Quantitative Untersuchungen im Internet
-
Optimierungsmethoden für Internet-Protokolle

Dissertation
zur
Erlangung des Doktorgrades
der Naturwissenschaften
(Dr. rer. nat.)

dem
Fachbereich Mathematik und Informatik
der Philipps-Universität
vorgelegt von

Oliver Dippel
aus Marburg/Lahn

Marburg/Lahn 2000

Vom Fachbereich Mathematik und Informatik
der Philipps-Universität Marburg
als Dissertation angenommen am : Oktober 2000
Erstgutachter : Prof. Dr. Manfred Sommer
Zweitgutachter : Prof. Dr. Bernhard Seeger
Tag der mündlichen Prüfungen am : 16.11.2000

1 EINLEITUNG UND MOTIVATION 1

1.1 Historie des Internets 1

1.2 Entwicklung des Internets in Zahlen 5

1.2.1 Hosts 5

1.2.2 Volumen 7

1.2.3 Protokolle 8

2 GRUNDLAGEN 10

2.1 Netzwerke 10

2.1.1 OSI-Modell 11

2.1.2 Visualisierung des Internets 15

2.2 Protokolle 21

2.2.1 Sicherungsschicht 21

2.2.2 Vermittlungsschicht 25

2.2.3 Transportschicht 30

2.2.4 Anwendungsschicht 37

2.3 Topologie 42

2.3.1 Bus 43

2.3.2 Stern 43

2.3.3 Ring 43

2.3.4 Baum 43

2.4 Medien 43

2.4.1 10BASE-5, ThickWire 44

2.4.2 10BASE-2, ThinWire, Cheapernet 44

2.4.3 10BASE-T, Twisted-Pair 44

2.4.4 10BASE-F, Lichtleiter 45

2.5 Verbindungskomponenten (Repeater, Bridges, Router) 45

2.5.1 Repeater 45

2.5.2 Bridge 46

2.5.3 Switch 46

2.5.4 Router 46

2.6 Daten 47

2.6.1 MIME 47

2.6.2 HTML und XHTML 48

2.6.3 Kompression 49

2.7 Client-Server 50

3 METHODEN DER OPTIMIERUNG 52

3.1 Klassifikation von Protokolloptimierungen 52

3.2 Optimierung der Verbindungssteuerung 53

3.2.1 Vermittlungsschicht 53
3.2.2 Transportschicht 54
3.2.3 Anwendungsschicht 57

3.3 Optimierung der Datenrepräsentation 58
3.3.1 Vermittlungsschicht 58
3.3.2 Transportschicht 60
3.3.3 Anwendungsschicht 62

3.4 Zusammenfassung 69

4 MESSUNGEN UND SIMULATIONEN 71

4.1 Vorüberlegungen 71

4.2 Vermittlungsschicht 72
4.2.1 IPv4 und IPv6 72
4.2.2 IPComp 74

4.3 Transportschicht 75
4.3.1 UDP, TCP und T/TCP 75

4.4 Anwendungsschicht 80
4.4.1 HTTP Pipelining und Persistente Verbindungen 80
4.4.2 HTTP-Caching 81
4.4.3 HTTP Content-Encoding 84

5 ERGEBNISSE UND DISKUSSION 85

5.1 Vermittlungsschicht 86

5.2 Transportschicht 87

5.3 Anwendungsschicht 89

5.4 Diskussion und Zusammenführung der Ergebnisse 92

6 ZUSAMMENFASSUNG 95

6.1 Überblick 95

6.2 Einblick 95

6.3 Ausblick 96

7 ANHANG 97

7.1 Abkürzungsverzeichnis 97

7.2 Abbildungsverzeichnis 99

7.3 Formelverzeichnis 101

7.4 Tabellenverzeichnis 101

7.5 Literatur und Verweise 102

1 Einleitung und Motivation

Das Internet ist das größte Netzwerk der Welt. Über dieses Netzwerk sind Anfang 2000 über 70 Millionen Computer [47] miteinander verbunden und ein Ende des Wachstums ist nicht abzusehen. Ein Grund dafür ist, daß es neue Kommunikationsformen erschließt, die den bisherigen oftmals trägen Kommunikationsmitteln in Geschwindigkeit, Ausbreitung und Kosten zumeist überlegen ist. Ein weiterer wichtiger Grund ist der öffentliche Zugang zu dem Medium Internet, jeder kann sich an der Kommunikation beteiligen. Im besonderen ist dabei die Kommunikationsform WWW (World Wide Web) zu erwähnen. Sie gestattet, nahezu jede Art von Informationen zwischen zwei Personen auf der Welt auszutauschen oder technisch gesagt, Daten zwischen einem Client (Computer, der eine Anfrage stellt) und einem Server (Computer, der eine Anfrage beantwortet) auszutauschen. Waren die angebotenen Inhalte in den Anfängen des Internets noch statisch, so können sie nun abhängig vom Benutzer und dessen Vorgaben dynamisch präsentiert werden. Dadurch lassen sich Produkte darstellen und auch zum Verkauf anbieten, was den kommerziellen Markt erschließt. Da jeder Benutzer des Internets die Möglichkeit hat, Informationen zur Verfügung zu stellen, wächst auch das Informationsangebot ins Gigantische. Daraus erwächst wiederum der Anreiz oder die Notwendigkeit für neue Benutzer das Internet ebenfalls zu nutzen oder intensiver zu nutzen.

Diese und noch viele andere Faktoren haben in den letzten Jahren zu einem exponentiellen Wachstum des Internets geführt. Der weitaus größte Teil der Kommunikation im Internet wird über das WWW geführt. Aus den Ausführungen läßt sich schon erahnen, daß diese Entwicklung enorme technische Anforderungen stellt. Besonders die Leistungsfähigkeit (Kapazität, Verarbeitungsgeschwindigkeit) des Netzwerks muß dem exponentiellen Wachstum folgen.

Zusätzlich zu der Maßnahme die Leitungskapazitäten zu erhöhen, ist es jedoch auch nötig, bisher verwendete Kommunikationsstandards zu überdenken, zu ergänzen oder zu ersetzen, da sich über die jahrzehntelange Entwicklung des Internets die Voraussetzungen und Ziele der Internetnutzung zum Teil entscheidend verändert haben.

Den Schwerpunkt der folgenden Betrachtungen sollen die an der Kommunikation beteiligten Netzwerkkomponenten und Protokolle bilden und zwar in Hinblick auf die Nutzung durch das WWW. Dabei werden Optimierungsstrategien auf unterschiedlichen Protokollebenen der Internetkommunikation vorgestellt, diskutiert und bewertet.

Nach einer Einleitung in die Thematik werden im Grundlagenteil die Komponenten und Protokolle in einem Netzwerk besprochen. Daraufhin werden unterschiedliche Optimierungsansätze erläutert und in Theorie und Praxis klassifiziert und bewertet. Es folgt ein Ergebnisteil mit anschließender Zusammenfassung der Ergebnisse.

1.1 Historie des Internets

Im Jahre 1964 wurde von der RAND Corporation (1948 gegründete, nicht kommerzielle Forschungs-Organisation) das sogenannte "RAND proposal" veröffentlicht, welches aus zwölf separaten Berichten [4] zum Thema „Verteilte Kommunikation“ besteht. Darin beschäftigte man sich

mit den Auswirkungen eines nuklearen Angriffs auf ein landesweites Kommando- und Kontrollnetzwerk und zeigte eine strategische Lösung für dieses Problem auf.

In diesem Dokument kam man zu dem Schluß, daß unabhängig von den Sicherheitsmaßnahmen, die man für ein wichtiges Netzwerk vornimmt, durch einen nuklearen Angriff doch immer ein Teil des Netzwerkes zerstört werden würde. Zudem wäre insbesondere der zentrale Knoten des Netzwerkes das Hauptziel eines gegnerischen Angriffs. Daraus ergeben sich einige sinnvolle Strategien für den Aufbau eines landesweiten Netzwerkes, das auch nach einem nuklearen Angriff zumindest noch in Teilen funktionieren soll. Die entscheidenden Ansätze bei diesem Überlegungen lauten:

- Dezentral: Das Netzwerk ist nicht zentral organisiert. Alle Netzwerkknoten sind in Funktion und Status gleichberechtigt.
- Annahme der Unsicherheit: Das Netzwerk muß zu jedem Zeitpunkt als ungesichert betrachtet werden.
- Paketorientiert: Der Datentransfer findet in Paketen statt, die individuell im Netzwerk transportiert werden.
- Routing: Der Pfad, den die Pakete von Knoten zu Knoten beschreiben, ist nicht festgelegt. Nur Quell- und Zielinformationen werden dem Paket mitgegeben.

Durch diese Regeln sollte sichergestellt werden, daß auch nach dem Ausfall von Teilen des Netzwerks Nachrichten, wenn auch nicht auf dem kürzesten Weg, ihr Ziel erreichen. Bei der Implementierung mußte somit eine große Dynamik des Netzwerkes berücksichtigt werden. Ursprünglich eher für den Ausfall von Netzwerkknoten gedacht, kommt diese geforderte Flexibilität gerade heute dem Ausbau von Netzwerken zugute.

Das erste Netzwerk, das den oben genannten Richtlinien entsprach, wurde 1968 im "National Physical Laboratory" in Großbritannien unter Leitung von Donald Davies [19] aufgebaut (NPL Data Network). Im September 1969 [82] wurde in USA auf eine Initiative der "Pentagons Advanced Research Projects Agency" hin in der Universität von Kalifornien (UCLA) der erste Knoten eines weiteren Netzwerkes installiert, bis Ende 1969 drei weitere Knoten (*Abbildung 1*).

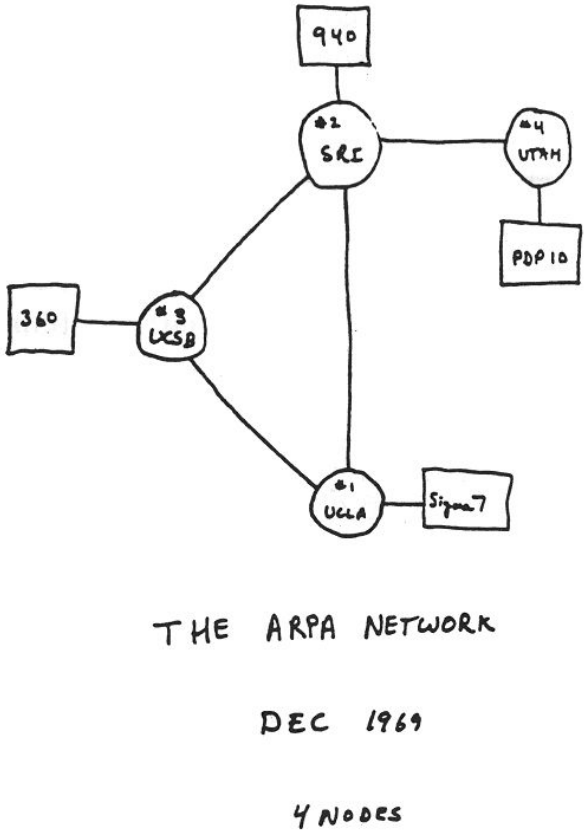

Abbildung 1: Zeichnung des Netzwerks im Dezember 1969 von Alex McKenzie vom BBN [1]

Dieses Netzwerk wurde nach dem Pentagon Sponsor ARPANET getauft. Im Jahre 1971 waren es bereits 15 Knoten, deren Anzahl sich im Jahre 1972 bereits auf 37 Knoten erhöht hatte. Weitere Netzwerke wurden 1977 zum ARPANET verbunden, dessen Dominanz sich trotz weiteren Wachstums im Netz der Netzwerke verringerte und seit 1989 nominell nicht mehr existiert. Der militärische Teil des ARPANET spaltete sich dann 1983 als MILNET ab. Die Summe der untereinander verbundenen Netzwerke wurde als "Internet" bekannt.

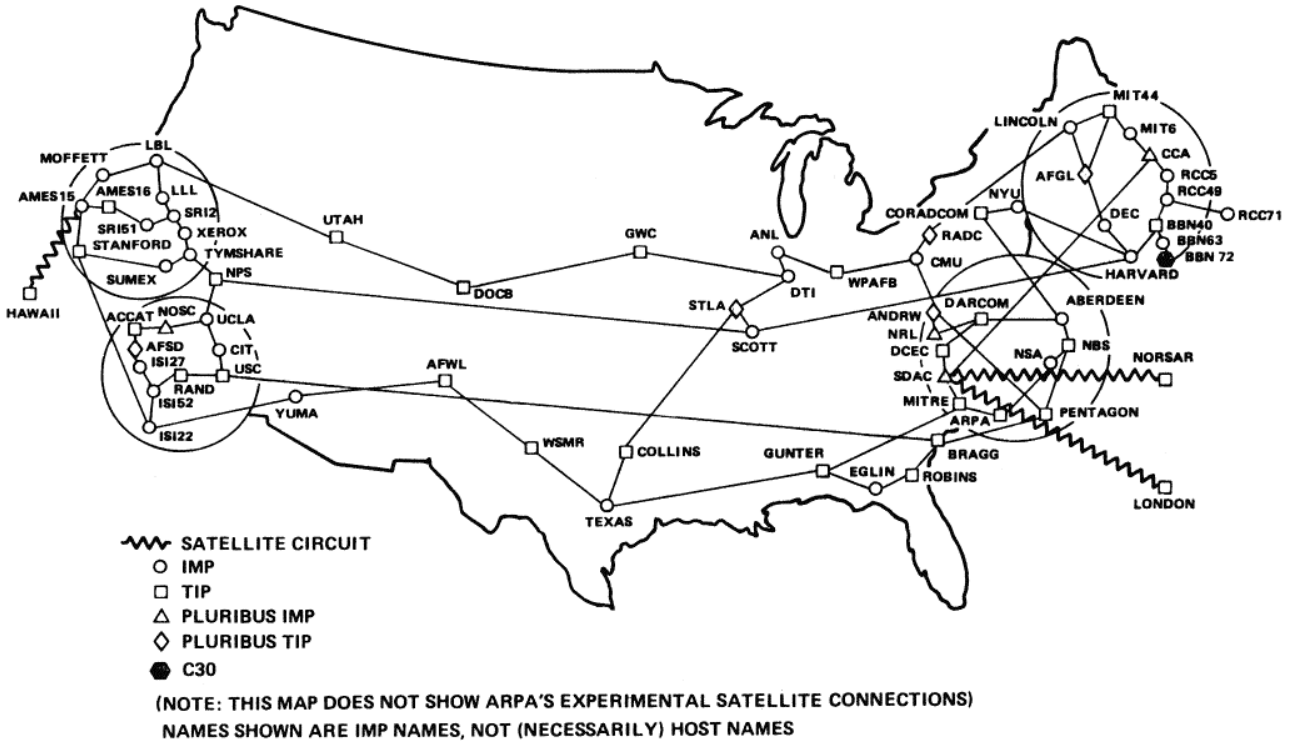

Abbildung 2: ARPANET Oktober 1980 [3]

Nicht zuletzt aufgrund der zur Verfügung gestellten Kapazitäten war das NSFNET ein ebenfalls bedeutendes Netzwerk innerhalb des Internets. Es wurde 1984 von der "National Science Foundation" gegründet und im Frühjahr 1995 aufgelöst und durch mehrere andere Institutionen ersetzt.

Durch den dezentralen Aufbau und die anarchische Struktur des Verbunds der Netzwerke war eine Kontrolle der neuen Knoten nur schwer möglich und im Grunde auch gar nicht gewünscht. Dadurch ergibt sich ein zunehmend buntes Erscheinungsbild der Interessensgruppen im Internet. War das Internet ursprünglich auf die Gruppen Militär und Forschung beschränkt, so gehören nun zu den am Internet teilnehmenden Gruppen zunehmend Privatpersonen, kommerzielle Anbieter und nicht-kommerzielle Organisationen. Trotz der unterschiedlichen Ansprüche, die hinter den einzelnen Gruppen stehen, ist das Internet in einem recht stabilen Zustand, und somit eines der wenigen Beispiele einer funktionierenden, modernen Anarchie [98].

Mit der Verbreitung der Vernetzung von Rechnern wachsen auch die daraus entstehenden Möglichkeiten. In vielfacher Hinsicht ist diese Entwicklung mit der des Telefons zu vergleichen. Auch hier steigerten sich Nutzungsmöglichkeiten und Akzeptanz des Mediums mit der Verbreitung von einer technischen Idee zur Notwendigkeit für jeden Menschen. Ist die Notwendigkeit eines Zugangs zum Internet im Moment (Jahr 1999) auch nicht für jeden Menschen gegeben, so läßt doch das exponentielle Wachstum des Internets erahnen, daß dieser Zeitpunkt in nicht allzu weit entfernter Zukunft liegt. Mit der Anzahl der Systeme wächst auch das Datenvolumen, das über die Leitungen des Internets transportiert wird. Somit wachsen auch die Anforderungen an die Bandbreite (Datenvolumen pro Zeit) der Übertragungsmedien. Das betrifft sowohl die Übertragungsleitungen zwischen den einzelnen Netzen, als auch die vermittelnden Komponenten in den Netzen, wie

Router, Switches und Bridges. Die Problematik der Bandbreite ist in lokalen Netzen recht leicht zu lösen, z.B. durch die Migration eines 10 MBit Ethernets auf ein 100 MBit Netz. Eine solche Migration kann im einfachsten Fall durch den Austausch der Ethernetkarten in den Endgeräten realisiert werden und ist ausgesprochen kostengünstig. Die Vergrößerung der Bandbreite zwischen einzelnen Netzen ist i.a. mit wesentlich höherem Aufwand und Kosten verbunden: Hier werden je nach Bandbreite unterschiedliche Netzkomponenten und Protokolle verwendet. Standleitungen geringer Bandbreite (64 KBit/s oder 128 KBit/s) werden mit ISDN-Leitungen realisiert, mittlere Kapazitäten (2 MBit/s) mit X.25 und hohe Bandbreiten (34 MBit/s) mit ATM-Technologie. Die Verbindungstechnologie, die für den Anschluß zwischen den Netzen verwendet wird, stellt höchste Anforderungen an die beteiligten Komponenten. Das spiegelt sich unmittelbar in den Anschaffungskosten (Router, etc.) und den laufenden Kosten (Standleitung) wieder. Doch auch in Hinblick auf die Kommunikationsprotokolle ist Entwicklungsarbeit notwendig. Das wohl populärste Beispiel ist die der Mangel an IP-Nummern durch das Wachstum der Anzahl der im Internet angeschlossenen Systeme. Dieses Problem kann nur durch die Einführung eines neuen Protokolls (IPv6) gelöst werden. Ein weiteres Problem ist die Sicherheit der Protokolle. Im Zuge der Kommerzialisierung des Internets besteht der Bedarf an sicheren Übertragungsprotokollen zur Übermittlung von sensiblen Daten, wie Kreditkartennummern. Das Internetprotokoll IP ist in seiner Konzeption nicht dafür ausgelegt. All diese Probleme erfordern viel Arbeit, zudem sie nach Möglichkeit transparent für den Anwender gelöst werden müssen.

1.2 Entwicklung des Internets in Zahlen

Es gibt eine große Anzahl von Statistiken über die Entwicklung des Internets. Jedoch sind die meisten davon in ihrer Aussage räumlich, zeitlich und in der Anzahl der Parameter begrenzt. Die Ursache dafür ist die schon oben erwähnte Topologie die Dynamik in der Struktur des Internets. Dennoch sind einige wenige Parameter, wie die Anzahl der weltweit am Internet angeschlossenen Systeme, über den Zeitraum von der Entstehung des Internets bis zum heutigen Tage verfügbar. Andere Parameter, wie die Anzahl der übertragenen Bytes, oder noch genauer die Anzahl der übertragenen Bytes in Abhängigkeit von der Anwendung (Telnet, FTP, WWW, ..), sind nur über einen begrenzten Zeitraum und nur für bestimmte Teilnetze verfügbar. Eine der ausführlichsten Statistiken wurde im Rahmen des ”NSFNET Backbone Service Project" (National Science Foundation NETwork) [63] erstellt. Das NSFNET wurde in Zusammenarbeit der Institutionen National Science Foundation, ANS, IBM, MCI und dem Staat Michigan aufgebaut und gepflegt. Dieses Projekt bestand von 1987 bis zum April 1995. In dieser Zeit wurden umfangreiche Messungen über die transferierten Daten vorgenommen. Zu den weiteren, hier verwendeten Quellen gehören RIPE [90], eine Institution zuständig für die Vergabe und Verwaltung von IP-Nummern in Europa und das örtliche Rechenzentrum HRZ [41].

1.2.1 Hosts

Ein Wert, welcher die zeitliche Entwicklung des Internets aussagekräftig darstellt ist die Anzahl der Hosts, auch Hostcount genannt. Mit dem Begriff Host ist in diesem Kontext nicht nur ein PC oder Rechner gemeint, sondern alle Computer im Internet, die sich über einen DNS-Namen oder eine IP-Nummer ansprechen lassen, d.h. einen A-Record im DNS haben [76]. In einem LAN wie der Universität Marburg lassen sich solche Statistiken mit akzeptablen Aufwand sammeln, z.B. bei der Anmeldung eines Rechners für den Anschluß and das Netz der Universität. Auch in den

Registrierungsinstitutionen, wie dem RIPE für Europa sind die Informationen über angemeldete Hosts verfügbar. Eine weltweite Registrierungsinstitution gibt es jedoch nicht. Daher sammeln verschiedene Firmen und Institutionen Statistikdaten über verschiedene Kenngrößen des Internets und vermarkten diese teilweise, wie die Firma MIDS [68] oder ISC [47]. Die Datenerfassung für solche Statistiken erfolgt meist über selbstgeschriebene Hilfsprogramme. Diese Statistiken können zwar aufgrund der Größe des Internets und der verwendeten Algorithmen der Datenakquirierung nicht exakt sein, liefern jedoch sehr gute Annäherungen an die Größenordnung der Meßwerte. In *Abbildung 3* ist die Anzahl der Hosts für das LAN der Philipps-Universität Marburg [41], der Domains für Europa [90] und für die ganze Welt [47] in zeitlicher Entwicklung aufgetragen. Die Daten für die weltweite Hostanzahl wurden von der nicht kommerziellen Organisation ISC [48] übernommen. Die Methode der Datenakquisition ist unter [49] beschrieben, mit der technischen Durchführung der Zählung ist die Firma Network Wizards [75] betraut. Den Datenreihen für die Hostanzahl in Marburg und Europa ist jeweils über eine Ausgleichsrechnung eine sog. Trendlinie zugeordnet, der ein polynomialer Ansatz zweiter Ordnung ($y = b + c_1 x + c_2 x^2$) zugrundeliegt. Der Datenreihe für die weltweite Anzahl der Hosts ist eine Trendlinie dritter Ordnung zugeordnet. Wie man der Abbildung entnehmen kann, nähern sich die Trendlinien der Entwicklung über den Zeitraum von sechs Jahren recht gut an.

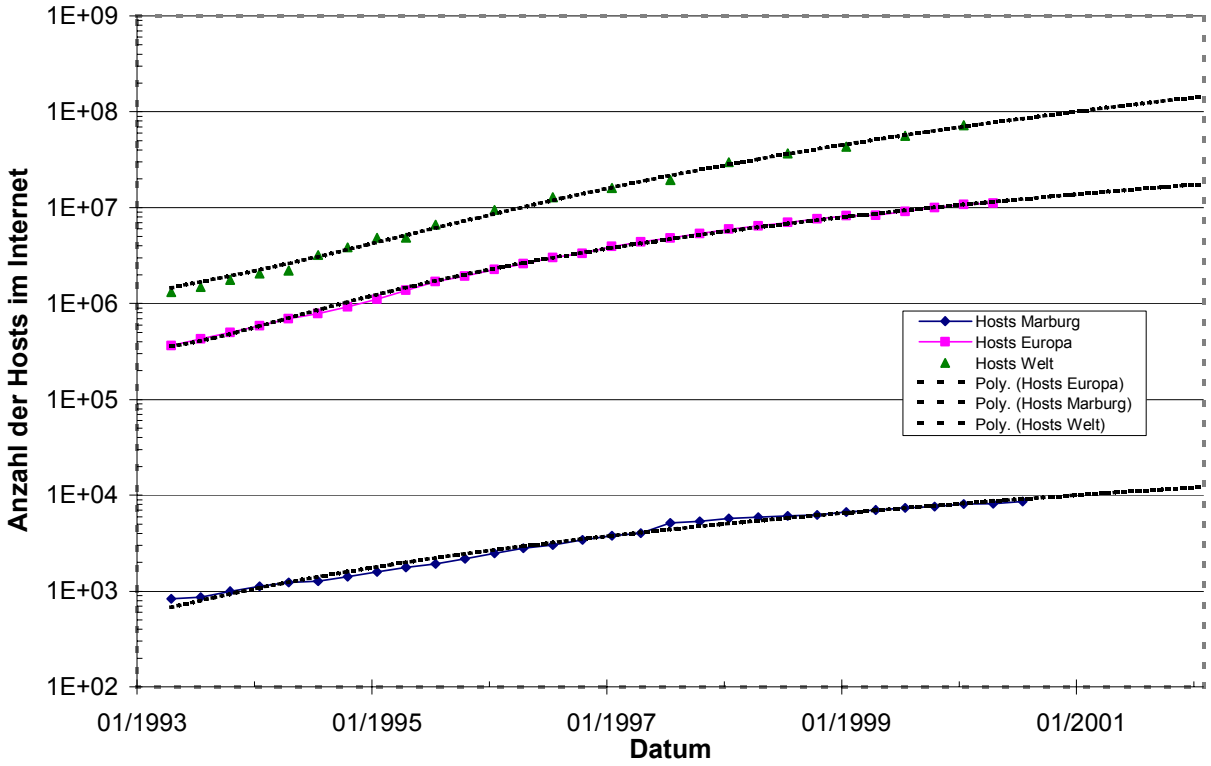

Abbildung 3: Entwicklung Hostcount Marburg, Europa, Welt

Dadurch läßt sich über die Jahrtausendwende hinaus eine gute Abschätzung über die Anzahl der Host treffen. Die Werte in *Tabelle 1* sind aus den Trendlinien (*Abbildung 3*) grafisch entnommen.

Datum	Jahr 2000	Jahr 2002	Jahr 2005
Marburg	8074	12000	17000
Europa	10 Mio	16 Mio	30 Mio
Welt	72 Mio	130 Mio	340 Mio

Tabelle 1: Abschätzung Hostcount Marburg, Europa, Welt (Stand: Mitte 2000)

In *Tabelle 1* wird die Abschätzung (Stand: Mitte 2000) durchgeführt und bezieht sich jeweils auf den Anfang der genannten Jahre. Die Zahlen für das Jahr 2005 haben aufgrund der Unsicherheiten in der Annäherung und der nur schwer einzuschätzenden Entwicklung der Netzwerkbranche eher hypothetischen Charakter. Jedoch zeigte eine zwei Jahre zuvor durchgeführte Abschätzung, daß die Genauigkeit einer solchen Aussage zumindest höher ist als eine Abschätzung der Größenordnung.

1.2.2 Volumen

Wie auch die Anzahl der Hosts im Internet ist das Transfervolumen im Internet einer rasanten Steigerungsrate unterworfen. Nicht zuletzt aufgrund der guten Verfügbarkeit werden hier exemplarisch die monatlichen Transfervolumina der Universität Marburg vom und ins Internet betrachtet. In *Abbildung 4* wird das Volumen in Abhängigkeit von der Zeit aufgetragen und ist mit einer polynomialen Trendlinie vierter Ordnung versehen. Eine Trendlinie zweiter Ordnung hätte im Gegensatz zur Hostanzahl das zeitliche Verhalten nur unzureichend getroffen.

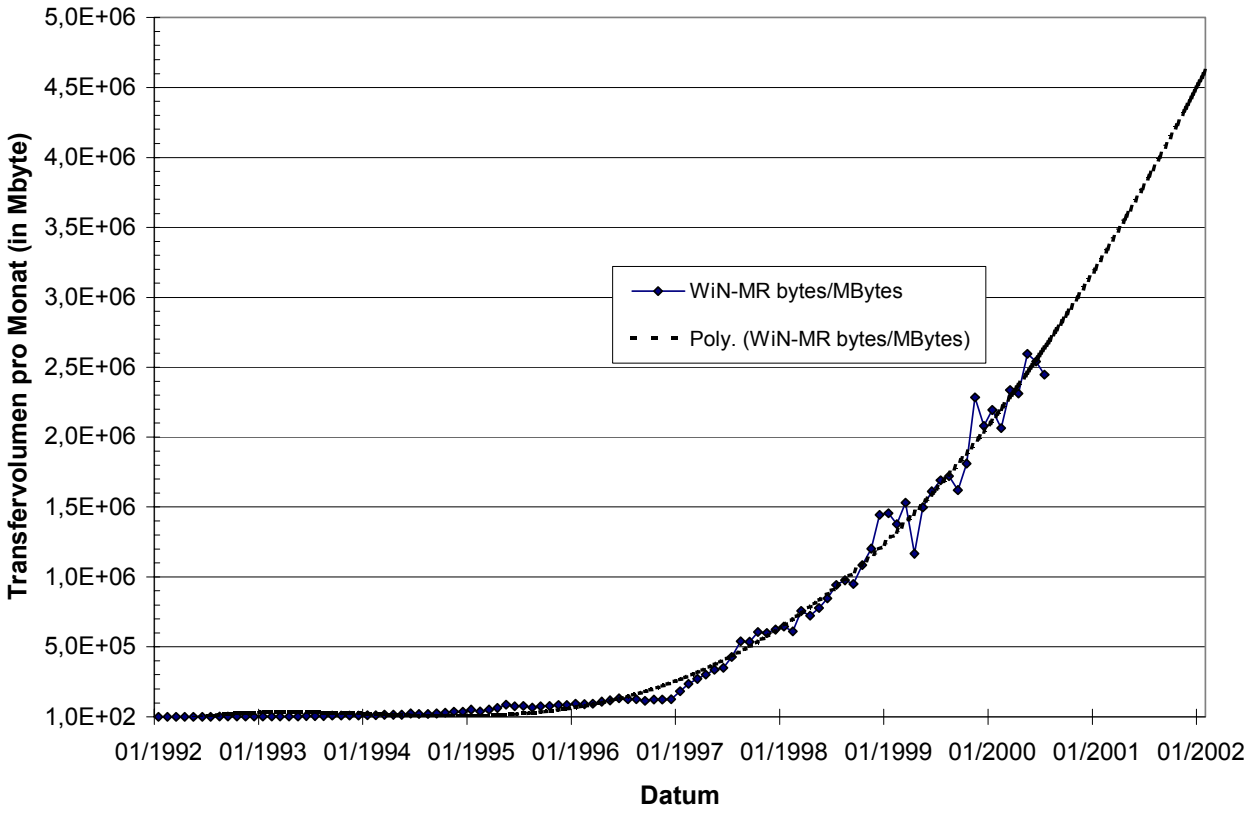

Abbildung 4: Monatliches Transfervolumen Marburg in MByte

Als Abschätzungen für die folgenden Jahre ergeben sich hier Werte (*Tabelle 2*), die mit noch größerer Unsicherheit als die Werte der Hostanzahl versehen sind.

