	9910,41	11,29	43,81
5	9910,32	10,94	43,76
Durchschnitt	**9910,29**	**11,07**	**43,72**

Tabelle 3.7: 1. Testdurchlauf – UDP-STREAM-Test

Testnummer	Seite	Datenrate/ Mbit/s	Sender-CPU-Auslastung/%	Empfänger-CPU-Auslastung/%
1	Sender	9918,40	50,02	
	Empfänger	9472,90		49,98
2	Sender	9918,50	50,02	
	Empfänger	9012,60		50,04
3	Sender	9918,70	50,03	
	Empfänger	9039,30		50,05
4	Sender	9915,00	50,02	
	Empfänger	9024,80		50,06
5	Sender	9917,70	50,02	
	Empfänger	9051,60		49,99
Durchschnitt	Sender	**9917,66**	**50,02**	
	Empfänger	**9120,24**		**50,02**

Im nächsten Testdurchlauf wurde ein MTU von 1500 verwendet.

2. Testdurchlauf (Tabelle 3.8, Tabelle 3.9)
WCFiFo = 0; **MTU = 1500**; IRQ coalescing = 75 µs; Allocation Order = 0; Timestamps = 1

Tabelle 3.8: 2. Testdurchlauf – TCP-Tests

Testnummer	Datenrate/ Mbit/s	Sender-CPU-Auslastung/%	Empfänger-CPU-Auslastung/%
TCP-STREAM			
1	7445,08	49,39	45,89
2	7151,87	53,68	44,39
3	7414,06	49,54	45,25
4	7395,45	50,17	45,41
5	7336,80	49,21	44,80
Durchschnitt	**7348,65**	**50,40**	**45,15**
TCP-SENDFILE			
1	8016,58	10,48	49,99
2	8076,12	9,83	49,99
3	8039,07	9,89	49,99
4	8020,39	9,83	50,04
5	8003,46	9,93	50,07
Durchschnitt	**8031,12**	**9,99**	**50,02**

Tabelle 3.9: 2. Testdurchlauf – UDP-STREAM-Tests

Testnummer	Seite	Datenrate/ Mbit/s	Sender-CPU-Auslastung/%	Empfänger-CPU-Auslastung/%
1	Sender	3186,60	60,91	
	Empfänger	3059,70		48,01
2	Sender	3254,90	61,09	
	Empfänger	3254,90		45,58
3	Sender	3247,50	61,50	
	Empfänger	3247,50		45,94
4	Sender	3533,30	50,33	
	Empfänger	2613,10		49,99
5	Sender	3470,40	50,03	
	Empfänger	2667,70		50,09
Durchschnitt	Sender	**3338,54**	**56,77**	
	Empfänger	**2968,58**		**47,92**

3.3.2 Vergleich MTU 9000 mit MTU 1500

Es wurden insgesamt 14 Testdurchläufe mit unterschiedlichen Konfigurationen vorgenommen. 14 entspricht nicht der maximalen Anzahl an Möglichkeiten mit den verschiedenen Parametern. Während die Parameter *WCFiFo, Timestamps* und MTU nur zwei mögliche Werte haben, ist es für die *Allocation Order* möglich vier Werte einzustellen. Das *Interrupt Coalescing* ist frei einstellbar. Dabei stellt der Wert, der für diesen Parameter eingegeben wird, eine Zeitgröße in µs dar (siehe 2.2.4). Übliche Werte sind 25 µs, 50 µs, 75 µs und 100 µs. 75 µs ist der Standardwert und bei einer Eingabe von 0 wird die *Interrupt Coalescing* Funktion ausgeschaltet. Nimmt man nur die fünf üblichen Werte für das *Interrupt Coalescing* ergeben sich bereits 160 Kombinationen. Diese wurden aus praktischen Gründen nicht alle durchgeführt.

Die vollständigen Tabellen sind im Anhang A.1 zu finden.

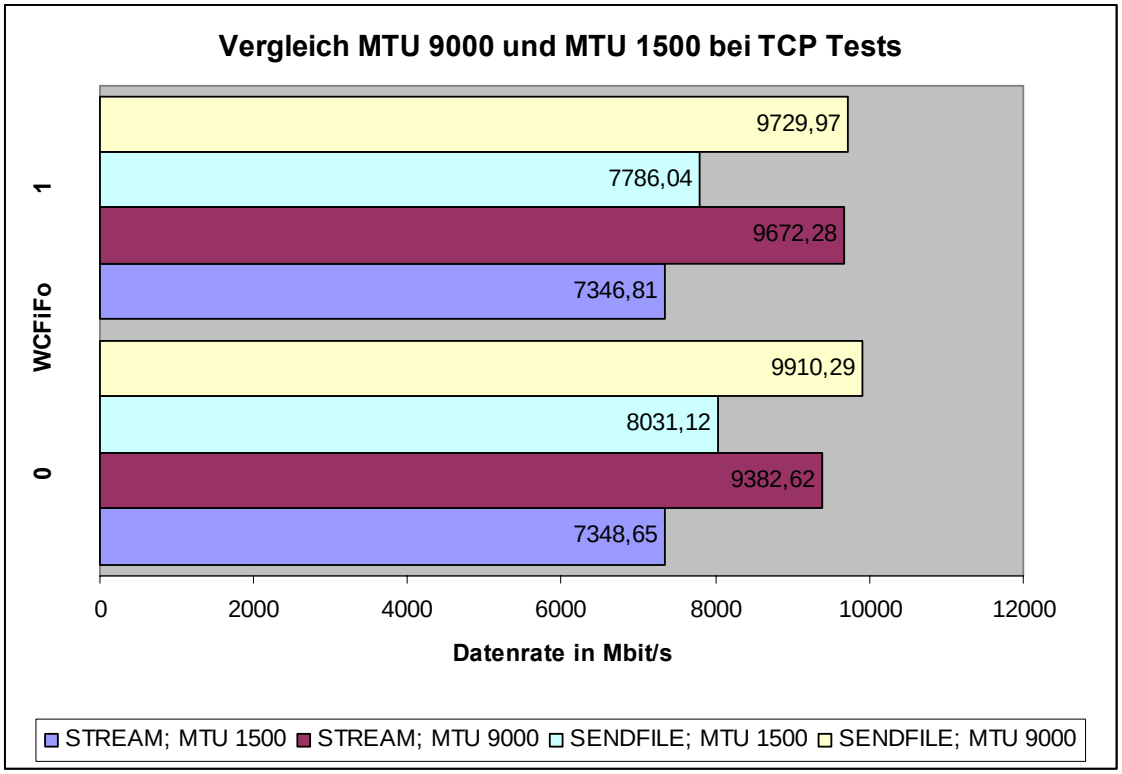

Bild 3.1: Vergleich MTU 9000 und MTU 1500 bei TCP Tests

Bild 3.1 zeigt den Vergleich von Messungen der Datenrate unter Benutzung der MTU von 1500 und der MTU von 9000. Für beide MTUs wurden auch Tests mit aktiviertem und deaktiviertem *WCFiFo* durchgeführt. Es wird deutlich, dass bei einer MTU von 1500 eine geringere Datenrate erreicht wird. Die Messwerte mit der kleineren MTU sind 20-25 % kleiner als die mit der großen MTU von 9000. In Bild 3.1 sind sowohl TCP-Sendfile-Ergebnisse als auch TCP-Stream-

Ergebnisse dargestellt. Bei den TCP-Sendfile-Messungen hat die aktivierte *WCFiFo*-Option negative Auswirkungen auf die Datenrate, während bei den TCP-Stream-Tests das Gegenteil der Fall ist.

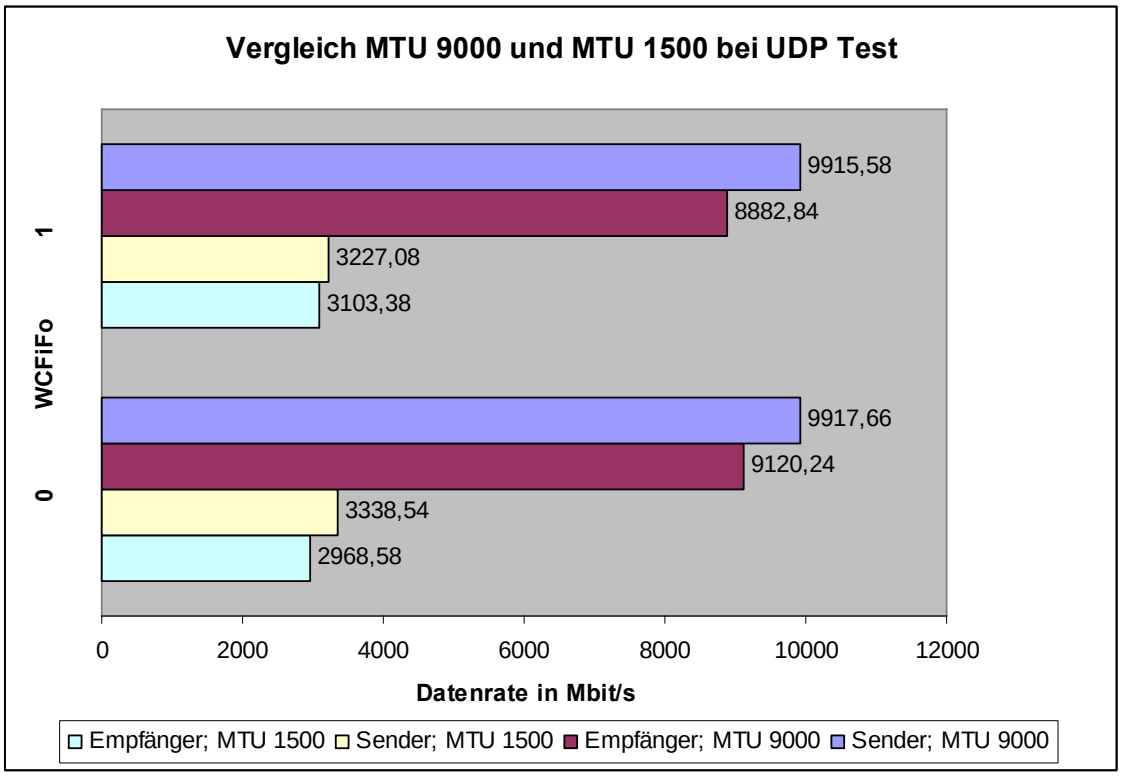

Bild 3.2: Vergleich MTU 9000 und MTU 1500 bei UDP-Test

Bild 3.2 zeigt den Vergleich von MTU 9000 und MTU 1500 bei UDP-Tests. Es galten die gleichen Testbedingungen wie bei den TCP-Tests, die in Bild 3.1 dargestellt sind. Auf Bild 3.2 ist zu beachten, dass in diesem Fall die Datenrate, die beim Sender gemessen wurde, oberhalb der Datenrate am Empfänger steht. Der Unterschied zwischen den verschiedenen MTUs ist bei den UDP-Messungen größer. Die Datenraten mit einer MTU von 1500 betragen nur 30-35 % der Datenraten mit einer MTU von 9000. Die Variation der *WCFiFo*-Option zeigt keinen merklichen Unterschied in der Sendeleistung. Über die Empfangsleistung lässt sich keine Aussage hinsichtlich eines Vorteils oder Nachteils machen, da es sich für beide MTUs einmal zum Vorteil und einmal zum Nachteil erweist.

3.3.3 Versuche mit verschiedenen WCFiFo- und Timestamps-Optionen

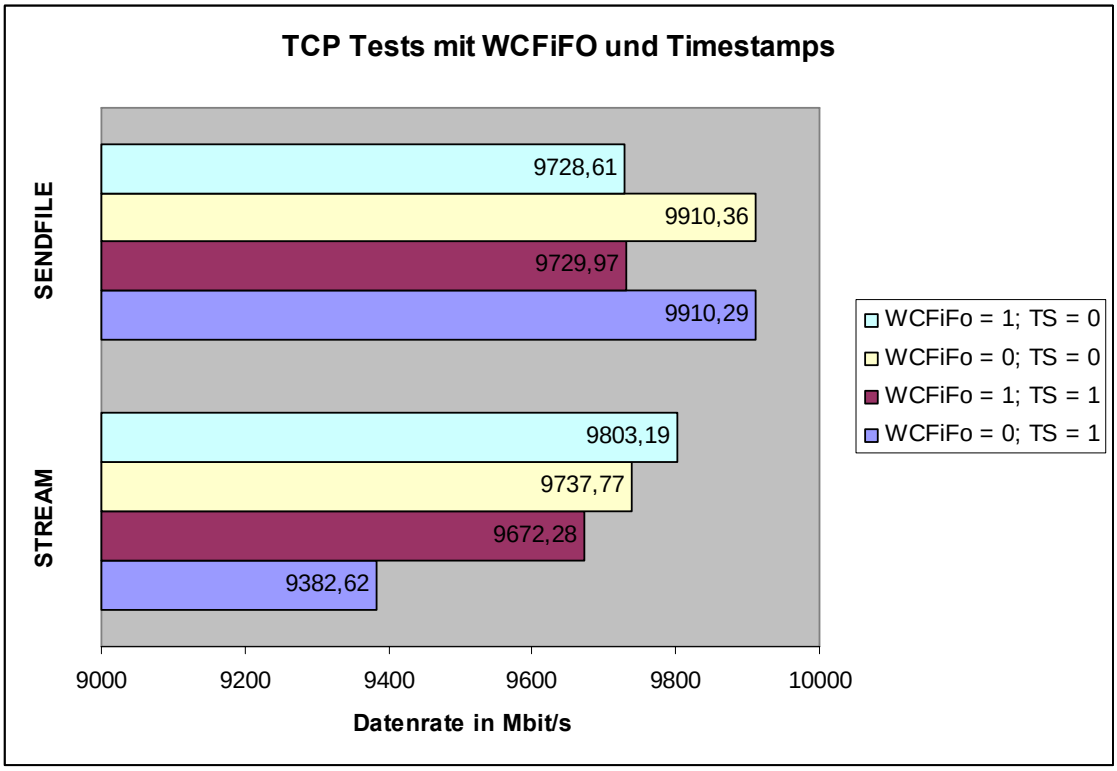

Bild 3.3: TCP Tests mit Variation der WCFiFo- und Timestamps-Optionen

Bild 3.3 zeigt die Messungen mit den vier Kombinationen bei der Einstellung der *WCFiFo*- und der *Timestamps*-Option jeweils für TCP-Stream- und TCP-Sendfile-Tests. Wie schon in Bild 3.1 zu sehen ist, führt eine Aktivierung der *WCFiFo*-Option bei TCP-Sendfile-Tests zu einer Verringerung der Datenrate von 1,8 %. Die Werte bleiben von der Veränderung der *Timestamps*-Option unberührt. Für TCP-Stream-Tests erweisen sich beide Optimierungen als positiv. Das Ausschalten der *Timestamps*- und die Aktivierung der *WCFiFo*-Option führen zu einer Verbesserung der Datenrate.

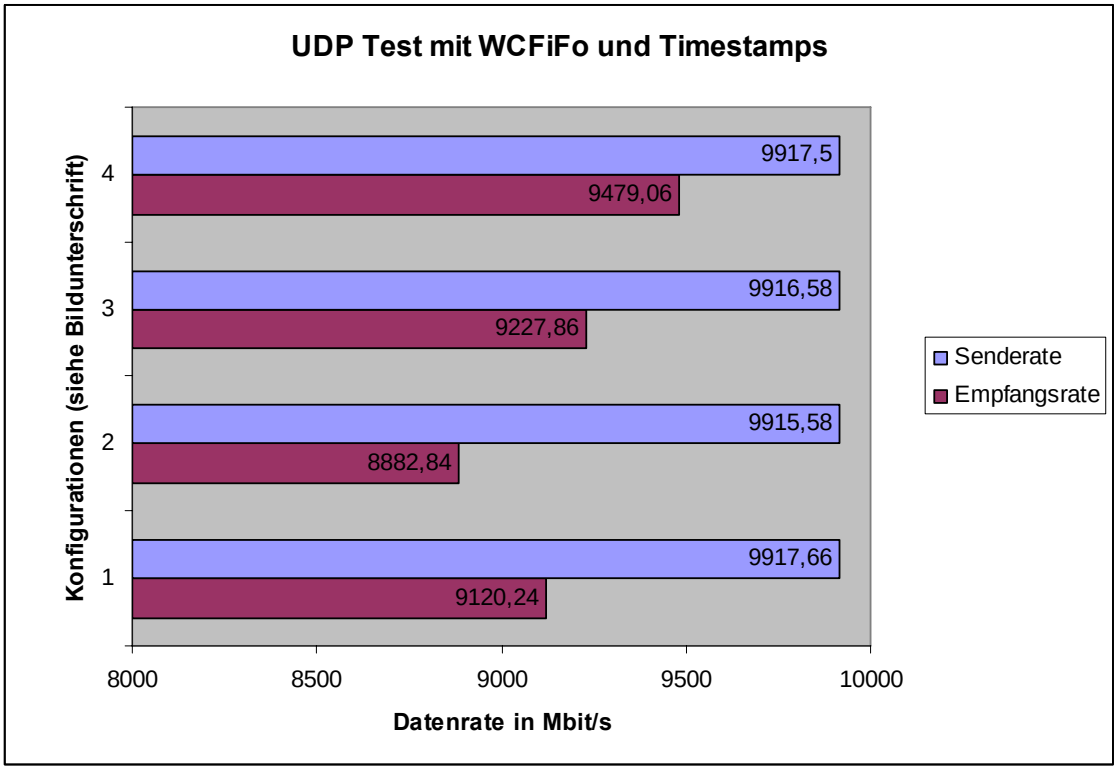

Bild 3.4: UDP Test mit Variation der WCFiFo und Timestamps Optionen – Konfigurationen:
1: WCFiFo = 0; Timestamps = 1
2: WCFiFo = 1; Timestamps = 1
3: WCFiFo = 0; Timestamps = 0
4: WCFiFo = 1; Timestamps = 0

Bild 3.4 zeigt die Ergebnisse der gleichen Testdurchführung wie die auf Bild 3.3. Es wurde mit verschiedenen Einstellungen der *WCFiFo*- und *Timestamps*-Optionen gemessen. Die *Timestamps*-Option hat keinen Einfluss auf UDP-Übertragungen, da es eine optionale Funktion des TCPs ist. Über die Auswirkungen durch die Veränderung der *WCFiFo*-Option lässt sich keine eindeutige Aussage machen. Bei der ersten Aktivierung von der 1. zur 2. Konfiguration (siehe Bild 3.4) verringert sich die Datenrate um 237,4 Mbit/s, während bei dem Wechsel der 3. zur 4. Konfiguration sich eine Verbesserung von 251,2 Mbit/s einstellt.