Datum	Jahr 2000	Jahr 2002	Jahr 2005
Marburg	2,2 TByte	4,6 TByte	9,6 TByte

Tabelle 2: Abschätzung monatliches Transfervolumen Marburg

Die Gründe dafür sind sowohl technischer, als auch finanzieller Natur: Zum einen kann es vorkommen, daß der Provider, welcher die Universität versorgt, technische Schwierigkeiten mit

dem Anschluß an den Rest des Internets hat, zum anderen könnte es vorkommen, daß die Universität nicht genug Geld zur Verfügung hat, um die notwendigen Kapazitäten einzukaufen. Beides führt zwangläufig dazu, daß das Transfervolumen der Universität zumindest zeitweise unter dem typischen, bzw. vom Benutzer gewollten Internetverhalten liegt. Ein Beispiel dafür ist die Stagnation des Wachstums gegen Ende 1996. Diese wurde Anfang 1997 behoben, als die Universität von einem 2 MBit X.25 Anschluß an das WiN auf einen 16 MBit ATM Anschluß (Mitnutzung eines 34 MBit Anschlußes) umgestellt worden ist. Gegen Ende 2000 soll dann der Anschluß der Universität Marburg an das G-WiN (Gigabit-WiN) mit max. 155 Mbit/s erfolgen. Das G-WiN des DFN-Vereins ist ein optisches Netzwerk mit Übertragungsraten von bis zu 2,4 GBit/s. Das Basis-Protokoll ist nicht wie im B-WiN ATM, sondern SDH.

1.2.3 Protokolle

Dank der ausführlichen Statistikdaten des NFSNET Projektes kann man auch eine Aussage über die Nutzung der Protokolle im NSFNET bis Anfang 1995 machen. Die Nachfolgeorganisationen des NSFNET führen leider keine so umfangreichen Messungen durch. Dadurch ist die statistische Auswertung auch auf den Zeitraum bis zum April 1995 beschränkt. Eine logarithmische Darstellung dieser bereits oben erwähnten Statistikdaten wurde z.B. von MIDS [67] durchgeführt. Eine alternative Darstellung ist in *Abbildung 5* zu sehen. Hier wurden nur die Protokolle mit den größten Anteilen aufgetragen. Dabei ist klar das rapide Wachstum des WWW-Verkehrs innerhalb des Internets (hier: NSFNET) zu erkennen. Bereits zwei Jahre nach seiner Einführung hat das WWW seinen Vorgänger Gopher überholt und ist bis heute zum dominanten Protokoll geworden.

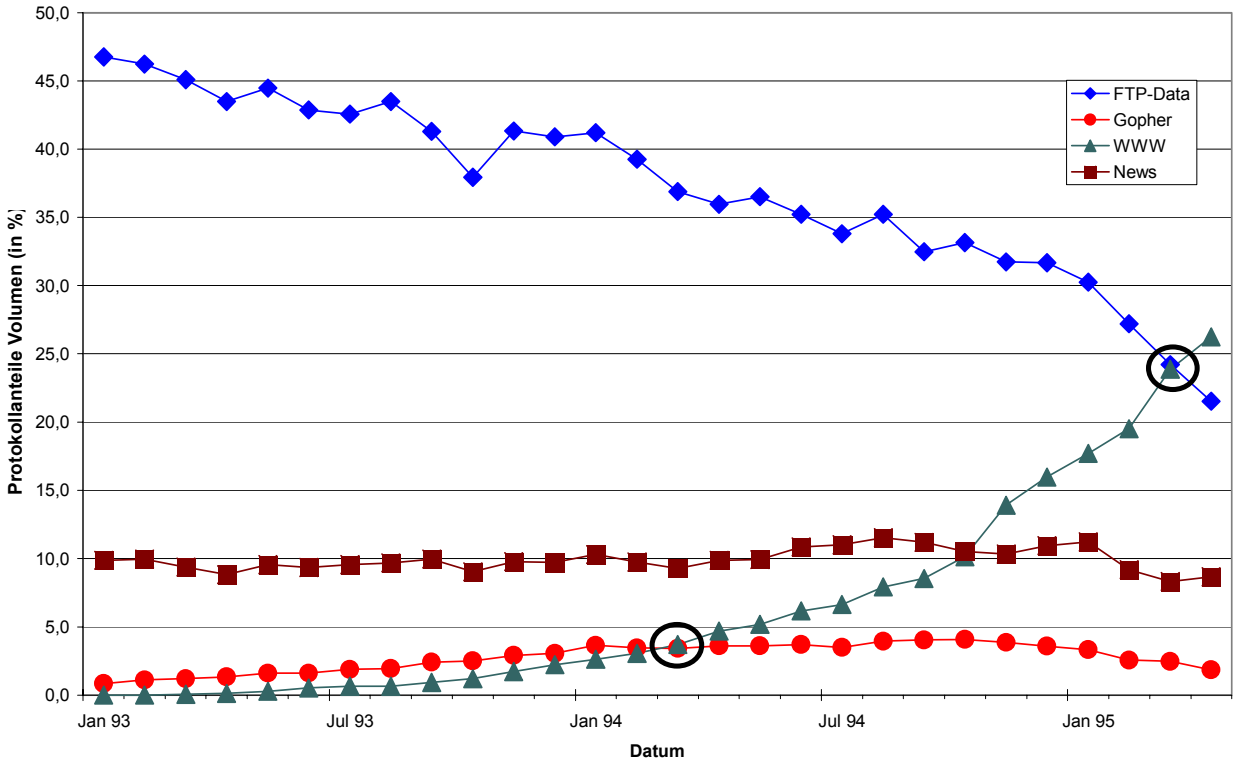

Abbildung 5: Protokollanteile des NSFNET Verkehrs [63]

In *Abbildung 5* sind zwei entscheidende Schnittpunkte markiert: Zum einen ist das der März 1994, als der WWW-Anteil den Gopher-Anteil überholt und zum anderen ist das der März 1995. Zu

diesem Zeitpunkt wird das WWW-Protokoll zum dominanten Protokoll im Internetverkehr. Diese Aussagen sind natürlich auf das NSFNET beschränkt.

Damit bestätigt sich die Bedeutsamkeit des WWW-Protokolls für das Internet. Doch nicht allein ein einziges Protokoll ist für den Transport der WWW-Daten zuständig, genau genommen ist das WWW-Protokoll eine Protokoll-Familie, die nicht nur aus dem HTTP-Protokoll besteht, sondern auch aus den im OSI-Modell darunter liegenden Schichten. Damit impliziert eine Analyse des WWW-Verkehrs eine Analyse aller damit verbundenen Protokolle, im speziellen sind das IP und TCP. Der Zusammenhang Protokolle, Internet, WWW, IP und TCP wird im nächsten Kapitel (*Kapitel 2: Grundlagen*) erläutert. Dabei werden auch die notwendigen Grundlagen der einzelnen Protokolle vermittelt, um Optimierungsstrategien zu analysieren.

2 Grundlagen

Die Frage, was das Internet eigentlich ist und wonach man es charakterisieren kann, läßt sich nur schwer beantworten. Ein Erklärungsansatz war im vorigen Abschnitt zu lesen: Das Internet ist ein Netzwerk aus Netzwerken. Dieser Erklärungsansatz ergibt sich aus der Betrachtung der räumlichen Anordnung und der Infrastruktur der einzelnen Netzknoten. Man kann jedoch das Internet auch unter sozialen Aspekten [52] als Zusammenschluß von Interessensgruppen definieren, bestehend aus Netzwerkbetreibern und Netzwerkbenutzern. Eine weitere Charakterisierung ist aufgrund der verwendeten Protokolle zur Kommunikation zwischen den einzelnen Internet-Teilnehmern möglich. Im folgenden sollen nun weniger die sozialen Aspekte der weltweiten Vernetzung diskutiert werden, sondern die technischen Grundlagen zum Verständnis über den Aufbau eines globalen Netzwerkes gelegt werden. Dazu wird auf das dem real existierenden Netzwerk zugrunde liegende OSI-Modell eingegangen und auf die überwiegend eingesetzten Protokolle im LAN- und WAN-Bereich.

2.1 Netzwerke

Ein Netzwerk ist nach Tanenbaum [105] definiert als: „mehrere miteinander verbundene, unabhängige Computer“. Dabei sei es unerheblich, ob die Verbindung elektrisch, optisch oder akustisch realisiert ist. Unter der Unabhängigkeit sei zu verstehen, daß es kein eindeutiges Master/Slave Verhältnis zwischen den Computern gibt. Damit wollte Tanenbaum ausschließen, daß z.B. ein Computer (Master) mit einem oder mehreren entfernt angeschlossenen Terminals oder Druckern (Slaves) als Netzwerk betrachtet werden. Diese Beispiele sind mittlerweile jedoch problematisch geworden, da Netzwerkdrucker und auch Terminals heute soviel „Eigenintelligenz“ besitzen, daß ein vom zugeordneten Computer unabhängiger Betrieb durchaus möglich ist. So haben einige Drucker schon Telnet-Server und WWW-Server zu Konfigurationszwecken eingebaut. Auch die aktuellen Forschungsarbeiten und Entwicklungen auf dem Gebiet der Agentensysteme zielen darauf, daß die ehemals „dummen“ Endgeräte einen autonomen und aktiven Status innerhalb eines Netzwerks erhalten. Spätestens an dieser Stelle muß die Master-Slave Definition beziehungsweise deren Verwendung überdacht werden.

Dieses Beispiel zeigt wieder einmal, daß auch die Definition des Begriffs des Netzwerkes nicht zwangsläufig eine Konstante in der Entwicklung der Netzwerke ist. Dennoch ist diese Definition natürlich in Ihrer Grundaussage zutreffend und wird in diesem Verständnis hier auch weiterhin verwendet.

Zum weiteren Verständnis wird das in der Informatik weit verbreitete OSI-Modell der Protokolle vorgestellt, welches in Anlehnung an das Prinzip „divide et impera !“ der römischen Außenpolitik im Latinerkrieg aufgebaut ist.

Dann folgen einige Beispiele für eine Visualisierung des Internets. Daran soll aufgezeigt werden, daß ein Netzwerk abhängig vom Kontext durchaus verschiedene Strukturmerkmale besitzen kann. So wird ein lokaler Adminstrator hauptsächlich die Computerstandorte der Computer und die zugehörigen IP-Nummern und DNS-Namen verwalten, wohingegen ein Netzprovider seine Betrachtung primär auf die Bandbreite in Abhängigkeit von seinen Standorten richtet. Ein Telekommunikationsunternehmen indessen, welches die Leitungen verlegt, wird hauptsächlich in den

geologischen und infrastrukturellen Gegebenheiten in Anhängigkeit von den Standorten interessiert sein, wohingegen ein Anwender, der z.B. an Inhalten einer Newsgroup interessiert ist, eine neuronale Visualisierung der Newsgroups hilfreich finden wird.

2.1.1 OSI-Modell

Das Open Systems Interconnect (OSI) Referenz Modell ist der Vorschlag der „International Standards Organization“ (ISO) für eine universelle Netzwerkarchitektur und wurde in ISO7498 definiert.. Dieses Modell findet seine Umsetzung in den OSI-Protokollen ISO8823, ISO8327, ISO8073 und ISO8473. Die Implementierung der OSI-Protokolle hat jedoch im Vergleich mit den Internetprotokollen TCP/IP nur eine geringe Bedeutung. Ursachen dafür sind die geringe Verbreitung und der späte Entwicklungs- und Implementationszeitpunkt. Zudem wurde der Beschluß der US-Regierung am 16. September 1995, die OSI-Protokolle nicht mehr verbindlich für die US-Verwaltung vorzuschreiben, mehrfach als "Todesstoß" für die Implementation der OSI-Protokolle angesehen, da rund 20% der Computer auf dem amerikanischen Markt im staatlichen Besitz sind [93].

Im OSI-Modell wird eine Protokollfamilie in sieben Schichten unterteilt, die durch ihre Funktion charakterisiert werden.

Schicht	Name (*englisch*)	Beschreibung
7	Anwendungsschicht (*application layer*)	Anwendungsspezifische Aufgaben
6	Darstellungsschicht (*presentation layer*)	Zuständig für Datenrepräsentation
5	Kommunikationssteuerungsschicht (*session layer*)	Kontrolle der Kommunikationkanäle
4	Transportschicht (*transport layer*)	Gesicherter Datentransport
3	Vermittlungsschicht (*network layer*)	Routing der Datenpakete
2	Sicherungsschicht (*data link layer*)	Gesicherter transport der Daten über das Medium
1	Bitübertragungsschicht (*physical layer*)	Eigenschaften des Transportmediums

Tabelle 3: OSI-Schichtenmodell

Jede Schicht in diesem Modell stellt der darüberliegenden Schicht einen Dienst zur Verfügung. Dazu kommuniziert diese Schicht mit der gleichen Schicht des Kommunikationspartners unter Zuhilfenahme der darunterliegenden Schicht. Die Orte, wo die Schichten impementiert sind, sind sehr unterschiedlich: Einige Schichten sind in der Hardware des sendenden Gerätes implementiert, andere im Betriebssystem und wieder andere in den Anwendungsprogrammen selbst.

Die Bitübertragungsschicht ***(physical layer)*** transportiert Bits von einem Ende des physikalischen Mediums zum anderen. Diese Schicht beinhaltet die **Spezifikation des Mediums** (Koax-Kabel, Lichtwellenleiter, Lasersignale, etc.), die Verbindungsstellen, die Modulation der Signale, die Beschränkungen des Einsatzgebietes, u.s.w.

Die Sicherungsschicht ***(data link layer)*** benutzt die Bitübertragungsfähigkeit des physical layers, um eine **gesicherte Verbindung** zwischen zwei Systemen aufzubauen. Dazu werden Verfahren wie Fehlererkennung und –korrektur und Kontrollmechanismen für den Medienzugriff verwendet.

Die Vermittlungsschicht ***(network layer)*** implementiert **Routingfähigkeiten** für die Zustellung von Datenpaketen im Netz. Die Routingprotokolle suchen in dieser Schicht den optimalen Pfad in einem Netzwerk.

Die Transportschicht ***(transport layer)*** stellt für die höheren, anwendungsbezogenen Schichten einen Datentransportdienst zur Verfügung, der nach Möglichkeit keine Implementationsdetails der Implementierung des Datentransports erkennen läßt. Die Aufgabe dieser Schicht ist es, einen **gesicherten Datentransport** im Netzwerk zu gewährleisten.

Die Steuerungsschicht ***(session layer)*** bietet die Möglichkeit, Synchronisationspunkte im Fluß der Datenpakete zu setzen, um dann im Bedarfsfall zu einem vorherigen Synchronisationspunkt zurückzukehren. Der session layer verfügt lediglich über die Fähigkeit, **eine wiederholte Sendung zu synchronisieren**, die Abwicklung selbst des Aufsetzens auf einen vorherigen Synchronisationspunkt muß von der Anwendung erfolgen.

Die Darstellungsschicht ***(presentation layer)*** legt die **Datenrepräsentation** fest, welche die Anwendung erhält, wie z.B. die Umsetzung von UNIX Zeichensatz auf einen DOS-Zeichensatz.

Die Anwendungsschicht ***(application layer)*** erfüllt **anwendungsspezifische Aufgaben**. Im Falle eines **E-Mail** Programmes könnte diese Schicht die Aufgabe der **Adressauflösung** und der Behandlung des **Transfers von sehr großen Dateien** erfüllen.

Eine Kommunikation im OSI-Modell durchläuft mindestens eine Schicht, im Maximalfall alle sieben Schichten. So sieht z.B. der Datenfluß einer Kommunkation zwischen zwei Anwendungen mit einem dazwischenliegenden IP-Router folgendermaßen aus:

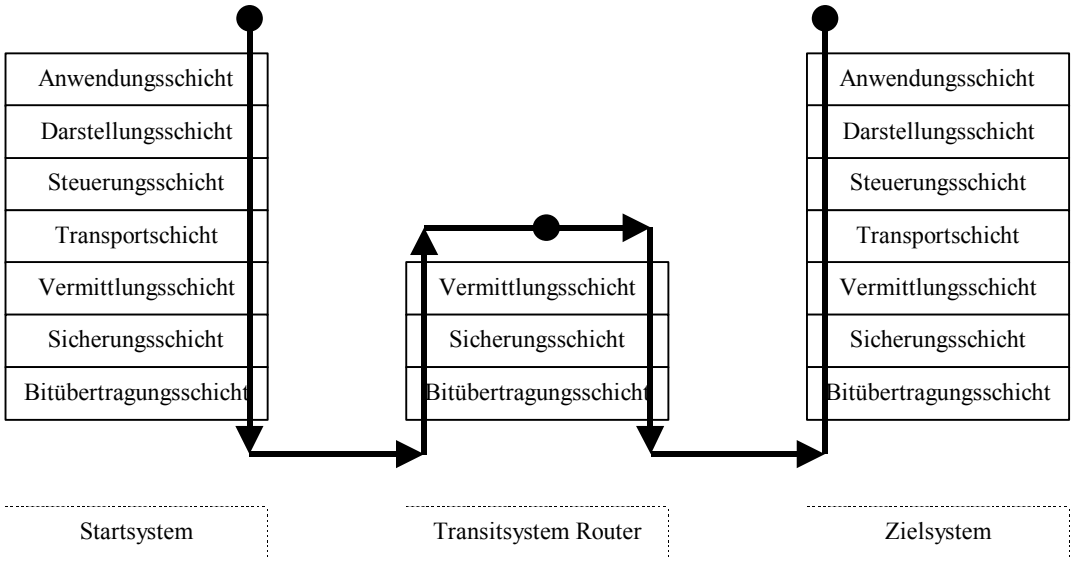

Abbildung 6: Kommunikation über Router

Die Anwendung schickt ihre Daten vom Startystem zum Zielsystem. Dabei müssen die Daten über das Transitsystem Router geleitet werden. Bei diesem Vorgang gelangen die Daten zunächst über die Anwendungsschicht in die Bitübertragungsschicht des Startsystems. Die Bitübertragungsschicht ist das physikalische Transportsystem zwischen den Systemen und kann z.B. ein 10 MBit Ethernet sein. Die Daten gelangen nun zum Router. Der Router ist ein System, das Transferdaten in der Vermittlungsschicht analysiert und bearbeitet. Daher steigen nun die Daten von der Bitübertragungsschicht des Routers in die Vermittlungsschicht des Routers, wo sie bearbeitet werden. Danach steigen die Daten wieder in die Bitübertragungsschicht hinab und werden zum Zielsystem transferiert, wo sie von der Bitübertragungsschicht aufsteigen zur Anwendungsschicht, dem Ziel der Datenreise: Die Schnittstelle zu der zweiten, an der Kommunikation beteiligten Anwendung.

Obwohl die ISO-Protokolle als Implementation des OSI-Modells an Bedeutung verloren haben, wird das OSI-Modell weiterhin als Referenz für die Einordnung von Protokollfamilien (z.B. AppleTalk, DECnet, Internet) verwendet.

In *Tabelle 4* sind einige wichtige Protokollfamilien in die OSI-Schichten eingeordnet, darunter auch die TCP/IP Protokollfamilie mit den zugehörigen Protokollen auf den höheren Schichten.

Das Prinzip des Schichten-Treiber Konzeptes lautet: Implementation der Funktionalität der Schicht und Bereitstellung einer definierten Schnittstelle sowohl zu der darüber, als auch zu der darunterliegenden Schicht. Dabei ist es jedoch nicht notwendig, daß die Implementation eines Schichtentreibers für eine Protokollfamilie zu der gleichen Implementation eines anderen Herstellers passt und somit austauschbar ist. Zumeist ist die Implementation einer vollständigen TCP/IP Protokollfamilie in drei tatsächlich austauschbare Blocks (mehrere Schichten) unterteilt: Der Hersteller der Netzwerkkarte (Schicht 1) liefert den passenden Treiber für die Sicherungsschicht (Schicht 2), der TCP/IP-Stack (Schicht 3 und 4) wird vom Betriebssystem zur Verfügung gestellt und die darüber liegenden Schichten (Schichten 5 bis 7) werden von den Anwendungsprogrammen der Softwarehersteller implementiert.

In der untersten Schicht der TCP/IP Protokollfamilie sind die möglichen Transportmedien für die Bitübertragungsschicht eingetragen. Wie aus der Tabelle zu entnehmen ist, sind diese unabhängig

von der betrachteten Protokollfamilie. Es können optische Lichtleiter wie auch elektrische Leiter (z.B. Koax-Kabel oder Twisted-Pair Kabel) zum Einsatz kommen.

Auch in der darüber liegenden Abstraktionsschicht der Sicherungsschicht ist man in der Wahl des Protokolls noch nicht eingeschränkt: So kann man für den Einsatz der TCP/IP Protokollfamilie zwischen den Protokollen Token-Ring, Ethernet und dessen Varianten wählen. Die Anbindung dieser Schicht in die Protokollfamilie wird über NIC-Treiber (Network Interface Cards-driver) realisiert.

In der Vermittlungsschicht wird das IP (Internet Protocol) implementiert, welches die Routingfähigkeiten der TCP/IP Protokollfamilie realisiert.

Darauf setzt dann in der Transportschicht das TCP (Transmission Control Protocol) auf, welches eine gesicherte Verbindung realisiert.

Darauf basierend folgen in der Steuerungsschicht Protokolle wie HTTP und SMTP.

Die darin erhaltenen Daten werden in der Darstellungsschicht von (optionalen) anwendungsspezifischen Protokollen - wie dem HTML - bearbeitet und für die darüber liegende Schicht vorbereitet.

In der Anwendungsschicht werden schließlich diese Daten entsprechend der Aufgabe des Anwendungsprogramms verarbeitet. Im Falle eines WWW-Browsers werden die HTML-Daten dem Anwender präsentiert.

OSI-Schicht	Apple Computer	Banyan Systems	DEC DECnet	IBM SNA	Microsoft Networking	Novell NetWare	TCP/IP Internet	Xerox XNS	OSI Protocols
Application Layer 7					Application Programs and Protocols for file transfer, electronic mail, etc.		Applications (WWW-Browser)		
Presentation Layer 6	AppleTalk Filing Protocol (AFP)	Remote Procedural Calls (Net RPC)	Network Management Network Application	Transaction Services Presentation Services	Server Message Block (SMB)	NetWare Core Protocols (NCP)	Application Specific Protocols (HTML)	Control and Process Interaction	ISO 8823
Session Layer 5	AppleTalk Session Protocol (ASP)		Session	Data Flow Control	Network Basic Input/Output System (NetBIOS)	Network Basic Input/Output System (NetBIOS)	(Telnet, FTP, SNMP, SMTP, ..)		ISO 8327
Transport Layer 4	AppleTalk Transaction Protocol (ATP)	Vines Interprocess Communic. (VIPC)	End Communic.	Transmission Control	Network Basic Extended User Interface (NetBEUI)	Sequenced Packet Exchange (SPX)	Transmission Control Protocol (TCP)	Sequenced Packet Protocol (SPP)	ISO 8073 (TP0-4)
Network Layer 3	Data Delivery Protocol (DDP)	Vines Internet Protocol (VIP)	Routing	Path Control		Internet Packet Exchange (IPX)	Internet Protocol (IP)	Internet Datagram Protocol (IDP)	ISO 8473 (CLNP)
Data Link Layer 2				Network Interface Cards:	Ethernet, Token-Ring, etc.	NIC drivers: ODI NDIS			
Physical Layer 1				Transport Media:	Twisted-Pair, Coax, etc.				

Tabelle 4: Einordnung verschiedener Protokollfamilien in das OSI-Modell [74]

2.1.2 Visualisierung des Internets

Es existieren mehrere Ansätze die Topologie des Internets zu beschreiben. So kann man die physikalische Struktur (Schicht 1 des OSI-Modells) eines TCP-basierten Netzes auf eine logische Struktur (oberhalb von Schicht 2 des OSI-Modells) abbilden.

Es gibt eine große Anzahl sinnvoller Abbildungen, die z.B. durch grafische Darstellung helfen können spezifische Strukturen des Internets hervorzuheben. Dazu sollen an dieser Stelle drei verschiedene Ansätze kurz vorgestellt werden: Die geographische Visualisierung, die topologische Visualisierung und die Visualisierung des Informationsraums.

2.1.2.1 Struktur des Internets für OSI-Schichten

Das Internet besteht aus einer Vielzahl von Rechnern, die in einzelnen Netzen organisiert sind. Diese Netze sind wiederum untereinander in Netzen organisiert. Die Topologie der Netze ist nicht festgelegt, es gibt stern-, bus- und ringförmige Netze, sowie Kombinationen davon. Kurz gesagt, das Internet ist in seiner physikalischen Struktur (mit Leitungen, Rechnern und zugehörigen Transfersystemen, wie Routern, Switches, etc.) ein Graph. Bis auf wenige Ausnahmen ist den einzelnen Knoten im Graphen (Kommunikationsendpunkte, wie einzelne PCs, Router, etc.) ein internetspezifisches Identifikationsmerkmal zugeordnet: Die IP-Nummer, die aus einem Tupel von vier Bytewerten (jeweils 0..255) besteht. Es bestehen Bestrebungen, diese IP-Nummern auf 16 Bytes zu erweitern (IPv6), um damit dem Wachstum des Internets und der damit verbundenen Knappheit an IP-Nummern Rechnung zu tragen.

Dieser physikalischen Struktur steht eine logische Struktur gegenüber, die hierarchisch gegliedert ist: Das Domainname System (DNS [70]). Damit wird der Graph der physikalischen Struktur auf den Baum der logischen Struktur abgebildet. Die Knoten im Baum sind die Domainnamen, die Blätter sind Kommunikationsendpunkte. Jedem Knoten im Graph der physikalischen Struktur sind ein oder auch mehrere Blätter im Baum der logischen Struktur zugeordnet. Es ist jedoch auch möglich, daß einem Blatt der logischen Struktur mehrere Knoten der physikalischen Struktur zugeordnet sind. Das ist jedoch nicht die Regel.

Der Zusammenhang zwischen der logischen und physikalischen Struktur soll nun an einem Beispiel verdeutlicht werden. Dieses Beispiel zeigt einen Auszug aus dem Netz der Universität Marburg im Jahre 1998, über welches zwei weitere Domains (scm.de und ipp.de) verbunden wurden. Demzufolge entsprechen die hier gezeigten Namen, Hierarchien und IP-Nummern nicht mehr dem derzeitigen Stand. In *Abbildung 7* wird die Hierarchie der physikalischen Struktur gezeigt. Die Hierarchie wird bei dieser Abbildung durch die Anzahl und Abfolge der Transitsysteme bestimmt. Eine solche Skizze kann man durch das Verfolgen eines IP-Paketes anfertigen (traceroute Befehl). Hierbei ist zu beachten, daß nur solche Systeme erfaßt werden, die auf der IP-Ebene oder höher arbeiten. Dem gegenübergestellt (*Abbildung 8*) ist die Hierarchie der logischen Struktur des DNS. Die Hierarchie des DNS ergibt sich ausschließlich aus den zugeteilten Rechnernamen und läßt keine Rückschlüsse auf die physikalische Struktur zu.