3.3.4 Versuche mit verschiedenen Allocation Order Werten

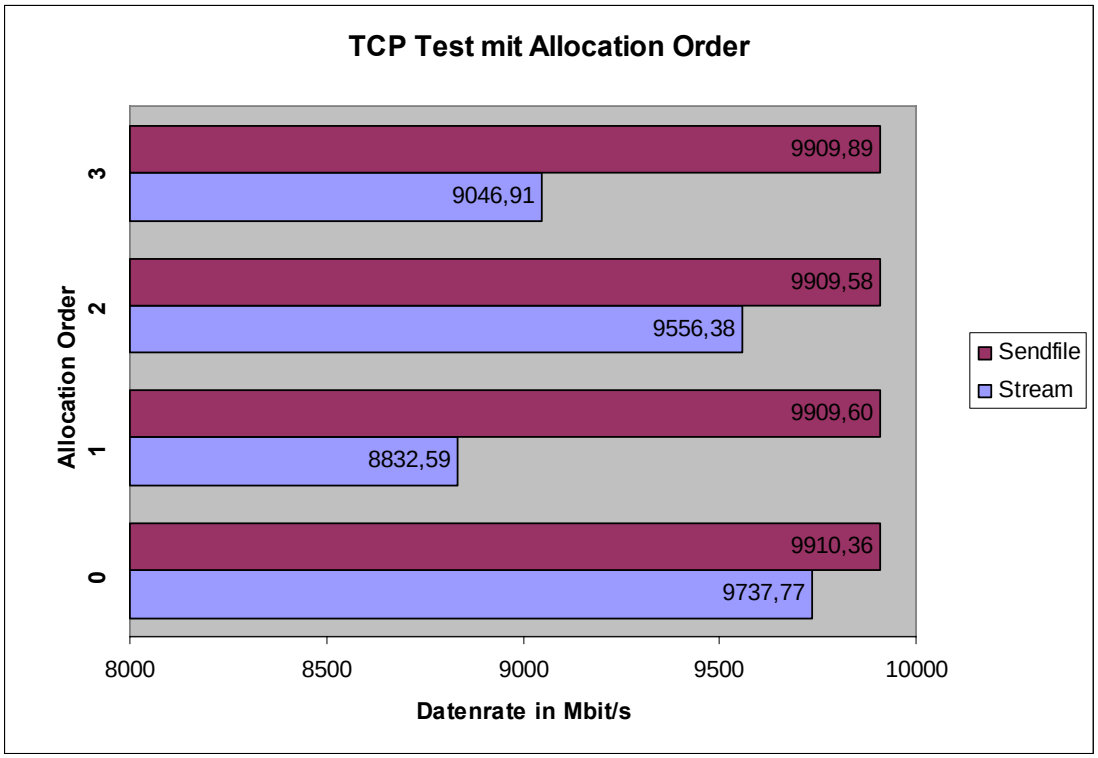

Bild 3.5: TCP Tests mit Variation der Allocation Order - WCFiFo = 0; Timestamps = 0

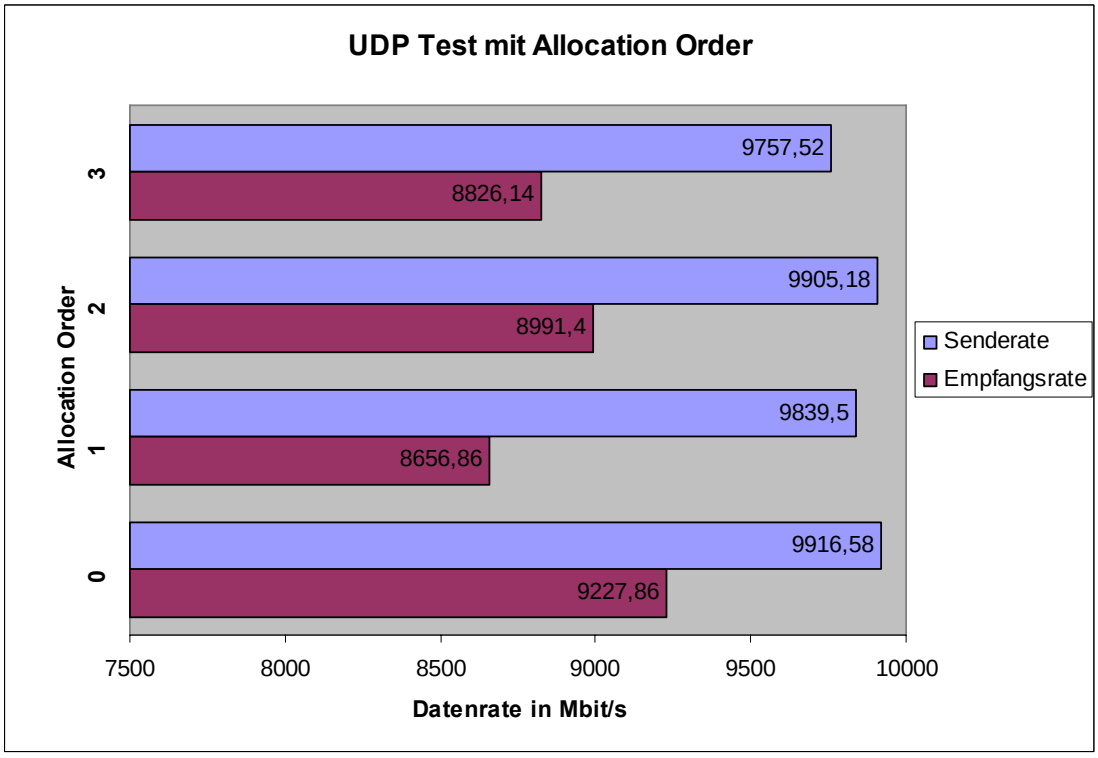

Bild 3.6: UDP Test mit Variation der Allocation Order

Bild 3.5 zeigt den Einfluss der *Allocation Order* auf die Datenrate bei TCP-Stream- und Sendfile-Tests. Für einen TCP-Stream-Test führt die Standardeinstellung der *Allocation Order* mit dem Wert 0 zu den besten Ergebnissen. Bei den TCP-Sendfile-Tests hat der Wert der *Allocation Order* keinen deutlichen Einfluss auf die Datenrate.

Die Standardeinstellung der *Allocation Order* führt ebenso bei den UDP-Messungen zu den besten Ergebnissen (siehe Bild 3.6).

3.3.5 Versuche mit verschiedenen Interrupt Coalescing Zeiten

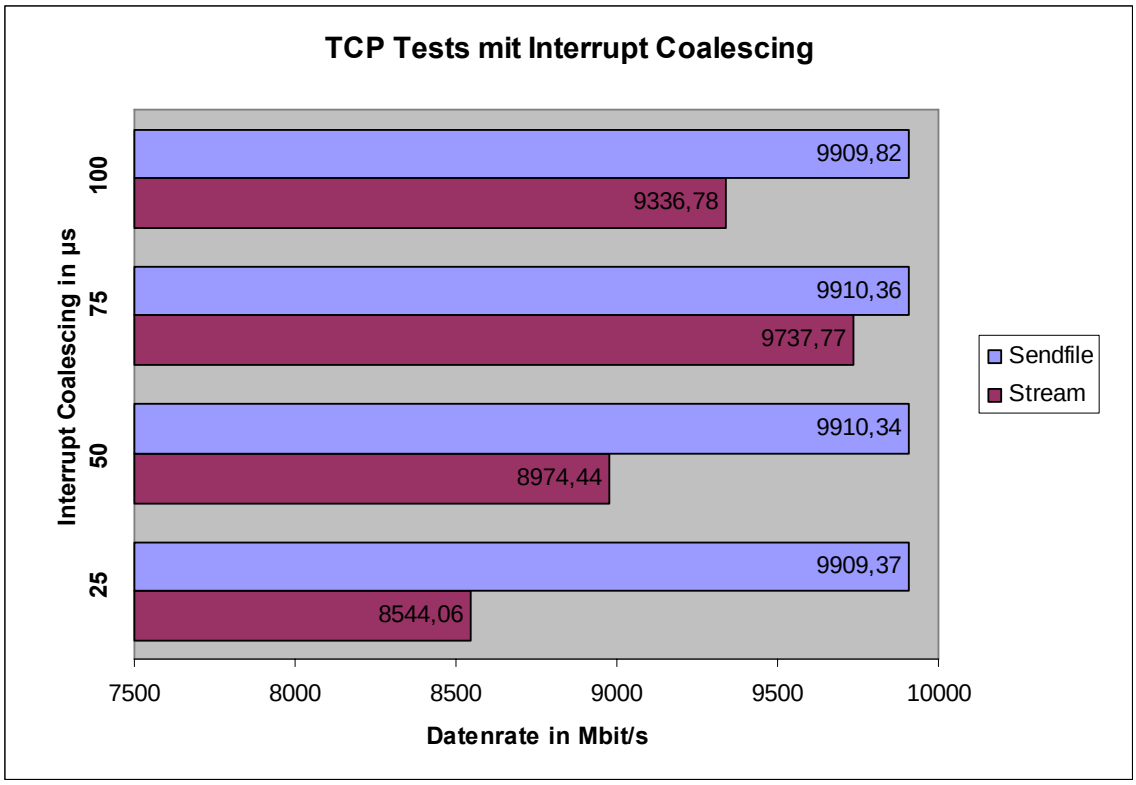

Bild 3.7: TCP Tests mit verschiedenen Interrupt Coalescing Zeiten

Auch bei TCP Tests mit verschiedenen *Interrupt Coalescing* Zeiten (siehe Bild 3.7) ist die Standardeinstellung der Myricom-Treiber mit 75 µs die beste, wobei die verschiedenen Einstellungen keinen deutlichen Einfluss auf die TCP-Sendfile Tests haben. In Bild 3.8 kann man jedoch sehen, dass eine *Interrupt Coalescing*-Zeit von 25 µs zu einer höheren CPU-Auslastung bei dem Sender und dem Empfänger führt. Es ist ebenso auf Bild 3.9 zu sehen. Dort hat es eine stärkere Auswirkung auf die Sender-CPU-Auslastung. Diese hat bei der Standardeinstellung einen kleinen Wert von 11,5 %. Bei einer *Interrupt Coalescing*-Zeit von 25 µs vervierfacht sich annähernd die Sender-CPU-Auslastung.

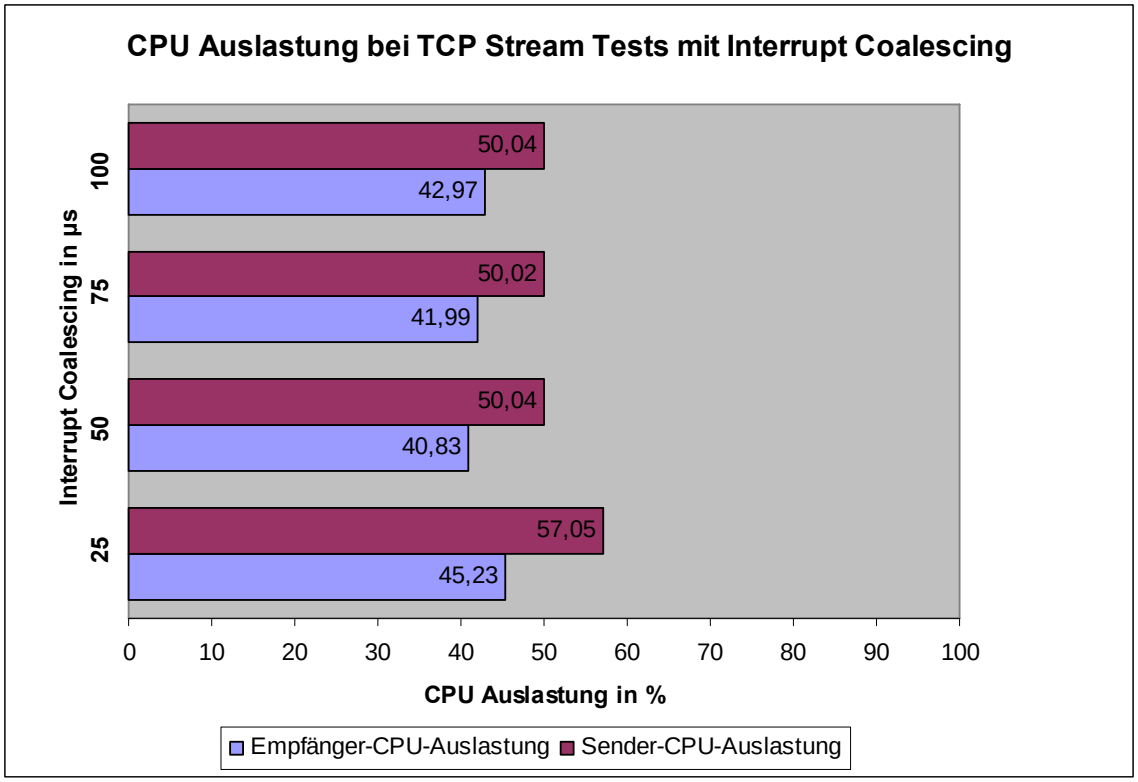

Bild 3.8: CPU Auslastung beim TCP Stream Test mit verschiedenen Interrupt Coalescing Zeiten

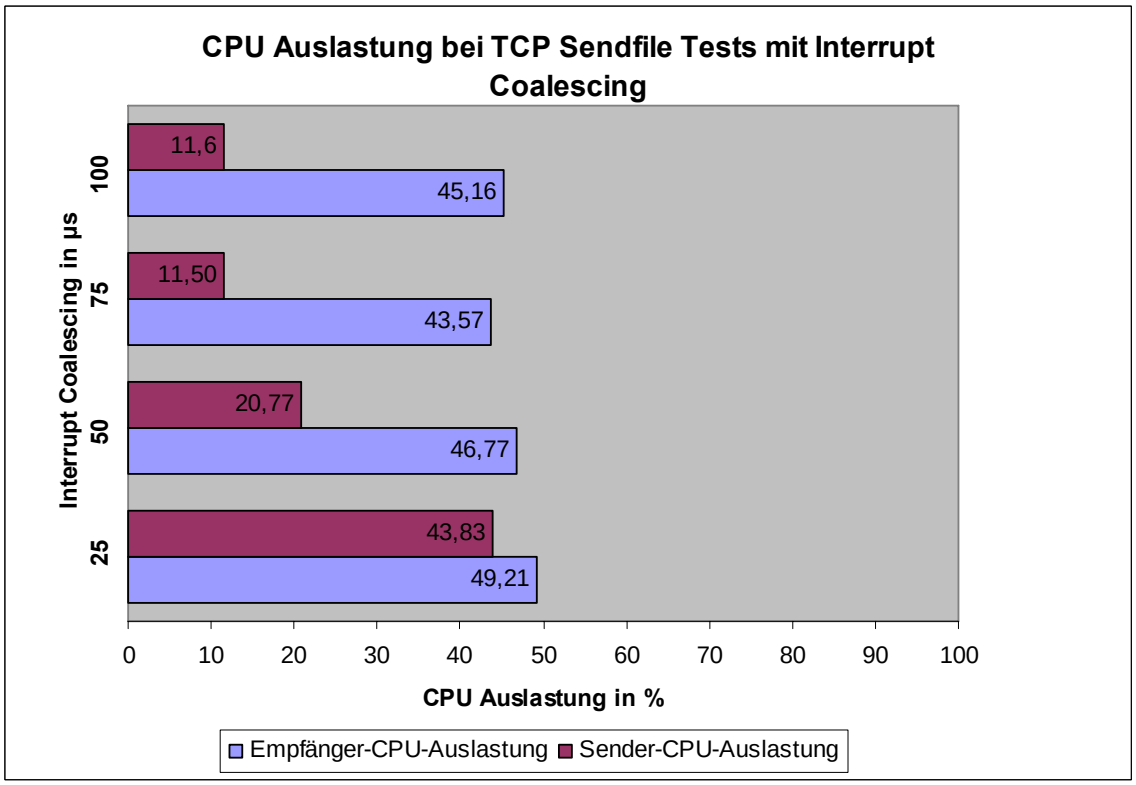

Bild 3.9: CPU Auslastung bei TCP Sendfile Test mit verschiedenen Interrupt Coalescing Zeiten

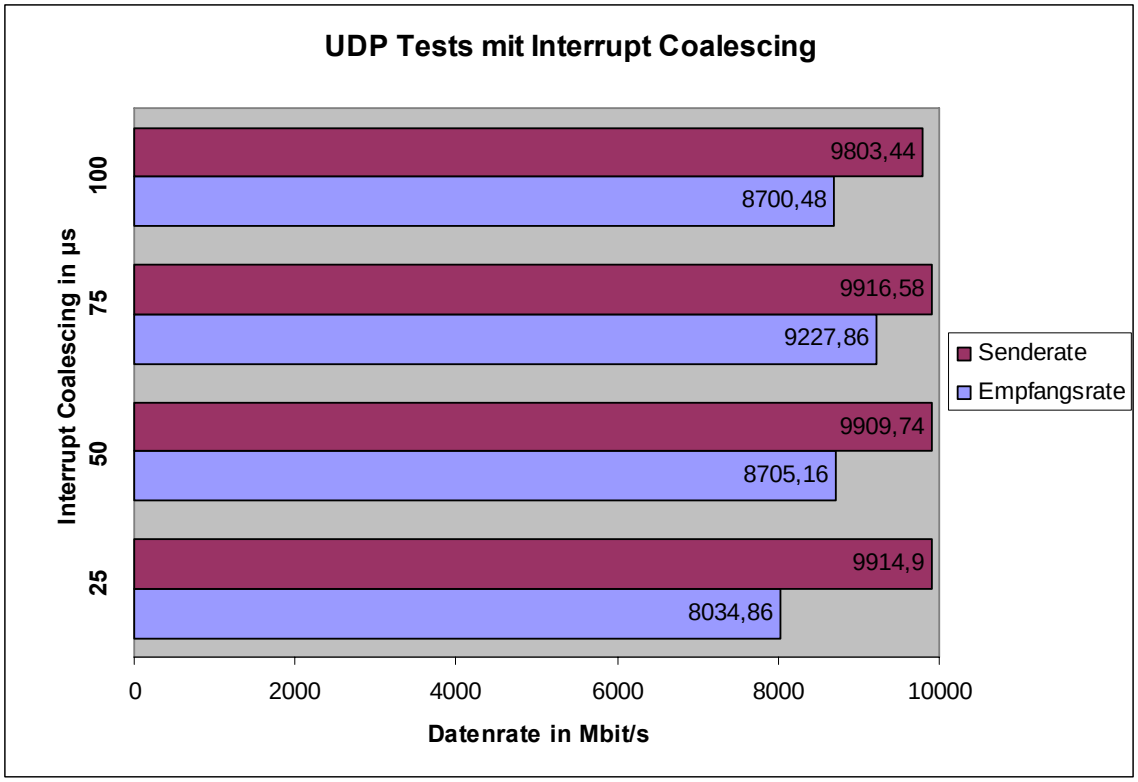

Bild 3.10: UDP Tests mit verschiedenen Interrupt Coalescing Zeiten

Auf Bild 3.10 sieht man die Ergebnisse der entsprechenden UDP-Messungen mit verschiedenen *Interrupt Coalescing*-Zeiten. Auch hier ist die Standardeinstellung von 75 µs die beste. Auffällig hierbei ist, dass bei 100 µs die Datenrate abgefallen ist. Auf Bild 3.11 kann man sehen, dass die CPU-Auslastung des Senders bei dieser Einstellung um annähernd 20 % angestiegen ist.