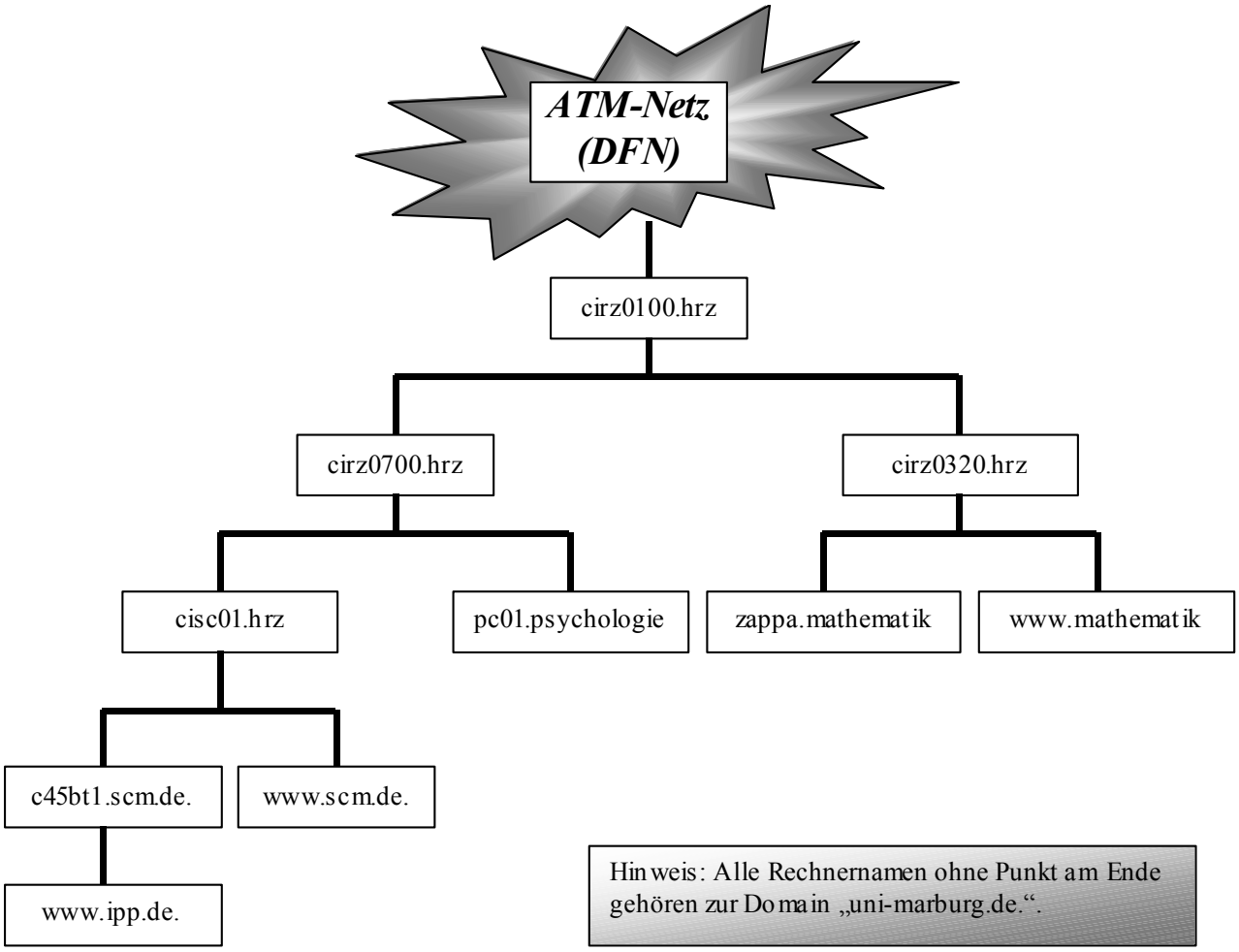

Abbildung 7: Hierarchie der physikalischen Struktur (Auszug, Stand 1998)

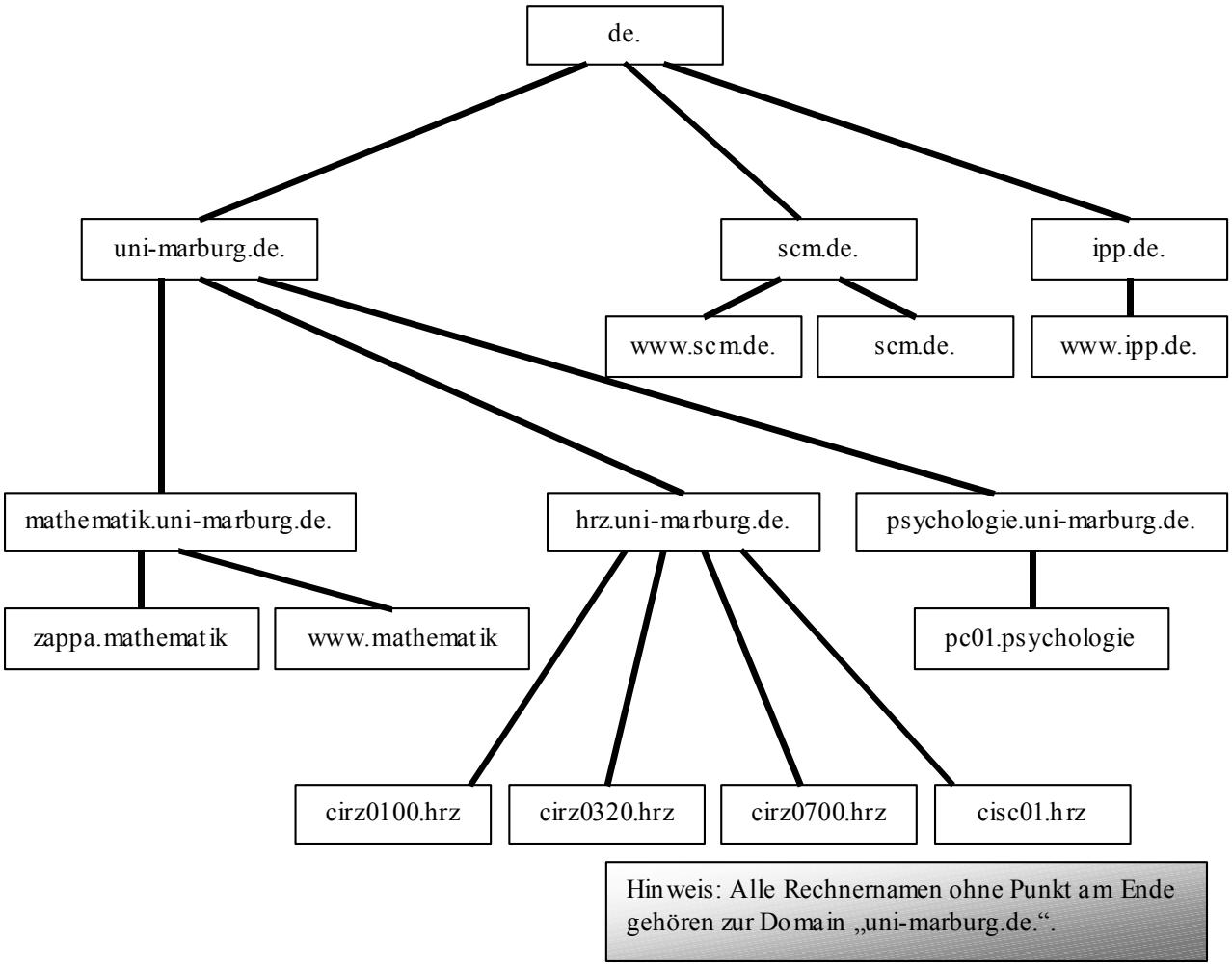

Abbildung 8: Hierarchie der DNS Struktur (Auszug, Stand 1998)

Weitere Skizzen zur Struktur lassen sich über den Verlauf der Verbindungsleitungen zwischen den einzelnen Rechnern und über die IP-Nummern anfertigen. Obwohl die Abbildung der IP-Nummern auf die Rechnernamen im DNS i.A. ein-eindeutig ist, haben die Skizzen zur IP-Nummernhierarchie und DNS-Hierarchie meist nur wenig gemeinsam. Der Grund dafür liegt in der Verteilung der Rechner einer Domain (alle Rechner einer Domain haben per Definition eines gemeinsam: den Domainnamen, z.B. mabi.de) im Internet: So kann z.B. der Mailserver einer ausgewählten Domain MeineDomain.de in einem Subnetz der Domain scm.de (IP-Nummern: 194.95.72.0) betrieben werden, während der WWW-Server in einem Subnetz des Internetproviders isp.com in USA (IP-Nummern: 207.240.40.0) lokalisiert ist.

Das Internet ist wie bereits erwähnt, nur in wenigen Aspekten zentral organisiert. Zu diesen zentralen Organisationen gehört die ICANN (Internet Corporation for Assigned Names and Numbers), welche für die IP-Adressraumvergabe, Protokoll-Definitionen, DNS Management und Rootserver zuständig ist. Diese Organisation hat im Oktober 1998 die IANA (Internet Assigned Number Authority) abgelöst, wo auch die Wurzeln der Domainnamenstruktur verwaltet wurden. Diese einst strikt zentrale Verwaltung der IP- und Domainnamen wurde über die Jahre immer weiter an dezentrale Organisationen delegiert, wie z.B. seit Dezember 1996 an das DE-NIC, welches u.a. für die Registrierung der .de-Domains in Deutschland zuständig ist.

2.1.2.2 Geographische Visualisierung

Das NCSA führte eine Visualisierung der Daten über das Transfervolumen des NSFNet durch [73]. Dazu wurde der Datenverkehr in den ANS/NSFNET T3 Backbone für den Monat Dezember 1994 betrachtet. Das Ergebnis ist eine Landkarte (*Abbildung 9*), bei der die Verkehrsdichte durch Farben visualisiert wurde (Lila: geringer Verkehr, Weiß: 1 Trillion Bytes).

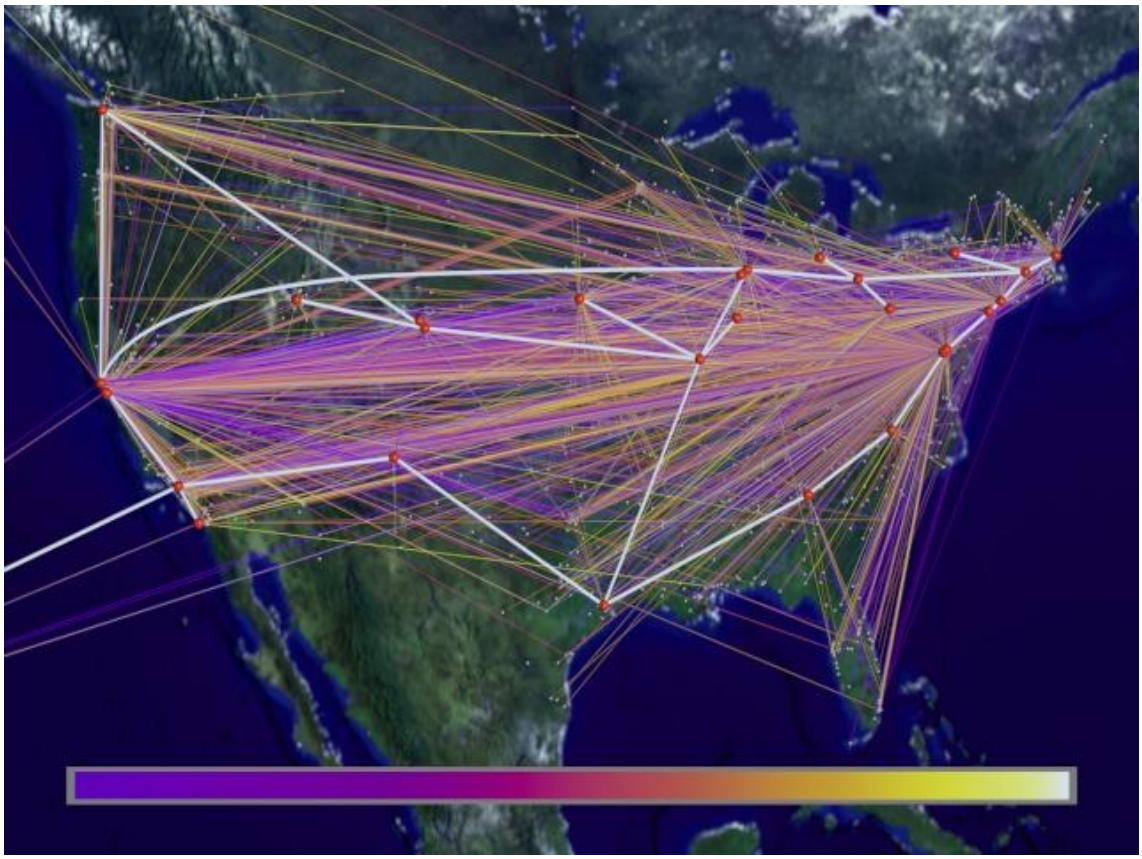

Abbildung 9: Transfervolumen des NSFNet Backbone (Stand 12/1994)

2.1.2.3 Topologische Visualisierung

Beispiele für eine topologische Visualisierung findet man oftmals bei den Netzanbietern, wie UUNET. Dort wird die Topologie der physikalischen Verbindungen vor dem Hintergrund einer Landkarte dargestellt (*Abbildung 10*).

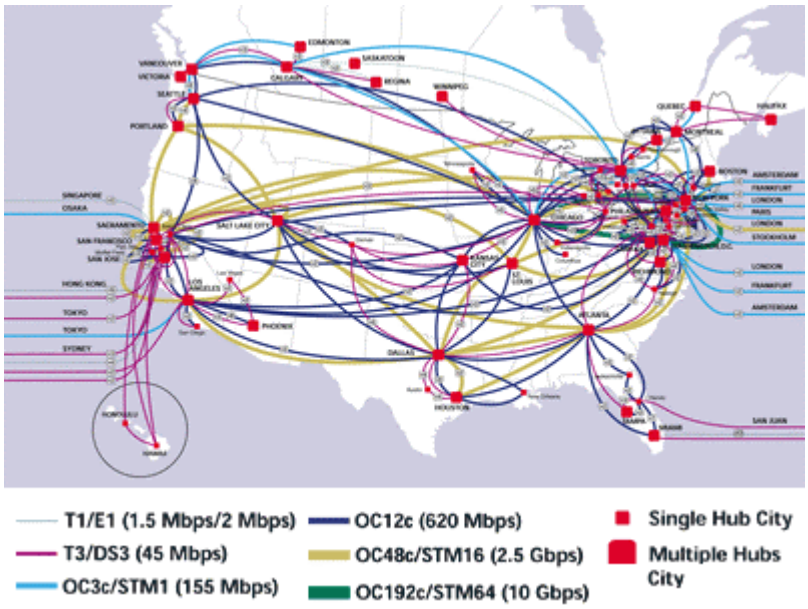

Abbildung 10: Topologie des ISP UUNET in USA (Stand 06/2000) [107]

2.1.2.4 Visualisierung des Informationsraums

Eine weitere Möglichkeit, die Topologie des Internets darzustellen, ist die Visualisierung der im Internet vorhandenen Informationsquellen. Mit diesem Thema beschäftigen sich zahlreiche Forschungsgruppen aus der Künstlichen Intelligenz. Das Prinzip besteht darin, über neuronale Netze die Information zu klassifizieren und dann in geeigneter Weise darzustellen.

So hat z.B. Luc Girardin vom "Graduate Institure of International Studies" [34] den Versuch unternommen, den Cyberspace (der Raum der im Internet verfügbaren Informationen) geografisch zu visualisieren (*Abbildung 11*). Er hat ein Kohonennetz verwendet, um eine Darstellung des Internets, bzw. dessen Inhalts zu erzeugen. Der Ausgangspunkt dieser Überlegung ist ein hochdimensionaler Raum, in dem die Informationen als Punkte enthalten sein. Zur Visualisierung muß nun die Dimension reduziert werden. Diese Aufgabe wird vom Algorithmus der selbstorganisierenden Karten ausgeführt. Die Orte der Informationsquellen wird in dem resultierenden Bild mit Kreuzen dargestellt. Die Farbe wurde durch empirische Daten, wie die Anzahl der Links dieser Informationsquelle, ermittelt.

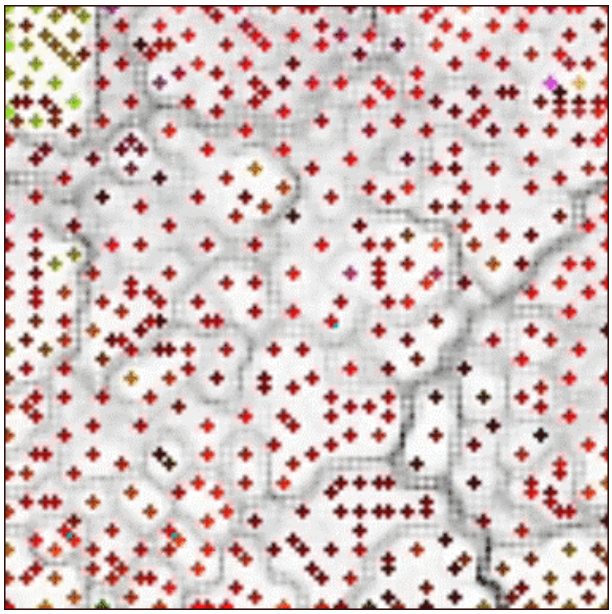

Abbildung 11: Karte von verschiedenen Informationsquellen in der Domain .ch

Ein weiterer Ansatz aus der Künstlichen Intelligenz, die Informationsflut aus dem Internet in einer Struktur zu organisieren, kommt aus dem ”Neural Networks Research Centre of Helsinki University of Technology” [51]. Hier hat man selbstorganisierende Karten verwendet, um verschiedene Dokumente in Abhängigkeit von ihrer Ähnlichkeit auf einer Karte darzustellen. Als Ergebnis erhält man eine Karte (*Abbildung 12*), auf der sich ähnliche Dokumente nahe beieinander befinden. Zudem sind charakterisierende Stichworte eingefügt. Als Beispiel wurden die Artikel einer News-Gruppe (comp.ai.neuronal-nets) auf eine solche Karte abgebildet.

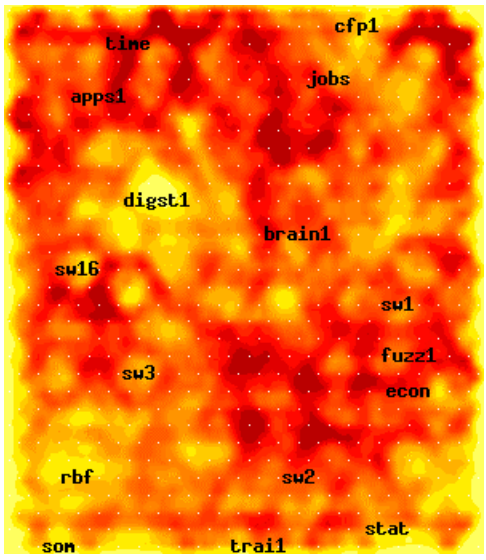

Abbildung 12: Karte der Newsgruppe comp.ai.neural-nets

2.2 Protokolle

In diesem Kapitel werden die Protokolle besprochen, die mit der WWW-Kommunikation in Internet und Intranet in Verbindung stehen. Die Protokolle werden dabei nach der Reihenfolge der Schichten im OSI-Modell von der Sicherungsschicht (Schicht 2) bis zur Anwendungsschicht (Schicht 7) vorgestellt. Die Bitübertragungsschicht (Schicht 1) wird nicht weiter betrachtet, weil eine solche Diskussion aufgrund der Hardwareabhängigkeit nur wenig zum Thema der Optimierung von Internet-Protokollen beitragen kann.

2.2.1 Sicherungsschicht

Die Sicherungsschicht (engl.: data link layer) hat fünf Aufgaben [105]: Sie legt die Schnittstelle zu der im OSI-Modell darüberliegenden Vermittlungsschicht fest. Das ist notwendig, um von der Vermittlungsschicht transparent auf den Datenfluß zuzugreifen. Desweiteren regelt sie die Zusammenstellung der Bits aus der unterliegenden Bitübertragungsschicht in Rahmen, sie nimmt eine Fehlerbehandlung bei der Übertragung dieser Rahmen vor und regelt den Rahmenfluß, sorgt also nach bestimmten Algorithmen dafür, daß bei mehreren Stationen im Netz ein kooperativer Datenaustausch gewährleistet wird. Schließlich ist sie noch für die Verbindungsverwaltung zuständig, sorgt somit für den korrekten Auf- und Abbau der Verbindungen auch im Falle eines Fehlers.

Es werden hier zwei Beispiele für Sicherungsprotokolle diskutiert: Ethernet und ATM. Ethernet wird ausschließlich im LAN-Bereich eingesetzt, nicht zuletzt aufgrund der geringen Kosten. ATM hingegen ist ein typisches WAN-Protokoll und wird z.B. im WiN des DFN-Vereins eingesetzt.

Weitere Protokolle, die man der Sicherungsschicht zuordnet, sind Token Ring, FDDI (Fiber Distributed Data Interface), ATM, SLIP (Serial Line IP) und PPP (Point-to-Point Protocol).

2.2.1.1 Ethernet

Der Ethernet Standard spielt eine zentrale Rolle im LAN-Bereich (Local Area Network). Die überwiegende Zahl der PCs (Blätter im Baum der Netzstruktur) im Internet sind über einen Ethernetadapter an ein LAN angeschlossen. Das heute verwendete Ethernetprotokoll arbeitet im Basisband und verfügt über eine Kollisionserkennung. Übertragungsraten sind 10 MBit, 100 MBit und 1 GBit.

Der Vorläufer des Ethernet Standards wurde in den Labors der Firma Xerox (Xerox Palo Alto Research Center) im Rahmen von Forschungsarbeiten über das "Büro der Zukunft" entwickelt. Die Spezifikation wurde als Ethernet Patent #4063220 der Erfinder Robert M. Metcalfe, David R. Boggs, Charles P. Thacker und Butler W. Lampson im Jahre 1977 unter dem Titel "Multipoint data communication system with collision detection" registriert [66]. Das erste experimentelle System hatte eine Übertragungrate von 3 Mbps. Der Begriff Ethernet spezifiziert die Verkabelung und das Signalverhalten eines Netzwerkes für die OSI-Schichten 1 (physical) und 2 (data link). Diese Definition wurde ursprünglich von Xerox in den späten 70ern für ein Basisband (baseband), CSMA/CD-basiertes Computernetzwerk über Koaxkabel vorgenommen.

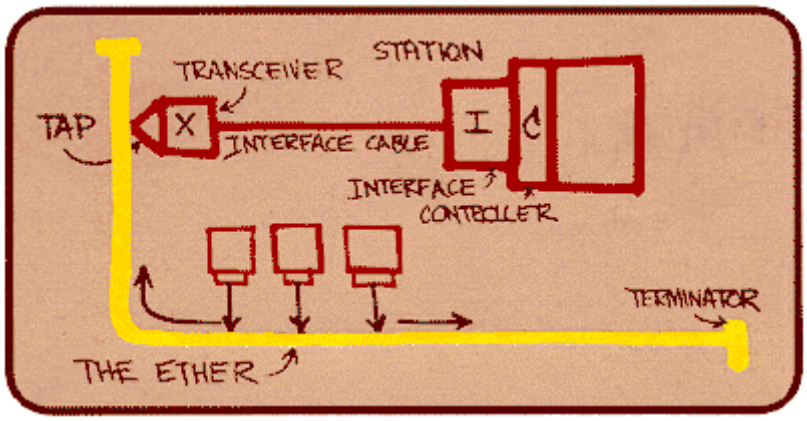

Abbildung 13: Skizze des ersten Ethernetsystems von Dr. Robert M. Metcalfe, 1976 [65]

In einer Kooperation der Firmen DEC, Intel und Xerox wurde 1980 der Standard Ethernet I im "Blue Book Standard" (DIX-Ethernet) festgelegt und im Jahre 1985 in Ethernet II erweitert. Das IEEE (Institute of Electrical and Electronic Engineers) entwickelte dann auf Basis der Ethernet II Definition den Ethernet 802.3 CSMA/CD Standard, der kompatibel zum Ethernet II Standard ist und sich im wesentlichen nur im Paketkopf (Network Packet Header) unterscheidet. Der IEEE Ethernet Standard mit dem formalen Titel "IEEE 802.3 Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications" wird ständig um neue Technologien, wie Twisted-Pair als Übertragungsmedium, 100 Mbit Fast Ethernet und GBit Ethernet erweitert.

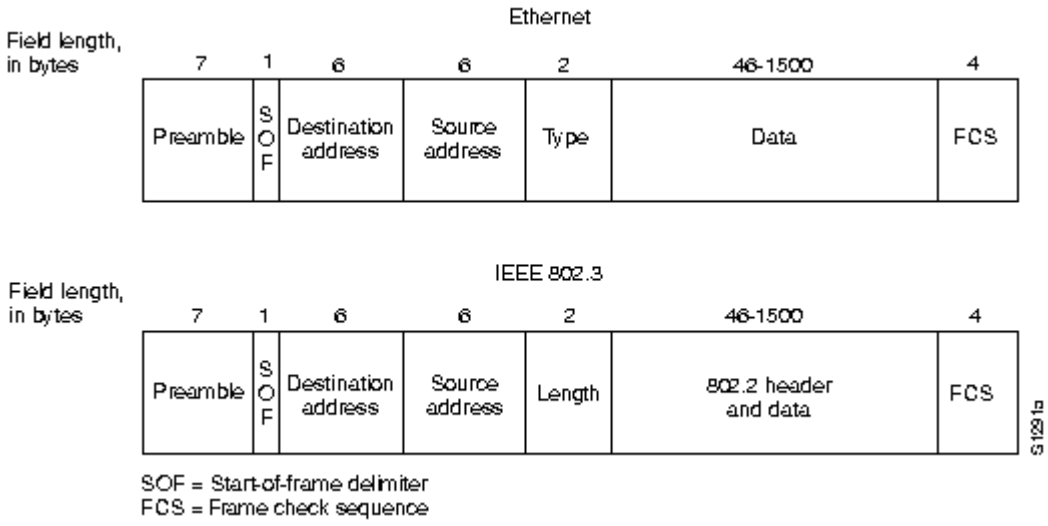

Abbildung 14: Ethernet Packetformat: Ethernet II und IEEE 802.3 [16]

Das Ethernet ist ein Basisband-Protokoll. Beim Basisband Netzwerk wird ein einziger Kanal zur Übertragung zur Verfügung gestellt, sodaß nur ein Gerät zu einem Zeitpunkt Daten übertragen kann. Das übertragende Gerät kann die gesamte Bandbreite des Übertragungsmediums nutzen (Vergleich: Telefonleitung). Im Gegensatz dazu wird beim Breitbandnetzwerk (broadband network) das physikalische Übertragungsmedium in mehrere logische Kanäle unterteilt, z.B. duch Verwendung unterschiedlicher Trägerfrequenzen (Vergleich: Kabelfernsehen) [64].

Das bereits angesprochene CSMA/CD Verfahren ist ein Kontrollmechanismus für den Zugriff auf das Transportmedium, legt also fest, in welcher Weise ein Paket in das Netzwerk eingespeist wird. Es wird bei Ethernet und 802.3 Netzwerken verwendet. CSMA/CD steht für ”Carrier Sense Multiple Access with Collision Detection”. Bei diesem Verfahren horcht zunächst ein am Netz angeschlossenes Gerät, ob eine Datenübertragung stattfindet. Ist dies nicht der Fall, so speist es das zu übertragende Datenpaket in das Netz ein, während es weiterhin am Netz darauf achtet, ob ein anderes Gerät ebenfalls zu senden angefangen hat. Ist dies der Fall, so ist der Zustand der Kollision eingetreten. Bei einer Kollision versuchen nun alle betroffenen Geräte, ihr Datenpaket erneut zu senden. Dazu warten die Geräte zunächst eine zufällige Zeit lang, horchen dann am Netz, ob eine Übertragung stattfindet und senden sobald das Netz frei ist.

Das Ethernet Paket ist aus drei Teilen aufgebaut: Der Kopf, der Datenbereich und eine abschließende Prüfsumme. Der Kopf und die Prüfsumme wird i.a. von der Ethernet Hardware direkt erzeugt. Der Datenbereich mit Ethernet Quell- und Zieladresse, Länge und Typ des Paketes, sowie den zu transferierenden Daten wird von der steuernden Software (Netzwerk-Treiber) erzeugt. Der Aufbau des Ethernetpakets ist in der folgenden Tabelle zu sehen.

Länge	Generator	Inhalt
62 Bit	Hardware	Präambel: Eine Serie von Bits, die vom Ethernet Empfänger zur Synchronisation benutzt werden
2 Bit	Hardware	Start Of Frame Delimiter – Trennzeichen, um gerade Bytegrenzen zu erzeugen
6 Byte	Software	Zieladresse des Ethernet Empfängers
6 Byte	Software	Quelladresse des Ethernet Senders
2 Byte	Software	Paketlänge (IEEE 802.3) oder Typenfeld (Ethernet I und II) – Ethernet Typen sind > 1500 (IP-Typ: 0x800)
46 Byte - 1500 Byte	Software	Datenbereich, kurze Pakete müssen bis 46 Byte aufgefüllt werden
4 Byte	Hardware	Frame Check Sequence – 32 Bit CRC Prüsumme mit dem AUTODIN II Polynom

Tabelle 5: Aufbau Ethernetpaket

Damit ergibt sich für die minimale Ethernet-Paketlänge 6 + 6 + 2 + 46 Byte = 60 Byte und für die maximale Paketlänge 6 + 6 + 2 + 1500 Byte = 1514 Byte.

2.2.1.2 ATM

ATM (Asynchronous Transfer Mode) ist eine Übertragungstechnologie, die dafür vorgesehen wurde, alle Kommunikationsarten, vom Telefon über Daten bis zum Video, über ein gemeinsames Netz zu transportieren [89]. Dazu werden die zu übertragenden Daten in Paketen, sogenannten

ATM-Zellen, mit der festen Länge von 53 Bytes, wovon 48 Byte Nutzlast (Payload) sind, in einem asynchronen Zeitmultiplexingverfahren übermittelt. Im Gegensatz zum synchronen Zeitmultiplexingverfahren werden hierbei die einzelnen Felder mit Empfängeradressen versehen, sodaß man keine fixierte Bandbreite hat, sondern auf variable Kapazitätsanforderungen reagieren kann. Zeitmultiplexing bedeutet hier, daß über einen realen Übertragungskanal in zeitlicher Abfolge mehrere virtuelle Kanäle Daten übermitteln.

Im ATM-Standard sind drei Schichten zum Transport definiert: Der Physical Layer, der ATM Layer und der ATM Adaption Layer. Die Einordnung der ATM-Schichten in das OSI-Modell ist in *Abbildung 15* zu sehen.

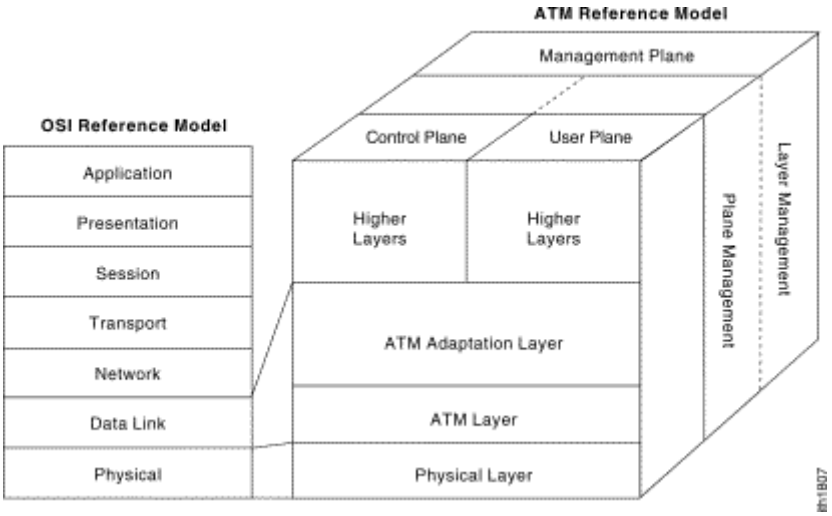

Abbildung 15: ATM Reference Modell [14]

Der Physical Layer bildet die physikalische Infrastruktur. Es sind eine Reihe von Protokollen in Abhängigkeit von Bandbreite und Kabeltyp definiert. Ein üblicher Standard ist STM-1, der auch im WAN des DFNs für den Betrieb des Wissenschaftsnetzes B-WiN eingesetzt wird. Der physikalische Träger ist ein Single- oder Multimode-Lichtwellenleiter und die Bandbreite beträgt 155,520 MBit/s. Weitere Medien sind Shielded Twisted-Pair und Koaxial, Bandbreiten liegen im Bereich zwischen 1,544 Mbit/s und 622 Mbit/s.

Der ATM-Layer hat die Aufgabe, virtuelle Verbindungen aufzubauen und die einzelnen ATM-Zellen an ihr Ziel weiterzuleiten. Dazu müssen die Zellen in die virtuellen Verbindungen mit Hilfe von (De-)Multiplexen ein- bzw. aussortiert werden, die Verbindungsparameter in den ATM-Switches angepasst werden, der ATM-Header vor der Weiterleitung an die darüberliegende Schicht entfernt bzw. hinzugefügt werden und die Datenflußkontrolle ausgeübt werden.

Der ATM-Adaption Layer wird wie auch der ATM-Layer der Schicht 2 des OSI-Modells zugeordnet. Er nimmt eine Anpassung zwischen den höheren Protokollschichten (im Falle der Datenkommunikation kann das z.B. IP sein) und den ATM-Zellen vor. Je nach Anwendung, die sich durch das Timingverhalten, Bitrate und Verbindungsmodus charakterisieren läßt, muß es eine spezifische Anpassung zwischen dem ATM-Layer und den höheren Verbindungsprotokollen geben. Diese wird mit den ATM-Adaption Layern AAL1 bis AAL5 vorgenommen. Diese Zuordnung mit den Charakterisierungen der Dienstklassen ist in Tabelle 6 zu sehen.

Klasse	A	B	C	D
Inhalt	Sprache	Video	Verb.orientierter Datenservice	Verb.loser Datenservice
Verbindungsmode	Verb.orientierter	Verb.orientierter	Verb.orientierter	Verb.loser
Bitrate	Konstant	Variabel	Variabel	Variabel
Timing	Synchron	Synchron	Asynchron	Asynchron
AAL-Layer	AAL1	AAL2	AAL3, AAL5	AAL4, AAL5

Tabelle 6: ATM-Dienstklassen und zugehörige AAL-Schichten [89]

2.2.2 Vermittlungsschicht

Die Aufgabe der Vermittlungsschicht [105] ist es, Pakete vom Ursprung zum Ziel zu bringen. Dabei kann im Gegensatz zur Sicherungsschicht das Springen der Pakete über mehrere Netzknoten notwendig sein, also über Subnetzgrenzen hinaus. Daher muß die Vermittlungsschicht Informationen über den Aufbau der betroffenen Subnetze haben und anhand dieser Informationen geeignete Pfade für die Pakete durch das Netz wählen. Naheliegenderweise werden nun exemplarisch für die Vermittlungsschicht die Internetprotokolle IPv4 und IPv6 besprochen.

2.2.2.1 IP und IPv4

Das IPv4-Protokoll, oder allgemein IP (engl.: Internet Protocol) gehört zu der TCP/IP Internet Protokollfamilie. Es ist im OSI-Schichtenmodell in der Vermittlungsschicht (network layer) positioniert. Somit sind die IP-Pakete (engl.: datagram) in den Paketen der Sicherungsschicht (data link layer) eingekapselt. Das IP-Protokoll ist die einzige Schnittstelle der Protokolle ICMP, IGMP, UDP und TCP in die Richtung des Transportmediums. Will somit eines der genannten Protokolle mit einem anderen Netzelement kommunizieren, so muß es seine Daten der Schnittstelle des Internetprotokolls übergeben. Das Durchlaufen jeder Schicht ist jedoch nicht in jedem Fall notwendig. So kann z.B. eine Anwendung die Transportschicht TCP/UDP überspringen und direkt mit der Schnittstelle IP in der Vermittlungsschicht kommunizieren. In der folgenden Abbildung werden die Kommunikationkanäle zwischen den einzelnen Protokollen in den Schichten dargestellt.

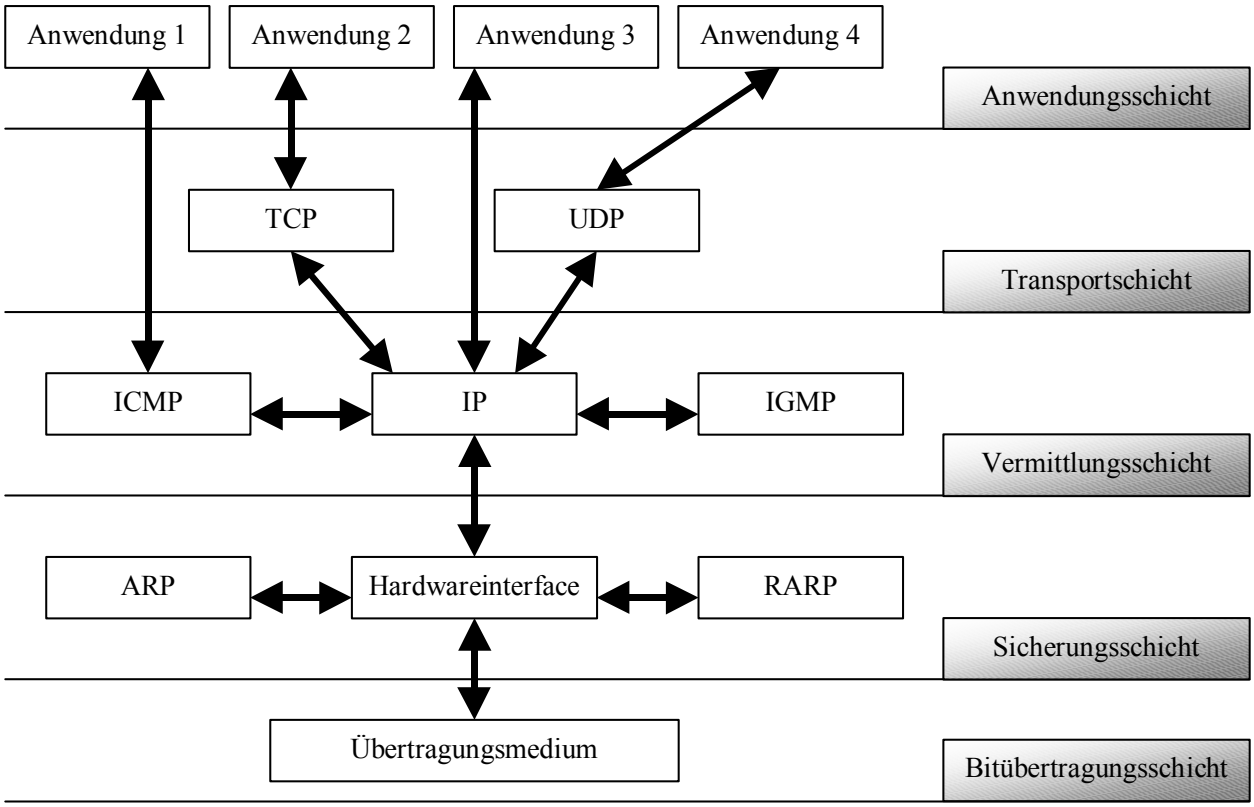

Abbildung 16: Kommunikationskanäle in den Schichten der TCP/IP Protokollfamilie [102]

Zwei wesentliche Eigenschaften des Internetprotokolls IP sind die Unsicherheit der übertragenen Pakete und die Verbindungslosigkeit der IP-Datenübermittlung. Unsicherheit meint in diesem Zusammenhang, daß es keine Garantie für die erfolgreiche Zustellung eines IP-Pakets gibt. Tritt ein Fehler in der Übermittlung eines Paketes auf, so wird das Paket ignoriert und eine ICMP-Nachricht zur Quelle zurückgeschickt. Die Sicherheit, daß jedes Paket beim Empfänger ankommt, muß von den übergeordneten Schichten (z.B. TCP) gewährleistet werden. Ein verbindungsloser Übertragungservice gewährleistet nicht, daß die einzelnen Pakete in der richtigen logischen Abfolge beim Empfänger ankommen. So kann es beim IP-Transport vorkommen, daß Paket T2 zwar nach Paket T1 vom Sender weggeschickt wird, jedoch vor Paket T1 beim Empfänger ankommt. Die Empfangsreihenfolge der Pakete beim Empfänger muß somit nicht geordnet sein.

Ein IP-Paket besteht aus einem Kopf (header) und einem Datenteil. Der IP-Header hat eine Länge von 20 Bytes, wenn keine zusätzlichen IP-Optionen übertragen werden. Das gesamte IP-Paket kann eine Länge von bis zu 65535 Bytes haben. Der genaue Aufbau eines IP-Paketes ist in *Tabelle 7* zu sehen.

Länge	Feldname	Beschreibung
4 Bit	IP version	IP Protokoll Version, i.A. = 4, auch IPv4 genannt
4 Bit	IP header length	Anzahl der 32 Bit-Wörter im IP header, die Optionen mit eingerechnet. Durch die Begrenzung auf 4 Bit ergibt sich für den Header eine maximale Länge von 15 * 4 Byte = **60 Byte**.
8 Bit	TOS	TOS bedeutet „type of service“. Das Feld beginnt mit einem 3 Bit Feld, das heute nicht mehr verwendet wird, dann folgt das 4 Bit lange TOS Feld, das von einem einzelnen Bit gefolgt ist, welches immer Null sein muß. Die vier TOS-Bits stehen für: „minimize delay“, „maximize througput“, „maximize reliability“ und „minimize monetary cost“. Nur eines dieser Bits darf gesetzt sein. Das RFC 1340 [88] spezifiziert, wie die Bits bei Standardanwendungen gesetzt werden sollten.
16 Bit	Total length	Die gesamte Länge des IP Datenpaketes in Bytes. Daraus ergibt sich auch die maximale IP Paketlänge von 65535 Bytes.
16 Bit	Identification	Identifiziert das IP-Paket eindeutig.
3 Bit	Flags	Kennzeichen für die Fragmentierungseigenschaften des IP-Pakets: „don’t fragment“, „may fragment“, „more fragments“ und „last fragment“.
13 Bit	Fragment offset	Das ist der Offset dieses Paketes zum Anfang des Orginalpaketes, welches fragmentiert wurde.
8 Bit	TTL	Wird auf die maximale Anzahl von Routerdurchläufen gesetzt. Durchläuft das Paket einen Router, so wird dieses Feld dekrementiert. Wird es Null, so wird das Paket verworfen.
8 Bit	Protokoll	Spezifiziert, von welchem Protokoll die Daten an das IP weitergereicht wurden. Z.B. TCP= 6, UDP= 17 [88]
16 Bit	Header checksum	Prüfsumme über den IP-Header. Dabei ist der Datenbereich nicht eingeschlossen. Für die Berechnung der Prüfsumme wird dieses Feld zunächst auf Null gesetzt. Dann wird die 16 Bit Einer-Komplement-Summe über den Header berechnet, wobei der Header als Reihe von 16 Bit Worten angesehen wird. Das Ergebnis wird in diesem Feld gespeichert. Der Empfänger bildet erneut die Prüfsumme, diesmal über den vollständigen Header, einschließlich dieses Prüfsummenfeldes. Wenn sich als Ergebnis nur Einsen in Bitdarstellung ergeben, so wurde der Header nicht verändert. Wurde der Header verändert, so wird das Paket kommentarlos gelöscht. Die anderen Protokollschichten müssen sich selbst um eine erneute Anforderung des Paketes kümmern.
32 Bit	Source IP address	IP-Adresse des Absenders
32 Bit	Destination IP address	IP-Adresse des Empfängers
4 Bytes*n	Options	IP-Optionen der Länge 4 Bytes * n, wobei n= 0, .., 10
?	Daten	Hier befindet sich der Datenbereich ..

Tabelle 7: IPv4 Paket

2.2.2.2 IPv6

Vor allem aufgrund des exponentiellen Wachstums des Internets und dem damit verbundenen Mangel an IP-Adressen hat man sich Gedanken zur Erweiterung des Internetprotokolls IPv4 gemacht. Das Resultat dieser Überlegungen ist das Protokoll IPv6 [20], bei dem gegenüber dem Protokoll IPv4 die folgenden Änderungen vorgenommen wurden:

Die Größe der IP-Adressen wurde von 32 Bit auf 128 Bit verändert. Damit erhält man mehr Ebenen bei der Adressierungshierarchie, eine größere Anzahl von adressierbaren Rechnern und eine einfachere Autokonfiguration von Adressen. Das Multicast-Routing wird durch die Einführung eines „Scope"-Feldes verbessert. Der neue Typ der „Anycast"-Adressen wird eingeführt, um ein Paket zu jedem Rechner einer Gruppe zu senden.

Der Header des IP Paketes wird vereinfacht, indem einige Felder des IPv4-Headers entfallen und andere optional werden. Damit wird sowohl der Verarbeitungsaufwand bei Transport gemindert. Inwieweit sich das Volumen des ganzen Paketes verändert, wird später betrachtet werden.

Die Unterstützung für Erweiterungen und Optionen wird verbessert, um das Weiterleiten der Pakete effizienter zu gestalten, die Beschränkungen der Optionslänge aufzuheben und eine größere Flexibilität für die Einführung zukünftiger Erweiterungen zu gewährleisten.

Durch die Einführung einer Datenflußbezeichnung können nun zusammengehörige Pakete, die zu einem speziellen Datenfluß gehören, eine Markierung in Form einer Bezeichnung erhalten. Dadurch ist es möglich, diesem Datenfluß bestimmte Dienstmerkmale wie QOS (Quality Of Service) oder Echtzeit zuzuweisen.

Es werden Erweiterungen für die Authentifizierung (authentication), Datenintegrität (integrity) und optionaler Datenverschlüsselung (confidentiality) in das Protokoll mit aufgenommen. Diese Anforderungen haben unter anderem durch die Kommerzialisierung des Internets gerade in letzter Zeit zur Kritik des IPv4 Protokolls geführt, welches diese Anforderungen nicht oder nur mit aufwendigen Erweiterungen unterstützt.

Ansonsten verhält sich das IPv6 Protokoll wie das IPv4 Protokoll: Das IPv6 Paket kapselt das Paket des darüberliegenden Protokolls (z.B. TCP) ein, indem es diesem Paket einen Header voranstellt. Beim IPv6 können nun zusätzlich zu dem IPv6 Header weitere optionale Header vorkommen. Diese werden dann dem eingekapselten Paket vorangestellt. In einem IPv6 Paket können auch mehrere dieser sogenannten Erweiterungsheader vorkommen. Sie werden über eine einfach verkettete Liste miteinander verbunden. Der Performancegewinn bei dieser Lösung gegenüber den im IPv4 Header fest eingebauten Optionen ergibt sich daraus, daß einige Optionen des IPv4s in die Erweiterungsheader des IPv6 ausgelagert wurden und die Erweiterungsheader beim Transitsystem nicht analysiert werden. Allerdings gibt es auch hier eine Ausnahme: Der Hop-By-Hop Header, der dem IPv6 Header unmittelbar zu folgen hat, muß natürlich vom Transitsystem beachtet werden. Eine Option dieses Headers ist die „Jumbo Payload Option", welche die Paketgröße für Pakete größer als 64 KByte enthält. In der folgenden Abbildung sind zwei Beispiele für ein IPv6 Paket zu sehen. Das eine Paket kommt ohne Erweiterungsheader aus, während das andere Paket zwei zusätzliche Erweiterungsheader enthält.

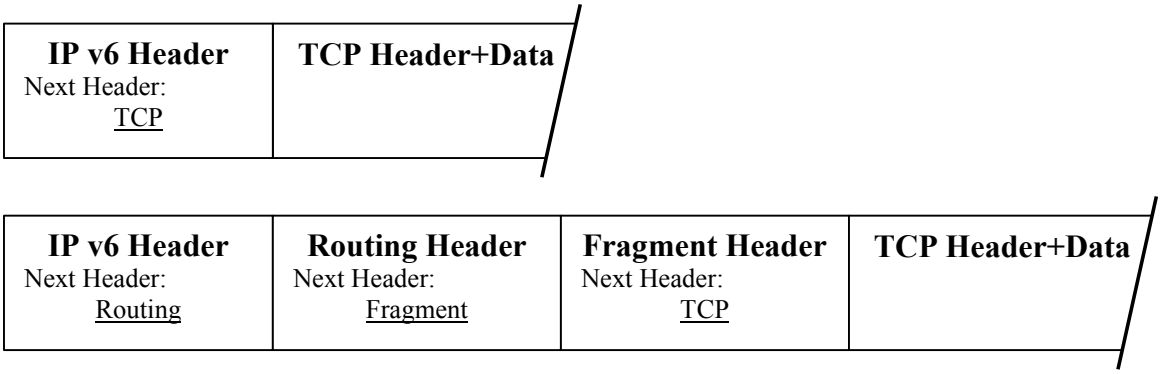

Abbildung 17: IPv6 Header und optionale Erweiterungsheader

Der IPv6 Header selbst (nicht zu verwechseln mit den optionalen Erweiterungsheadern) ist in der nachfolgenden Tabelle zu sehen.

Länge	Feldname	Beschreibung
4 Bit	IP version	IP Protokoll Version, für IPv6 hier = 6
4 Bit	Priority value	Prioritätsstufe 0 bis 7 für verbindungsorientierte Dienste und 8 bis 15 für verbindungslose Dienste. Die Werte 0 bzw. 8 stehen für die niedrigste Priorität.
24 Bit	Flow Label	Dieses Feld dient der Markierung der Pakete, für die eine besondere Behandlung, wie „Quality Of Service“, vorgesehen ist. Diese Option ist noch in einem experimentellen Stadium.
16 Bit	Payload Length	Die Länge des IP Datenpaketes in Bytes ohne IPv6 Header. Es ergibt sich auch die maximale IP Paketlänge von 65535 Bytes. Ist dieses Feld = 0, so ist die Paketlänge in der „Jumbo Payload Option“ enthalten.
8 Bit	Next Header	Identifiziert den Typ des nächsten Headers, entsprechend den Spezifikationen des entsprechenden IPv4 Feldes.
8 Bit	Hop Limit	Maximale Anzahl der Hops, die das Paket durchlaufen kann. Wird bei jedem Transitsystem dekrementiert.
128 Bit	Source Address	Die Adresse des Paketsenders.
128 Bit	Destination Address	Die Adresse des Paketempfängers.

Tabelle 8: IPv6-Paket Header

Läßt man die Länge der im Header enthaltenen IP-Adressen unberücksichtigt, so ist der IPv4 Header mit einer Länge von 12 Bytes 50% größer als der 8 Byte lange IPv6 Header.

Zur Zeit (Anfang 2000) gibt es bereits zahlreiche Implementationen vom IPv6 [2] für nahezu alle Betriebssysteme, jedoch überwiegend nur als Option. Mit diesen Implementationen ist es möglich,

lokal ein IPv6-Netzwerk aufzubauen. Zudem kann man sich über Initiativen wie den Freenet6-Service [109] über einen IPv4-Tunnel zum 6BONE [44] verbinden. 6BONE ist ein virtuelle Netzwerk innerhalb des Internets, welches mit IPv6 betrieben wird. Die Verbindung der einzelnen Hosts, die an diesem Netzwerk teilnehmen, sind über IPv4-Tunnel miteinander verbunden, werden jedoch langsam auf „reinen" IPv6-Transport umgestellt. Der Betrieb und die Weiterentwicklung der Spezifikationen wird von der „IETF IPng working group" [39] überwacht. IPng ist der Arbeitstitel für das Nachfolgeprotokoll von IPv4. Der formale Name für das Protokoll ist IPv6.

Der Entwicklungsstand von IPv6 wird als stabil bezeichnet, sowohl was die Spezifikationen der IPng Arbeitsgruppe angeht, als auch die Implementationen des Protokolls für Hosts und Router. Von einem flächendeckenden Einsatz ohne Verwendung von IPv4-Systemen ist man jedoch noch weit entfernt, auch wenn bereits weltweit Systeme am 6BONE angeschlossen sind [54] (Deutschland: 71 Systeme, Stand August 2000). In *Abbildung 18* sind die Netze bzw. Institutionen mit den Verbindungen aufgetragen, die am 6BONE angeschlossen sind.

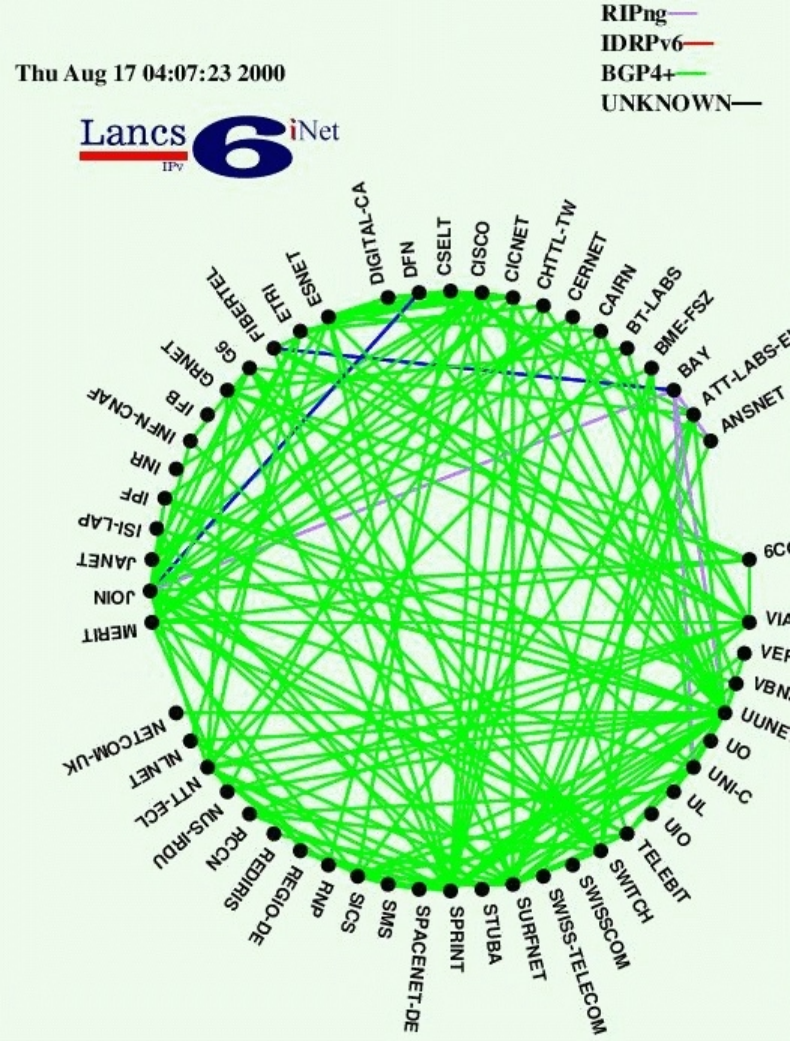

Abbildung 18: 6BONE konnektierte Netze, Stand August 2000 [44]

2.2.3 Transportschicht

Die Aufgabe der Transportschicht ist es, eine Schnittstelle zur Verfügung zu stellen, mit der die Daten von einem beliebigen Quellrechner zu einem Zielrechner übermittelt werden. Für die

Internetkommunikation über die TCP/IP Protokollfamilie bedeutet das, daß der Transport gegenüber dem Anwender völlig transparent abläuft, d.h. die Anwendungsprogramme in der über der Transportschicht liegenden Anwendungsschicht können über das TCP oder UDP kommunizieren, ohne das Transportmedium berücksichtigen zu müssen. Damit sind auch schon die beiden Protokolle der TCP/IP Familie in der Transportschicht genannt, die nun vorgestellt werden.

2.2.3.1 UDP

Das User Datagram Protocol (UDP) ist im OSI-Modell eine Schicht über dem IP angeordnet. Somit werden die Pakete des UDP dem IP übergeben. Das Internetprotokoll packt das UDP-Paket als Datenteil in ein IP-Paket ein und übergibt das IP-Paket der darunter liegenden Schicht. Die Einkapselung des UDP-Paketes ist in *Abbildung 19* zu sehen.

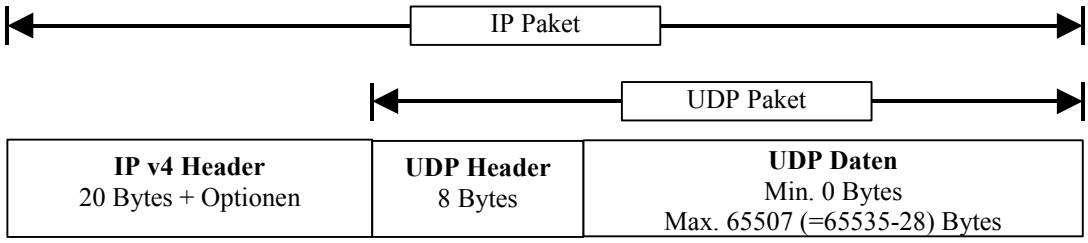

Abbildung 19: UDP Einkapselung

Das User Datagram Protocol [81] ist, wie auch das darunterliegende Internet Protocol, unsicher: Es wird nicht garantiert, daß die Pakete ihr Ziel erreichen, auch nicht, daß die Reihenfolge der Pakete beibehalten wird. Insgesamt sind die Eigenschaften mit denen des IP identisch. Nur im UDP Header stecken noch weitere Informationen, wie aus der folgenden Tabelle ersichtlich wird.

Länge	Feldname	Beschreibung
16 Bit	Source Port	Die Portnummer des Absenders
16 Bit	Destination Port	Die Portnummer des Empfängers
16 Bit	UDP length	Die Länge des vollständigen UDP Paketes (Header + Daten). Diese Information ist redundant, da die Länge bereits aus dem IP Header bekannt ist.
16 Bit	UDP checksum	Das ist die (optionale) Prüfsumme über das gesamte UDP Paket (Header + Daten). Im Gegensatz dazu bezog sich die Prüfsumme im IP Header nur auf den Header, nicht auf den Datenteil des IP Paketes.
?	Daten	Hier befindet sich der Datenbereich, min. 0 Byte, max. 65507 Byte

Tabelle 9: UDP Paket

Die Portnummern dienen der Zuordnung der UDP- und TCP-Pakete zu einem Dienst. So hat z.B. der „daytime“-Service (dieser liefert Datum und Tag in ASCII-Repräsentation) die Portnummer 13. Es findet somit eine zweites Demultiplexing auf dem Weg des IP Paketes zur Anwendung im Empfänger statt: Das erste Demultiplexing ist die Unterscheidung, welches Protokoll im Datenteil des IP-Paketes eingekapselt ist (UDP oder TCP). Das zweite Demultiplexing ist die Unterscheidung nach der Portnummer. Je nach Protokoll (UDP oder TCP) und Portnummer (0,..,65535) wird das

Paket an das für diese Kombination zuständige Programm, bzw. an den zuständigen Dienst weitergeleitet (z.B. der „daytime“-Daemon). Zudem ist eine optionale Prüfsumme für das gesamte Datenpaket vorgesehen.

Damit beschränken sich die Unterschiede des UDP zum IP auf die Einführung von Portnummern und die optionale Paketprüfsumme.

Zu den Anwendungen die UDP benutzten gehören der Internet Nameserver, das TFTP (Anwendung: Booten über das Netzwerk) und vor allem das NFS (Network File System), welches gemeinsamen Zugriff auf verteilte Dateisysteme im LAN realisiert.

2.2.3.2 TCP

Das TCP Paket ist, wie auch das UDP Paket, in ein IP Paket eingekapselt.

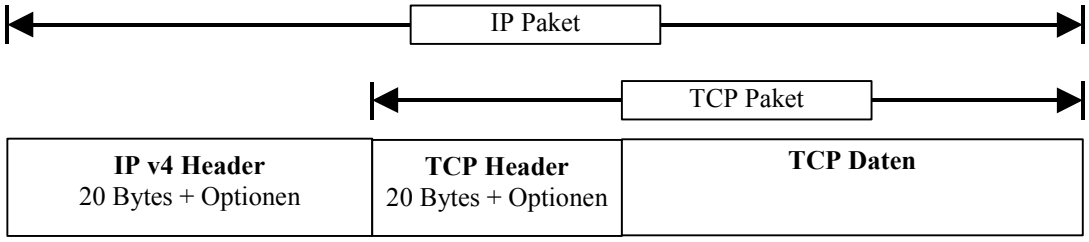

Abbildung 20: TCP Einkapselung

Das TCP [45] unterscheidet sich in seinen Eigenschaften erheblich vom IP und UDP. So werden bei diesem Protokoll kein verbindungsloser, unzuverlässiger Paketdienst zur Verfügung gestellt, sondern ein verbindungsorientierter, zuverlässiger Byte orientierter Datenstrom. In der folgenden Tabelle werden die Eigenschaften von IP, UDP und TCP gegenübergestellt.

Eigenschaft	IP	UDP	TCP
Verbindungsorientiert	Nein	Nein	Ja
Nachrichtenbegrenzung	Ja	Ja	Nein
Prüfsumme über Paket	Nur Header	Optional	Ja
Positive Bestätigung	Nein	Nein	Ja
Timeout mit erneuter Übertragung	Nein	Nein	Ja
Erkennung doppelter Übertragung	Nein	Nein	Ja
Überwachung der Reihenfolge	Nein	Nein	Ja
Überwachung des Datenflusses	Nein	Nein	Ja

Tabelle 10: Protokolleigenschaften von IP, UDP, TCP [99]

Aus dem Vergleich der Eigenschaften läßt sich bereits erkennen, daß die TCP-Schicht eine erhebliche zusätzliche Logik aufbringen muß, um eine zuverlässige, virtuelle Verbindung für die Anwendung bereitzustellen.

Der Aufbau eines TCP Paketes ist in *Tabelle 11* einzusehen.

Länge	Feldname	Beschreibung
16 Bit	Source Port	Die Portnummer des Absenders
16 Bit	Destination Port	Die Portnummer des Empfängers
32 Bit	Sequence number	Diese Nummer identifiziert die Bytes im Datenstrom vom Sender zum Empfänger. Dabei entspricht die sequence number dem ersten Byte in diesem Datensegment.
32 Bit	Acknowledgment number	Diese Feld enthält die nächste sequence number, die der Sender des Acknowledgements zu empfangen erwartet. Dieses Feld ist nur gültig, wenn der ACK-Marker gesetzt ist.
4 Bit	Header length	Anzahl der 32 Bit-Wörter im TCP header, die Optionen mit eingerechnet. Durch die Begrenzung auf 4 Bit ergibt sich für den Header eine maximale Länge von 15 * 4 Byte = **60 Byte**.
6 Bit	Reserved	Reservierte Bits.
1 Bit	Flag URG	Der urgent pointer ist korrekt.
1 Bit	Flag ACK	Die Acknowledgement number ist korrekt.
1 Bit	Flag PSH	Der Empfänger soll die Daten möglichst schnell an die Anwendung weitergeben.
1 Bit	Flag RST	Die Verbindung wird zurückgesetzt (Reset).
1 Bit	Flag SYN	Dieser Marker zeigt an, daß die sequence number zu Begin einer Verbindung synchronisiert werden soll.
1 Bit	Flag FIN	Der Sender hat die Datenübertragung abgeschlossen.
16 Bit	Window size	Anzahl der Bytes, die der Empfänger entgegen nehmen kann.
16 Bit	TCP checksum	Das ist die Prüfsumme über das gesamte TCP Paket (Header + Daten). Im Gegensatz dazu bezog sich die Prüfsumme im IP Header nur auf den Header, nicht auf den Datenteil des IP Paketes.
16 Bit	Urgent pointer	Der Offset, der zu der sequence number addiert werden muß, um die Position des letzten Bytes der urgent data zu bestimmen. Dieses Feld ist nur gültig, wenn der Marker URG gesetzt ist.
4 Bytes * n	Options	Diverse TCP Optionen, z.B. MSS (maximum segment size)
?	Daten	Hier befindet sich der Datenbereich, min. 0 Byte, max. 65507 Byte.