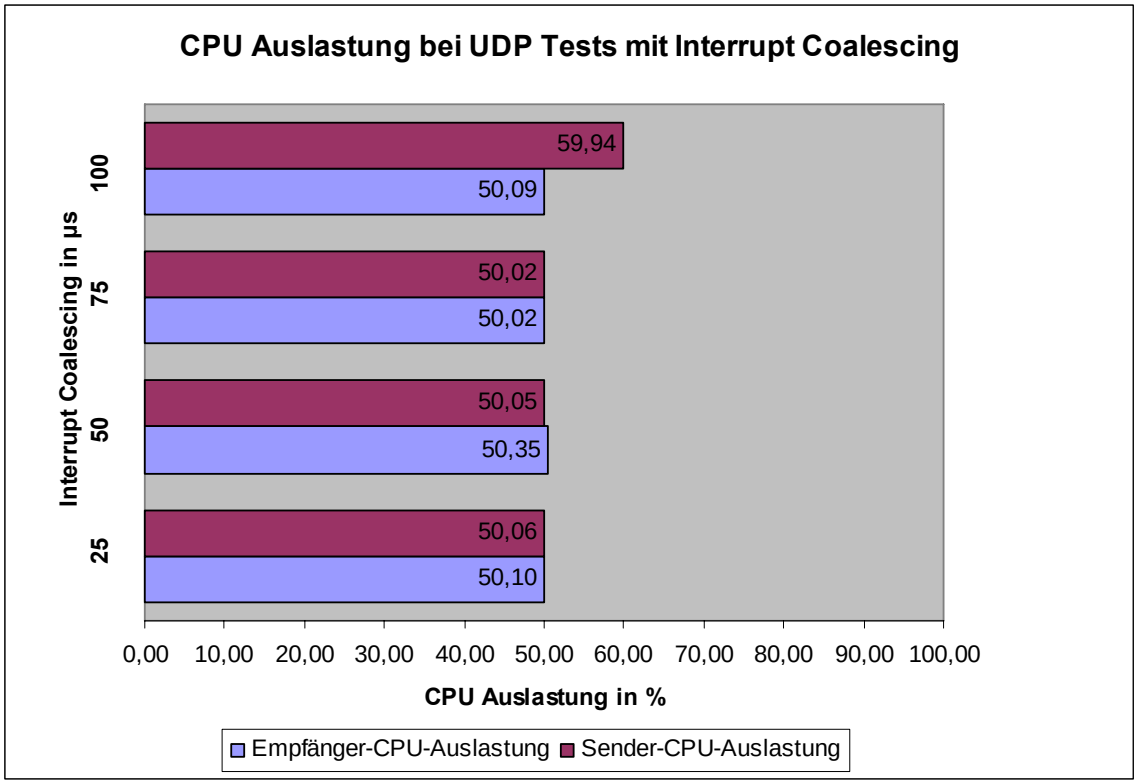

Bild 3.11: CPU Auslastung bei UDP Tests mit verschiedenen Interrupt Coalescing Zeiten

3.3.6 Die CPU Auslastung bei verschiedenen Versuchen

Die Parameter *Timestamps, WCFiFo* und *Allocation Order* haben keinen starken Einfluss auf die CPU-Auslastung. Die Auswirkungen von verschiedenen *Interrupt Coalescing*-Zeiten sind oben beschrieben. Den Einfluss der MTU auf die CPU-Auslastung zeigen Bild 3.12 und Bild 3.13. Auf Bild 3.12 kann man sehen, dass die CPU-Auslastung für beide TCP-Testarten beim Sender mit einer kleineren MTU sinkt, jedoch beim Empfänger steigt. Bei den UDP-Tests, wie das Bild 3.13 zeigt, tritt der umgekehrte Fall ein. Die Sender-CPU-Auslastung steigt bei kleinerer MTU, während die Empfänger-CPU-Auslastung sinkt. Die CPU-Auslastung bei den restlichen Messungen, wo für die MTU und die *Interrupt Coalescing*-Zeit die Standardwerte ausgewählt wurden, ist in Tabelle 3.10 zu sehen. Bis auf einige wenige Ausreißer befinden sich die Werte für die CPU-Auslastung in diesen Bereichen.

Tabelle 3.10: CPU-Auslastungsbereiche für alle Testarten mit einer MTU von 9000 und einer Interrupt Coalescing-Zeit von 75 µs – Angaben in %

	Sender	Empfänger
TCP Stream	50-50,20	39-42
TCP Sendfile	11-12,6	42,5-43,5
UDP Stream	50-50,10	50-50,10

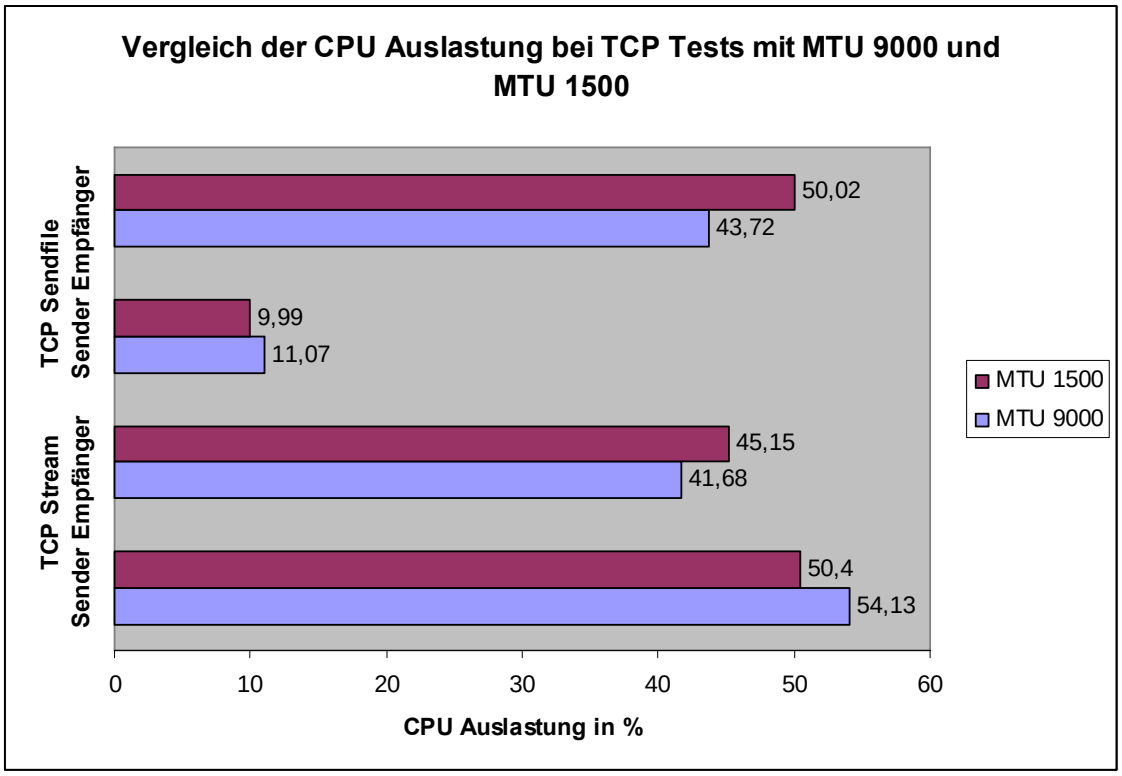

Bild 3.12: Vergleich der CPU Auslastung bei TCP Tests mit einer MTU von 9000 und einer MTU von 1500

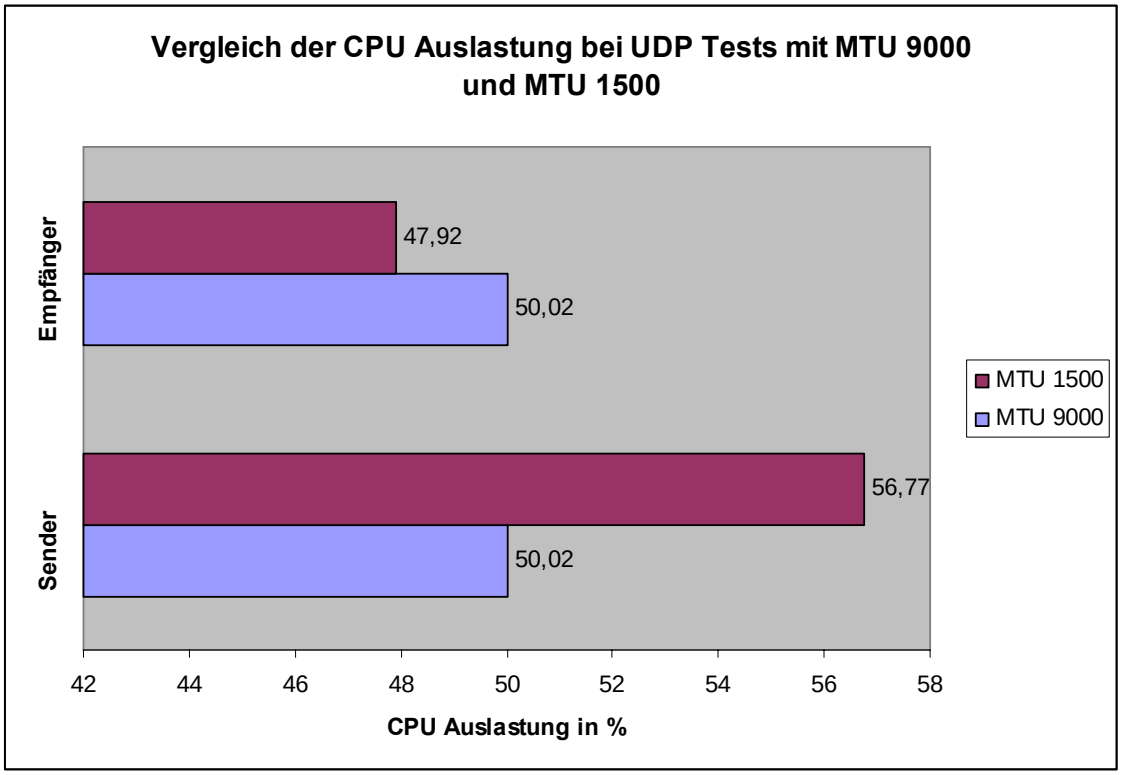

Bild 3.13: Vergleich der CPU Auslastung bei UDP Tests mit einer MTU von 9000 und einer MTU von 1500

3.3.7 Das optimale Ergebnis

Die oben dargestellten Ergebnisse führen zu folgender Schlussfolgerung. Das primäre Ziel ist eine hohe Datenrate bei UDP-Übertragungen. Daher wird die optimale Konfiguration primär nach den Ergebnissen der UDP-Messungen gerichtet. Das bedeutet die Standardwerte von *Interrupt Coalescing, Allocation Order* und MTU bleiben erhalten. Die *Timestamps*-Option wird ausgeschaltet und die *WCFiFo*-Option wird aktiviert. Mit den optimalen Einstellungen gehen immer noch ungefähr 5 % der Pakete bei UDP-Übertragungen verloren. Um den Verlust zu verringern, wurde die Linux-Kernel-Version 2.6.20.14 konfiguriert, compiliert und installiert. Bild 3.14 zeigt den Durchschnitt der Messungen der Datenrate bei UDP-Übertragungen. Bei der Übertragung gehen ungefähr 1 % der Pakete verloren.

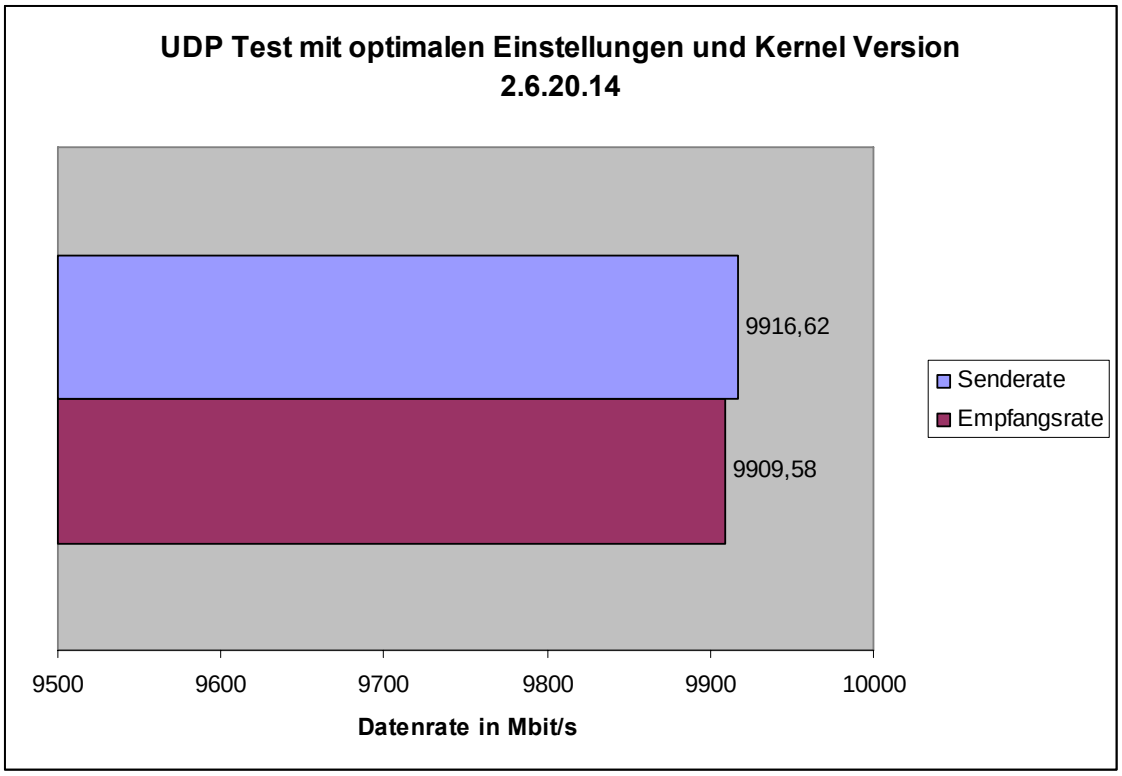

Bild 3.14: UDP Test mit optimalen Einstellungen und Linux-Kernel-Version 2.6.20.14

Mit dieser Kernel-Version wurden mit 9904,44 Mbit/s auch die besten Ergebnisse bei TCP-Stream-Messungen erzielt (siehe Tabelle 3.11). Die Datenrate bei TCP-Sendfile-Tests, die mit Linux-Kernel-Version 2.6.18.5 mit verschiedenen Einstellungen Höchstwerte von 9910 Mbit/s besaß, ist mit der neueren Kernel-Version um 1 % gesunken. Die Sender-CPU-Auslastung ist hierbei auch um mehr als 60 % gestiegen (vgl. Tabelle 3.10 und Tabelle 3.11). Insgesamt sind die Sender- und Empfänger-CPU-Auslastungen bei TCP-Tests angestiegen. Bei den UDP-Tests stellt sich eine CPU-Auslastung wie bei UDP-Messungen mit Kernel-Version 2.6.18.5 unter Benutzung einer MTU von 1500 ein (vgl. Bild 3.13 und Tabelle 3.11).

Tabelle 3.11: Mittelwerte der Datenrate und CPU Auslastungen bei TCP Stream, Sendfile und UDP Stream Messungen mit optimaler Einstellung und der neuern Linux-Kernel-Version 2.6.20.14

Testart	Seite	Datenrate/ Mbit/s	Sender-CPU-Auslastung/%	Empfänger-CPU-Auslastung/%
TCP Stream		9904,44	51,70	44,87
TCP Sendfile		9896,55	19,55	44,90
UDP Stream	Sender	9916,62	57,65	
	Empfänger	9909,58		46,79

3.3.8 UDP-Datenrate in Abhängigkeit der Sende- und Empfangspuffer

In diesem Abschnitt werden die Ergebnisse der Messungen zur Ermittlung der benötigten Sende- und Empfangspuffer dargestellt. Es wurden die optimalen Einstellungen, die im vorigen Kapitel ermittelt worden sind, übernommen. Im ersten Teil dieses Kapitels wird der benötigte Empfangspuffer gezeigt. Dafür wurde bei den Messungen mithilfe der ‚-S' Option des Programms Netperf (siehe 3.2.3) die Sende- und Empfangspuffer[8] auf der Server-Seite verändert. Der Sendepuffer hatte bei den Messungen einen konstanten Wert von 1000 kB.

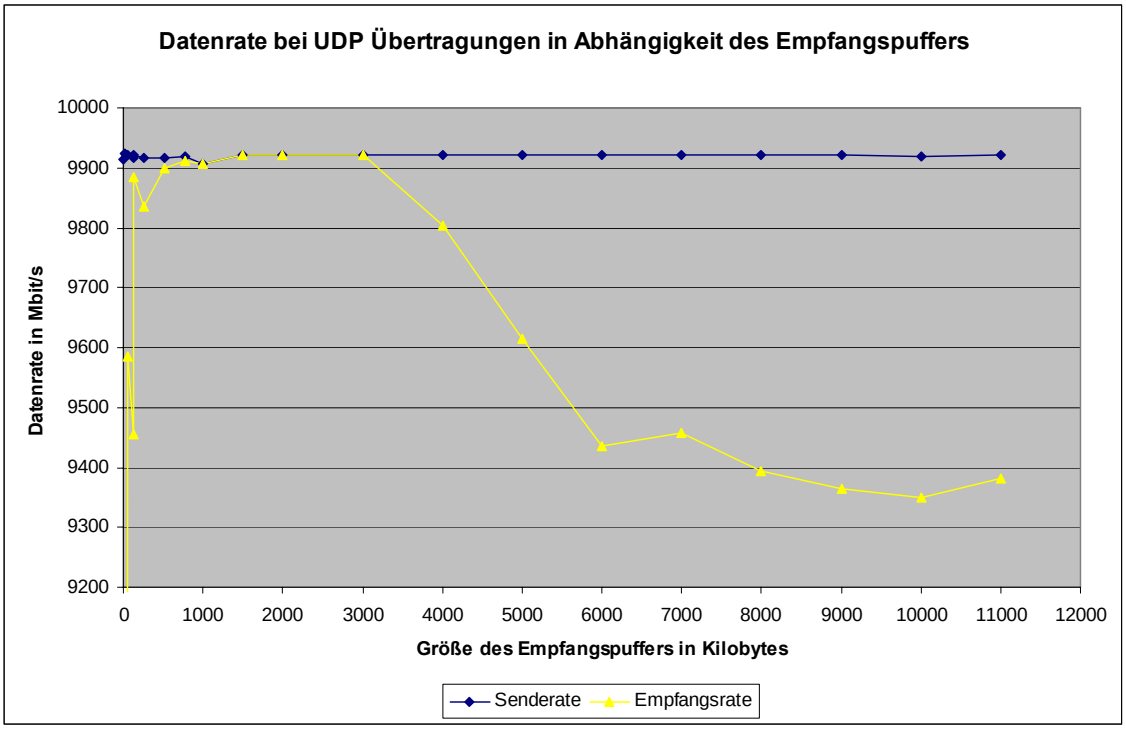

Bild 3.15: Datenrate bei UDP Übertragungen in Abhängigkeit des Empfangspuffers

Bild 3.15 zeigt einen Ausschnitt von den Ergebnissen, die bei den Versuchen zur Ermittlung des benötigten Empfangspuffers gemessen wurden. Der Tabelle 3.12 lassen sich die Messwerte entnehmen. Diese zeigen, dass ab einer Empfangspuffergröße von 1000 kB alle versendeten Pakete beim Empfänger ankommen. Der Bereich, in dem eine 100 %-Übertragung stattfindet, reicht bis zu einem Empfangspuffer von 3000 kB. Danach sinkt die Anzahl ankommender

[8] In dieser Arbeit wird als Sendepuffer der Sende- und Empfangspuffer der Client-Seite und als Empfangspuffer der Sende- und Empfangspuffer der Server-Seite bezeichnet. Bei einem Netperf-Test, bei dem die Daten von Client zu Server übertragen werden, ist hauptsächlich der Sendepuffer der Client-Seite und der Empfangspuffer der Server-Seite für die Datenrate maßgebend, daher die Bezeichnungen.

Pakete und die Datenrate pendelt sich in einen Bereich von 9300-9400 Mbit/s ein.

Tabelle 3.12: Datenraten bei UDP Übertragungen in Abhängigkeit des Empfangspuffers

Empfangspuffer/ Kilobytes	Datenrate am Sender/ Mbit/s	Datenrate am Empfänger/ Mbit/s
8	9914,5	945,8
16	9922,7	2829,4
32	9921,4	6312,8
64	9921,5	9584,5
126,876	9916,4	9455,9
128	9921,6	9885,6
256	9916	9836,8
512	9917,2	9900,1
768	9920	9912,4
1000	**9905,9**	**9905,9**
1500	9922,2	9922,2
2000	9922,4	9922,4
3000	9920,9	9920,9
4000	9920,9	9803,1
5000	9921,7	9615,3
6000	9921,2	9436,6
7000	9920,7	9457,8
8000	9920,9	9393,8
9000	9920,6	9364

Im zweiten Teil werden die Ergebnisse zur Ermittlung des benötigten Sendepuffers dargestellt. Die Messungen sollen zeigen, wie viel Puffer auf der Senderseite benötigt wird, um die mit den verwendeten Einstellungen höchstmögliche Datenrate zu erreichen. Dazu wurde mithilfe der ‚-s' Option des Programms Netperf (siehe 3.2.3) die Sende- und Empfangspuffergröße des Clients verändert. Der Empfangspuffer hatte bei den Messungen einen konstanten Wert von 1000 kB.

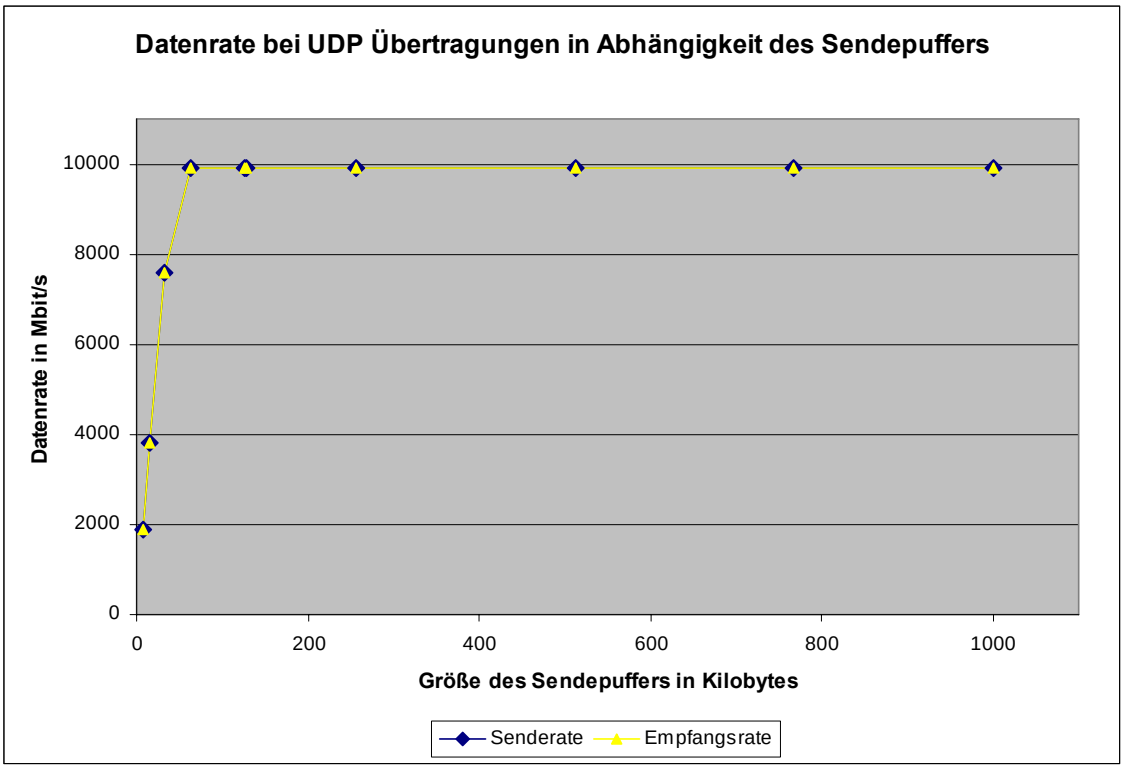

Bild 3.16: Datenrate bei UDP Übertragungen in Abhängigkeit des Sendepuffers

Auf Bild 3.16 kann man den Verlauf der Datenrate in Abhängigkeit des Sendepuffers sehen. Ab der vierten Messwertgruppe wird der Bereich der höchsten Datenraten auf der Sender -Seite erreicht. Es werden mindestens 64 kB Sendepuffer benötigt (siehe Tabelle 3.13). Bei den Messungen kommen annähernd alle versendeten Pakete am Empfänger an.

Tabelle 3.13: Datenraten bei UDP Übertragungen in Abhängigkeit des Sendepuffers

Sendespuffer/ Kilobytes	Datenrate am Sender/ Mbit/s	Datenrate am Empfänger/ Mbit/s
8	1899,7	1899,7
16	3798,4	3798,4
32	7590,3	7590,3
64	**9910,2**	**9910,1**
126,976	9907,7	9906,3
128	9917,3	9917,3
256	9916,8	9915,3
512	9923,3	9922,6
768	9923,6	9923,6
1000	9916,3	9916,3

3.3.9 UDP-Datenrate in Abhängigkeit der Nachrichtengröße

In diesem Abschnitt wird gezeigt, welches die optimale Größe der Nachrichten ist, die von den Applikationen an das UDP überreicht werden, damit möglichst alle versendeten Pakete den Empfänger erreichen. Dabei wurde die ‚-m'-Option des Programms Netperf (siehe 3.2.3) verändert. Es wurden die optimalen Einstellungen übernommen. Sendepuffer und Empfangspuffer wurden auf eine Größe von 1000 kB eingestellt.

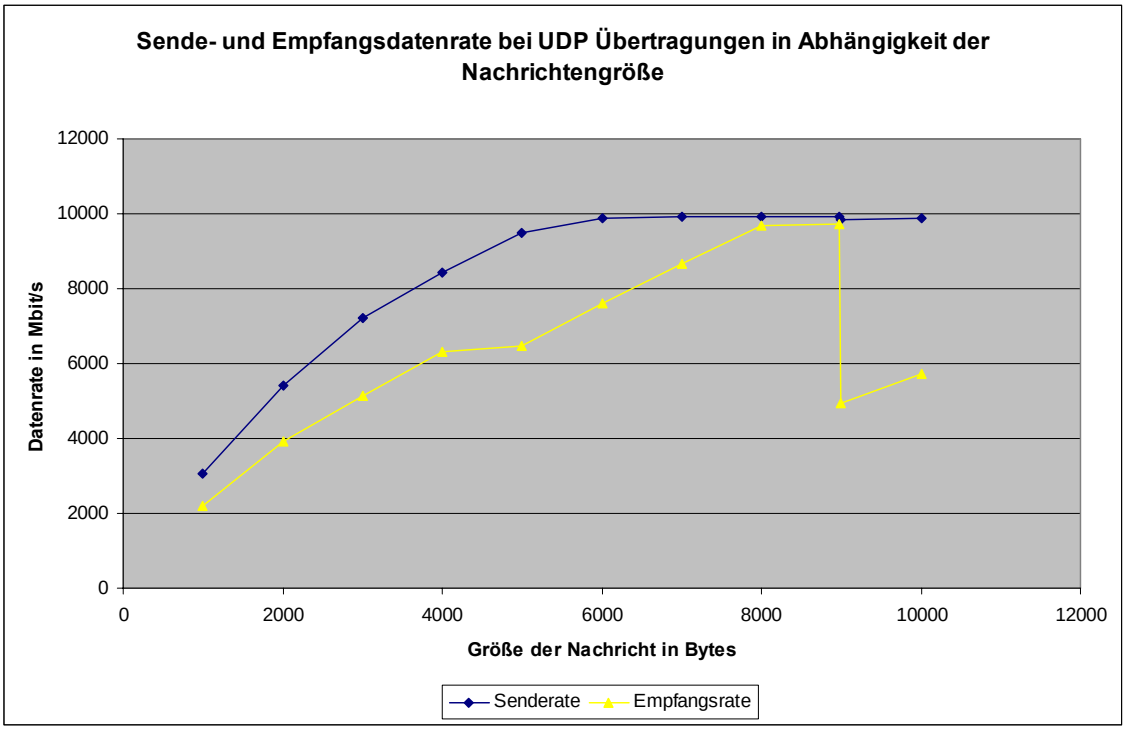

Bild 3.17: Sende- und Empfangsdatenrate bei UDP Übertragungen in Abhängigkeit der Nachrichtengröße

Aus Bild 3.17 kann man die Senderate und die Empfangsrate mit verschiedenen Nachrichtengrößen ablesen. Man erkennt, dass die Senderate mit der Nachrichtengröße steigt. Ab einer Größe von 6000 Bytes wird die optimale Datenrate auf der Sender-Seite erreicht. Bei einer Nachrichtengröße von 8972 Bytes ist die Datenrate sowohl auf der Sender-Seite als auch auf der Empfänger-Seite am größten. Wird die Größe der Daten, die UDP übergeben werden, um 28 Bytes größer als das Optimum, fällt die Anzahl der am Sender ankommenden Pakete um annähernde 50 % ab. Danach steigt die Empfangsrate wieder an.

3.4 Diskussion

Die zentrale Frage bei der Untersuchung der 10-Gbit-Übertragungstrecke war, ob mindestens 8 Gbit/s über eine Punkt-zu-Punkt-Verbindung mit 10-Gbit-Ethernet übertragen werden können und welche Einstellungen vorgenommen werden müssen, um dies zu bewerkstelligen. 8 Gbit/s schnell ist der Datenstrom, nachdem das zu übertragene Analogsignal von einem ADBoard digitalisiert wurde. Die Erfahrungen aus dieser Untersuchung sollen bei der Entwicklung des FiLa 10G Boards genutzt werden. Es soll die gleiche Datenrate bei einer Übertragungsstrecke erreicht werden, wo mindestens ein Endpunkt aus einem FiLa 10G Board besteht. Für die zukünftige optische Verbindung werden die Protokolle UDP, IP und Ethernet benutzt. Mit der optimalen Einstellung für UDP wurden Datenraten von über 9910 Mbit/s auf der Senderseite erreicht und auf der Empfängerseite waren es mindestens 99 % der Senderate.

3.4.1 Der Weg zur optimalen Konfiguration

Die ersten Versuche mit der Linux-Kernel-Version 2.6.16.13, die mit der Linux Distribution Suse 10 installiert wurde, führten zu keinen plausiblen Ergebnissen. Für verschiedene Konfigurationen wurden bei TCP-Messungen fortwährend nur ungefähr ein zehntausendstel der theoretischen Datenrate einer 10-Gigabit-Ethernet-Verbindung gemessen. Bei UDP-Messungen hatten verschiedene Einstellungen ebenfalls zu keiner Verbesserung der Ergebnisse geführt. Ein defektes Glasfaserkabel als Grund für die schlechten Datenraten konnte durch einen Austausch ausgeschlossen werden. Da die zu dem Zeitpunkt neuesten Gerätetreiber verwendet wurden, konnte dies ebenso nicht der Grund sein. Eine neue Linux-Kernel-Version brachte schließlich plausible Datenraten (siehe Tabelle 3.6-Tabelle 3.9).

Nach ersten Versuchen mit der neueren Linux-Kernel-Version in Standardkonfiguration (siehe Tabelle 3.4) wurden Versuche mit einer MTU von 9000 Bytes und von 1500 Bytes durchgeführt. Die Ergebnisse zeigten deutlich Leistungseinbußen bei der Benutzung einer kleineren MTU von 1500. 1500 Bytes ent-

spricht der MTU von Ethernet ohne QTAG-Präfix. Als Ethernet entwickelt wurde, wurden noch keine großen Datenmengen übertragen. Der Nachteil einer kleineren MTU liegt in der größeren Verarbeitung des Overheads durch den Host und durch die weniger effiziente Ausnutzung der Bandbreite des Netzwerkes durch erhöhten Overheadbedarf. Jeder Ethernet-Rahmen mit einer MTU von 1500 Bytes wird ein 18 Bytes großer Header angehängt. Zusätzlich sind innerhalb der Nutzdaten mindestens 28 Bytes Overhead für IP und UDP oder mindestens 40 Bytes bei der Nutzung von TCP vorhanden. Das bedeutet, dass das Verhältnis von Nutzdaten zu Overhead kleiner ist als bei einer MTU von 9000 Bytes. Dementsprechend muss mehr Bandbreite für den Overhead aufgewendet werden. Zusätzlich kommt der Mehraufwand bei der Verarbeitung der Header durch die Protokoll-Stacks und der Netzwerkkarten an den Endgeräten hinzu. Dies hat aber nicht in allen Fällen eine höhere CPU-Auslastung zur Folge. Aus Bild 3.12 kann man sehen, dass bei TCP-Übertragungen der Wechsel zu einer kleineren MTU zu einer geringeren CPU-Auslastung des Senders aber zu einer erhöhten des Empfängers führt. Die integrierte Flusssteuerung und die Zuverlässigkeit von TCP könnten unter anderem der Grund dafür sein. Der Host auf der Empfängerseite muss mehr Arbeit bei der Verarbeitung einer größeren Anzahl von Paketen leisten. Wenn der Empfänger überlastet ist und keine Pakete mehr verarbeiten kann, sendet er über die Flusssteuerung eine kleinere Fenstergröße zurück an den Sender. Mit der kleineren Fenstergröße weiß der Sender, dass er weniger Daten an den Empfänger senden muss. Dazu kommt das Pakete, die nicht vom Empfänger verarbeitet werden konnten, verworfen werden. Da der Sender keine Bestätigung für sein versendetes Paket erhält, schickt er das Paket noch einmal. Der Sender hat dadurch keinen Mehraufwand. Die Daten, die versendet werden sollen, befinden sich zu diesem Zeitpunkt immer noch im Sendepuffer. Damit verschickt der Sender weniger Nutzdaten an den Empfänger.