Tabelle 11: TCP Paket

Die meisten Anwendungen im Internet tauschen ihre Daten über das TCP aus. Dazu gehören Anwendungen, wie Telnet, Rlogin, FTP (File Transfer Protocol) und SMTP (Simple Mail Transfer Protocol).

2.2.3.3 T/TCP

Das T/TCP [11] (Transaction TCP) ist historisch betrachtet ein Nachfolger des TCP. In seiner Funktion liegt es jedoch zwischen Protokollen UDP und TCP, in seiner Einordnung in das OSI-Modell in der Transportschicht, also auf der Ebene von UDP und TCP. T/TCP steht für

Transaction/TCP, im Vergleich zum „normalen“ verbindungsorientiertem TCP und dem verbindungslosen UDP. Das Wort „Transaction“ soll andeuten, wo die Stärken des Protokolls liegen: Im Transaktionsbereich, also überall dort, wo eine große Anzahl von (kleinen) voneinander unabhängigen Request/Response-Kommunikationsverbindungen verlangt wird. Das können WWW-Verbindungen sein, aber ebenso FTP-Verbindungen oder Datenbankabfragen. Zu beachten ist, daß der Begriff der Transaktion in der deutschen Sprache kontextabhängig mehrfach überladen ist. Im Kontext dieser Arbeit soll der Begriff jedoch primär im Sinne des RFC 1644 [11] für die Transportschicht verstanden werden: Eine Transaktion ist eine atomare Sequenz aus einem request/response-Paar (Anfrage/Antwort-Paar) in der Transkationsschicht.

T/TCP ist vom Design [10] eine Erweiterung des TCP und zu diesem abwärtskompatibel, auch wenn es in den Implementierungen verschiedener Hersteller Performanceschwächen bei Verbindungen zwischen TCP und T/TCP Rechnern gibt [101].

Beim Filetransfer von großen Dateien mit FTP liegt das Augenmerk darauf, eine Datei von einem Rechner auf einen anderen Rechner zu transferieren. Dabei spielt z.B. das Antwortverhalten beim Auf- und Abbau der Verbindung mit zunehmender Dateigröße eine kleinere Rolle. Im Gegensatz dazu gibt es jedoch eine Reihe von Anwendungen, wie z.B. Datenbankabfragen, die darauf basieren, eine große Anzahl von Verbindungen in kurzer Zeit aufzubauen, eine (kleine) Datenmenge zu transferieren und dann die Verbindung wieder abzubauen. Solche Anwendungen sind auf ein gesichertes Protokoll angewiesen, somit scheidet UDP als Transportprotokoll von vorneherein aus. Jedoch erscheint auch die Alternative TCP für diese Anforderungen ungeeignet zu sein: Der Protokoll Overhead für den Transfer eines Datenpaketes ist aufgrund des Three-Way Handshakes erheblich. Zudem wird die Verbindung zu dem Server auf der Clientseite im TIME_WAIT Zustand für 240 Sekunden offen gehalten, was eine der 16 Bit Portadressen für diese Zeit blockiert. Somit ergibt sich eine Verbindungsrate von 64512 Verbindungen / 240 Sekunden = 268 Verbindungen pro Sekunde. Diese Schwächen des TCP und UDP in der Ausführung von Transaktionen wurden schon früh erkannt. So wurde von T. Miller 1985 das IRTP [69] (Internet Reliable Transaction Protocol) spezifiziert, ein Voll-Duplex transaktionsorientiertes Protokoll, das eine gesicherte Zustellung von Paketfolgen gewährleistet.

Im gleichen Jahr wurde von R. Braden Designkriterien [12] für ein transaktionsorientiertes Protokoll vorgestellt. Die Konzepte für eine Realisierung auf Basis des TCP folgten mit RFC 1379 im November 1992. Die erste Implemetierung des T/TCP wurde an einem SunOS [104] 4.1.3 Kernel im September 1994 durchgeführt, in FreeBSD [25] ist T/TCP seit der Version 2.0 im März 1995 enthalten. Für BSD/OS, welches wie auch FreeBSD und NetBSD auf 4.4BSD-Lite Quelltexten basiert, ist ebenfalls ein Kernelpatch vorhanden. Mittlerweile gibt es auch für Linux eine Implementation [97].

Das T/TCP versucht nun die oben erwähnten Schwächen des TCP zu korrigieren, indem im wesentlichen zwei Modifikationen vorgenommen werden: Das Three-Way Handshake wird vermieden und der TIME_WAIT-Zustand von 240 Sekunden auf 12 Sekunden verkürzt. Diese Änderungen wurden durch die Einführung drei neuer TCP Optionen realisiert, die sich mit der Behandlung des T/TCP Parameters CC (Connection Count) beschäftigen. Die Verbindungszahl CC ist eine pro Rechner eindeutige Zahl, die einer Verbindung zwischen einem Rechnerpaar zugeordnet wird. Somit muß jeder Rechner eine Tabelle pflegen, bestehend aus den beiden Spalten Rechner und Verbindungszahl CC. Dort sind alle Rechner und die zugehörigen Verbindungszahlen aufgeführt, mit denen er eine T/TCP Verbindung aufgebaut hat.

Die zur Zeit gültigen TCP Optionen mit den zusätzlichen T/TCP Optionen (Typ 11, 12 und 13) sind in der folgenden Tabelle aufgelistet.

RFC	TCP Option	Typ (1 Byte)	Länge (1 Byte)	Inhalt
793	End of option list (EOL)	0		
793	No operation (NOP)	1		
793	Maximum segment size (MSS)	2	4	Maximum segment size (2 Byte)
1323	Window scale factor	3	3	Shift count (1 Byte)
1323	Timestamp	8	10	Timestamp value (4 Byte), Timestamp echo reply (4 Byte)
1644	CC	11	6	Connection count (4 Byte)
1644	CCNEW	12	6	New connection count (4 Byte)
1644	CCECHO	13	6	Connection count echo (4 Byte)

Tabelle 12: TCP Optionen

Beim Verbindungsaufbau über das T/TCP unterscheidet man zwischen zwei Fällen: Der Verbindungsaufbau zu einem Ziel-Rechner, zu dem vorher noch keine Verbindung bestand und der Verbindungsaufbau für den Fall, daß bereits vorher schon einmal eine T/TCP-Verbindung bestanden hat. Der Unterschied liegt darin, daß nur beim ersten Verbindungsaufbau das vom TCP her bekannte Three-Way Handshake vorgenommen wird, dieses jedoch bei allen weiteren Verbindungen entfallen kann.

Beim ersten Verbindungsaufbau wird im Zuge des Three-Way Handshakes eine hostspezifische Verbindungsnummer CC ausgehandelt, die bei weiteren Verbindungen zur Identifikation des T/TCP-fähigen Hosts verwendet wird. Daraus folgt auch, daß z.B. der Server Host einen Cache von hostspezifischen CC-Nummern anlegen muß. Erhält dieser Server einen Request für einen Verbindungsaufbau von einem Client Host, so überprüft er zunächst, ob die Verbindungszahl CC vom Client größer als die von ihm gecachte CC-Nummer ist. Ist das der Fall, so kann das Three-Way Handshake entfallen. Ist keine solche Nummer vorhanden, oder seine gecachte CC-Nummer größer als die neu übermittelte, so wird ein Three-Way Handshake ausgeführt. Zudem wird durch das verbindungsabhängige Inkrementieren der Verbindungsnummer CC sichergestellt, daß keine Pakete in falscher Reihenfolge oder Duplikate entgegengenommen werden.

Einen anschaulichen Vergleich zwischen den beiden Fällen ist in *Abbildung 21* zu sehen, wo man auch in den TCP-Optionen die inkrementierten Verbindungszahlen wiederfindet.

T/TCP erste Verbindung

```
Packet 1
 IP   : srcip=137.248.121.166 destip=137.248.121.226 hlen=20 TOS=0x00
  TCP : srcport=1024 destport=8055 seqno=0x00a53f6d ackno=0x00000000
  TCP : hlen=48 (data=20) UAPRSF=0b001011 wnd=16728 cksum=0x8E59
  TCP : timestamp=0x72 timeecho=0x0 ccnew=3
   d  : 0x000 "GET / HTTP/1.0....."
Packet 2
 IP   : srcip=137.248.121.226 destip=137.248.121.166 hlen=20 TOS=0x00
  TCP : srcport=8055 destport=1024 seqno=0x88db8539 ackno=0x00a53f6e
  TCP : hlen=56 (data=0) UAPRSF=0b010010 wnd=17280 cksum=0x4871
  TCP : timestamp=0xf1e972 timeecho=0x72 cc=247 ccecho=3
   d  : <No data>
Packet 3
 IP   : srcip=137.248.121.166 destip=137.248.121.226 hlen=20 TOS=0x00
  TCP : srcport=1024 destport=8055 seqno=0x00a53f82 ackno=0x88db853a
  TCP : hlen=40 (data=0) UAPRSF=0b010001 wnd=17280 cksum=0xA326
  TCP : timestamp=0x72 timeecho=0xf1e972 cc=3
   d  : <No data>
Packet 4
 IP   : srcip=137.248.121.226 destip=137.248.121.166 hlen=20 TOS=0x00
  TCP : srcport=8055 destport=1024 seqno=0x88db853a ackno=0x00a53f83
  TCP : hlen=40 (data=0) UAPRSF=0b010000 wnd=17260 cksum=0xA246
  TCP : timestamp=0xf1e972 timeecho=0x72 cc=247
   d  : <No data>
Packet 5
 IP   : srcip=137.248.121.226 destip=137.248.121.166 hlen=20 TOS=0x00
  TCP : srcport=8055 destport=1024 seqno=0x88db853a ackno=0x00a53f83
  TCP : hlen=40 (data=169) UAPRSF=0b011001 wnd=17280 cksum=0xAB68
  TCP : timestamp=0xf1e972 timeecho=0x72 cc=247
   d  : 0x000 "HTTP/1.0 404 Not found....<HTML><HEAD></HEAD"
   d  : 0x010 "><BODY><H1>Error 404</H1>Not found - file doesn't exist or "
   d  : 0x020 " no permission..</BODY></HTML>...."
Packet 6
 IP   : srcip=137.248.121.166 destip=137.248.121.226 hlen=20 TOS=0x00
  TCP : srcport=1024 destport=8055 seqno=0x00a53f83 ackno=0x88db85e4
  TCP : hlen=40 (data=0) UAPRSF=0b010000 wnd=17111 cksum=0xA325
  TCP : timestamp=0x72 timeecho=0xf1e972 cc=3
   d  : <No data>
```

T/TCP weitere Verbindungen

```
Packet 1
 IP   : srcip=137.248.121.166 destip=137.248.121.226 hlen=20 TOS=0x00
  TCP : srcport=blackjack destport=8055 seqno=0x0121c959 ackno=0x00000000
  TCP : hlen=48 (data=20) UAPRSF=0b001011 wnd=17280 cksum=0x026F
  TCP : timestamp=0xca timeecho=0x0 cc=4
   d  : 0x000 "GET / HTTP/1.0....."
Packet 2
 IP   : srcip=137.248.121.226 destip=137.248.121.166 hlen=20 TOS=0x00
  TCP : srcport=8055 destport=blackjack seqno=0x895b7703 ackno=0x0121c96f
  TCP : hlen=56 (data=169) UAPRSF=0b011011 wnd=17280 cksum=0xD42C
  TCP : timestamp=0xf1e9ca timeecho=0xca cc=248 ccecho=4
   d  : 0x000 "HTTP/1.0 404 Not found....<HTML><HEAD></HEAD"
   d  : 0x010 "><BODY><H1>Error 404</H1>Not found - file doesn't exist or "
   d  : 0x020 " no permission..</BODY></HTML>...."
Packet 3
 IP   : srcip=137.248.121.166 destip=137.248.121.226 hlen=20 TOS=0x00
  TCP : srcport=blackjack destport=8055 seqno=0x0121c96f ackno=0x895b77ae
  TCP : hlen=40 (data=0) UAPRSF=0b010000 wnd=17111 cksum=0x25C1
  TCP : timestamp=0xca timeecho=0xf1e9ca cc=4
   d  : <No data>
```

Abbildung 21: Verbindungsaufbau T/TCP

Der Ablauf einer Verbindung während einer HTTP-Transaktion mit Gegenüberstellung der Protokolle T/TCP, TCP und UDP wird in (*Kapitel 3.2.2: Transportschicht*) diskutiert.

2.2.4 Anwendungsschicht

In der TCP/IP Internetkommunikation setzt die Anwendungsschicht direkt auf die Transportschicht auf, das heißt die Anwendungsprogramme rufen die Socketfunktionen von den Protokollen UDP und TCP auf. Die OSI-Schichten der Kommunikationssteuerung und Darstellung entfallen. Es gibt zahllose Beispiele für Anwendungen der Anwendungsschicht, denn jedes Programm, welches über die unterliegende Transportschicht kommuniziert, kann als ein solches Beispiel angesehen werden. Jede kommunizierende Anwendung implementiert für sich erneut ein Protokoll, im Falle der Anwendungsschicht ein Anwendungsprotokoll. Einige der populärsten Programme und zugehörigen Protokolle sind in *Tabelle 13* als Beispiele aufgeführt.

Anwendung	**Protokoll**
WWW-Server/-Browser	HTTP
E-Mail	SMTP
File Transfer	FTP
Netzwerk Verwaltung	SNMP
Terminal Emulation	TELNET

Tabelle 13: Protokolle der Anwendungsschicht

Das für diese Arbeit relevante Protokoll ist das des World Wide Webs, das Hypertext Transfer Protocol (HTTP), was im folgenden Abschnitt vorgestellt wird.

2.2.4.1 HTTP

Das Hypertext Transfer Protokoll (HTTP) bildet die Basis für das World Wide Web (WWW). Das HTTP ist ein einfaches Protokoll: Über das TCP wird eine Verbindung von einem Client zu einem Server aufgebaut, eine Anfrage gestellt und die Antwort darauf empfangen. Das empfangene Dokument ist eine Datei beliebigen Typs, im Normalfall ein Hypertext Dokument in der Beschreibungssprache HTML (Hypertext Meta Language) mit optionalen Links zu anderen Dokumenten. Der Client, auch Browser genannt, kann nun die im Dokument enthaltenen Referenzen auflösen, indem er weitere Serveranfragen stellt. Die Funktionalität des Servers beschränkt sich somit weitgehend auf den Transfer von angeforderten Dateien. Die wesentliche Arbeit wird dem Client überlassen, der die angeforderten Daten analysieren muß und in entsprechender Form darzustellen hat, bzw. weitere Daten anfordern muß. Vergleicht man das Verhältnis des Sourcecodes von Server zu Client, so kommt man im Fall eines UNIX-Systems auf ein Verhältnis von Server: 6500 Zeilen zu Client: 80000 Zeilen [100].

Die Entwicklung des Hypertext Transfer Protokolls (HTTP) hat ihre Ursprünge in Designüberlegungen von Tim Berners-Lee im Jahre 1991 [6]. Kurz darauf folgte vom W^3C eine Prototypen Implementation des HTTP. Diese Version wurde als HTTP/0.9 bekannt und umfaßte

einen Teil des vollen HTTP/1.0, welches im Jahre 1992 als Internet Draft von der Internet Engineering Task Force (IETF) veröffentlicht wurde. Es dauerte allerdings vier Jahre, bis die Spezifikation in der siebenten und letzten Version [7] als RFC 1945 im Mai 1996 freigegeben wurde. Bereits im Januar 1997 folgte dann die Veröffentlichung der Version HTTP/1.1. als RFC 2068 [24].

Das Protokoll HTTP folgt dem Anfrage-Antwort-Schema einer Client-Server Architektur. Ein Beispiel mit zwei Servern ist in *Abbildung 22* zu sehen. Bei diesem Beispiel einer HTTP-Kommunikation fordert der WWW-Browser (Client) vom WWW-Server 1 in einer Anfrage (1) ein Dokument an und erhält dieses in einer Antwort (2). Nach dem Parsen des Dokumentes löst der WWW-Browser einen im Dokument enthaltenen Hyperlink auf, indem er eine weitere Anfrage (3) an einen WWW-Server 2 stellt und von diesem ebenfalls ein Dokument in einer Antwort (4) enthält.

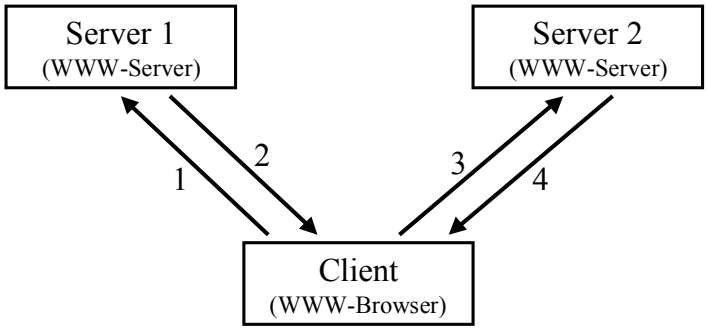

Abbildung 22: HTTP Client-Server Beispiel mit zwei Servern

Betrachtet man dieses Kommunikation näher, so kann man auch quantitative Aussagen über eine HTTP-Transaktion machen. In *Abbildung 23* ist die zeitliche Abfolge der Nachrichten aus *Abbildung 22* aufgetragen. Zu beachten ist dabei, daß diese Abbildung aus Sicht der Anwendungsschicht erstellt wurde.

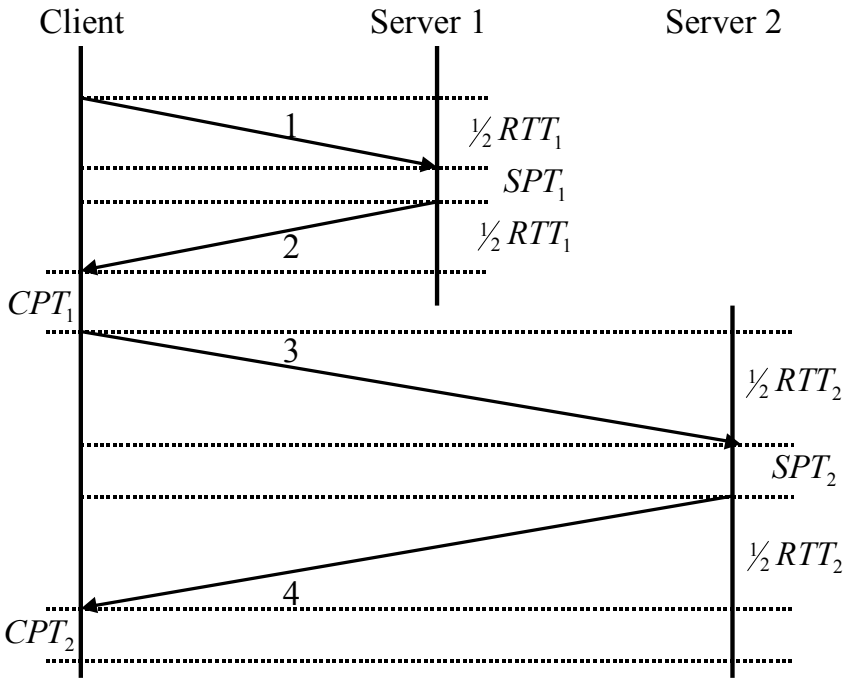

Abbildung 23: HTTP Client-Server Beispiel mit zwei Servern (Verbindungsdiagramm)

Dabei bezeichnet RTT_i (Round Trip Time) die Zeit, die eine Nachricht vom Client zum Server und wieder zurück benötigt, SPT_i (Server Processing Time) die Zeit, die der Server zum Bearbeiten der Anfrage benötigt und CPT_i (Client Processing Time) die Zeit, die der Client zum Bearbeiten des Dokumentes (Parsen, darstellen, etc.) benötigt. Somit ergibt sich allgemein als Zeit T_N für die eine WWW-Verbindung mit N URLs

$$T_N = \sum_{i=1}^{N} RTT_i + SPT_i + CPT_i$$

Beim HTTP gibt es zwei verschiedene Nachrichtentypen: Anfragen (request) und Antworten (response). Sie werden im ASCII-Format übertragen, genauer gesagt im 8 Bit ISO-Latin1 Zeichensatz. Die Anfrage besteht aus einer Anfrage-Zeile, Optionszeilen und einem optionalen Inhaltsteil. Die Antwort besteht aus einer Status-Zeile, einem Block aus weiteren Informationen über die Antwort und ebenfalls einem optionalen Inhaltsteil. Diese Nachrichtentypen werden im folgenden in EBNF formuliert und kurz erläutert.

Eine Anfrage in EBNF Notation hat die folgende Form:

```
Request=  Request-Line
                *( general-header
                | request-header
                | entity-header )
                CRLF
                [ message-body ]
```

Dabei ist `Request-Line` definiert als:

```
Request-Line    = Method SP Request-URI SP HTTP-Version CRLF
```

Method ist dabei eine der folgenden Anfrage-Methoden: `OPTIONS`, `GET`, `HEAD`, `POST`, `PUT`, `DELETE` und `TRACE`. Die Methode `OPTIONS` ermittelt die Kommunikationsoptionen, die bei der gegebenen Anfrage möglich sind. In der Antwort des Servers ist nicht das angeforderte Objekt (Request-URI) enthalten. Die am häufigsten verwendete Methode ist `GET`. Damit wird ein Objekt

angefordert. Je nach zusätzlichen Optionen ist die Anfrage ein bedingtes GET (If-Modified-Since, ..) oder ein beschränktes GET (Range, ..). Die Syntax und das Ergebnis der `HEAD`-Anfrage ist mit dem der GET-Anfrage identisch, mit der Ausnahme, daß das angeforderte Objekt nicht von Server übertragen wird. Diese Anfrage wird zumeist zur Überprüfung von Aktualität, etc. verwendet. Die `POST`-Methode sendet ein Objekt an den Server zur Weiterverarbeitung. Ein Einsatzgebiet ist der Transfer von Informationen aus Formularen (FORM). Die `PUT`-Methode dient dem Transfer eines Objektes zu dem spezifizierten Ziel Request-URI. Die DELETE-Methode löscht ein durch Request-URI benanntes Objekt. Mit der `TRACE`-Methode wird dem Client die beim Server angekommene eigene Anfrage übermittelt. Damit wird es ermöglicht, den Kommunikationsweg einer Anfrage vom Client aus zu verfolgen.

`Request-URI` (Uniform Resource Identifier) ist eine eindeutige Angabe des Objektes und HTTP-Version ist die verwendete Version des HTTPs. Das Token `HTTP-version` ist eine Zeichenkette, welche die verwendete Version des HTTPs repräsentiert. Zur Zeit gültige Werte sind: `HTTP/0.9`, `HTTP/1.0` und `HTTP/1.1`.

Über die Optionen `general-header` werden ausschließlich Optionen für die Nachricht selbst, nicht für den übertragenen Inhaltsteil (message-body) angegeben. Dazu gehören u.a. die Felder `Cache-Control` und `Pragma`.

Der `request-header` spezifiziert zusätzliche Optionen für die Anfrage. Dazu gehören die für den Client zulässigen Kodierungen und Sprachen (`Accept-Encoding`, `Accept-Language`, ..) ebenso wie zusätzliche Bedingungen, die mit der Anfrage verknüpft sein können (`If-Modified-Since`, ..).

Der `entity-header` beschreibt den Inhaltsteil der Nachricht. Die Eigenschaften werden dabei über Felder wie `Content-Length` und `Expires` beschreiben.

Ein Beispiel für eine Anfrage:

```
GET /~dippel/test.html HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.5 [en]C-CCK-MCD UMR/FB12  (WinNT; I)
Pragma: no-cache
Host: www
Accept: image/gif, image/jpeg, image/pjpeg, image/png, */*
Accept-Encoding: gzip
Accept-Language: en,de
Accept-Charset: iso-8859-1,*,utf-8
```

Die Antwort des Servers enthält Statusinformationen über die Ausführung der Anfrage und im Erfolgsfall die angeforderte Information sowie zusätzliche Informationen über das Dokument. Die Antwort läßt sich in EBNF-Notation beschreiben als

```
Response=  Status-Line
      *( general-header
      | response-header
      | entity-header )
      CRLF
      [ message-body ]
```

In der Statuszeile gibt der Server Auskunft über das Resultat der Anfrage. Die Syntax wird beschrieben durch:

```
Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF
```

Der Status-Code besteht aus drei Ziffern, welche die Kategorie der Antwort beschreiben. Dabei kann eine Einordnung bereits mit der ersten Ziffer vorgenommen werden. Diese Einteilung nach der Spezifikation des HTTPs [24] ist in *Tabelle 14* zu sehen.

Status-Code	Kategorie	Beschreibung
1xx	Informational	Request received, continuing process
2xx	Success	The action was successfully received, understood, and accepted
3xx	Redirection	Further action must be taken in order tocomplete the request
4xx	Client Error	The request contains bad syntax or cannot be fulfilled
5xx	Server Error	The server failed to fulfill an apparently valid request

Tabelle 14: Status-Codes des HTTP-Servers

Beispiel für eine Antwort:

```
HTTP/1.1 200 OK
Date: Tue, 13 Apr 1999 14:22:35 GMT
Server: Apache/1.3.3 (Unix)
Last-Modified: Tue, 13 Apr 1999 14:07:42 GMT
Accept-Ranges: bytes
Content-Length: 84
Keep-Alive: timeout=30, max=1000
Connection: Keep-Alive
Content-Type: text/html

<HTML>
<HEAD><TITLE>Minimal-Seite</TITLE></HEAD>
<BODY>
Antwort
</BODY>
</HTML>
```

Die Entwicklung der Version 1.1 des HTTPs ist ein entscheidender Schritt gewesen. Die Ziele der Änderungen von HTTP/1.0 zu HTTP/1.1 wurden wie folgt definiert [110]: Höhere Client-Performanz (WWW-Browser) unter Beibehaltung der bisherigen Stabilität und Datenintegrität mit Hilfe von persistenten Verbindungen, Pipelining, Caching und reduzierter Verwendung von IP-Adressen. Die Änderungen [53] im einzelnen:

Bei sogenannten multi-homed WWW-Servern wurde es durch die Verwendung des Host Request-Headers nun möglich, verschiedene Domainnamen und damit mehrere WWW-Server unter einer IP-Nummer anzusprechen. Bisher war es nur möglich mehrere WWW-Server (virtuelle Server) auf einer physikalischen Maschine laufen zu lassen, indem dem Ethernetinterface mehrere IP-Nummern zugewiesen wurden. Diese Neuerung bewirkt eine erhebliche Einsparung von IP-Adressen, da ansonsten pro Domainname eine IP-Adresse verbraucht würde und zudem der Betriebssystemkern (Kernel) einen nicht zu unterschätzenden Verwaltungsoverhead für jeden einzelnen Domainnamen (sprich: IP-Adresse) verarbeiten müsste.

Deutliche Effizienzsteigerungen konnten durch zwei Protokollerweiterungen erreicht werden: Durch persistente Verbindungen wird das unnötige Öffnen und Schließen einzelner TCP-Verbindungen vermieden. Dadurch entfällt u.a. auf TCP-Ebene das an anderer Stelle erläuterte Three-Way Handshake. Zudem werden die nur begrenzt zur Verfügung stehenden Ports geschont. Die andere Protokollerweiterung zur Effizienzsteigerung ist das Pipelining, welches eine einzelne Verbindung effizienter nutzt, indem mehrere Anfragen hintereinander in eine einzelne Verbindung „hineingeschoben“ werden.

Das Caching-Modell des HTTP wurde erst in Version 1.1 so spezifiziert, daß es von Server-Betreibern und Nutzern des Internets effizient benutzt werden konnte. Die beim HTTP/1.0 vorgenommenen Definitionen waren zu ungenau für den reibungslosen Betrieb eines Caching-Netzwerkes.

Desweiteren wurden weitere Request-Methoden („PATCH“, „LINK“, „UNLINK“) im HTTP/1.1 definiert und Header-Fields hinzugenommen („ALTERNATES“ , „CONTENT-VERSION“, „DERIVED-FROM“, „LINK“, ..), auf die hier aber nicht weiter eingegangen werden soll. Um die Stabilität des Web-Verkehrs zu gewährleisten, wurde bei der Ausarbeitung der Spezifikation besonders Wert gelegt auf die Kompatibilität zwischen den Protokollen HTTP/0.9, HTTP/1.0 und HTTP/1.1 mit besonderer Betrachtung des HTTP/1.0 mit persistenten Verbindungen.

Für weitere Versionen des Protokolls stehen bereits die Planungen und Entwicklungsarbeiten. Dazu gehört z.B. die Effizienzsteigerung des Cachings im Bezug auf Seitenzähler (Hit-Counter). Desweiteren ist man beim Design eines neuen Protokolls, dem Message Multiplexing. Dieses Protokoll ist oberhalb von TCP und unterhalb der Anwendungen in der Transportschicht angesiedelt. Es stellt gemultiplexte, sichere und bidirektionale Datenströme über eine TCP-Verbindung her.

2.3 Topologie

Unter der Netzwerk-Topologie versteht man die Art des Graphen, der die physikalischen Verbindungen der Geräte eines Netzwerks beschreibt. Die Topologien sind zum Teil an die Transportmedien gebunden und hängen auch von den im Netz verwendeten Verbindungskomponenten ab. Man unterscheidet gewöhnlich zwischen vier verschiedenen Grundtypen [106], die in einem Netzwerk jedoch auch gemischt vorkommen können.