Bei UDP-Übertragungen tritt der umgekehrte Fall ein. In Bild 3.13 kann man sehen, dass der Sender durch die Benutzung einer MTU von 1500 Bytes mehr CPU-Ressourcen benötigt, während der Empfänger weniger verbraucht. Eine Dienstanforderung an UDP von höheren Schichten führt genau zu einem UDP-Datagramm. Damit muss UDP um den Faktor 6,095 mehr UDP-Datagramme erstellen, um die gleiche Menge Nutzdaten zu versenden wie bei einer MTU von 9000 Bytes. Die erhöhte Anzahl von Dienstaufrufen an UDP ist an der höheren CPU-Auslastung zu erkennen. Jedoch erreicht der UDP/IP-Protokoll-Stack des Senders etwa nur ein Drittel der Datenrate bei einer Übertragung mit den optimalen Einstellungen. Ein Grund dafür könnte an der Art der Nutzung des Sendepuffers durch den Myricom-Gerätetreiber liegen. Durch die erhöhte Anzahl an Zuweisungen von physikalisch zusammenhängendem Speicher für das Versenden von Paketen ist der Sendepuffer schneller überlastet als bei einer

geringeren Anzahl an Zuweisungen. Dadurch müssen Dienstanforderungen an UDP warten, was die Datenrate auf der Senderseite verringert. Auf der Empfängerseite hingegen kommen wie oben erklärt etwas nur ein Drittel der maximal möglichen Nutzdaten an. Die Anzahl der Pakete pro Nutzdaten ist jedoch etwa sechsmal größer, deswegen verringert sich die CPU-Auslastung des Empfängers nur um 2,1 % von 50,02 % auf 47,92 % (siehe Bild 3.13).

Bild 3.3 zeigt die Auswirkungen der *WCFiFo-* und der *Timestamps*-Optionen auf TCP-Stream- und TCP-Sendfile-Tests. Die Standardwerte sind für ersteres ein Wert von 0 und für letzteres ein Wert von 1. Die Deaktivierung der *Timestamps*-Option sorgt für eine Verbesserung der Datenrate bei TCP-Stream-Tests, während bei TCP-Sendfile-Test keine merklichen Beeinträchtigungen zu sehen sind. Bei Aktivierung wird in das *options*-Feld des TCP-Headers ein Zeitstempel an die Gegenseite mitgeschickt. Dadurch vergrößert sich der Overhead bei TCP-Paketen und deren Verarbeitung benötigt mehr Ressourcen. Letzteres ist nicht ausschlaggebend, die Messungen zeigten keine Ersparnis von CPU-Ressourcen. Die zusätzlichen Bytes im Overhead eines TCP-Pakets reichen jedoch aus, um die Datenrate merklich zu sinken.

Die Veränderung der *WCFiFo*-Option führt zu unterschiedlichen Ergebnissen bei TCP-Stream- und TCP-Sendfile-Tests. Wie in Kapitel 2.2.10 beschrieben, werden bei aktiviertem *Write Combining* mehrere Daten in eine Zwischenspeicherzeile geschrieben, um sie mit einer Bustransaktion zu übertragen. Bei der *WCFiFo*-Option wird das *First-in-First-out*-Prinzip angewendet. Dabei werden die ersten Daten, die in den Zwischenspeicher geschrieben wurden, auch als erstes wieder ausgelesen. TCP-Stream führt bei einem Speicherzugriff, um Daten an den Netzwerk-Socket zu schicken, Standardbefehle aus (siehe 2.2.5). Diese haben einen großen internen Kopieraufwand im Hauptspeicher zur Folge. Durch die *WCFiFo*-Option werden diese Vorgänge beschleunigt [vgl. 6]. Dies macht sich in der Datenrate bei TCP-Stream-Tests bemerkbar. TCP-Sendfile-Tests hingegen führen einen *sendfile*-Befehl aus. Dieser hat den *Zero-Copy*-Mechanismus implementiert, der dafür sorgt, dass die internen Kopiervorgänge eingespart werden können. Damit bleibt der Vorteil, der die *WCFiFo*-Option bringen kann, aus. Der TCP-Sendfile-Test bleibt jedoch auch nicht davon unberührt. Die Datenrate sinkt bei eingeschalteter *WCFiFo*-Option.

Bei UDP-Übertragungen kann die *Timestamps*-Option keinen Einfluss auf die Messungen haben, da sie eine TCP-Option ist. Aus Bild 3.4 ist dies zu erkennen. Über die Auswirkungen der *WCFiFo*-Option lässt sich keine Aussage machen. UDP-Stream-Versuche mit aktivierter *WCFiFo*-Option ergaben in einem Test eine Verbesserung und in einem anderem Test eine Verschlechterung.

Auf der Support-Seite der Firma Myricom wird empfohlen mit der *Allocation Order* zu experimentieren [6]. Man kann Werte von 0 bis 3 für diese Einstel-

lungsmöglichkeit wählen. Der Zahlenwert ist der Exponent zur Basis 2. Das Ergebnis wird mit 4 kB multipliziert und das Endprodukt ist die Größe des zusammenhängenden Speichers, der durch den Gerätetreiber als Empfangspuffer bereitgestellt wird. Bei einer *Allocation Order* von 0 wird dem Betriebsystem die Wahl der Größe überlassen. Standardmäßig ist dieser Wert eingerichtet und führt zu einer Empfangspuffer von 9 kB, was der Größe eines Jumbo-Rahmens entspricht. Bild 3.5 zeigt die Ergebnisse von TCP-Tests mit den verschiedenen *Allocation Order*-Werten. Die Standardeinstellung liefert die besten Ergebnisse. Auf der Webseite der Firma Myricom wird ein Wert von 2 empfohlen. Das führt nach der oben erklärten Berechnung zu einer Größe des zusammenhängenden Speichers von 16 kB. Mit dieser Einstellung werden die zweitbesten Ergebnisse erzielt. Der Vorteil einer größeren Zuweisung liegt in der Ersparnis von Speicherzuweisungsbefehlen und darin, dass ein großes Paket komplett auf einer Seite zusammenhängenden Speichers geschrieben werden kann. Da die Pakete maximal 9000 Bytes groß sind, bleibt dieser Vorteil aus. In einen Empfangspuffer mit 16 kB Größe passen ebenso nicht 2 Pakete hinein. Der Nachteil bei großen Zuweisungen ist, dass bei vielen Speicherzugriffen verschiedener Anwendungen eventuell nicht genügend große Seiten zusammenhängender Speicher für die Netzwerkkarte bereitgestellt werden können. Das führt zu einem insgesamt kleineren Empfangspuffer und dadurch zu Paketverlusten, was bei UDP und TCP eine Verringerung der Datenrate bedeutet. Mit einer *Allocation Order* von 1 werden 1 kB große Seiten zugewiesen. Folglich nimmt ein Jumbo-Rahmen 2 Seiten des Empfangspuffers ein. Diese Einstellung führt zu den schlechtesten Ergebnissen. Bei UDP-Messungen (siehe Bild 3.6) sind die gleichen Auswirkungen zu beobachten.

Mit der *Interrupt Coalescing*-Option sollen hauptsächlich die Zeiten für die Bearbeitung eines Interrupts durch den Host verringert werden. Dadurch bleiben der CPU mehr Ressourcen übrig, um andere Aufgaben auszuführen. In der Netzwerkanwendung muss ein Kompromiss zwischen Latenz und CPU-Auslastung gefunden werden. Die Standardeinstellung des Myricom-Gerätetreibers beträgt 75 µs. Das bedeutet für das erste Paket, welches in diesem Zeitabschnitt eintrifft, eine maximale Wartezeit bis zur Bearbeitung gerade von diesem Wert. Für die Host CPU ist es jedoch von Vorteil, da sie für alle in dieser Zeit eingetroffenen Pakete nur einen Interrupt auslösen muss. Dies kann man aus Bild 3.8 und Bild 3.9 ersehen. Dort führt eine Verringerung der *Interrupt Coalescing*-Zeit zu einer erhöhten CPU-Auslastung. Bei UDP-Messungen bleibt die CPU-Auslastung von dieser Option annähernd unberührt (siehe Bild 3.11). Mit der Veränderung der *Interrupt Coalescing*-Option verändert sich aber nicht nur die CPU Auslastung, sondern auch die Datenrate variiert. Mit kleineren Zeiten verringert sich ebenfalls der Durchsatz. Grund dafür ist die benötigte Zeit für die Bearbeitung der Interrupts. Die Pakete werden zwar von der Netz-

werkkarte schneller weiter verarbeitet, indem von der Hardware ein Interrupt ausgelöst wird, jedoch verbleiben die Daten in dem Empfangspuffer und warten darauf, dass sie von der Host CPU bearbeitet werden. Diese braucht aber nun mehr Ressourcen für die Behandlung der Interrupts. Die Netzwerkkarte empfängt weiter Pakete, die sie jedoch nicht an den Empfangspuffer schicken kann. Die Datenrate bei TCP-Sendfile-Messungen bleibt hingegen annähernd unverändert. Durch die Einsparung an CPU-Ressourcen wegen dem *Zero-Copy*-Mechanismus kann dieser Test die höhere Anforderung an die Host CPU kompensieren. Es ist deutlich in Bild 3.9. Die Veränderung der CPU-Auslastung ist bei TCP-Sendfile-Tests ist größer als bei TCP-Stream-Tests.

Diese Ergebnisse führten schließlich zu dem optimalen Ergebnis, welches in Kapitel 3.3.7 beschrieben wird.

3.4.2 Die Datenrate in Abhängigkeit der Puffer- und Nachrichtengröße

In Kapitel 3.3.8 wurden die Ergebnisse für die UDP-Datenrate in Abhängigkeit der Sende- und Empfangspuffer dargestellt. Ein entscheidender Faktor für die Datenrate bei einer UDP-Übertragung ist die Größe des Empfangspuffers. Geht man davon aus, dass der Sendepuffer mit der maximal möglichen Datenrate die Pakete versendet, wird die Datenrate anhand der ankommenden Pakete am Empfänger gemessen. Steht dem Empfänger nicht genügend Puffer zur Verfügung, kann dieser weiterhin eintreffende Pakete nicht bearbeiten und da UDP kein zuverlässiges Protokoll ist, werden die Daten unwiederbringlich verworfen. Aus Bild 3.15 und Tabelle 3.12 kann man sehen, dass eine Empfangspuffergröße von 1000 kB nötig ist, damit der Empfänger alle versendeten Pakete empfängt. Ab einer Puffergröße von 3000 kB sinkt die Anzahl der ankommenden Pakete und somit die Datenrate. Ein Grund könnten die internen Prozesse bei der Zuweisung des Empfangspuffers durch die Host CPU sein. Die Host CPU muss mehr Speicher für die Netzwerkkarte bereitstellen, was zu mehr Aufwand bei der Zuweisung von Speicher und schließlich bei den Lese- und Schreibprozessen führt.

In Bild 3.16 und Tabelle 3.13 werden die Ergebnisse der Messungen der UDP-Datenrate in Abhängigkeit des Sendepuffers dargestellt. Es lässt sich erkennen, dass im Gegensatz zum Empfangspuffer nur 64 kB benötigt werden, um die maximal mögliche Datenrate auf der Senderseite zu erreichen. In der Hauptanwendung des FiLa 10G Boards ist dieser nur als Sender von Daten gedacht, aber es lassen sich keine konkreten Aussagen über den Zusammenhang des benötigten Sendepuffers des FiLa 10 Boards und der Myri-10G Netzwerkkarte machen. Die TCP/IP- oder UDP/IP-Protokoll-Suite wird bei ersterem von einem Xilinx FPGA und bei letzterem von einem AMD Opteron mit einem NVIDIA nForce Professional 2200 Chipsatz behandelt. Das FiLa 10G Board mit einem Xi-

linx FPGA ist eine spezielle Anwendung, über deren interne Prozesse zum Versenden von Paketen erst mit der Untersuchung eines Prototyps Erfahrungen gesammelt werden können.

Die Datenrate bei UDP-Übertragungen hängt ebenfalls stark von der Nachrichtengröße, die von den Anwendungen an den Netzwerk-Socket übergeben werden, ab. Bei der Nutzung von UDP wird bei jedem Sendeaufruf ein UDP-Datagramm erzeugt, während bei TCP Nachrichten einer Anwendung zusammengefasst oder geteilt werden können. Das bedeutet für eine kleine Nachricht, die mit UDP versendet werden soll, ein komplettes UDP-Paket mit einem UDP- und IP-Header von 28 Bytes. Werden viele kleine Nachrichten versendet, ist das Verhältnis von Nutzdaten zu Overhead klein, was zu einer schlechten Ausnutzung der verfügbaren Bandbreite des Netzwerks führt. Aus Bild 3.17 ist der deutliche Anstieg der Datenrate mit der Annäherung an die optimale Nachrichtengröße zu sehen. Die optimale Nachrichtengröße bei UDP unter Nutzung von Jumbo-Rahmen, was eine MTU von 9000 Bytes bedeutet, beträgt 8972 Bytes. Das sind 28 Bytes weniger als die MTU. Diese 28 Bytes werden für den 20 Bytes großen IP- und für den 8 Bytes großen UDP-Header benötigt. Man kann ebenso erkennen, dass bei einer Nachrichtgröße von 9000 Bytes die Datenrate stark abfällt. Bei dieser Größe werden für jede Nachricht zwei UDP-Datagramme erstellt. Das erste ist 8972 Bytes groß. Das zweite ist folglich 28 Bytes groß. Das führt zu oben beschriebenem Problem der schlechten Ausnutzung der verfügbaren Bandbreite.

3.4.3 Vergleich der gemessenen Werte mit den Ergebnissen der Firma Myricom

Die Firma Myricom führte eine Reihe von Leistungstests ihrer 10-Gbit-Netzwerkkarten durch, deren Ergebnisse auf der Firmenwebseite veröffentlich sind [23]. Sie testeten ihre Netzwerkkarten mit einer großen Anzahl verschiedener Prozessoren, Motherboards und Betriebsystemen. Das Max-Planck-Institut nutzte die Ergebnisse der Firma Myricom, um Bauteile für eigene geeignete Endsysteme auszuwählen und schließlich aufzubauen. Tabelle 3.14 zeigt die maximalen Werte für TCP-Stream-, TCP-Sendfile- und UDP-Stream-Tests, die von Mitarbeitern der Firma Myricom gemessen wurden. Das Max-Planck-Institut verwendete für die Treiber und die Testarten annähernd die gleichen Einstellungen. Es wurde anstatt der Linux-Kernel-Version 2.6.20.14 mit der Ubuntu 7.04 Linux-Distribution eine Linux-Kernel-Version 2.6.20.14 mit Suse 10 benutzt. Über die genutzte Firmware-Version und der Nutzung der *WCFiFo*-Option sind keine Informationen auf deren Webseite vorhanden.

Tabelle 3.14: Ermittelte Werte der Firma Myricom mit einer vergleichbaren Übertragungsstrecke inklusive der Endgeräte [23]

Netperf Test	MTU/ Byte	Bandwidth/ Mbits/s	TX_CPU/ %	RX_CPU/ %
TCP_STREAM	9000	9910,67	10,74	9,07
TCP_SENDFILE	9000	9903,59	3,24	9,11
UDP_STREAM_TX	9000	9871,30	12,50	0
UDP_STREAM_RX	9000	9871,30	0	9,29

Die UDP-Stream-Tests des Max-Planck-Instituts für Radioastronomie ergaben bis zu 1 % bessere Ergebnisse, während sie bei TCP-Sendfile-Tests um den gleichen Prozentsatz geringer sind. Die TCP-Stream-Tests der Firma Myricom sind nur geringfügig besser. Eine erhebliche Leistungssteigerung ist jedoch bei der CPU-Auslastung zu sehen. Myricom nutzte einen Intel quad-core dual-processor 2,66 GHz Xeon X5355s mit einem Supermicro X7DB8 Motherboard. Das Max-Planck-Institut verwendete hingegen zwei AMD single-core Opteron Prozessoren mit 2,6 GHz Taktrate auf einem Tyan S2895 Motherboard. Die Taktraten der verschiedenen Prozessoren ist ähnlich, das bedeutet, dass die geringe CPU-Auslastung des Intel Systems nur auf den Unterschied der Technologien dieser Prozessoren und Motherboards zurückzuführen ist.

4 Das Backend

In diesem Kapitel sollen die Möglichkeiten für ein Backend untersucht werden. Für die konstante Übertragung von Daten mit einer Geschwindigkeit von 10 Gbit/s spielen das Verhalten und die Leistung der Rechnerarchitektur und der Netzwerkkarte eine wichtige Rolle. Im Allgemeinen müssen Daten zwischen der Netzwerkkarte und dem Arbeitsspeicher transferiert werden. Daher sind die Geschwindigkeiten der Arbeitsgänge und der Datenleitungen von der Netzwerkkarte, dem Arbeitsspeicher, dem Speicher-Bus, dem Chipsatz, dem I/O-Bus (in diesem Fall PCI-E) und natürlich der CPU von großer Bedeutung. Mit der Entwicklung von 10-Gigabit-Ethernet verlagert sich der Engpass bei einem Datenaustausch auf einer Netzwerkverbindung von ursprünglich der Netzwerkkarte zu den übrigen Elementen, die bei einer Netzwerkübertragung betroffen sind.

Bei der Speicherung auf nicht-flüchtigen Speichern, wie Festplatten, kommt der Weg vom Arbeitsspeicher über den Chipsatz, dem I/O-Bus (in diesem Fall IDE, SCSI oder SATA) und der Schreibgeschwindigkeit der Festplatte bei der Berechnung der kompletten Übertragungsgeschwindigkeit von Anwender zu Anwender dazu.

Nachfolgend steht eine kleine Einführung in *Very Long Baseline Interferometry* (VLBI), um das Backend in der Radioastronomie besser verstehen zu können.

4.1 Einführung in Very Long Baseline Interferometry (VLBI)

Very Long Baseline Interfermetry (VLBI) ist eine bestimmte Art von astronomischer Interferometrie, die in der Radioastronomie genutzt wird. Durch Interferometrie ist es möglich das Auflösungsvermögen eines Teleskops zu erhöhen. Als Auflösungsvermögen wird die Unterscheidbarkeit von zwei nahe beieinander liegenden Punkten bezeichnet. Die Auflösung steigt mit dem Durchmesser des Teleskops und der kürzer werdenden Wellenlänge der elektromagnetischen Strahlung. Ein konventionelles Radiointerferometer besteht aus zwei oder mehr Radioteleskopen, die über Kabel, Funk oder Lichtwellenleiter miteinander verbunden sind. Als Basislinie wird die Distanz zwischen den Teleskopen bezeichnet. Mit dem Abstand der Teleskope wird die Auflösung vergrößert.