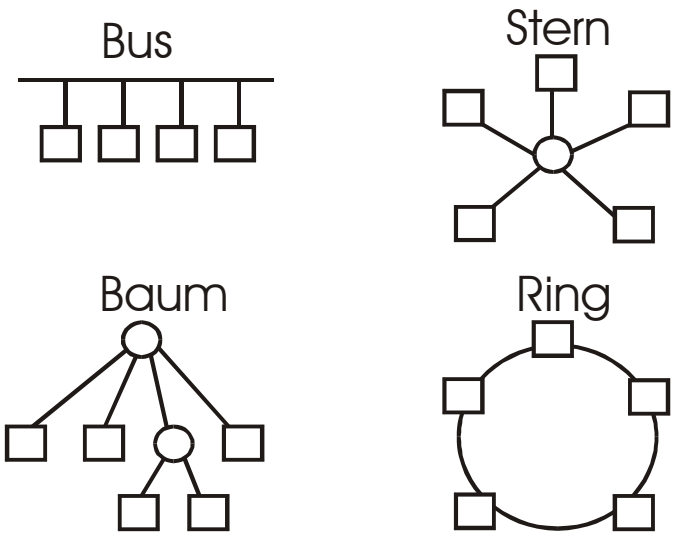

Abbildung 24: Netzwerk-Topologien

2.3.1 Bus

Bei der Bustopologie kommunizieren die Netzwerkstationen über ein gemeinsames Kabel. Ein typische Vertreter dieser Topologie sind das ThickWire-Kabel und das ThinWire-Kabel, die vor allem früher bei Ethernet-Netzen zum Einsatz kamen.

2.3.2 Stern

Von einer Sterntopologie spricht man, wenn von einem zentralen Punkt (Hub, Konzentrator) eine Punkt-zu-Punkt-Verbindung mit den einzelnen Netzwerkstationen besteht. Als Beispiel kann hier ein Ethernet Twisted-Pair Hub aufgeführt werden, an dem alle Netzwerkstationen angeschlossen sind.

2.3.3 Ring

Wie der Name schon sagt, ist bei dieser Topologie die Verkabelung, mit der die einzelnen Netzwerkstationen miteinander verbunden werden, als Ring ausgeführt. Typische Vertreter dieser Topologie sind Token-Ring- und FDDI-Netze.

2.3.4 Baum

Eine Baumstruktur wird z.B. erreicht, indem einzelne Hubs oder Konzentratoren über Punkt-zu-Punkt-Verbindungen kaskadiert werden. Die Netzwerkstationen werden dabei an nicht für die Kaskadierung benötigten Ports angeschlossen.

2.4 Medien

Das physikalische Transportmedium wird von der Bitübertragungsschicht benutzt, um den Transport von einzelnen Bits vorzunehmen. Das verwendete Medium und die Topologie implizieren zumeist das Sicherungsschichtprotokoll, so spezifiziert z.B. das IBM Token Ring Protokoll die Stern-Topologie in Verbindung mit einem Twisted-Pair Kabel [17]. Das Token-Protokoll IEEE 802.3

hingegen spezifiert weder Medium noch Topologie. Um Verwirrungen an dieser Stelle auszuschließen, muß hinzugefügt werden, daß die Stationen sternförmig an eine sogenannte MSAU (multistation access unit) angeschlossen werden, die jedoch ihrerseits ringförmig miteinander verbunden sind. Der Paketfluß ist somit ringförmig. Im folgenden sind die gängigen Medien für Ethernet aufgeführt, die entsprechenden Verbindungsstecker sind in *Abbildung 25* zu sehen.

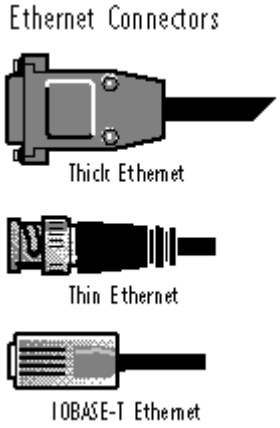

Abbildung 25: Ethernet Verbindungsstecker

2.4.1 10BASE-5, ThickWire

Dieser Urtyp des Ethernetkabels hat nur noch historische Bedeutung und wird bei Neuverkabelungen nicht mehr eingesetzt. Dieses Ethernet-Kabel ist bis 10 MBit/s spezifiziert und darf maximal 500 m lang sein, dabei dürfen maximal 100 Transceiver angeschlossen werden. Der Abstand zwischen zwei Transceivern muss 2.5 m oder ein Vielfaches davon betragen. Der Anschluß eines Transceivers ist keine Arbeit für den Laien: Der Transceiver wird and das ThickWire-Kabel angeschlossen, indem das Kabel angebohrt wird und der Transceiver über einen Dorn mit der Innenader des Koax-Kabels elektrisch verbunden wird.

2.4.2 10BASE-2, ThinWire, Cheapernet

Dieses sehr populäre Ethernet-Medium ist bis 10 Mbit/s spezifiziert und ähnelt auch sonst dem 10BASE5-Medium. Es basiert jedoch auf einem wesentlich dünnerem und auch preiswerterem Medium. Weitere Vorteile neben dem Preis des Mediums liegen in der leichteren Verlegung des Mediums und vor allem dem leichten Anschluß von Geräten an das Medium, da kein Transceiver mehr nötig ist. Die Nachteile von 10BASE-2 sind die Beschränkung auf 185 m und 30 Stationen und dem notwendigen Unterbrechen des Mediums beim Anschluß einer weiteren Station. Hinzu kommt der Nachteil, daß die Fehlersuche sehr aufwendig ist, da der gesamte Bus nach offenen Verbindungen abgesucht werden muß.

2.4.3 10BASE-T, Twisted-Pair

Das Twisted-Pair Kabel ist zweifelsohne der Nachfolger des Koax-Verkabelung. Das Twisted-Pair Kabel besteht aus acht paarweise verdrillten Kabeln, die je nach Ausführung noch zusätzlich abgeschirmt sind. Durch die Verdrillung werden Störsignale zwischen den Leitern verhindert. Beim Twisted-Pair Kabel ist es möglich, Bandbreiten bis zum Gigabit-Bereich zu übertragen. Anfangs

für 10 MBit/s Ethernet vorgesehen, wird es nun auch in der sogenannten Verkabelungskategorien 5 für Fast Ethernet, Gigabit Ethernet, FDDI und ATM verwendet. Das 10BASE-T wird stets für Punkt-zu-Punkt Verbindungen eingesetzt, dadurch wiegt der Nachteil der maximalen Kabellänge von 100m nicht ganz so schwer. Als Verbindungskomponenten kommen Repeater oder Switches in Frage. Weil jedes Kabel ein eigenes Segment ist, erhöht sich die Sicherheit gegenüber Störungen erheblich. Zudem wird durch den Einsatz von Verbindungskomponenten die Fehlersuche enorm erleichtert, weil diese stets den Zustand der angeschlossenen Segmente anzeigen.

2.4.4 10BASE-F, Lichtleiter

Lichtwellenleiter sind von den Spezifikationen her identisch mit den hochqualitativen Versionen der Twisted-Pair Verkabelung. Dieses Medium wird hauptsächlich zur Verbindung von Switches und Bridges verwendet. Der klare Vorteil liegt in der nahezu beliebig skalierbaren Bandbreite, die lediglich von den Endgeräten abhängt, und den Einsatzorten, wo eine elektrisch unempfindliche (NMR-Geräte, etc.) und nicht leitende Verbindung (Outdoor-Bereich) nötig ist. Zudem kann das Kabel bis zu 2 km lang sein. Die Nachteile liegen in der schwierigeren Verlegbarkeit und schlicht und ergreifend im Preis.

2.5 Verbindungskomponenten (Repeater, Bridges, Router)

Ein Netzwerk besteht im allgemeinen nicht nur aus Kommunikationsendpunkten (Hosts) und einem Transportmedium. Wenn das Netz eine bestimmte Topologie (Stern) aufweist, eine bestimmte Größe überschreitet oder mit anderen Netzen verbunden ist, wird es notwendig zusätzliche verbindende Komponenten im Netz einzusetzen. Allen diesen Verbindungskomponenten ist gemeinsam, daß sie Kommunikationsendpunkte oder auch weitere dieser Komponenten miteinander verbinden. Ein Kriterium zur Unterscheidung dieser Komponenten ist die OSI-Schicht, auf der sie arbeiten.

2.5.1 Repeater

Synonyme für den Ausdruck Repeater sind Hub oder auf deutsch Verstärkerstation. Der Repeater arbeitet auf der Bitübertragungsschicht. Er hat die Funktion eines Signalregenerators, er verstärkt und bereinigt das physikalische Signal, ohne es weiter auszuwerten. Es findet eine Überprüfung des physikalischen Signals statt, nicht jedoch der übertragenen Bits. Bei dieser Vorgehensweise werden zwar Spannungsspitzen gefiltert, jedoch werden fehlerhafte Bits (z.B. „gekippte“ Bits) oder Pakete weiter zu allen anderen angeschlossenen Komponenten und Hosts weitergeleitet. Ein Einsatzgebiet des Repeaters sind große Subnetze, bei denen eine Signalabschwächung durch große Kabellängen ausgeglichen werden soll (z.B. Überschreitung der max. Kabellänge von 185 m bei 10BASE-2). Ein weiteres Einsatzgebiet ist die Bereitstellung von mehreren Anschlußpunkten für Geräte an ein Kabel. Damit wird zusätzlich die Verfügbarkeit des Netzes verbessert. Den Repeater gibt es in unterschiedlichen Ausführungen. Im einfachsten Fall als Gerät mit zwei Anschlüssen zur Signalverstärkung bei großen Kabellängen, bis hin zum Gerät mit mehreren dutzend Anschlüssen, als zentralen Anschlußpunkt eines sternförmigen Netzes. Baut man ein Netz mit mehr als zwei

Kommunikationsendpunkten über Twisted-Pair Verkabelung auf, so ist der Repeater zwingend notwendig und zugleich die kostengünstigste Verbindungskomponente unter den Alternativen Repeater, Bridge und Router.

2.5.2 Bridge

Die Bridge, oder auf deutsch Brücke, arbeitet auf der Sicherungsschicht. Sie ist somit in der Lage, einzelne Pakete der Sicherungsschicht zu identifizieren und nötigenfalls zu modifizieren. Damit ist es möglich, Netze mit unterschiedlichen Sicherungsschichten, aber identischen Vermittlungsschichten miteinander zu verbinden [15]. Im Falle einer Verbindung zwischen Token Ring und Ethernet kann es z.B. Aufgabe der Bridge sein, die Token Ring Pakete in Ethernet Pakete umzupacken. Die Möglichkeiten und Einsatzgebiete einer Bridge sind jedoch weitaus vielfältiger. Die Bridge leitet nur Pakete weiter, die von ihr als korrekt erkannt werden. Dadurch werden alle defekten Pakete und Signalstörungen aus den anderen, separat angeschlossenen Netzen ferngehalten und damit die Datensicherheit erhöht. Man kann zudem Filter anwenden, um z.B. auf Basis von Ethernetadressen Sperren vorzunehmen. Weil die Protokolle der Sicherungsschicht meist ein Verweis auf den Typ des eingekapselten höheren Protokolls enthält, ist es auch möglich, z.B. AppleTalk aus bestimmten Segmenten zu filtern.

Bridges sind typischerweise in Software implementiert, was bedeutet, daß sie einen schnellen Prozessor mit genügend Arbeitsspeicher benötigen.

2.5.3 Switch

Ein Switch arbeitet wie ein Vermittlungsrechner beim Telefonsystem. Er stellt eine leitungsorientierte Vermittlung zwischen zwei Geräten her. Ein Switch arbeitet wie eine Bridge im OSI-Schichtenmodell auf der Sicherungsschicht und ist auch wie diese aufgebaut. Der Unterschied ist, daß der Switch i.A. mehr Anschlüsse (Ports) hat. Von den Leistungsbeschreibungen ist ein Switch mit einer Multiport-Bridge identisch. Die Funktionen, die bei einer Bridge in den 90er Jahren in Software implementiert wurden, werden heute in Hardware implementiert. Der Begriff „Switch“ wurde von der Firma Kalpana (inzwischen von Cisco aufgekauft) kreiert, weil ihre Multiport-Bridges nicht der IEEE-Spezifikation einer Bridge entsprachen und somit auch nicht Bridge genannt werden konnten [106].

Ein Switch wird dort eingesetzt, wo die Bandbreite des Ethernets nicht mehr ausreicht. Man teilt das Ethernet in mehrere Segmente auf und verbindet diese über einen Switch oder eine Multiport-Bridge mit schnellem internen Bus, auch „collapsed backbone“ oder Backplane genannt. Ein wesentliches Qualitätkriterium eines Switches ist die Bandbreite der Backplane.

2.5.4 Router

Der Router arbeitet eine Schicht höher als die Bridges, er arbeitet auf der Vermittlungsschicht. Ein Routing wird erforderlich, wenn zwei Hosts in unterschiedlichen Subnetzen miteinander kommunizieren sollen. Eine Kommunikation zwischen den Hosts kann nur über ein routingfähiges Protokoll erfolgen (TCP/IP, IPX, ..), für die Kommunikation über nicht-routingfähige Protokolle (NetBEUI, ..) haben die meisten Router noch eine zusätzliche Bridging-Fähigkeit. Routing ist gegenüber den Protokollen der Vermittlungsschicht nicht transparent, Router müssen alle

verwendeten Protokolle erkennen können und die Pakete modifizieren können. Zu den Modifikationen gehören Veränderungen der Paketlänge (Unterschiede bei Ethernet, Token-Ring, X.25, ..) und Geschwindigkeitsanpassungen (Übergang LAN zu WAN). Das Einsatzgebiet ist die Verbindung von einzelnen Subnetzen. Die Router bestimmen den Weg (engl.: route) der Pakete anhand von internen Routing-Tabellen, die auch zwischen den Routern über Routing-Protokolle ausgetauscht werden können.

2.6 Daten

Beim WWW werden Daten unterschiedlichster Art übermittelt. Das fängt an mit unformatiertem, ASCII-kodierten Text, geht über HTML-Text, Grafik und Audio bis hin zu Videoströmen und proprietären Anwendungsdaten.

Aus dem Bitmuster der Daten alleine kann man nicht entscheiden, welche Art von Daten vorliegen und wie diese zu behandeln sind. Diese zusätzliche Information muß beim Datenaustausch mit übermittelt werden oder über das Protokoll von vorneherein festgelegt sein. So wurde beim SMTP-Protokoll festgelegt, daß die Daten 7 Bit ASCII kodiert übermittelt werden. Mit dieser Kodierungsvorschrift ist das Zeichen ‚A' mit der Bitfolge ‚1000001' eindeutig kodiert, unabhängig von Land und Rechnertyp. Neben der Kodierung der Daten ist auch die verarbeitende Anwendung entscheidend für die Präsentation der Daten. So stellt ein Texteditor eine HTML-Datei im HTML-Quelltext dar, wohingegen ein Browser die HTML-Token verarbeitet und diese Datei entsprechend der Formatierungsinformationen darstellt.

Um das starre Konzept der festgelegten Kodierung und Zuordnung der Anwendung zu ersetzen wurden die Mail Erweiterungen MIME definiert. Mit diesen Erweiterungen ist es nun möglich, Daten nach einer bestimmten Vorschrift einen Typ und eine Kodierung zuzuordnen. Diese Information wird beim Datenaustausch mit übergeben und gewährleistet eine plattformunabhängige Darstellung der Daten.

Zu den Inhalts-Typ der Daten (Content-Type) gehört auch die Kompression. Es gibt verschiedene Kompressionsverfahren, die sich neben dem Kompressionsgrad und dem damit verbundenen Rechenaufwand zur Kompression und Dekompression auch in der Verbreitung unterscheiden.

2.6.1 MIME

MIME ist die Abkürzung für „Multipurpose Internet Mail Extensions" und wurde, wie der Name schon verrät, als Erweiterung des ursprünglichen Standards von 1982 zur E-Mail [18] entwickelt. Eine RFC 822 konforme E-Mail besteht aus 7-Bit ASCII-Zeichen und darf keine Zeilen mit mehr als 1000 Zeichen enthalten. Diesen Nachteilen begegnet MIME [26] 10 Jahre später mit der Definition von zusätzlichen Header-Zeilen in einer E-Mail, welche den Inhalt und die Struktur der E-Mail spezifizieren.

Dadurch ist es nun möglich, mehrere Objekte in eine einzelne E-Mail einzupacken, beliebig lange Zeilen in unbeschränkten E-Mails zu verwenden, andere Zeichensätze zu verwenden und beliebige Dateien in Binärform zu übertragen, die bei Bedarf einen Typ zugewiesen bekommen (z.B. Audio, Video, etc.). Dieser Standard ist in den RFCs 2045 bis 2049 [27-30; 72] festgelegt und wird

laufend um Funktionalitäten erweitert. Ein aktuelles Beispiel dafür ist die Definition von XML-Mediatypen [115], wie im folgenden Beispiel zu sehen.

```
Content-type: text/xml; charset="utf-8"
   <?xml version="1.0" encoding="utf-8"?>
```

Bei diesem Beispiel ist die erste Zeile der Content-Type MIME Header und die zweite Zeile bereits der XML-Inhalt. Für die meisten Parameter gibt es auch Default-Werte, so ist der Zeichensatz „us-ascii“, wenn der Parameter „charset“ weggelassen wird. Dies würde nebenbei bemerkt im obigen Beispiel zu Problemen führen, da der XML-Inhalt offensichtlich nach „utf-8“ kodiert ist, die Übertragung entsprechend des MIME-Headers ohne Parameter „charset“ jedoch in „us-ascii“ durchgeführt wird.

Die ursprüngliche Motivation für die Einführung von MIME war die Integration von zusätzlichen Medien in eine E-Mail Nachricht. Diese Konzept erwies sich jedoch auch beim WWW als sehr wertvolles Instrumentarium. Damit ist es dem WWW-Server möglich, dem Client zusätzlich zu den Daten den Datentyp und weitere Informationen, wie z.B. die Datenlänge, die Kodierung, den Zeichensatz, etc. mitzuteilen. Mit HTTP/1.0 folgt das Format von Client-Request und Server-Response beim HTTP dem MIME-Format von E-Mail Nachrichten und profitiert unmittelbar von dessen Flexibilität und Erweiterbarkeit. Beispiele für MIME-typische Header sind in *Kapitel 2.2.4.1: HTTP* zu sehen.

2.6.2 HTML und XHTML

HTML (Hypertext Markup Language) hat einen vielschichtigen Charakter, der je nach Betrachtungspunkt sehr unterschiedlich beschrieben werden kann. Für den Anwender im Netz ist es eine Sprache, die dafür sorgt, daß ein Dokument im Netz betrachtet werden kann. Für den WWW-Administrator hingegen ist es eine Ausdrucksform, in der die Netzwerkbenutzer ihre Dateien abspeichern (deren Inhalt ihn zumeist nicht interessiert), die dann Knoten in der Graphen-Struktur [38] seines WWW-Servers repräsentieren.

Bei einer Zusammenfassung dieser extremen Betrachtungsweisen erhält man eine Datei, die neben dem textuellen Inhalt auch Formatierungsinformationen und andere Medienelemente (Bilder, Sound, ..) enthält und zudem Verweise zu beliebigen anderen Dokumenten enthalten kann. Zudem muß die HTML-Datei nicht notwendigerweise in Baumstruktur angeordnet sein.

Der Ursprung von HTML ist historisch gesehen nicht genau zu datieren, die Abstammung ist klar auf die „Markup Language“ SGML aus den 60er Jahren zurückzuführen, wobei man HTML als SGML-Anwendung auffassen kann. Die Idee des Hypertextes wurde schon von verschiedenen Personen in unterschiedlichem Kontext aufgegriffen und diskutiert, Anfänge von computergestützten Hypertextsystemen [13] sind auf 1945 datiert [86], populäre Implementationen wie HyperCard für den Macintosh von Bill Atkinson [84] auf 1987.

Der entscheidende Schritt zu einer Form eines Hypertextsystems, welches wir nun als WWW kennen, wurde zweifelslos mit einem Proposal [5] von Tim Berners-Lee am CERN gemacht. In diesem Dokument wird ein Informations-Management System vorgeschlagen für den Einsatz innerhalb des CERN-Organisation. Der endgültige Durchbruch für das WWW kam mit der Entwicklung der Browser-Software vom NCSA.

Mit der Entwicklung des Internets durchlief auch die HTML-Definition eine Reihe von Versionen, die voraussichtlich letzte ist Version 4.01 vom Dezember 1999 [87]. Im weiteren Verlauf wird

HTML durch XHTML ersetzt werden. Diese Sprachdefinition liegt seit Januar 2000 in der Version 1.0 vor [85] und ist im wesentlichen eine Umformulierung von HTML Version 4 in XML-Syntax. Ziele für die Umformulierung ist eine XML-konforme Syntax und Modularisierung der Funktionen. Damit soll eine einfache Unterteilung in Teilsprachen und ebenfalls eine einfachere Erweiterung und Transformation in andere Dokument-Typen ermöglicht werden.

Das entscheidende Merkmal im Kontext dieser Arbeit ist die Struktur einer HTML-Seite, so wie sie am Bildschirm vom Browser dargestellt wird. Alle dafür notwendigen Informationen müssen dazu natürlich erst einmal über das Netz übertragen werden. Neben textueller Information enthalten HTML-Seiten zumeist eine Vielzahl von Grafiken oder anderen Objekten, die jeweils vom Browser zusätzlich geholt werden müssen. Wenn man sich daher eine Seite im Web anschaut, so generiert man damit nicht eine einzige Anfrage, sondern indirekt eine Reihe von Anfragen, wie schon in *Kapitel 2.2.4.1:HTTP* diskutiert wurde.

2.6.3 Kompression

Es gibt eine große Anzahl von Kompressionsalgorithmen [92]. Eines der wichtigsten Merkmale dieser Kompressionsalgorithmen ist, ob der Algorithmus verlustbehaftet oder verlustfrei arbeitet. Verlustbehaftete Kompression kann nur dort eingesetzt werden, wo ein gewisser Informationsverlust grundsätzlich vom Konsumenten in Kauf genommen wird. Typischerweise sind das Bereiche der Informatik, wo die Daten dem Konsumenten in analoger Form präsentiert werden. Dazu gehört vor allem der Audio- und Videobereich. Die z.B. beim MPEG- und JPEG-Dateiformat (MPEG: Videoströme, JPEG: Standbilder) verwendeten Algorithmen basieren darauf, die Informationen wegzulassen, die dem menschlichen Sinnen am wenigsten auffallen. Im Falle der JPEG-Kompression werden (über eine Fourier-Transformation) im Bild Bereiche kleiner Kontraste angeglichen. Durch diese Quantisierung erreicht man einen Informationsverlust, der durch anschließende verlustfreie Kompression dahingehend ausgenutzt werden kann, daß die resultierenden Daten ein kleineres Volumen haben.

Bei der Internetkommunikation können im Bereich der OSI-Schichten eins bis vier jedoch nur verlustfreie Kompressionsalgorithmen zum Einsatz kommen, da in diese Protokollschichten keine Informationen über den Typ der transportierten Daten (Audio, Video, Textdokument, Programmfragment, etc.) vorliegen.

Eine sehr einfache verlustfreie Kompression ist die RLE-Kodierung (Run Length Encoding). Bei dieser Kompression werden Sequenzen gleicher Zeichen durch ein Steuerzeichen, der Anzahl der Zeichen und dem Zeichen selbst ersetzt. Das Steuerzeichen, sowie der reservierte Platz für die Anzahl der Zeichen ist implementationsabhängig. RLE-Implementationen findet man u.a. in den Dateiformaten BMP, PCX, TIFF und PDF.

Eine weit verbreitete Klasse von Kompressionsalgorithmen sind die Huffman-Kodierungen [42]. Sie gehören zu der Familie der statistischen Kodierungen. Jedem Zeichen in den zu kodierenden Ausgangsdaten wird ein Kode zugeordnet. Der Kode ist eine Bitfolge variabler Länge. Die Länge des Kodes eines Zeichens richtet sich nach der Häufigkeit des Zeichens in den Ausgangsdaten. Das am häufigsten vorkommende Zeichen erhält z.B. den Wert 0, das nächst-häufigste den Wert 10, dann folgen die Kodes 110, 111, etc. Das Prinzip der Kodierung liegt darin, die Kodierung mit fester Länge (z.B. 8 Bit) durch eine Kodierung variabler Länge zu ersetzen, bei der den am häufigst vorkommenden Zeichen kürzere Kodes zugewiesen werden.

Zu den Wörterbuch basierten Algorithmen gehören die Algorithmen LZ77 [56] und LZ78 [57], die 1977 und 1978 von Abraham Lempel und Jakob Ziv entwicklet wurden. Sie bilden die Basis der Algorithmen LZSS, LZW und zahlreicher anderer Varianten. Ihnen ist gemein, daß sie eine substituierende Kompression [35] vornehmen. Bei dieser Funktionsweise werden einmal im Eingabedatenstrom vorgekommene Zeichenketten durch eine Referenz auf ihr vorheriges Auftreten ersetzt. Diese Algorithmen benötigen zur Dekompression keine weiteren Informationen über die Daten, da sie diese während des Dekodiervorgangs erhalten. Damit eignen sie sich besonders gut für eine für den Benutzer, bzw. für die Protokollschicht transparente Implementation.

Zu den weit verbreiteten Kompressionsverfahren gehört auch der DEFLATE-Algorithmus [22]. Dieses Verfahren komprimiert die Eingabedaten durch eine Kombination der LZ77- und Huffman-Algorithmen.

Ein weiteres Kompressionsverfahren ist die Stac Electronics LZS-Kompression. Sie basiert auf dem LZ77-Algorithmus [56] und wurde von der Firma Stac Electronics weiterentwickelt. Die LZS-Kompression findet in vielen Kommunikationsprotokollen im Internet Einsatz [32], [31].

Der GZIP-Algorithmus [33] implementiert den DEFLATE-Algorithmus, kann jedoch um weitere Kompressionalgorithmen erweitert werden. Dieser Algorithmus beschreibt auch das Datenformat, welches die komprimierten Daten repräsentiert. In diesem Sinne ist das GZIP-Projekt eher als „Umbrella“-Algorithmus zu verstehen, da es mehrere Standards vereinigt.

2.7 Client-Server

Die Client-Server Architektur ist nicht nur im Internet Grundlage der Kommunikation, auch im Alltag der Informatik lassen sich zahlreiche Beispiele für Client-Server basierte Anwendungen finden. Ein populäres Beispiel ist das Modell der Grafikausgabe bei UNIX Workstations. Das Grafikprogramm sendet seine Daten an den X-Server, der diese Daten auf dem Bildschirm darstellt. Bei diesem Beispiel ist das Grafikprogramm der Client, der seine Daten an den Server (X-Server) schickt, welcher die Daten in entsprechend verarbeiteter Form auf dem Bildschirm darstellt.

Wie in diesem Beispiel verhält es sich bei nahezu allen Netzwerkdiensten: Auf dem Server wartet ein Prozeß auf die vom Client initiierte Verbindung. Über ein dienst-spezifisches Protokoll erhält der Server eine Anfrage, die er bearbeitet und daraufhin reagiert. Üblicherweise wird dem Client das Ergebnis der wie auch immer gearteten Bearbeitung als Antwort übermittelt. Es folgen nun kurze Erläuterungen einiger Begriffe, die an anderer Stelle (*Kapitel 3: Methoden der Optimierung*) verwendet werden.

Als **Client** wird der Host bezeichnet, der die Kommunikation initiiert. Sinngemäß wird das Programm ebenfalls als Client bezeichnet.

Der **Server** ist der Host, der auf die Anfrage des Clients wartet. Die Anfrage wird vom Server bearbeitet und ein Ergebnis dem Client übermittelt.

Request und **Response** sind die in der englischen Literatur üblichen Begriffe für Anfrage und Antwort. Im übrigen nimmt man an, daß die Anfrage und die Antwort jeweils ununterbrochene Bytefolgen in einer Richtung der Kommunikation sind, die allerdings gegebenenfalls in mehrere Pakete aufgeteilt sein können.

Wichtige Kenngrößen für die quantitative Untersuchung sind in diesem Zusammenhang die SPT (**Server Processing Time**) und RTT (**Network Round-Trip Time**). Die SPT ist die Zeit, die der

Server zur Verarbeitung des Requests benötigt. Die RTT ist die Zeit, die ein Paket vom Client zum Server und wieder zurück benötigt. Der Einfachkeit halber wird die Kommunikation als symmetrisch angenommen, was jedoch nicht immer der Fall sein muß [103].

Der Begriff der **Transaktion** im Sinne der Internetkommunikation ist als eine Request-Response Abfolge in einer Client-Server Kommunikation definiert [11]. Die Transaktion beinhaltet die Anfrage des Clients, die Bearbeitung und die Antwort des Servers. Die minimale Zeit für eine Transaktion ist mit obigen Definitionen RTT + SPT.

3 Methoden der Optimierung

Nachdem in den vorangegangenen Kapiteln Maßnahmen zur Optimierung von Kommunikation im Internet motiviert wurden und die Grundlagen zur Kommunikation im Internet erläutert wurden, sollen an dieser Stelle Methoden diskutiert werden, welche die Kommunikation verbessern. Mit „verbessern“ sind alle Möglichkeiten gemeint, welche die benötigte Zeit des Verbindungsaufbaus und –abbaus oder das Volumen (und als Folge die Laufzeit) der Datenkommunikation reduzieren können.

Der Verbindungsaufbau und –abbau wird im folgenden **Verbindungssteuerung** genannt. Die Verbindungssteuerung legt fest, wie eine Verbindung aufgebaut wird, was nach einem Verbindungsaufbau im weiteren geschieht und wie eine Verbindung abgebaut wird. Ein Leistungsparameter für die Verbindungssteuerung ist z.B. die Transaktionsrate. Sie gibt an, wieviele Transaktionen pro Sekunde durchgeführt werden können.

Die Volumen und auch die Laufzeit der Daten sind Leistungsparameter für die **Datenrepräsentation**. Die Datenrepräsentation in einem Protokoll legt fest, wie die Daten kodiert, verschlüsselt und in Header verpackt werden und wie optionale Parameter der Verbindung übertragen werden.

Zum Vergleich und zur Einordnung wird zunächst eine Klassifikation der Optimierungsmethoden eingeführt. Im weiteren werden die Methoden im einzelnen erläutert und der Entwicklungsstand, sowie die Voraussetzungen für den Einsatz besprochen.