VLBI bedeutet Radiointerferometrie mit sehr langen Basislinien. Dadurch wird eine sehr hohe Auflösung erreicht. Die Teleskope sind dabei über verschiedene Länder oder Kontinente verteilt. Die Daten, die von mehreren Radioteleskopen empfangen werden, werden mit einem Zeitstempel von einer Atomuhr verse-

hen und auf Festplatten gespeichert. Die Festplatten werden zu einem Ort gebracht und korreliert. Bild 4.1 zeigt ein Blockschaltbild eines VLBI-Systems. Man kann erkennen, dass eine Atomuhr (*Atomic Clock*) die Referenz für den Oszillator und den Zeitstempel ist. Nachdem die Daten formatiert wurden, werden sie auf mehrere Festplatten, die mit einem Raid-Controller verbunden sind, gespeichert. Ältere Systeme verwendeten anstatt Festplatten Magnetbänder (*Mag Tape*).

In Entwicklung ist das sogenannte e-VLBI. Die Abkürzung steht für electronic-VLBI. Hierbei werden die Daten nicht an den Teleskopen gespeichert, sondern über Lichtwellenleiter eines Forschungsnetzwerks direkt an einen Korrelator geschickt. Damit ist es möglich Echtzeitmessungen vorzunehmen, die den Forschungsprozess vereinfacht und beschleunigt. Das FiLa 10G Board soll in einer weiteren Ausbausstufe auch für diese Anwendung benutzt werden können.

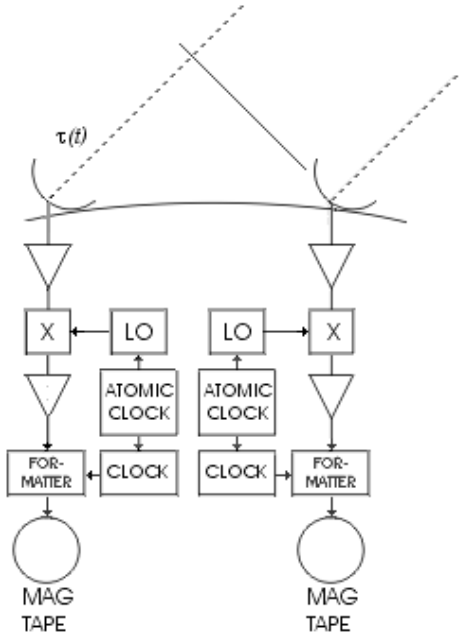

Bild 4.1: VLBI-System mit unabhängigen Atomaren Standard Oszillatoren [35]

Um möglichst hohe Auflösungen mit VLBI zu erhalten, müssen Radioteleskope in einer Messung benutzt werden, die weit von einander entfernt sind. Es werden Radioteleskope aus verschiedenen Ländern und Kontinenten verwendet. In Europa wurde dazu das Europäische-VLBI-Netzwerk (EVN) gegründet. Es wurde 1980 von den fünf größten Instituten für Radioastronomie in Europa, darunter das Max-Planck-Institut für Radioastronomie, und dem Institut für Geodäsie der Universität Bonn gegründet. Das EVN besteht derzeit aus 14 Instituten, wovon sich drei außerhalb Europas, in Südafrika, in der Volksrepublik China und in Puerto Rico, befinden. Es werden auch globale VLBI-Messungen mit Radioteleskopen in den USA und in Japan durchgeführt.

Im Mittelpunkt der Untersuchungen stehen die Zentren von aktiven galaktischen Kernen und deren Jets. Ein Schwerpunkt hierbei ist die bildliche Darstellung der direkten Umgebung der sog. „Kernmaschinen“ in aktiven Galaxienkernen. Man vermutet hier die sog. „supermassiven Schwarze Löcher“ [26].

VLBI-Messungen werden nicht nur für Untersuchungen in der Astronomie verwendet, sondern dienen auch für Messungen in der Geodäsie[9]. Durch die Messung des Zeitunterschieds zwischen der Ankunft der verschiedenen Signale an den Radioteleskopen ist es möglich die Rotation der Erde oder tektonische Plattenbewegungen im Zentimeterbereich zu vermessen [vgl. 21].

4.2 Diskussion über ein mögliches Backend

In diesem Projekt wird davon ausgegangen, dass die Sendeseite später das FiLa 10G Board ist, welches sich im digitalen Empfänger befindet und Daten in einer noch unbestimmten Form an den Kontrollraum sendet. Um einen Datenstrom mit der Geschwindigkeit von 10 Gbit/s in Echtzeit speichern zu können, müssen Raid-Kontroller und PC-Cluster eingesetzt werden. Die Kombination von beiden Systemen wird benötigt, da eine Technologie nicht ausreicht, ohne einen erheblichen Mehraufwand an Hardware oder Kosten in Kauf zu nehmen.

Die Art der Daten, die vom Empfänger versendet werden, ist noch unbestimmt. Das FiLa 10G Board wird nicht nur die HSI-Schnittstelle,sondern auch eine VSI-Schnittstelle besitzen. In erster Planung war der Einsatz des FiLa 10G Boards direkt hinter dem A/D-Wandler gedacht (siehe Bild 5.3). Ein Möglichkeit für das Backend ist wieder ein FiLa 10G Board in einem DBBC. Das soll die Daten entweder verarbeiten und z. B. VSI-gerecht formatieren und dann weiterleiten oder die Daten ohne Verarbeitung weiterleiten, z.B. vom Kontrollraum über eine weitere 10-Gbit-Übertragungs-strecke vom Radioteleskop in Effelsberg nach Bonn.

Eine andere Möglichkeit ist der Anschluss eines Switches mit einem 10 Gbit Uplink und entsprechendem Transceiver. Das Switch soll den Datenstrom auf mindestens 10 1-Gbit-Ports demultiplexen. An diesen Ports würden sich dann PCs befinden, die die Daten auf Festplatten speichern. Dies wäre ein PC-Cluster.

Diese PCs müssten Raid-Kontroller beinhalten. Jedoch selbst diese schaffen keine Schreibrate von 1Gbit/s. Daher müssten dementsprechend mehr PCs an das Switch angeschlossen werden, um die Daten in Echtzeit zu speichern.

---

[9] Geodäsie ist die Wissenschaft von der Ausmessung und Abbildung der Erdoberfläche.

Im Max-Planck-Institut für Radioastronomie steht ein Foundry Networks Edge Premium x448 zur Verfügung. Dies ist ein Switch mit passendem Transceiver am 10-Gbit-Uplink-Port. Es sollte überprüft werden, ob ein Versuch aufgebaut werden kann, der die Übertragungsrate von einer Quelle mit mehreren Senken, dem PC-Cluster, und die korrekte Übertragung der Daten testet.

Leider stehen dem Max-Planck-Institut zum Zeitpunkt der Arbeit nicht genügend PCs mit entsprechenden Raid-Kontrollern zur Verfügung, so dass dieser Versuch nicht aufgebaut werden kann. Hinzu kommt das Fehlen der Test-Software für den Test mit PC-Clustern. Würde man ein Programm verwenden wie Tsunami, welches FTP over UDP nutzt, um Daten von einer Festplatte über ein Netzwerk an ein anderen PC schickt, der diese wieder auf die Festplatte schreibt, müsste dieses Programm mindestens zehnmal gestartet werden. Jedes Programm adressiert dabei ein anderes Endsystem. Tsunami gibt nach der Übertragung die Übertragungsrate aus. Der Flaschenhals bei High-End-Rechnern wäre das Schreiben auf die Festplatte oder das Lesen von dieser. Darin involviert ist die Festplatte und der I/O-Bus. Das heißt die ausgegebene Übertragungsrate wäre die Schreibgeschwindigkeit auf die Festplatte.

Das Programm Netperf hingegen misst die Übertragungsgeschwindigkeit von Socket zu Socket, d. h. oberhalb der Schicht 4 des OSI-Referenzmodells. Damit würde man die reine Nutzdatenrate über das Kommunikationsnetz erhalten. Das Problem das Programm mindestens zehnmal starten zu müssen bleibt jedoch bestehen. Da das Programm mindestens zehnmal gestartet werden muss, laufen auf Anwendungsebene mindestens 10 Programme, die auf die CPU- und Netzwerkhardware-Ressourcen zugreifen. Leider kann keine Aussage darüber getroffen werden, welchem Programm wann und wie viele Ressourcen oder Dienste das Betriebssystem und die Protokolle der unteren Schichten den einzelnen Anwendungen zur Verfügung stellen. Möglicherweise wären die CPU und/oder die Netzwerkhardware überlastet, somit würden die Datenraten von einzelnen oder mehreren Testprogrammen sinken. Bei einem erneuten Versuch könnten die PCs andere Datenraten ausgeben.

5 Das FiLa 10G Board

Das Endziel des Projekts, zu welchem diese Diplomarbeit erstellt wurde, ist die Entwicklung des FiLa 10G Boards. Dieses Projekt ist eine Zusammenarbeit des Instituto di Radioastronomia in Noto auf Sizilien und des Max-Planck Institutes für Radioastronomie in Bonn. Das FiLa 10G Board soll eine flexible Schnittstellenkarte werden, dessen Hauptaufgabe es ist, Daten über eine optische 10-Gigabit-Ethernet-Übertragungsstrecke zu übermitteln. In erster Anwendung soll es die digitalisierten Daten vom Radioteleskop in Effelsberg zu dessen Kontrollraum transferieren. Diese Strecke ist ungefähr 350 m lang. Direkt am Radioteleskop soll sich ein digitaler Empfänger befinden. Dieser wird zurzeit noch vom Max-Planck-Institut entwickelt. Das FiLa 10G Board soll als Zusatzkarte in den digitalen Empfänger eingesteckt werden können. Über die optische Verbindung sollen die Daten dann in den Kontrollraum übertragen werden, wo sich ein entsprechendes Backend befindet. Dieses Backend kann aus einem FiLa 10G Board oder aus einem wie in Kapitel 3 aufgebauten Endsystem bestehen. Um die Verwendung des FiLa 10G Boards genauer verstehen zu können, werden nachfolgend der *Digital Baseband Converter* (DBBC), welcher vom Instituto di Radioastronomia entwickelt wurde, und der digitale Empfänger, welcher eine Entwicklung des italienischen Instituts und des Max-Planck-Instituts ist, einführend beschrieben. Das FiLa 10G Board soll als Zusatzkarte in diese beiden Systeme eingefügt werden können.

5.1 Einführung zum Digital Baseband Converter (DBBC)

Die Hauptidee hinter dem DBBC-Projekt ist die Ersetzung eines existierenden VLBI-Terminals durch ein komplettes und kompaktes System, welches mit jedem VLBI Standard Interface[10] (VSI) kompatiblen Recorder verwendet werden soll. Das DBBC stellt ein Backend System dar, welches analoge Hochfrequenz- oder Zwischenfrequenzsignale digitalisieren, die resultierenden digitalen Daten nach VSI formatieren und schließlich versenden kann. Das System basiert auf einer flexiblen Architektur, welches aus mindestens einem Analog-Digital-Board (ADBoard), einem Core-Board und einem FiLa-Board besteht. Die maximale Konfiguration besteht aus 4 ADBoards, 16 Core-Boards und 2 FiLa-Boards. Die Platinen können in paralleler Struktur kaskadiert werden [vgl. 32].

[10] VSI ist eine von mehreren Instituten der Radioastronomie entwickelte Standardschnittstelle für VLBI-Daten-Transmissionssysteme. Es soll bei der Datengewinnung und der folgenden Korrelation verwendet werden [3].

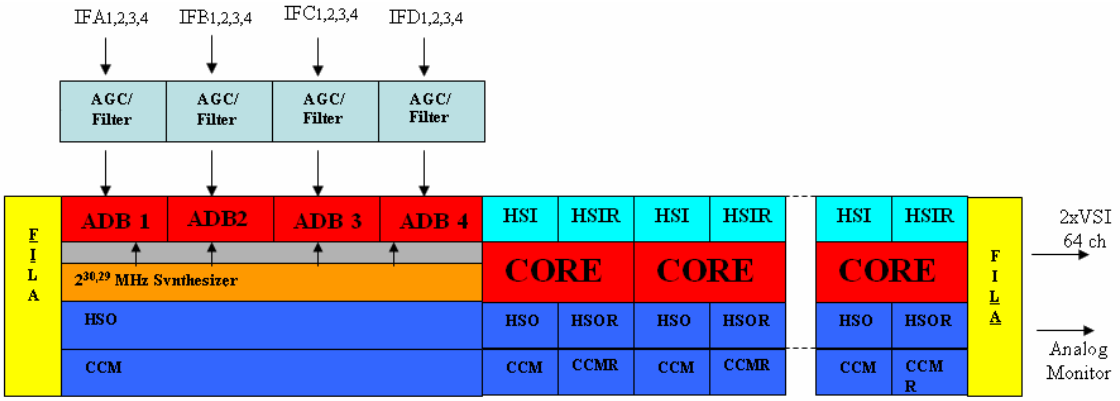

Bild 5.1: Blockschaltbild eines DBBC

Bild 5.1 zeigt das Blockschaltbild eines DBBC. Es können maximal vier Zwischenfrequenzsignale über *Automatic Gain Controls* (AGC, engl. für automatische Verstärkungsregelung) eingespeist werden. Auf den ADBoards befinden sich 8-bit-Analog/Digitalwandler, die mit einer Geschwindigkeit von 2^{30} Hz arbeiten. Mit den Core-Boards können je nach Bedarf verschiedene Anwendungen realisiert werden. Auf den Core-Boards befinden sich vollprogrammierbare *Field Programmable Gate Arrays* (FPGA). Die Daten werden parallel über *High Speed Input*-(HSI) und *High Speed Output*-(HSO)-Busse an die Coreboards weitergegeben. Zusätzlich befindet sich ein *Control/Configuration/Monitor*-(CCM)-Bus auf den Platinen. Das Ganze wird mit FiLa-Boards abgeschlossen. Das erste FiLa-Board, welches an das erste ADBoard angesteckt wird, besitzt Kommunikations- und Programmierschnittstellen und Kanäle zur Zeitsynchronisation und Taktgenerierung. Das zweite FiLa-Board, welches an das letzte Core-Board angeschlossen wird, besitzt zwei VSI-Stecker und einen Digital/AnalogWandler für den Anschluss eines Spektrumanalyzers für Testzwecke.

5.2 Einführung zum digitalen Empfänger

Der digitale Empfänger des Max-Planck-Instituts für Radioastronomie ist eine eigene Entwicklung mit Unterstützung des Instituto di Radioastronomia. Für den digitalen Empfänger werden das ADBoard, die FiLa-Boards und das Core-Board des DBBC verwendet. Im Gegensatz zum DBBC soll sich der digitale Empfänger jedoch nicht im Backend befinden, sondern direkt im Empfängerkopf des Radioteleskops. Die analogen Eingänge des Empfängers sind für zwei Frequenzbereiche, von 1200-1500 MHz und von 1500-1800 MHz, ausgelegt. Bild 5.2 zeigt die erste Ausbausstufe des digitalen Empfängers.

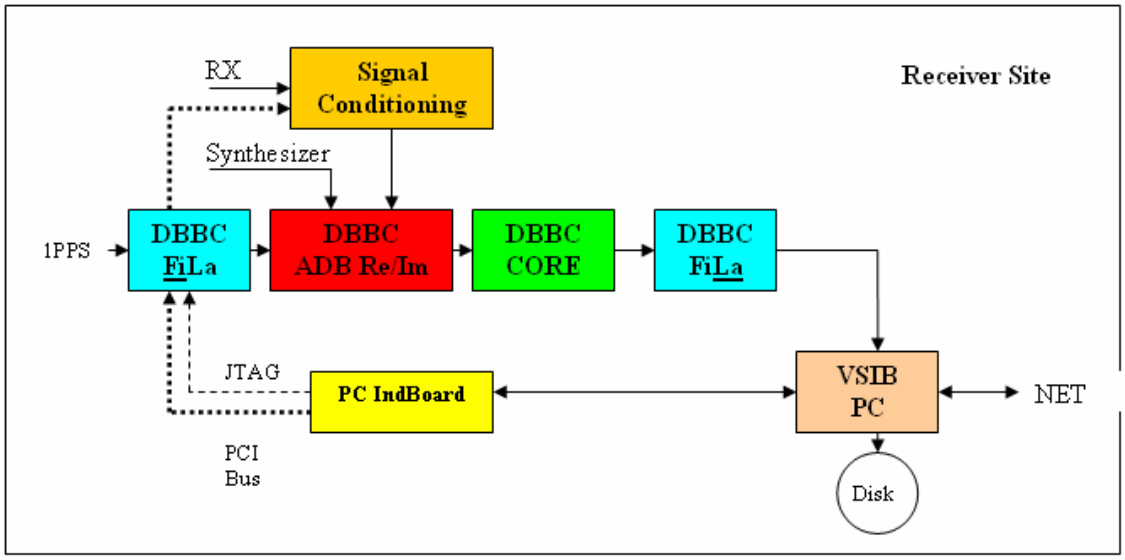

Bild 5.2: Blockschaltbild der ersten Ausbaustufe des digitalen Empfängers [33]

Die analogen Signale werden mit einem *Signal Conditioning*-Modul aufbereitet und dem Eingangslevel des ADBoards entsprechend angepasst. Das ADBoard erstellt, wie oben beschrieben, mit einer Geschwindigkeit von 2^{30} Hz eine 8-bit-Repräsentation des analogen Signals. Das digitale Signal wird dann an mindestens ein Core-Board weitergegeben. Das FPGA des Core-Board wurde für die spezielle Anwendung des digitalen Empfängers programmiert. Es reduziert den 8 Gbit/s schnellen Datenstrom auf 1 Gbit/s und formatiert es nach VSI. Dieser Datenstrom wird über ein FiLa Board mit VSI-Steckern an ein VSI-Board (VSIB), welches sich in einem handelsüblichen PC befindet, gesendet. Dieser PC versendet schließlich die Daten über eine 1-Gbit-Ethernet-Netzwerkkarte an das Backend. Die Daten werden mit einem *Pulse per second* (PPS) synchronisiert. Das *Joint Test Action Group* (JTAG) dient der Konfiguration der einzelnen Platinen.

5.3 Die Idee des FiLa 10G Boards

In der zweiten Ausbaustufe des digitalen Empfängers soll eine optische 10-Gbit-Ethernet-Übertragungsstrecke hinzugefügt werden, die die Rohdaten vom Empfänger direkt an ein entsprechendes Backend sendet. Als Rohdaten werden die digitalisierten Daten ohne Formatierung bezeichnet. Das bedeutet, dass das FiLa 10G Board als Sender direkt hinter dem ADBoard anzusiedeln ist.

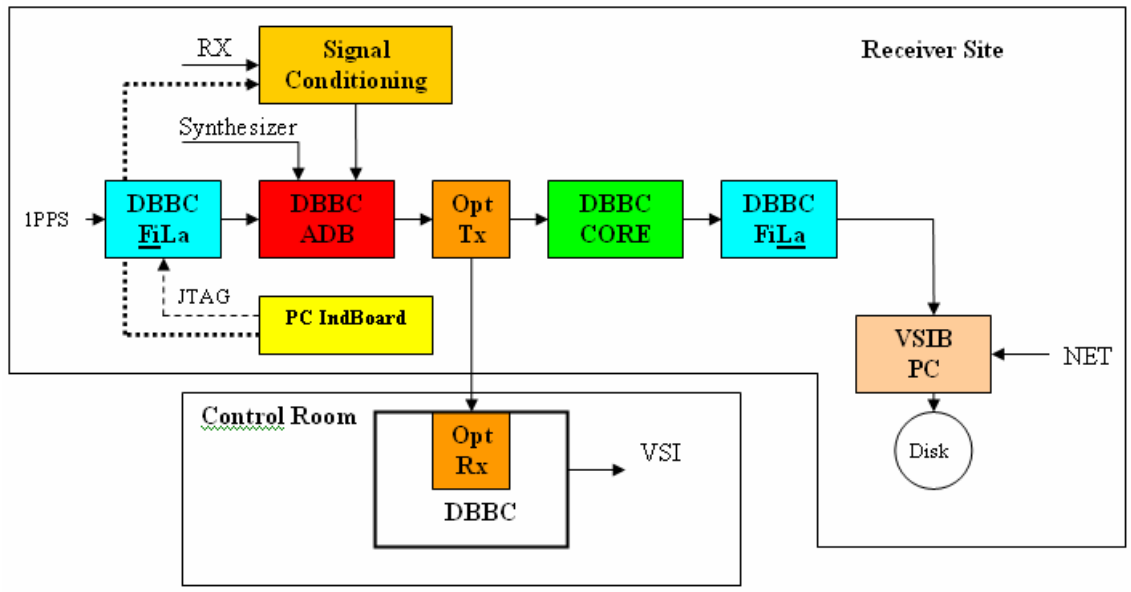

Bild 5.3: Blockschaltbild der zweiten Ausbaustufe des digitalen Empfängers [33]

Auf Bild 5.3 kann man die Lage der optischen 10-Gbit-Ethernet-Übertragungsstrecke sehen. Im Kontrollraum kann als Backend, als Empfänger, das DBBC mit einem FiLa 10G Board oder ein anderes entsprechendes Endsystem mit einem 10-Gbit-Ethernet Empfänger verwendet werden. Das FiLa 10G Board soll flexibel als eine aufsteckbare Zusatzkarte gestaltet werden. Es wird sich ein vollprogrammierbarer FPGA auf der Platine befinden, der für verschiedene Funktionen konfiguriert werden kann. Die Karte soll nicht nur an das ADBoard aufgesteckt werden können, sondern auch an die Coreboards. Bild 5.1 zeigt die möglichen Positionen des FiLa 10G Boards in einem DBBC. Da die gleichen Bauteile im digitalen Empfänger verwendet werden, kann dieses Bild repräsentativ für die Positionen im digitalen Empfänger genutzt werden. Das Bild zeigt nur die möglichen Lagen. Die Anzahl der FiLa 10G Boards, die benutzt werden sollen, ist nicht definiert.

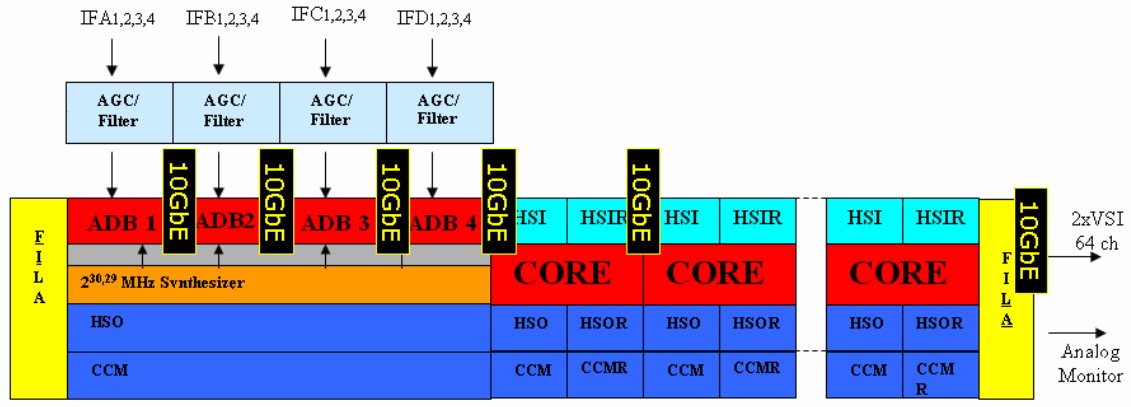

Bild 5.4: Blockschaltbild des DBBC als Bestandteil eines digitalen Empfängers mit möglichen Lagen des FiLa 10G Boards mit 10 GbE Schnittstelle (vgl. Bild 5.1)

Das FiLa 10G Board soll eine Schnittstellenkarte mit drei verschiedenen Schnittstellen werden, wovon die 10-Gbit-Ethernet-Schnittstelle die wichtigste ist (siehe Bild 5.5). Die Schnittstellen können für den Empfang und für den Versand miteinander kombiniert werden, daraus ergibt sich jeweils eine andere Funktionalität des FiLa 10G Boards. In ferner Zukunft kann eventuell diese Karte für Anwendungen des e-VLBIs genutzt werden. Mit einem 10GBase-ER (siehe 2.1.8) und der Nutzung von Repeatern ist es denkbar, eine 10-Gbit-Ethernet-Übertragungsstrecke von Effelsberg nach Bonn aufzubauen. Ein FiLa 10G Board könnte im Backend die Daten vom Radioteleskop empfangen. Das Backend könnte mit Coreboards dann falls nötig die Daten nach VSI formatieren und schließlich mit einem weiteren FiLa 10G Board an den Korrelator nach Bonn schicken.

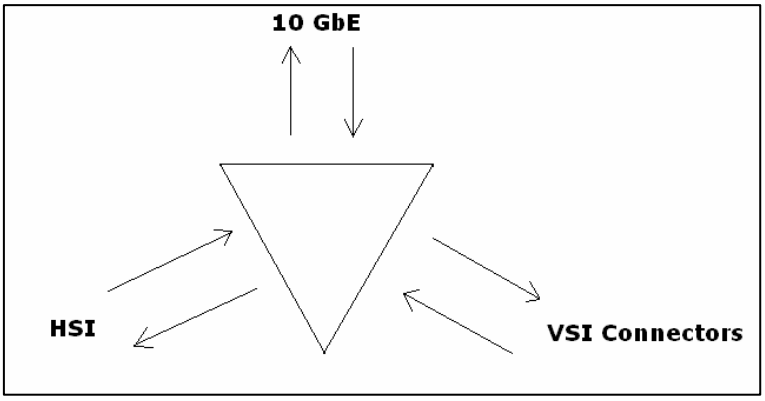

Bild 5.5: Schnittstellen des FiLa 10G Boards

6 Zusammenfassung und Ausblick

Der Kern dieser Diplomarbeit war die Untersuchung der optischen 10-Gigabit-Ethernet-Übertragungsstrecke. Das Ziel war das Ansammeln von Erfahrungen über die 10-Gigabit-Ethernet-Technologie mit ihren Schnittstellen, Protokollen und ihrer Hardware. Diese Erfahrungen sollen in naher Zukunft bei der Entwicklung des FiLa 10G Boards helfen. Außerdem soll die verfügbare Bandbreite, die diese neue Technologie bereitstellt, optimal ausgenutzt werden. Eine zentrale Frage bei den Untersuchungen war, welche Datenrate erreicht werden kann. Das Max-Planck-Institut für Radioastronomie möchte in der zweiten Ausbaustufe ihres digitalen Empfängers in diese eine optische 10-Gigabit-Ethernet-Übertragungsstrecke implementieren. Über die Punkt-zu-Punkt-Verbindung sollen mindestens 8 Gbit/s übertragen werden können. Die Untersuchungen der Testübertragungsstrecke zeigten, dass 9,9 Gbit/s mit TCP oder UDP möglich sind. Daraus folgt, dass das Max-Planck-Institut für Radioastronomie in Bonn und das *Instituto di Radioastronomia* in Noto weiterhin die Entwicklung des FiLa 10G Boards vorantreiben und in diese neue Technologie investieren.

Obwohl die 10-Gigabit-Ethernet-Technologie erst im Jahre 2002 standardisiert wurde und somit eine der neuesten und schnellsten Technologien in lokalen Netzwerken ist, wird sie schon bald in der Radioastronomie an ihre Grenzen stoßen. Die nächste Generation des digitalen Empfängers ist bereits in Entwicklung, welches ein Analogsignal vom Radioteleskop in einen 22 Gbit/s schnellen digitalen Datenstrom wandeln kann.

Für die Entwicklung des FiLa 10G Boards wird der Verfasser dieser Diplomarbeit für einige Monate beim Max-Planck-Institut für Radioastronomie festangestellt sein. In naher Zukunft sollen Untersuchungen mit einem 350 m langen Glasfaserkabel durchgeführt werden. Weiterhin wird er direkt mit der Entwicklung der Platine beauftragt, wo ihm die Erfahrungen, die er bis zu diesem Zeitpunkt gesammelt hat, von Vorteil sein werden. Bis zum Ende dieses Jahres soll ein Layout für einen Prototypen ausgearbeitet werden.

Literaturverzeichnis

[1] *10 Gigabit Ethernet Technology Overview White Paper.* Revision 2, Draft A. Gigabit Ethernet Alliance (www.10gea.org). April 2002

[2] *10 Gigabit Small Form Factor Pluggable Module*. Rev. 4.5. SFF Committee. www.xfpmsa.org. Saratoga, USA. August 2005

[3] **Cannon, Wayne et al.** *VLBI Standard Interface (VSI).* IVS Technology Report. www.haystack.edu/tech/vlbi/vsi/docs/vsi_vugraphs.html. August 2000

[4] **Craver, Thomas R.:** *Hyper-Threading Technology and Write Combining Store Buffers – Understanding, Detecting and Correcting Performance Issues.* www.intel.com. Intel Corporation.

[5] *Ethernet Device Driver Software Interface for Myricom Myri-10G Network Interface Cards.* www.myri.com. Myricom, Inc. Mai 2007

[6] *Frequently Asked Questions*. www.myricom.com. Myricom, Inc. July 2007

[7] **Geoffray, Patrick; Prylli, Loic**: *Myrinet Hardware and Software Overview.* www.myri.com. Myricom, Inc. 2005

[8] *Glasfaser*. http://www.elektronik-kompendium.de/sites/kom/1008271.htm

[9] **Grossman, Leonid:** *Large Receive Offload implementation in Neterion 10GbE Ethernet Driver.* Proceedings of the Linux Symposium. Volume One. Ottawa, Ontario. Canada July 20nd-23th, 2005

[10] **Hertweck, Thomas.** *Konfigurieren, Compilieren und Installieren eines Kernels 2.6 unter (open)SUSE Linux.* Version 0.8. www.thomashertweck.de. August 2006

[11] **Holtkamp, Heiko:** *Einführung in TCP/IP*. www.rvs.uni-bielefeld.de/~heiko/tcpip. Universität Bielefeld. Technische Fakultät. Februar 2002

[12] *IEEE STD 802.3-2005 Section Five.* The Institute of Electrical and Electronics Engineers, Inc. New York, USA. Dezember 2005

[13] *IEEE STD 802.3-2005 Section Four.* The Institute of Electrical and Electronics Engineers, Inc. New York, USA. Dezember 2005

[14] *IEEE STD 802.3-2005 Section Three.* The Institute of Electrical and

Electronics Engineers, Inc. New York, USA. Dezember 2005

[15] *IEEE STD 802.3-2005 Section Two.* The Institute of Electrical and Electronics Engineers, Inc. New York, USA. Dezember 2005

[16] *IEEE STD 802.3-2005.* The Institute of Electrical and Electronics Engineers, Inc. New York, USA. Dezember 2005

[17] *Intel® TXN18107 Product Brief.* www.intel.com. Intel Corporation

[18] **Jones, Rick:** *Care and Feeding of Netperf. Versions 2.4.3 and Later.* www.netperf.org. Hewlett-Packard Company. 2005-2007

[19] **Keller, Reinhard**: *Aufbau und Untersuchung einer 10Gbit/s Glasfaser-übertragungsstrecke zur Übertragung digitalisierter Analogsignale mit hohen Bandbreiten.* Max-Planck-Institut für Radioastronomie. Bonn 2007

[20] **Kurose, James F.;Ross, Keith W.:** *3.3 Connectionless Transport: UDP*. www-net.cs.umass.edu/kurose/transport/UDP.html. 1996-2000

[21] *Max-Planck-Institut für Radioastronomie.* http://www.mpifr-bonn.mpg.de.

[22] **Messmer, Hans-Peter:** *PC Hardwarebuch: Aufgaben, Funktionsweise, Programmierung.* 6. Aufl. München : Addison-Wesley Verlag, 2000

[23] *Myri-10G 10-Gigabit Ethernet Performance Measurements.* http://www.myricom.com/scs/performance/Myri10GE/. Myricom, Inc. August 2007.

[24] *Netzwerkkarte.* ZDNet Glossar > Netzwerke > Weitverkehrsnetze > Netzwerkkomponenten. www.zdnet.de. DATACOM Buchverlag GmbH

[25] **Orlamünder, Harald:** *Paket-basierte Kommunikationsprotokolle: OTH, Ethernet, RPR, GFP, IPv4, IPv6 und andere.* Landsberg : Hüthig Verlag, 2005

[26] *Radio-Interferometrie/VLBI – Radio Kontinuum.* http://www.mpifr.de/forschungsgruppen/antonZensus/index.html. Max-Planck Institut für Radioastronomie. Bonn. 2007

[27] **Siegmund, Gerd**: *Technik der Netze: 4., neubearbeitete und erweiterte Aufl.* Heidelberg : Hüthig Verlag, 1999

[28] **Stancevic, Dragan:** *Zero Copy I: User-Mode Perspective.* www.linuxjournal.com/article/6345. January 2003

[29] **Stevens, W. R.: TCP/IP:** *Der Klassiker: Protokollanalysen: Aufgaben*

und Lösungen. Landsberg : Hüthig Verlag, 2004

[30] *The European VLBI Network.* www.evlbi.org.

[31] **Tuccari, G. ; Keller R. ; Winkelmann P. u. a.:** *L-Band Digital Receiver*

[32] **Tuccari, Gino.** *DBBC – A Flexible Platform for VLBI Data Process.* Instituto di Radioastronomia . Noto, Italien

[33] **Tuccari, Gino:** *L Band Digital Receiver*

[34] **Valenzuela, Alejandro.** *Übertragungstechnik und Kommunikationsnetze ET 4. Semester (K+M).* FH-Bonn-Rhein-Sieg. Sommersemester 2005

[35] *Very Long Baseline Interferometry.* http://www.mpifr-bonn.mpg.de/div/vlbicor/index_e.html. Max-Planck Institut für Radioastronomie. Bonn

[36] *Write-Combining Memory in Video Miniport Drivers.* www.microsoft.com. Microsoft Corporation. November 2005

[37] www.wikipedia.org

Anhang

A.1 Messergebnisse

3. Testdurchlauf (Tabelle A.1-1, Tabelle A.1-2)
WCFiFo = 1; MTU = 9000; IRQ-coalescing = 75; Allocation Order = 0;Ts[11] = 1

Tabelle A.1-1: 3. Testdurchlauf – TCP-Tests

Testnummer	Datenrate/ Mbit/s	Sender-CPU-Auslastung/%	Empfänger-CPU-Auslastung/%
TCP-STREAM			
1	9753,77	50,05	43,46
2	9735,67	50,19	43,03
3	9737,62	50,02	42,99
4	9663,01	50,01	42,77
5	9471,31	50,02	37,99
Durchschnitt	**9672,28**	**50,06**	**42,05**
TCP-SENDFILE			
1	9730,38	10,96	43,05
2	9728,83	11,03	43,23
3	9729,90	11,43	42,98
4	9729,49	10,96	43,16
5	9731,25	11,23	43,65
Durchschnitt	**9729,97**	**11,12**	**43,21**

[11] Ts steht für *Timestamps*.