An dieser Stelle sollte darauf hingewiesen werden, daß die einzelnen Methoden zum Teil seit Jahren im Einsatz sind (van Jacobson IP Header compression), zum Teil seit Jahren in ihrer Spezifikation abgeschlossen sind, aber nur in bestimmten Betriebssystemen implementiert sind (T/TCP), einige erst seit kurzem die Spezifikationsphase beendet haben (IPComp) und einige lediglich in experimentellen Testumgebungen (IPv6) im Einsatz sind. Dementsprechend schwankt die Quantität und Qualität der Leistungsparameter für die untersuchten Methoden. Diese Problematik wird auch in den folgenden Kapiteln an den entsprechenden Stellen angesprochen, verhindert jedoch nicht eine aussagekräftige Analyse der Methoden.

3.1 Klassifikation von Protokolloptimierungen

Ziel der Arbeit ist es unter anderem, Methoden zur Optimierung der Kommunikation vorzustellen und miteinander zu vergleichen. Zur Kommunikation im Internet gehören zwei Kommunikationsendpunkte (zumeist ein Client und ein Server) und eine Verbindung zwischen beiden. Bei einer Kommunikation sendet der Kommunikationsendpunkt Client dem Kommunikationsendpunkt Server Daten. Diese werden in irgendeiner Weise dort verarbeitet. Danach werden optional weitere Daten ausgetauscht. In welcher Form und wann Daten ausgetauscht werden, wird von den Protokollen geregelt.

Die Verarbeitung auf den Kommunikationsendpunkten Client und Server hängt sowohl von der eingesetzten Software als auch von der Hardware ab. Daher besteht bei der „Server Processing Time“ meist eine lineare Abhängigkeit von der CPU-Leistung des Servers. Auch bei der Transfergeschwindigkeit der Daten besteht eine lineare Abhängigkeit von der Hardware, in diesem Fall von

der Bandbreite des Übertragungsmediums. Diese hardwareabhängigen Aspekte sollen jedoch nicht Teil dieser Untersuchung sein.

Hier sollen die hardwareunabhängigen Aspekte der Kommunikation betrachtet werden, insbesondere die Transferprotokolle. Zwei Aspekte der Aufgaben eines Transferprotokolls werden im folgenden als Klassifikation von Protokolloptimierungen dienen: Die Vorschriften zur Verbindungssteuerung und die Repräsentation der Daten, die zwischen den Kommunikationsendpunkten ausgetauscht werden.

Eine dazu orthogonale Klassifizierung kann man naheliegenderweise anhand der Protokollschichten vornehmen. Um Hardwareabhängikeiten zu eliminieren, sollen nur die den OSI-Schichten drei bis sieben betrachtet werden, also die Vermittlungsschicht (IP), die Transportschicht (TCP und UDP) und die darüber liegenden Schichten, im folgenden zusammengefaßt als Anwendungsschicht (HTTP) bezeichnet (im Gegensatz zum OSI-Modell, bei dem die Anwendungsschicht lediglich die Schicht 7 bezeichnet).

Anhand der oben definierten Einordnungen erhält man eine Klassifikationstafel, in der in den Zeilen die Protokollschichten (Vermittlungsschicht, Transportschicht und Anwendungsschicht) eingetragen werden und in den Spalten der Typ der Optimierungsansätze (Verbindungssteuerung und Datenrepräsentation).

Schicht	Verb.-Steuerung	Datenrepräsentation
3-Vermittlungsschicht		
4-Transportschicht		
5-Anwendungsschicht		

Tabelle 15: Klassifikationstafel

Im weiteren Verlauf des Kapitels wird diese Tafel mit den verschiedenen Optimierungsansätzen gefüllt werden.

3.2 Optimierung der Verbindungssteuerung

Hier werden wichtige Optimierungsstrategien, Protokollerweiterungen oder neue Protokolle aufgeführt, die zu einer Verbesserung der Verbindungssteuerung führen. Die Maßnahmen dazu ähneln sich auf allen Protokollebenen. Es wird versucht überflüssigen Overhead und redundante Informationen bei der Kommunikation zu vermeiden, besonders wenn eine Verbindung in der Schicht N durch mehrere Verbindungen in der Schicht N-1 realisiert wird.

3.2.1 Vermittlungsschicht

In der Vermittlungsschicht gibt es keine wesentlichen Verbesserungen der Performanz des IPv4 Protokolls. Ein Grund dafür ist sicherlich, daß IPv4 nur rudimentäre Dienste zur Verfügung stellt, wodurch das Protokoll selbst schon eine hohe Performanz erreicht. Ein Bereich, in dem viele Verbesserungen am IPv4 vorgeschlagen und durchgeführt worden sind, ist allerdings die Sicherheit. Dieser Aspekt der Kommunikation wurde beim Design der TCP/IP-Protokollsuite kaum beachtet und erfordert nun viele Nachbesserungen. Ein Grund für die etwas schleppende

Einführung des IPv6, welches im folgenden untersucht wird, ist sicherlich das performante und robuste (mit obigen Einschränkungen) Design von IPv4.

3.2.1.1 IPv6

In der Vermittlungsschicht betrachten wir das Protokoll IPv6, dessen Eigenschaften eine Verbesserung im Bereich der Verbindungssteuerung versprechen. Wie in *Kapitel 2.2.2: Vermittlungsschicht* bereits zu sehen war, ist der Header des IPv6 Paketes gegenüber dem von IPv4 deutlich geschrumpft. Das Ziel dieser Maßnahme ist [21], den Verarbeitungsaufwand und die benötigte Bandbreite für den allgemeinen Fall der Paketvermittlung mit IPv6 zu reduzieren. Es läßt sich auch durchaus leicht nachvollziehen, daß der IPv6-Header mit nur noch 8 Bytes statt 12 Bytes in IPv4 (ohne IP-Adressen) von Vermittlungssystemen (Router) wesentlich performanter weitergeleitet werden kann. Zumindest für den allgemeinen Fall, wenn keine weiteren Headeroptionen vorhanden sind, bzw. nicht beachtet werden müssen.

Ein weiterer Vorteil, der in den Bereich der Optimierung der Datenrepräsentation (*Kapitel **Error! Reference source not found.**: **Error! Reference source not found.***) fällt, ist die Jumbo-Payload Option beim IPv6. Damit ist es möglich IPv6 Pakete zu versenden, die größer als 65535 Bytes sind, auch „Jumbograms“ [9] genannt. Damit kann der Overhead von zusätzlichen IP-Paketen beim Versenden von großen Dateien eingespart werden. Die Jumbo Payload Option ist eine „Hop-By-Hop“-Option, muß also bei jedem Knoten auf dem Pfad zum Zielsystem beachtet werden. Die Jumbo-Payload Option ist sechs Bytes lang und besteht wie andere IP-Optionen aus einem Byte für die Options-Kennung, einem für die Länge der Option und zusätzlich aus 4 Bytes für die Länge des Jumbo-IP Paketes. Zudem kommen noch zwei Bytes für den Hop-By-Hop Options-Header dazu, falls nicht schon vorhanden. Im allgemeinen Fall muß man daher von einer Vergrößerung des Headers um 8 Bytes bei der Verwendung der Jumbo-Payload Option ausgehen. Inwieweit dies relevant ist, wird in den nächsten Kapiteln geklärt werden.

3.2.2 Transportschicht

Die Transportschicht gehört mit der Vermittlungsschicht zu den wichtigsten Protokollen im Internet, nicht nur was die Funktionalität angeht, sondern auch aufgrund der langjährigen Verwendung. Sowohl die darunter liegenden Schichten wie Tokenring, Ethernet und ATM, als auch die darüber liegenden Schichten oder Anwendungen wie Gopher und HTTP können und wurden über die Jahre der Existenz des Internets ausgetauscht. Daher ist es wie auch in der Vebindungsschicht schwierig, Optimierungen einzubauen, oder gar das Protokoll zu ersetzen. Ein Vorstoß in diese Richtung wurde unter Beibehaltung der Kompatibilität mit dem T/TCP vorgenommen.

3.2.2.1 T/TCP

Eine entscheidende Verbesserung der Verbindungssteuerung ist mit der Einführung des T/TCP gelungen. Dessen Erweiterungen und Modifikationen des TCPs zielen direkt auf eine Optimierung von Transaktionen. Zu den Verbindungen, die in die Klasse der Transaktionen einzuordnen sind, gehören neben dem DNS und anderen auch das HTTP, welches im Internet den größten Anteil am Verkehr für sich beansprucht.

Nun sollen Transaktionen für die Protokolle UDP, TCP und T/TCP verglichen werden. Dazu ist in *Abbildung 26* eine Transaktion für jedes der erwähnten Protokolle zu sehen. In zeitlicher Folge sind dort die einzelnen Schritte zum Abwickeln einer Verbindung aufgeführt. In der Initialisierungsphase (***Init***) startet der Client und der Server, welcher zunächst auf eine Verbindung von einem Client wartet.

Der Verbindungsaufbau folgt in der Verbindungsphase (***Connection***), ausgehend vom Client. Im Falle eines WWW-Browsers als Clientsystem wäre das der Zeitpunkt, zu dem die Ziel-URL eingegeben wurde. Beim TCP wird nun ein Three-Way Handshake zwischen Client und Server durchgeführt. Dieses entfällt beim T/TCP für die n-te Verbindung zwischen zwei Hosts (n>1) (*Abbildung 21*). Beim verbindungslosen UDP findet kein Verbindungsaufbau statt.

Danach wird vom Clientsystem der eigentliche Request an das Serversystem gesendet (***Request***) und im Falle des TCPs mit einem Acknowledge (Ack) bestätigt. Dieses Acknowledge wird beim T/TCP mit in das Antwortpaket hineingepackt, beim verbindungslosen UDP entfällt wieder die Bestätigung.

Die Anfrage wird dann vom Serversystem bearbeitet (***Processing***).

Die Antwort wird schließlich (***Response***) vom Server an den Client geschickt. Dabei wird beim TCP und T/TCP über Fin das halbseitige Schließen der Verbindung signalisiert. Bei den verbindungsorientierten Protokollen TCP und T/TCP wird dieses Paket erneut bestätigt.

Am Ende wird die Verbindung bei den verbindungsorientierten Protokollen geschlossen (***Close***), was beim TCP in einem letzten Fin-Paket mit Bestätigungs-Paket vom Server resultiert.

Die mit der Transaktion verbundenen API-Aufrufe sind in den entsprechenden Spalten der Protokolle in der Reihenfolge des Aufrufs eingetragen. Zudem sind die Pakete und deren Richtung mit Pfeilen zwischen Client- und Serversystem neben den hervorrufenden API-Aufrufen eingetragen.

	Anwendungsschicht		Transportschicht									
	Anwendung		TCP			T/TCP			UDP			
	Client	Server	Client		Server	Client		Server	Client		Server	Zeit
Init	Programm-Start	auf Verbindung warten	socket()		socket() bind() listen()	socket()		socket() bind() listen()	socket()		socket() bind()	
Connection	Verbindung aufbauen	Verbindung bestätigen	connect()	Sync Ack, Sync Ack	accept()			accept()				RTT
Request	Anfrage senden	Anfrage lesen	write()	Ack, Push Ack	read()	sendto()	Push, Sync, Fin	read()	sendto()	→	recvfrom()	RTT
Processing		Bearbeitung			_doing()			_doing()			_doing()	SPT
Response	Antwort lesen	Antwort senden	read()	Ack, Push, Fin Ack	write()	read()	Ack, Push, Sync, Fin Ack	write()	recvfrom()	←	sendto()	1/2 RTT
Close	Verbindung schließen	Verbindung schließen	close()	Ack, Push, Fin Ack	close()	close()		close()				

Abbildung 26: Transaktionen in der Transportschicht

3.2.3 Anwendungsschicht

Mit der Einführung des HTTP/1.1 wurden viele wichtige Änderungen in Bezug auf die Performanz des Protokolls vorgenommen, wie schon in *Kapitel 2.2.4.1: HTTP* erwähnt. Zu den wichtigsten Änderungen bezüglich Performanz gehören die persistenten Verbindungen und das Pipelining.

3.2.3.1 HTTP Persistente Verbindungen

Die persistenten Verbindungen sind nahezu identisch zu der „Keep-Alive“ Erweiterung des HTTP/1.0, der Unterschied zwischen den Implementationen besteht in dem verbesserten Verhalten bei Verbindungen über kaskadierte Proxyserver. Durch diese Neuerung beim HTTP/1.1 wird eine Verbindung auch nach dem erfolgreichen Abschluß einer Transaktion (z.B. dem Lesen eines HTML-Dokumentes) offengehalten, so daß weitere Transaktionen (wie z.B. Inline-Grafiken) ebenfalls über diese Verbindung abgewickelt werden können. Damit werden zusätzliche Verbindungen eingespart, die beim HTTP/1.0 nötig wären, um jedes Objekt einzeln vom Server über eine separate Verbindung abzuholen. Der Effekt davon ist vielfältig. Zum einen spart man sich die zusätzlichen TCP-Pakete, die jeweils zum Auf- und Abbau der Verbindung für jedes neue Objekt nötig wären, was das Transfervolumen durch einen geringeren TCP-Overhead reduziert. Zum anderen spart man jeweils die Zeit, die der Verbindungsaufbau für eine neue Verbindung normalerweise benötigt. Dies ist immerhin 1 RTT (round-trip time), die im Schnitt bei WAN-Verbindungen in der Größenordnung von 100 ms liegt. Bei einem ansprechenden Design einer HTML-Seite sind neun eingebundene Grafiken durchaus keine Seltenheit. In diesem Fall erreicht man mit persistenten Verbindungen eine Ersparnis im Sekunden-Bereich. Hinzu kommt noch ein weiterer Aspekt, der sich positiv auswirken kann: Das TCP Slow-Start Verhalten kann bei einer schnellen Folgen von Transaktionen über eine Verbindung, wie das im allgemeinen bei einer HTML-Seite mit vielen Grafiken der Fall ist, greift nur auf die Summe der Transaktionen, nicht auf jede einzelne Transaktion, wie das der Fall wäre, wenn man für jede Transaktion eine einzelne Verbindung öffnen müsste. Zur Erklärung: Beim TCP Slow-Start erwartet der Sender vom Empfänger in einer Verbindung nach N Bytes ein Acknowledge. Fällt dieses positiv aus, so wird das Fenster in der Größe verdoppelt, so daß nun erst nach 2N Bytes ein Acknowledge erwartet wird. Damit wird klar, daß kleine Pakete, die größer als die anfängliche Fenstergröße sind, jeweils zusätzliche Acknowledge-Pakete erzeugen. Je größer das Transfervolumen ist, desto positiver ist das Verhältnis zwischen Volumen und Acknowledge-Paketen.

3.2.3.2 HTTP Pipelining

Beim Pipelining werden mehrere Client-Request gepuffert und nach einem bestimmten Kriterium später versendet. Zwei mögliche Kriterien für das Senden des Puffer-Inhaltes an den Server sind das Überschreiten einer bestimmten Puffer-Größe und Überschreiten einer gewissen Zeit, in welcher der Puffer gefüllt wird.

Quantitative Aussagen über den Grad der Verbesserungen werden in *Kapitel 4.4.1: HTTP Pipelining und Persistente Verbindungen* getroffen.

3.3 Optimierung der Datenrepräsentation

Eine Maßnahme zur Optimierung der Datenrepräsentation ist die Kompression der Daten (*Kapitel 2.6.3: Kompression*). Sie kann in allen Protokollebenen realisiert werden. Kompressionsalgorithmen, die eine möglichst hohe Kompressionsrate zum Ziel haben, können ausgesprochen rechenintensiv sein und eignen sich damit nur eingeschränkt für echtzeitnahe Kompression von Daten, die mit hoher Datenrate über das Netzwerk übertragen werden müssen. Algorithmen mit moderater Kompression oder Dekompression von Daten können hingegen durchaus vom Systemprozessor mitgetragen werden kann. Allerdings könnte je nach Auslastung des Systems (z.B. Router) ein Koprozessor nötig werden, um für die restlichen Systemaufgaben (Datenkapselung, Filtering, Accounting, ..) keine Leistungseinbußen in Kauf nehmen zu müssen.

Insgesamt ist es recht erstaunlich, daß in der ursprünglichen Spezifikation der TCP/IP-Protokollfamilie keine Datenkompression vorgesehen war, noch nicht einmal eine einfache Run-Length Kodierung wie beim FAX.

Eine weitere Optimierung, die man vielleicht nicht unmittelbar unter diesem Punkt einordnen würde ist das Caching von Daten. Aber ebenfalls wie bei der Kompression wird auch hier eine Optimierung des Protokolls durch eine Einsparung beim Transfervolumen erreicht. Spezifiziert und realisiert ist diese Form der Optimierung jedoch nur auf der Anwendungsebene beim WWW-Protokoll HTTP.

3.3.1 Vermittlungsschicht

Eine Kompression in der Vermittlungsschicht hat einen bestechenden Vorteil: Von dieser Art der Kompression profitieren automatisch alle höherliegenden Protokolle, ohne daß ein einziges Byte im Sourcecode oder der Binärdatei einer Anwendung geändert werden muß. Durch das Schichtenmodell stellt sich die Umgebung für die TCP/IP-basierte Anwendung transparent dar. Die Kompression auf IP-Ebene erhöht den Datendurchsatz, ohne die Datenrepräsentation der darüberliegenden Schichten zu verändern.

3.3.1.1 IPv6

Der Einfluß von IPv6 auf die Verbindungssteuerung wurde bereits als ein Aspekt der folgenden Untersuchungen angesprochen. In diesem Zuge soll auch der Aspekt diskutiert werden, welche Auswirkungen der Einsatz dieses Protokolls auf die Datenrepräsentation hat. Zwar wurde das Design von IPv6 nicht darauf ausgelegt, das Volumen von IP-Paketen drastisch zu verringern, jedoch gibt es zwei gegensätzliche Veränderungen, die eine Untersuchung wert sind: zum einen ein größerer IP-Header und zum anderen die Einführung von sogenannten Jumbo-Paketen.

3.3.1.2 IPComp

Eine für jede IP-Verbindung gültige Kompression wurde von der „IP Payload Compression Protocol Working Group“ Mitte 1997 erstmals im Rahmen eines IETF-Drafts [95] vorgestellt. Dort wird das IPComp Protokoll vorgestellt, welches mit verlustloser Kompression die Inhalte von IP Datagrammen komprimiert. Dadurch soll die Performanz der Kommunikation zwischen zwei Knoten (Rechner oder Gateway) gesteigert werden. Das soll durch die Reduktion des Transport-

volumens erreicht werden, was zusätzlich eine verkürzte Antwortzeit der beteiligten Systeme mit sich bringt. Es wird aber ebenfalls darauf hingewiesen, daß dafür eine entsprechende Rechenleistung nötig ist, die entweder von der CPU des Systems zur Verfügung gestellt wird oder durch einen separaten Koprozessor. Das Protokoll hat seit Ende 1998 den Status eines RFCs [96] erlangt.

Besonders sinnvoll in der Anwendung wird die Kompression auf IP-Ebene, wenn auf IP-Ebene eine Verschlüsselung vorgenommen wurde. In diesem Fall wird eine Kompression auf unterliegenden Protokoll-Schichten wie dem PPP-Layer ineffizient, da dieser Schicht die verschlüsselten Daten zur Kompression vorliegen, die nach Definition der Verschlüsselung stets einen zufälligen Charakter haben. Dementsprechend sinnvoll ist das Zusammenspiel zwischen Kompressionsprotokoll IPComp und Verschlüsselungsprotokoll IPSec auf IP-Ebene. Dabei muß natürlich gewährleistet werden, daß zunächst die Kompression und erst dann die Verschlüsselung vorgenommen wird.

Die grundsätzliche Regel für die Anwendung der IPComp-Kompression ist, daß man möglichst früh innerhalb der Vermittlungsschicht die Kompression anwendet und erst danach weitere Verarbeitungsprozeduren in der Vermittlungsschicht anwendet. So muß die Kompression nicht nur vor der Verschlüsselung, sondern auch vor der Fragmentierung des IP-Paketes und im Falle von IPv6 vor dem Hinzufügen von Hop-By-Hop- und Routing-Informationen. Dies ist notwending, da diese Informationen möglicherweise auf jedem der beteiligten Router auf dem Zustellungspfad ausgewertet werden müssen.

Es können unterschiedliche Kompressions-Algorithmen angewendet werden, jedoch muß jeder Algorithmus prüfen, ob die komprimierte Form tatsächlich auch kleiner geworden ist. Ist dies nicht der Fall, wird das Paket unkomprimiert verschickt. Damit erhält man unabhängig vom Kompressionsalgorithmus die Gewährleistung, daß das Datenvolumen nicht ansteigt.

Bei IPv4 wird der IP-Header dahingehend abgeändert, daß das Protokoll-Feld auf den Wert 108 (für IPComp) gesetzt und dementsprechend noch die IP-Header Prüfsumme angepaßt wird. Zudem wird noch der IPComp-Header unmittelbar vor die komprimierte Payload des IPv4 Paketes eingefügt.

Bei IPv6 muß die Payload-Length korrigiert werden und das entsprechende Next-Header Feld auf den Wert 108 gesetzt werden. Es muß darauf geachtet werden, daß der IPComp-Header nach dem Fragmentation-Header eingefügt wird.

Der IPComp-Header hat die folgende Gestalt:

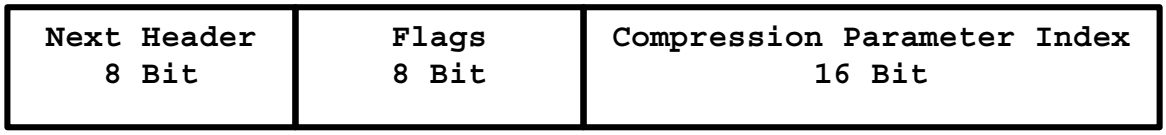

Abbildung 27: IPComp-Header

Dabei wird das NextHeader-Feld mit dem IPv4 Protokoll-Feld oder dem IPv6-NextHeader Feld belegt. Das Flag-Feld ist noch ungenutzt und muß mit dem Wert Null belegt werden. Im Feld Compression Parameter Index wird der Kompressionsalgorithmus eingetragen.

Die Auswahl welcher Kompressionsalgorithmus und welche zusätzlichen Parameter für die Verbindung gewählt werden, auch IPCA (IPComp Association) genannt, kann entweder manuell

oder dynamisch erfolgen. Die dynamische Auswahl im Zusammenhang mit IP Security kann über das Protokoll ISAKMP [61] erfolgen. Das ISAKMP (Internet Security Association and Key Management Protocol) übernimmt dabei die Verhandlung für die IPComp Association.

Zur Zeit sind zwei Kompressionsalgorithmen (*Kapitel 2.6.3: Kompression*) definiert [80]: IPCOMP_DEFLATE [79], der zlib deflate Algorithmus und IPCOMP_LZS [31], der Stac Electronics LZS Algorithmus. Weitere Algorithmen können implementiert werden, müssen jedoch in Form eines RFC vorliegen, um wie die beiden genannten Algorithmen als „well-known" Algorithmus in der Liste der bekannten Parameter für ISAKMP [80] aufgenommen zu werden. Neben diesen „well-known" Algorithmen, deren „IPCOMP Transform Identifiers" für den Compression Parameter Index im Bereich von 0 bis 63 liegt, gibt es auch noch die Möglichkeit lokal gültige Algorithmen einzusetzen, indem man diesem einen Transform Identifier im Bereich von 61440 bis 65535 zuweist, was insbesondere für VPNs interessant ist.

Der Vorteil von IPComp liegt klar auf der Hand: Man erreicht eine Datenkompression in IP-basierten Netzen, ohne auf die darunter liegende Schicht (Ethernet, ATM, PPP, etc.) Rücksicht nehmen zu müssen. Auch muß nicht jede Anwendung für sich selbst eine Kompression zu implementieren. Es wird nur eine Implementation für die Kompression an genau einer Stelle benötigt: In der IP-Schicht.

3.3.2 Transportschicht

Für die Transportschicht werden zwei Protokolle diskutiert: T/TCP und die Van Jacobson TCP/IP Header Compression. Dabei ist nur die Header Compression mit der Optimierung der Datenrepräsentation motiviert. Dennoch wird auch der Effekt der Anwendung von T/TCP auf die Datenrepräsentation in *Kapitel 4: Messungen und Simulationen* untersucht.

3.3.2.1 Van Jacobson TCP/IP Header compression

Für den Fall einer Kommunikation über eine serielle Leitung wurde von Van Jacobson im Jahre 1990 eine Header Kompression für TCP/IP Pakete definiert [108], mit der es möglich ist eine Kompression des Headers bis zum Faktor 8 zu erreichen. Die TCP/IP-Header Kompression ist weltweit eine Standard-Option für die seriellen Protokolle SLIP und PPP.

Der Effekt des reduzierten Datenvolumens beruht auf einer Verkleinerung des TCP/IP-Headers. Der Ausdruck Kompression ist in diesem Falle ein wenig irreführend, weil man damit zumeist einen Kompressionsalgorithmus wie z.B. LZW assoziiert. Tatsächlich werden jedoch zum einen Teil redundante Informationen aus dem TCP-Header weggelassen und zum anderen Teil werden nur die Differenzen der Header-Informationen übertragen. Daß man mit diesem relativ einfachen Schema der differentiellen Kodierung zumindest für kleine Pakete hohe Kompressionsraten erzielen kann, liegt vor allem an dem typischen Einsatzgebiet von seriellen Leitungen, den Point-To-Point Verbindungen. So bestehen bei diesen physikalischen Verbindungen im allgemeinen nur wenige logische TCP-Verbindungen (diese sind durch das Tupel <Source-IP, Destination-IP, Source-Port, Destination-Port> gekennzeichnet). Auf der anderen Seite ist die Zeit der kleinen Datenpakete (Telnet, Chat, Talk, IRC, ..) schon vorbei. Das WWW dominiert auch beim häuslichen Anwender, das Transfervolumen wächst und damit auch die durchschnittliche Paketgröße. Und mit wachsender Paketgröße verringert sich die Effektivität der Header-Kompression.

Dennoch an dieser Stelle einige Worte zum Verfahren: Sender und Empfänger pflegen eine Liste der logischen Verbindungen in Form von Verbindungsnummern. Dadurch ist es möglich die über eine logische Verbindung konstanten Felder – in *Abbildung 28* grau unterlegt – wegzulassen und statt dessen nur die verbindungsspezifische Verbindungsnummer (connection number) zu übertragen. Alleine dadurch werden 20 Bytes eingespart. Bei Verwendung eines Link-Level Protokolls, welches die Paketlänge kennt, kann im weiteren auch noch das Feld „Total Length" entfallen. Auch die Prüfsumme (IP Header Checksum) kann entfallen, muß jedoch beim Emfänger nach der Vervollständigung des Paketes wieder neu generiert werden.

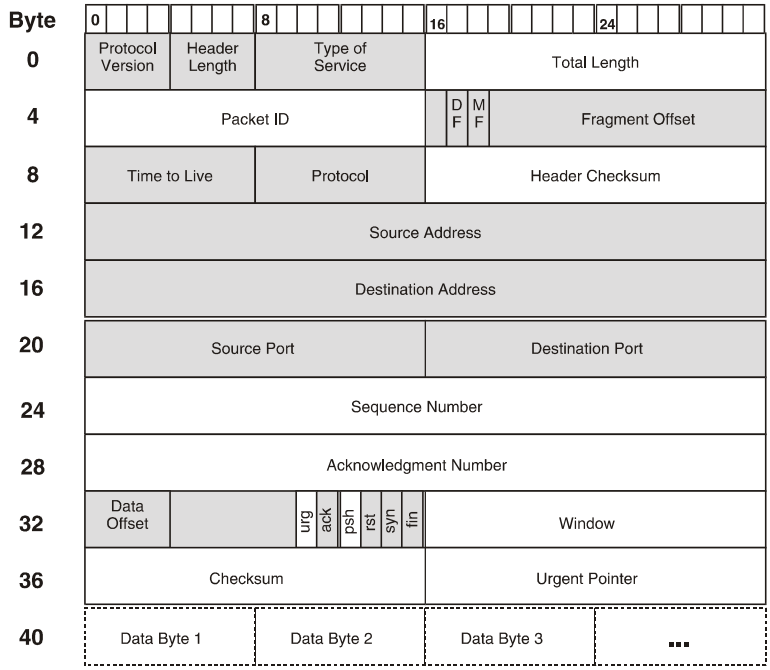

Abbildung 28: TCP/IP-Datagramm mit konstanten Header-Feldern [108]

Die übrigen Informationen sind während einer Verbindung nicht grundsätzlich konstant. Eine Veränderung dieser Werte wird dem Empfänger daher über Flags im ersten Byte des Paketes signalisiert. Des weiteren kann man die 16-Bit Differenz von Packet-ID und Sequence Number übertragen, statt des vollständigen Feldes. Zusammen mit der Berücksichtigung von einigen Spezialfällen kann man die Größe des Headers meist auf drei bis fünf Bytes reduzieren. Dies ist auch aus *Abbildung 29* zu ersehen. In Byte 0 sind einige Flags enthalten, die signalisieren, ob die entsprechenden Differenzfelder im weiteren Header vorkommen oder nicht. Dementsprechend fängt der Datenbereich des Paketes bei Byte $n, 3 \leq n \leq 16$ an.

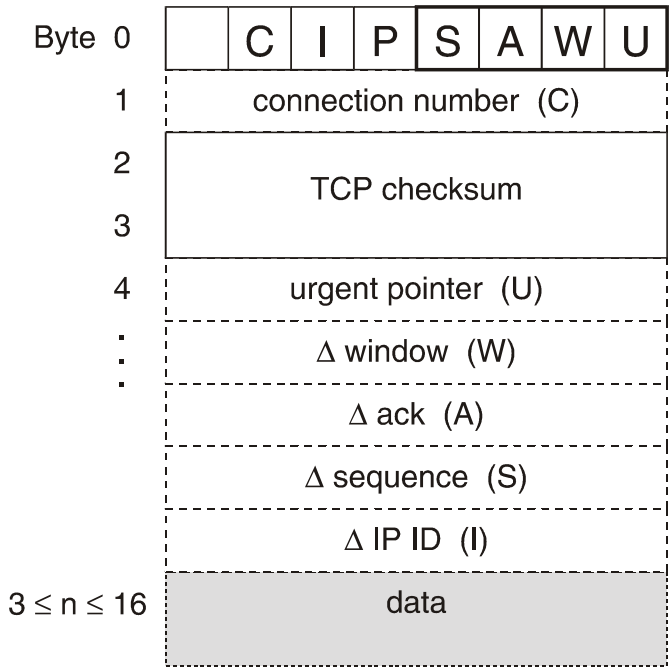

Abbildung 29: TCP/IP-Datagramm mit komprimiertem Header [108]

Durch das begrenzte Einsatzgebiet soll diese spezielle Methode der Optimierung hier nicht detaillierter erläutert werden, ist aber dennoch erwähnenswert, da es ein sehr früher Ansatz der Optimierung ist und hier bereits eine „Connection Number“ verwendet wird, wie sie sinngemäß als „Connection Count“ auch im T/TCP wiederzufinden ist. Zudem ist die Implementation mit ca. 250 Zeilen C-Quelltext sehr einfach, wie man an der Beispiel-Implementation im Anhang von [108] sehen kann.