Tabelle A.1-2: 3. Testdurchlauf – UDP-STREAM-Test

Testnummer	Seite	Datenrate/ Mbit/s	Sender-CPU-Auslastung/%	Empfänger-CPU-Auslastung/%
1	Sender	9914,00	50,02	
	Empfänger	8957,40		50,07
2	Sender	9916,50	50,02	
	Empfänger	8869,50		50,07
3	Sender	9914,30	50,01	
	Empfänger	8851,80		50,07
4	Sender	9918,60	50,03	
	Empfänger	8840,10		50,07
5	Sender	9914,50	50,03	
	Empfänger	8895,40		50,06
Durchschnitt	Sender	**9915,58**	**50,02**	
	Empfänger	**8882,84**		**50,07**

4. Testdurchlauf (Tabelle A.1-3, Tabelle A.1-4)
WCFiFo = 1; **MTU = 1500**; IRQ-coalescing = 75; Allocation Order = 0;Ts = 1

Tabelle A.1-3: 4. Testdurchlauf – TCP-Tests

Testnummer	Datenrate/ Mbit/s	Sender-CPU-Auslastung/%	Empfänger-CPU-Auslastung/%
TCP-STREAM			
1	7203,73	49,61	44,00
2	7075,33	50,03	42,48
3	7768,29	46,29	49,99
4	7199,53	49,40	44,05
5	7487,15	48,31	46,82
Durchschnitt	**7346,81**	**48,73**	**45,47**
TCP-SENDFILE			
1	7793,33	9,08	50,00
2	7769,39	9,34	49,99
3	7778,07	8,81	49,99
4	7792,25	8,80	49,99
5	7797,15	8,75	50,01
Durchschnitt	**7786,04**	**8,96**	**50,00**

Tabelle A.1-4: 4. Testdurchlauf – UDP-STREAM-Test

Testnummer	Seite	Datenrate/ Mbit/s	Sender-CPU-Auslastung/%	Empfänger-CPU-Auslastung/%
1	Sender	3333,90	50,06	
	Empfänger	3139,10		49,02
2	Sender	3355,90	50,05	
	Empfänger	3126,10		49,58
3	Sender	3115,00	62,46	
	Empfänger	3092,50		42,89
4	Sender	3242,80	56,31	
	Empfänger	3071,40		46,71
5	Sender	3087,80	63,20	
	Empfänger	3087,80		42,65
Durchschnitt	Sender	**3227,08**	**56,42**	
	Empfänger	**3103,38**		**46,17**

5. Testdurchlauf (Tabelle A.1-5, Tabelle A.1-6)
WCFiFo = 0; MTU = 9000; IRQ-coalescing = 75; Allocation Order = 0;**Ts = 0**

Tabelle A.1-5: 5. Testdurchlauf – TCP-Tests

Testnummer	Datenrate/ Mbit/s	Sender-CPU-Auslastung/%	Empfänger-CPU-Auslastung/%
TCP-STREAM			
1	9624,79	50,04	42,27
2	9774,02	50,01	42,99
3	9734,91	50,01	39,52
4	9800,35	50,01	42,63
5	9754,80	50,03	42,54
Durchschnitt	**9737,77**	**50,02**	**41,99**
TCP-SENDFILE			
1	9910,16	11,35	43,35
2	9910,20	11,58	43,74
3	9910,14	11,51	43,43
4	9910,50	11,65	43,56
5	9910,78	11,43	43,76
Durchschnitt	**9910,36**	**11,50**	**43,57**

Tabelle A.1-6: 5. Testdurchlauf – UDP-STREAM-Test

Testnummer	Seite	Datenrate/ Mbit/s	Sender-CPU-Auslastung/%	Empfänger-CPU-Auslastung/%
1	Sender	9915,10	50,02	
	Empfänger	9216,70		50,10
2	Sender	9916,60	50,02	
	Empfänger	9249,50		50,06
3	Sender	9916,60	50,01	
	Empfänger	9168,40		50,06
4	Sender	9918,30	50,01	
	Empfänger	9234,10		49,99
5	Sender	9916,30	50,03	
	Empfänger	9270,60		49,87
Durchschnitt	Sender	**9916,58**	**50,02**	
	Empfänger	**9227,86**		**50,02**

6. Testdurchlauf (Tabelle A.1-7, Tabelle A.1-8)

WCFiFo = 1; MTU = 9000; IRQ-coalescing = 75; Allocation Order = 0;**Ts = 0**

Tabelle A.1-7: 6. Testdurchlauf – TCP-Tests

Testnummer	Datenrate/ Mbit/s	Sender-CPU-Auslastung/%	Empfänger-CPU-Auslastung/%
TCP-STREAM			
1	9760,64	50,50	42,09
2	9802,03	50,03	42,54
3	9822,30	50,01	42,45
4	9790,23	50,02	43,01
5	9840,76	50,04	42,84
Durchschnitt	**9803,19**	**50,12**	**42,59**
TCP-SENDFILE			
1	9728,22	11,11	41,95
2	9729,63	10,84	41,75
3	9727,66	11,09	42,10
4	9728,42	11,04	42,92
5	9729,11	11,43	43,91
Durchschnitt	**9728,61**	**11,10**	**42,53**

Tabelle A.1-8: 6. Testdurchlauf – UDP-STREAM-Test

Testnummer	Seite	Datenrate/ Mbit/s	Sender-CPU-Auslastung/%	Empfänger-CPU-Auslastung/%
1	Sender	9918,50	50,03	
	Empfänger	9433,00		49,90
2	Sender	9917,60	50,02	
	Empfänger	9497,40		49,65
3	Sender	9916,10	50,01	
	Empfänger	9476,60		49,84
4	Sender	9918,50	50,01	
	Empfänger	9487,10		48,99
5	Sender	9916,80	50,03	
	Empfänger	9501,20		49,91
Durchschnitt	Sender	**9917,50**	**50,02**	
	Empfänger	**9479,06**		**49,66**

7. Testdurchlauf (Tabelle A.1-9Tabelle A.1-10)

WCFiFo = 0; MTU = 9000; IRQ-coalescing = 75; **Allocation Order = 2**;**Ts =0**

Tabelle A.1-9: 7. Testdurchlauf – TCP-Tests

Testnummer	Datenrate/ Mbit/s	Sender-CPU-Auslastung/%	Empfänger-CPU-Auslastung/%
TCP-STREAM			
1	9672,43	50,06	43,18
2	9722,63	50,04	42,79
3	9561,79	50,01	42,48
4	9575,02	50,03	41,66
5	9250,03	50,02	41,07
Durchschnitt	**9556,38**	**50,03**	**42,24**
TCP-SENDFILE			
1	9908,64	12,86	43,98
2	9910,35	12,63	44,30
3	9910,44	12,44	44,36
4	9908,29	12,96	44,34
5	9910,17	12,39	43,71
Durchschnitt	**9909,58**	**12,66**	**44,14**

Tabelle A.1-10: 7. Testdurchlauf – UDP-STREAM-Test

Testnummer	Seite	Datenrate/ Mbit/s	Sender-CPU-Auslastung/%	Empfänger-CPU-Auslastung/%
1	Sender	9921,20	50,04	
	Empfänger	9148,40		50,03
2	Sender	9914,90	50,06	
	Empfänger	8859,90		50,07
3	Sender	9913,40	50,02	
	Empfänger	8932,20		50,07
4	Sender	9915,30	50,07	
	Empfänger	9069,60		49,99
5	Sender	9861,10	50,02	
	Empfänger	8946,90		50,08
Durchschnitt	Sender	**9905,18**	**50,04**	
	Empfänger	**8991,40**		**50,05**

8. Testdurchlauf (Tabelle A.1-11, Tabelle A.1-12)
WCFiFo = 0; MTU = 9000; IRQ-coalescing = 75; **Allocation Order = 3**;**Ts = 0**

Tabelle A.1-11: 8. Testdurchlauf – TCP-Tests

Testnummer	Datenrate/ Mbit/s	Sender-CPU-Auslastung/%	Empfänger-CPU-Auslastung/%
TCP-STREAM			
1	9171,42	50,06	40,96
2	8898,76	50,25	37,71
3	8744,49	63,13	38,10
4	9150,19	50,03	40,55
5	9269,69	50,02	41,47
Durchschnitt	**9046,91**	**52,70**	**39,76**
TCP-SENDFILE			
1	9910,42	12,19	44,12
2	9909,08	12,31	44,24
3	9909,41	12,63	43,95
4	9910,30	12,33	44,47
5	9910,25	12,13	44,17
Durchschnitt	**9909,89**	**12,32**	**44,19**

Tabelle A.1-12: 8. Testdurchlauf – UDP-STREAM-Test

Testnummer	Seite	Datenrate/ Mbit/s	Sender-CPU-Auslastung/%	Empfänger-CPU-Auslastung/%
1	Sender	9919,60	59,89	
	Empfänger	8650,10		50,10
2	Sender	9918,00	52,46	
	Empfänger	8757,70		50,06
3	Sender	9820,60	60,09	
	Empfänger	8832,40		50,04
4	Sender	9500,00	59,65	
	Empfänger	9016,40		49,76
5	Sender	9629,40	59,55	
	Empfänger	8874,10		50,03
Durchschnitt	Sender	**9757,52**	**58,33**	
	Empfänger	**8826,14**		**50,00**

9. Testdurchlauf (Tabelle A.1-13, Tabelle A.1-14)

WCFiFo = 0; MTU = 9000; IRQ-coalescing = 75; **Allocation Order = 1**;**Ts = 0**

Tabelle A.1-13: 9. Testdurchlauf – TCP-Tests

Testnummer	Datenrate/ Mbit/s	Sender-CPU-Auslastung/%	Empfänger-CPU-Auslastung/%
TCP-STREAM			
1	8987,65	50,06	39,11
2	8682,39	56,20	37,90
3	8774,97	56,64	38,41
4	8903,97	50,03	39,32
5	8813,96	54,00	38,62
Durchschnitt	**8832,59**	**53,39**	**38,67**
TCP-SENDFILE			
1	9906,79	12,39	43,81
2	9910,30	12,24	43,56
3	9910,44	12,36	43,98
4	9910,37	12,18	43,76
5	9910,10	12,18	43,49
Durchschnitt	**9909,60**	**12,27**	**43,72**

Tabelle A.1-14: 9. Testdurchlauf – UDP-STREAM-Test

Testnummer	Seite	Datenrate/ Mbit/s	Sender-CPU-Auslastung/%	Empfänger-CPU-Auslastung/%
1	Sender	9917,30	50,04	
	Empfänger	8688,30		50,03
2	Sender	9918,00	50,01	
	Empfänger	8673,20		50,00
3	Sender	9910,10	50,04	
	Empfänger	8665,90		50,22
4	Sender	9886,90	50,02	
	Empfänger	8518,60		50,02
5	Sender	9565,20	59,73	
	Empfänger	8738,30		49,94
Durchschnitt	Sender	**9839,50**	**51,97**	
	Empfänger	**8656,86**		**50,04**

10. Testdurchlauf (Tabelle A.1-15, Tabelle A.1-16)
WCFiFo = 0; MTU = 9000; **IRQ-coalescing = 25**; Allocation Order = 0;**Ts = 0**

Tabelle A.1-15: 10. Testdurchlauf – TCP-Tests

Testnummer	Datenrate/ Mbit/s	Sender-CPU-Auslastung/%	Empfänger-CPU-Auslastung/%
TCP-STREAM			
1	8720,10	50,16	45,73
2	8448,07	65,89	44,79
3	8417,64	66,15	44,68
4	8607,34	53,04	45,47
5	8527,15	50,03	45,46
Durchschnitt	**8544,06**	**57,05**	**45,23**
TCP-SENDFILE			
1	9909,93	50,06	49,07
2	9909,78	50,05	48,99
3	9909,41	18,98	49,05
4	9909,54	49,99	49,36
5	9908,21	50,07	49,57
Durchschnitt	**9909,37**	**43,83**	**49,21**

Tabelle A.1-16: 10. Testdurchlauf – UDP-STREAM-Test

Testnummer	Seite	Datenrate/ Mbit/s	Sender-CPU-Auslastung/%	Empfänger-CPU-Auslastung/%
1	Sender	9912,80	50,08	
	Empfänger	7967,70		50,09
2	Sender	9916,70	50,02	
	Empfänger	7960,90		50,11
3	Sender	9915,10	50,09	
	Empfänger	8086,00		50,09
4	Sender	9914,00	50,07	
	Empfänger	8167,60		50,11
5	Sender	9915,90	50,04	
	Empfänger	7992,10		50,10
Durchschnitt	Sender	**9914,90**	**50,06**	
	Empfänger	**8034,86**		**50,10**

11. Testdurchlauf (Tabelle A.1-17Tabelle A.1-18)
WCFiFo = 0; MTU = 9000; **IRQ-coalescing = 50**; Allocation Order = 0;**Ts = 0**

Tabelle A.1-17: 11. Testdurchlauf – TCP-Tests

Testnummer	Datenrate/ Mbit/s	Sender-CPU-Auslastung/%	Empfänger-CPU-Auslastung/%
TCP-STREAM			
1	8945,15	50,07	41,31
2	8990,46	50,04	41,34
3	9002,33	50,04	42,15
4	9029,11	50,02	41,76
5	8905,17	50,02	37,60
Durchschnitt	**8974,44**	**50,04**	**40,83**
TCP-SENDFILE			
1	9910,57	13,55	46,40
2	9910,31	13,54	47,30
3	9909,95	50,05	46,45
4	9910,35	13,20	47,06
5	9910,51	13,52	46,64
Durchschnitt	**9910,34**	**20,77**	**46,77**

Tabelle A.1-18: 11. Testdurchlauf – UDP-STREAM-Test

Testnummer	Seite	Datenrate/ Mbit/s	Sender-CPU-Auslastung/%	Empfänger-CPU-Auslastung/%
1	Sender	9911,40	50,04	
	Empfänger	8744,10		50,10
2	Sender	9915,30	50,04	
	Empfänger	8753,00		50,09
3	Sender	9908,60	50,06	
	Empfänger	8734,00		50,10
4	Sender	9907,00	50,07	
	Empfänger	8765,00		51,31
5	Sender	9906,40	50,02	
	Empfänger	8529,70		50,14
Durchschnitt	Sender	**9909,74**	**50,05**	
	Empfänger	**8705,16**		**50,35**

12. Testdurchlauf (Tabelle A.1-19, Tabelle A.1-20)
WCFiFo = 0; MTU = 9000; **IRQ-coalescing = 100**; Allocation Order = 0;**Ts = 0**

Tabelle A.1-19: 12. Testdurchlauf – TCP-Tests

Testnummer	Datenrate/ Mbit/s	Sender-CPU-Auslastung/%	Empfänger-CPU-Auslastung/%
TCP-STREAM			
1	9277,02	50,03	42,62
2	9338,89	50,06	43,13
3	9426,25	50,02	43,30
4	9372,86	50,03	43,04
5	9268,86	50,05	42,75
Durchschnitt	**9336,78**	**50,04**	**42,97**
TCP-SENDFILE			
1	9909,11	11,96	45,19
2	9910,32	11,52	44,95
3	9910,33	11,59	45,26
4	9910,23	11,23	45,18
5	9909,13	11,69	45,22
Durchschnitt	**9909,82**	**11,60**	**45,16**

Tabelle A.1-20: 12. Testdurchlauf – UDP-STREAM-Test

Testnummer	Seite	Datenrate/ Mbit/s	Sender-CPU-Auslastung/%	Empfänger-CPU-Auslastung/%
1	Sender	9918,30	59,95	
	Empfänger	8578,50		50,10
2	Sender	9916,30	60,05	
	Empfänger	8653,60		50,10
3	Sender	9710,80	59,90	
	Empfänger	8781,10		50,10
4	Sender	9900,70	60,08	
	Empfänger	8634,30		50,09
5	Sender	9571,10	59,73	
	Empfänger	8854,90		50,05
Durchschnitt	Sender	**9803,44**	**59,94**	
	Empfänger	**8700,48**		**50,09**

13. Testdurchlauf (Tabelle A.1-21, Tabelle A.1-22) – Linux Kernel Version 2.6.20.14; Firmware-Treiber-Version 1.4.14
WCFiFo = 1; MTU = 9000; IRQ-coalescing = 75; Allocation Order = 0;**Ts = 0**

Tabelle A.1-21: 13. Testdurchlauf – TCP-Tests

Testnummer	Datenrate/ Mbit/s	Sender-CPU-Auslastung/%	Empfänger-CPU-Auslastung/%
TCP-STREAM			
1	9893,41	52,52	45,39
2	9907,59	50,32	44,65
3	9907,50	50,26	44,46
4	9907,01	50,60	44,81
5	9907,32	50,30	44,40
Durchschnitt	**9904,57**	**50,80**	**44,74**
TCP-SENDFILE			
1	9891,46	19,38	44,47
2	9893,76	19,76	44,66
3	9893,50	19,25	44,94
4	9893,70	19,52	44,37
5	9893,20	19,54	44,77
Durchschnitt	**9893,12**	**19,49**	**44,64**

Tabelle A.1-22: 13. Testdurchlauf – UDP-STREAM-Test

Testnummer	Seite	Datenrate/ Mbit/s	Sender-CPU-Auslastung/%	Empfänger-CPU-Auslastung/%
1	Sender	9644,60	57,56	
	Empfänger	9644,60		45,89
2	Sender	9885,10	57,59	
	Empfänger	9724,10		48,65
3	Sender	9917,80	57,47	
	Empfänger	9725,50		48,90
4	Sender	9898,20	57,64	
	Empfänger	9693,70		48,95
5	Sender	9913,00	55,50	
	Empfänger	9699,40		50,15
Durchschnitt	Sender	**9851,74**	**57,15**	
	Empfänger	**9697,46**		**48,51**

14. Testdurchlauf (Tabelle A.1-23, Tabelle A.1-24) – Linux Kernel Version 2.6.20.14; Firmware-Treiber-Version 1.4.21
WCFiFo = 1; MTU = 9000; IRQ-coalescing = 75; Allocation Order = 0;**Ts = 0**

Tabelle A.1-23: 14. Testdurchlauf – TCP-Tests

Testnummer	Datenrate/ Mbit/s	Sender-CPU-Auslastung/%	Empfänger-CPU-Auslastung/%
TCP-STREAM			
1	9903,44	50,27	44,78
2	9905,74	50,22	44,52
3	9904,95	50,30	45,03
4	9899,55	50,48	44,91
5	9908,50	57,24	45,10
Durchschnitt	**9904,44**	**51,70**	**44,87**
TCP-SENDFILE			
1	9895,18	19,89	44,80
2	9895,13	20,18	44,60
3	9902,21	17,84	45,17
4	9895,15	19,73	44,91
5	9895,07	20,10	45,03
Durchschnitt	**9896,55**	**19,55**	**44,90**

Tabelle A.1-24: 14. Testdurchlauf – UDP-STREAM-Test

Testnummer	Seite	Datenrate/ Mbit/s	Sender-CPU-Auslastung/%	Empfänger-CPU-Auslastung/%
1	Sender	9915,90	57,73	
	Empfänger	9915,90		46,80
2	Sender	9918,40	57,61	
	Empfänger	9885,80		47,02
3	Sender	9918,90	57,38	
	Empfänger	9918,90		46,86
4	Sender	9922,00	57,62	
	Empfänger	9919,40		46,76
5	Sender	9907,90	57,93	
	Empfänger	9907,90		46,52
Durchschnitt	Sender	**9916,62**	**57,65**	
	Empfänger	**9909,58**		**46,79**