3.3.3 Anwendungsschicht

Für die Anwendungsschicht werden hier die beiden wesentlichsten Möglichkeiten besprochen, das Volumen zu reduzieren. Bei der ersten Möglichkeit nutzt man mit dem Caching von Dokumenten aus, daß dieselben Dokumente von einem bestimmten Zielserver mehrfach angefordert werden, entweder von demselben Client nacheinander oder von mehreren verschiedenen Clients. Dabei erkennt ein zwischen Client und Server zwischengeschaltetes System, der HTTP-Cache, identische Dokumentanfragen und beantwortet diese direkt anstatt sie zum Server weiterzuleiten. Der Effekt ist eine Reduktion des teuren Datenvolumens zwischen Cacheserver und dem Zielserver. Eine andere Möglichkeit der Volumenreduktion besteht in der Kompression der übertragenen Inhalte. In diesem Fall beantwortet der Server eine Dokumentenanfrage mit dem Dokument in komprimierter Form, wobei im Header der Antwort vermerkt ist, daß das nun folgende Dokument in komprimierter Form übertragen wird. Beide Systeme sind etabliert und werden im folgenden vorgestellt.

3.3.3.1 HTTP Caching

Das HTTP-Caching ist allgemein beschrieben [62] ein Prozess, den Zugriff auf WWW-Dokumente effizienter zu gestalten. Dies soll erreicht werden durch lokale Kopien häufig benutzter Dokumente. Die Kopien der WWW-Dokumente werden als Cache bezeichnet. Der Prozess des HTTP-Cachings wird mit einem Caching Proxyserver erreicht.

Ein Proxyserver ist eine Zwischenstation in der Kommunikation zwischen Clientsystem und Serversystem, welche für das Clientsystem die Kommunikation mit dem Serversystem übernimmt und das (Zwischen-)Ergebnis an das Clientsystem liefert. In diesem Sinne ist auch jeder Cacheserver ein Proxyserver, jedoch gilt nicht die Umkehrung, da nicht jeder Proxyserver die weitergeleiteten Informationen in einem lokalen Cache abspeichert. Proxyserver werden typischerweise von Firmen, Institutionen oder kleineren Arbeitsgruppen eingesetzt, um aus dem lokalen Intranet heraus im weltweiten Internet über den Proxyserver zu surfen. Das hat den Vorteil, daß nur der Proxyserver einen Zugang (und registrierte IP-Nummer) zum Internet haben muß. Neben diesen anbindungsspezifischen Aufgaben kann ein Proxyserver auch sicherheitsrelevante Funktionen erfüllen und kann je nach Einsatzgebiet auch dazu verwendet werden, unerwünschte URL's zu filtern.

Beim Caching werden lokale Kopien von Dokumenten dem Clientsystem übermittelt. Diese lokalen Kopien sollten möglichst nah am Clientsystem im Cacheserver vorgehalten werden. Dadurch wird im Falle eines „Cache-Hits" sowohl der Datenverkehr zwischen Cacheserver und Serversystem reduziert, als auch die Antwortzeit für die Antwort vermindert. Der Begriff des Cache-Hits bedeutet, daß ein vom Clientsystem angefordertes Dokument im lokalen Cache des Cacheservers vorliegt (und entsprechend der Anfrage gültig ist) und direkt an das Clientsystem geliefert werden kann. Vom „Cache-Miss" spricht man, wenn keine gültige lokale Kopie des angeforderten Dokumentes vorliegt. In diesem Fall erhöht sich die Latenzzeit für die Antwort ein wenig, weil der Cacheserver ein zusätzliches Glied in der Kommunikationskette zwischen Clientsystem und Zielserver ist.

Das Caching von WWW-Dokumenten beruht auf einer Baumstruktur der Kommunikationswege, bei der die Wurzel der Cacheserver ist und dieser mindestens jeweils einen Kindknoten von zwei Kindknotentypen hat. Der eine Knotentyp sind die Clientsysteme, der andere Typ sind die Serversysteme. Im einfachsten Fall sind Clientsysteme WWW-Browser und Serversysteme WWW-Server (*Abbildung 30*). In diesem Fall stellt das Clientsystem dem Cacheserver seine Anfrage. Der Cacheserver schaut nun in seinen lokalen Cache nach, ob dort bereits die Antwort auf diese Anfrage vorhanden ist. Ist sie vorhanden, erhält das Clientsystem sofort die Antwort, ansonsten holt der Cacheserver für das Clientsystem die Antwort, übermittelt dem Clientsystem die Antwort und speichert diese zusätzlich in seinem lokalen Cache ab.

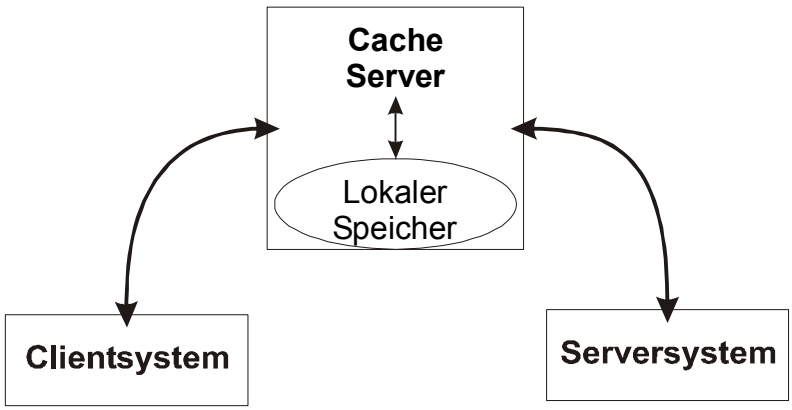

Abbildung 30: WWW-Cacheserver Topologie, einfach

Um die Aktualität der Daten zu gewährleisten, ist der Cacheserver mit diversen Schwellwerten konfiguriert. In der einfachsten Variante gibt es zwei Schwellwerte: MIN_AGE gibt das max. Alter an, zu dem Dokumente aus dem Cache sofort an den Client geliefert werden, ohne das Serversystem nach einer neueren Version zu fragen. MAX_AGE gibt das Alter an, nach dem das Dokument in jedem Fall ungültig ist und somit vom Serversystem neu angefordert werden muß. Zu den ersten WWW-Cacheservern gehört zweifellos der CERN-httpd [58], ein WWW-Server mit der Option des (caching) Proxyservers. Seine Entwicklung ist jedoch 1996 zugunsten des Java basierten WWW-Servers Jigsaw [59] eingestellt worden.

Mittlerweile haben sich jedoch weitaus verfeinerte Caching-Algorithmen als oben beschrieben etabliert. Besonders gut zu verfolgen ist diese Entwicklung am Projekt Squid. Der Squid Web Proxy Cache [113] ist einer der am weitest verbreitetsten Cacheserver. Das Programm ist im Quelltext frei verfügbar (free, Open-Source software) und stammt von dem Projekt „Harvest Information Discovery and Access Project“ [46] ab. Der Algorithmus des Squid WWW-Cacheservers in der Version 1.1 ist in [8] skizziert, ein entsprechendes Diagramm für Squid Version 2.x wurde nach [112] erstellt und ist in *Abbildung 31* zu sehen.

In diesem Algorithmus soll durch eine Reihe von Abfragen gewährleistet werden, daß nur dann ein Dokument von dem Server geholt wird, wenn es unbedingt notwendig ist. Zwei eindeutige Kriterien für den Status des Dokumentes sind die optionalen Parameter CLIENT_MAX_AGE und EXPIRES. Sie werden über den HTTP-Header an den Client geliefert.

Das Ergebnis der weiteren Abfragen hängt von den konfigurierten Refresh-Parametern des Cacheservers ab. Diese Parameter sind CONF_MIN, CONF_PERCENT und CONF_MAX. CONF_MIN gibt an, wieviele Minuten ein Objekt ohne explizite EXPIRES-Zeit als gültig (FRESH) angesehen werden soll. CONF_PERCENT ist die Prozentzahl des Objektalters (Zeit seit der letzten Veränderung), innerhalb der ein Objekt als gültig angesehen wird. CONF_MAX ist das maximale Alter eines gültigen Dokumentes.

Zu den berechneten Parametern gehören: OBJ_DATE ist der Zeitpunkt, zu dem das Objekt vom Server geliefert wurde. OBJ_LASTMOD ist der letzte Modifikationszeitpunkt. OBJ_AGE ist das Alter, seitdem das Objekt vom Server empfangen wurde. LM_AGE gibt an, wie alt das Objekt war, als es vom Server empfangen wurde. LM_FACTOR ist der Quotient aus OBJ_AGE und LM_AGE.

Diese Parameter werden dazu verwendet, um die Wahrscheinlichkeit abzuschätzen, mit der das Dokument noch gültig ist. Dazu dient auch der Faktor LM_FACTOR, der das Verhältnis aus dem Cache- und dem Modification-Alter ist. Übersteigt dieser Dokument-spezifische Wert einen vorkonfigurierten Schwellwert, so wird das Dokument als ungültig angesehen. Der LM_FACTOR eines Dokumentes vergrößert sich mit dem Cache-Alter des Dokumentes. Zudem wird er umso größer, je kleiner der Abstand zwischen Erstellung und Modifikation des Dokumentes ist. Dies kann als Parameter für die Dynamik eines Dokumentes betrachtet werden.

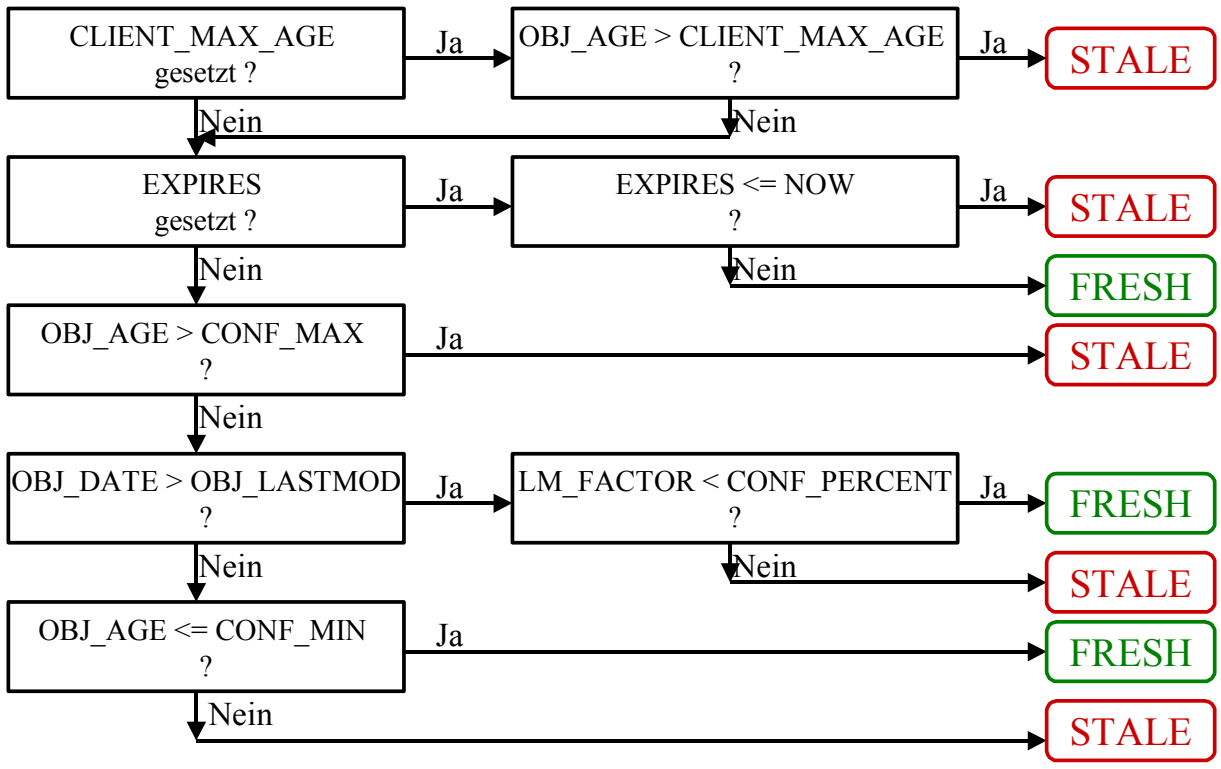

Abbildung 31: Squid-2 caching algorithm [8]

Der Harvest Cacheserver war auch das erste System, das eine hierarchische Cache-Topologie unterstützte. Auch diese Technologie wurde im Squid-Cacheserver weiterentwickelt. Es sind nun Multilevel-Hierarchien von Cacheservern möglich. Zudem kann man die verbundenen Cacheserver als „neighbors“ (gleiche Ebene) oder „parents“ (darüber liegende Ebene) definieren und diesen darüberhinaus Domainzuständigkeiten (Cacheserver1 liefert nur .com-Domains, etc.) zuweisen. Anwendung finden diese komplexen Topologien bereits in mehreren Projekten, wie z.B. dem DFN-Cache-Service [83], dem europäischem Projekt DESIRE [40] und dem NSF gefördertem Projekt IRCACHE [114]. Die vom DFN-Verein verwalteten Cacheserver sind in *Abbildung 32* zu sehen. Der Multilevel-Charakter ergibt sich aus den an die eingezeichneten Cacheserver angeschlossenen Servern der Universitäten und Institute, die wiederum interne Cacheserver betreiben können.

Im folgenden wird eine konkrete Konzeption einer hierarchischen Cache-Topologie beschrieben. Als Beispiel dient das Cacheserver Projekt des DFN-Vereins, der ein solches Projekt für das WiN realisiert und dokumentiert [37] hat.

Das Ziel des Projektes bestand in der Planung und Realisierung eines Cache-Services für das WiN. Dazu sollten zehn Cacheserver in den verhältnismäßg schnellen Backbone des WiN plaziert werden und die angeschlossenen Cacheserver der Kunden (i.A. Institute und Hochschulen) versorgen. Caching-Software sollte beurteilt und Konfigurationshinweise ausgesprochen werden.

Als Caching-Software entschied man sich für den Squid-Cache, der über ACL's (Access Control Lists) so konfiguriert wurde, daß der Zugriff auf die Server auf angemeldete Cacheserver der Universitäten begrenzt war. Dies diente dem Schutz vor Mißbrauch der Cacheserver.

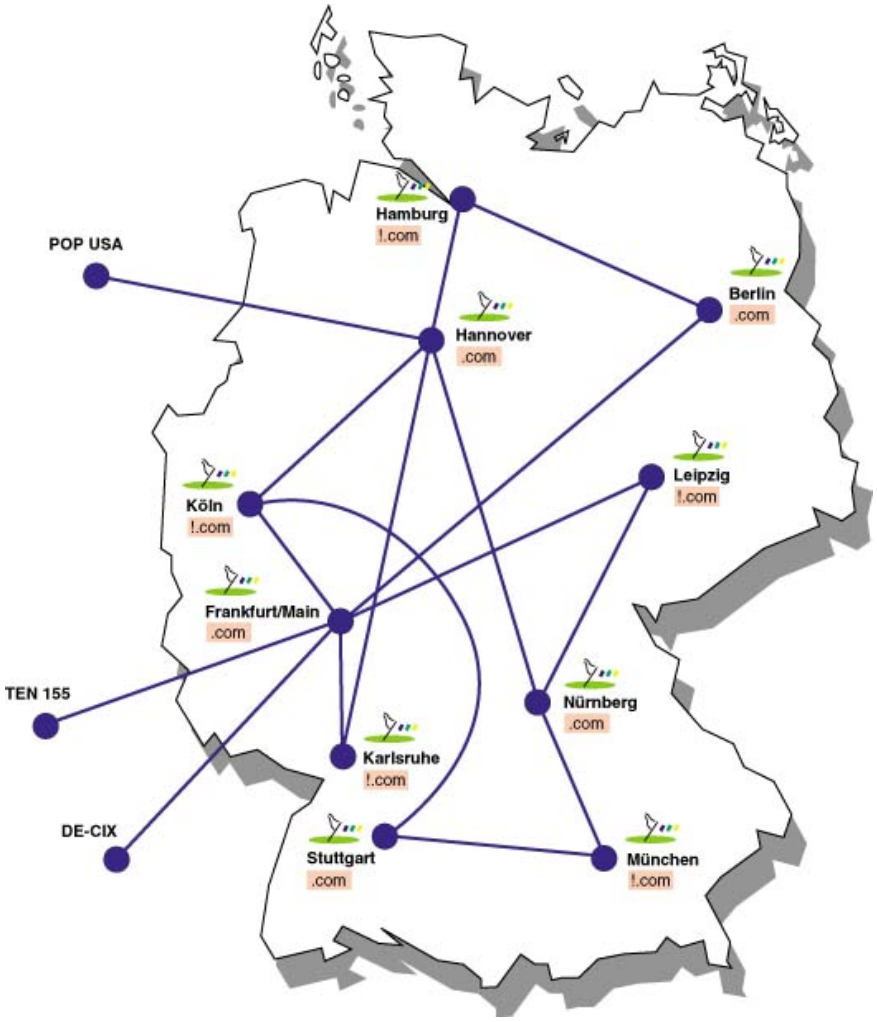

Abbildung 32: Karte der DFN Cacheserver [83]

Die geografische Topologie der DFN-Cacheserver ergab sich schon fast zwangsweise aus der Topologie des DFN-Backbones und ist in *Abbildung 32* zu sehen.

Im ersten Ansatz wurde eine Multilevel-Hierarchie (*Abbildung 33*) realisiert, indem den lokalen Cacheservern (Ebene 1) jeweils zwei DFN-Caches (Ebene 2) als Parent-Server zugeordnet wurden. Diese konnten dann auf die DFN-Caches (Ebene 3) zugreifen, denen jeweils bestimmte Toplevel Domains zugeordnet waren. Das heißt, daß nur bestimmte Cacheserver z.B. Anfragen für Seiten der Domain www..*..com entgegengenommen haben. Zusätzlich zu den HTTP-Verbindungen sind in

dieser Abbildung auch noch die Verbindungen eingezeichnet, die sich aus dem ICP (Internet Cache Protocol) ergeben können. Dieses Protokoll dient der Synchronisierung der Server untereinander.

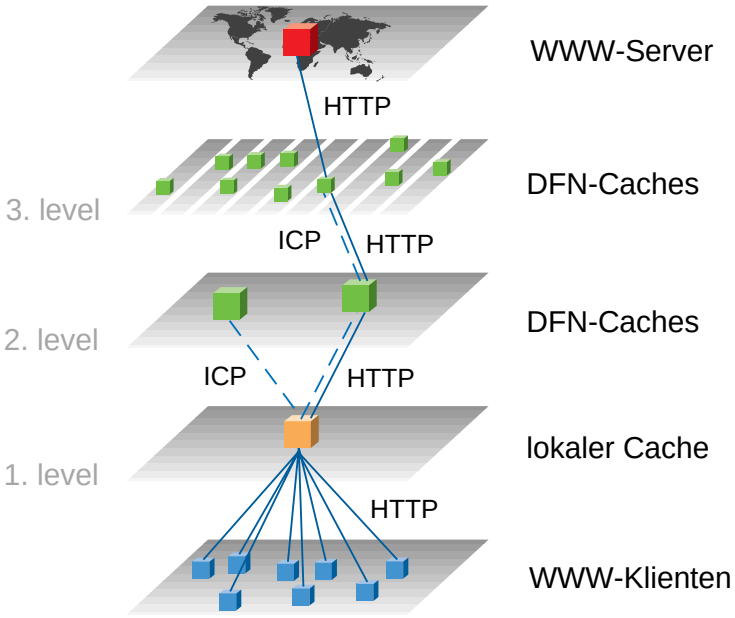

Abbildung 33: DFN Cacheserver Topologie, 3-stufig [37]

Bereits nach kurzer Zeit erkannte man, daß die gewählte Architektur zu Problemen führt.

So waren für die Toplevel Domain .com nur zwei Cacheserver vorgesehen, der Anteil der Anfragen nach .com Domains betrug jedoch 50%. Das führte dazu, daß die Systeme überlastet waren und der Ausfall einer der beiden Cacheserver den verbleibenden Cacheserver derart überlastete, daß überhaupt kein sinnvoller Betrieb mehr gewährleistet war.

Zudem war für die übrigen Domains jeweils nur ein einziger Server zuständig. Bei Ausfall eines Systems waren alle Domains, für die er zuständig war, im DFN-Netz nicht mehr über Cacheserver erreichbar.

Aber auch im störungsfreien Normalbetrieb hat sich ergeben, daß der Fall eines Cache-Miss (die Seite ist auf keinem der Cacheserver vorhanden) die Verzögerungszeit über die dreistufige Hierarchie dem Anwender nicht mehr zumutbar war.

Diese Kritikpunkte wurden in einem neuen Konzept berücksichtigt. In dieser neuen Konzeption der des DFN Cache-Verbundes wurde die Hälfte aller Cacheserver (vier) für die Domain .com vorgesehen, die andere Hälfte für die übrigen Toplevel Domains. Damit wurde dem Verkehrsanteil für .com-Domains und der Ausfallsicherheit für alle Domains Rechnung getragen. Die Hierarchie innerhalb der DFN Cacheserver wurde durch den Verzicht auf Parent-Beziehungen zugunsten von Sibling-Beziehungen abgeschafft und damit eine Stufe in der Hierarchie eliminiert. Die Topologie ist in *Abbildung 34* zu sehen.

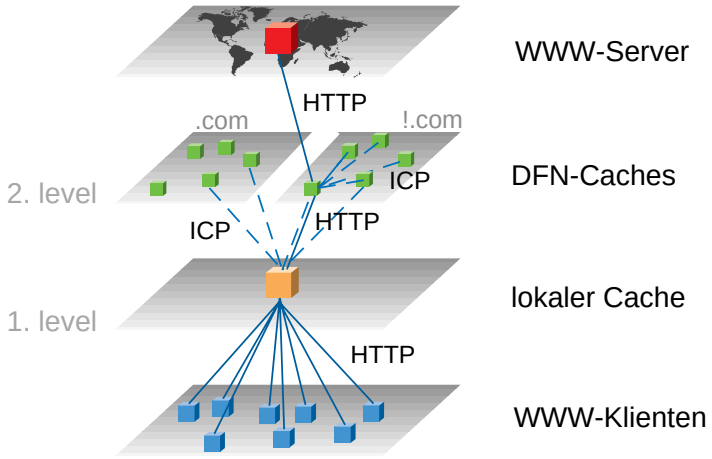

Abbildung 34: DFN Cacheserver Topologie, 2-stufig [37]

Das Ergebnis dieses zweiten Ansatzes ist die Verdopplung des Datenvolumens, welches an die lokalen Cacheserver ausgeliefert werden konnte unter Reduzierung der Wartezeiten bei Steigerung der Ausfallsicherheit.

3.3.3.2 HTTP Prefetching

Eine weitere Strategie zur Steigerung der Effizienz eines Cacheservers ist das HTTP-Prefetching. Das Prefetching basiert darauf, Web-Inhalte von einem Zielserver anzufordern und lokal zu speichern, noch bevor das Clientsystem diese anfordert. Damit werden die bei Anwendung von HTTP-Caching üblichen zusätzlichen Latenzzeiten nicht nur vermieden, sondern sogar die Performanz einer Anfrage gegenüber einem direkten Abruf vom Zielserver deutlich gesteigert.

Die Problematik einer solchen Optimierungsstrategie liegt zum einen darin, daß der Cacheserver nicht sicher weiß, welche Web-Inhalte in den folgenden Anfragen vom Clientsystem angefordert werden. Desweiteren werden vom Cacheserver teure Datentransfers durchgeführt, die eventuell nicht vom Clientsystem benötigt werden.

Entscheidenden Einfluß auf den Nutzen dieser Optimierungsstrategie hat die dem Prefetching zugrunde liegende Heuristik. Sie bestimmt, welche Webinhalte (URLs) im voraus angefordert werden, und mit welcher Rekursionstiefe diese URLs geholt und lokal gespeichert werden.

Der triviale Ansatz, alle Dokumente, die in einer vom Clientsystem angeforderten URL referenziert sind, im voraus zu laden, ist bei vorhandener Bandbreite sicherlich für das Clientsystem sehr angenehm, verursacht jedoch schon bei zehn Referenzen pro Seite eine Verzehnfachung der teuren Datentransfers.

Ein Ansatz, diese Problematik zu entschärfen, kommt von Markatos und Chronaki. In Ihrem Top-10 Ansatz [60] beschränken sie das Prefetching auf die populärsten Dokumente, die aus den Logdateien des Cacheservers ermittelt werden. In ihren Experimenten erreichten sie eine Hitrate von bis zu 60%, , wobei der Datentransfer lediglich um bis zu 20% ansteigt.

HTTP-Prefetching wird z.B. in dem HTTP-Server Viking [91] von RobTex unterstützt. Eine Analyse des HTTP Prefetchings findet im Rahmen dieser Arbeit nicht statt.

3.3.3.3 HTTP Content-Encoding

Ab der Version 1.1 des HTTP wird nun auch die Transportkompression von Dokumenten unterstützt, sodaß z.B. HTML-Dokumente auf dem Transportweg zum Client (WWW-Browser) komprimiert werden können. Als Kompressionsalgorithmen für die Kompression sind zunächst die Verfahren gzip, compress und deflate vorgesehen. Alle diese Verfahren basieren auf Varianten einer Lempel-Ziv Kodierung des Datenstroms. Dabei werden beim Kodieren häufig vorkommende Zeichenketten durch einen kürzeren Kode ersetzt. Diese Kodes werden bei der Kodierungsphase wieder in die entsprechenden Zeichenketten umgewandelt. Durch diese Kodierung erhält man eine Reduktion des zu übertragenden Datenvolumens.

Eine Steigerung der Effizienz bei der Kompression ist z.B. auch durch die konsistente Anwendung von Groß- oder Kleinschreibung für die HTML-Tags möglich, wie von Nielsen und Prud'hommeaux in [78] beschrieben . Dies ist auch ausgesprochen plausibel, wenn man das oben angesprochene Prinzip der Kodierung kennt. Bestimmte HTML-Tags tauchen in einem HTML-Dokument häufiger auf. So benötigt das zweifach vorkommende Element „<TABLE>“ bei der Kodierung nur einen Platz in der Code-Tabelle des Kodierers, während die Elemente „<TaBlE>“ und „<tAbLe>“ vom Kodierer als unterschiedlich angesehen werden und zu einer schlechteren Kodierung führen werden.

3.4 Zusammenfassung

Im Laufe des Kapitels wurde eine Auswahl an verschiedenen Optimierungsansätzen für unterschiedliche Schichten des OSI-Modells vorgestellt. Diese Auswahl soll repräsentativ sein und die wichtigsten und bekanntesten Optimierungsansätze beinhalten, soll und kann aber nicht den Anspruch auf Vollständigkeit suggerieren. In *Tabelle 16* sind die angesprochenen Ansätze eingetragen. Messungen und Simulationen zu diesen Ansätzen werden im nächsten Kapitel diskutiert.

Schicht	Verb.-Steuerung	Datenrepräsentation
3-Vermittlungsschicht	IPv6	IPv6 IPComp
4-Transportschicht	T/TCP	TCP/IP-Header Kompression T/TCP
5-Anwendungsschicht	HTTP pers. Verb. HTTP Pipelining	HTTP Caching HTTP Content-Encoding

Tabelle 16: Klassifikationstafel Optimierungsansätze

Es wurde eine Methode zur Klassifikation vorgestellt, die eine Einordnung von Protokolloptimierungen in eine Klassifikationstafel erlaubt. Aus dieser Tafel läßt sich unmittelbar durch Produktbildungen der Optimierungsfaktoren der einzelnen Methoden der Grad der Optimierungen für eine Protokollschicht oder den vollständigen Protokollstack ermitteln.

Wichtige Optimierungsansätze der letzten Jahre wurden mit Hilfe dieser Methode eingeordnet. Alternative oder künftige Methoden können leicht in dieses Klassifikationsschema eingefügt

werden. Damit wird man in die Lage versetzt, Protokolländerungen oder Erweiterungen zu klassifizieren und die Auswirkung auf Performanz und Datenvolumen des vollständigen Protokollstacks zu bestimmen.

4 Messungen und Simulationen

In diesem Kapitel werden die Messungen und Simulationen beschrieben, mit deren Hilfe eine quantitative Aussage über einen Teil der im vorigen Kapitel beschriebenen Optimierungsstrategien getroffen werden soll. Zu ausgewählten Optimierungsmethoden wurden Messungen durchgeführt. Die Meßumgebung wurde so gewählt, daß die Resultate auch mit Ergebnissen aus der Literatur [36; 71; 77; 78; 108] verglichen werden können.

4.1 Vorüberlegungen

Die Arbeit befaßt sich mit verschiedenen Optimierungsstrategien zur Kommunikation im Internet. Wie bereits in *Kapitel 1: Einleitung und Motivation* festgestellt wurde, ist das WWW der lohnenste Kandidat Optimierungen durchzuführen, weil der Anteil der WWW-Kommunikation im Internet dominiert. Jedoch sollte man bei der Analyse noch einen Schritt weiter gehen und betrachten, wie dieser Verkehr aussieht. Wichtige Quellen dazu sind WWW- und WWW-Cacheserver. Über die Zugriffsstatistiken dieser Server kann man sich ein Bild machen, welche Dateien beim Webzugriff den größten Anteil haben. Ein Überblick über solche Daten wurde in einer Praktikumsarbeit am FB Mathematik/Informatik [50] erarbeitet.

Ein Kriterium für eine Einordnung solcher Zugriffsdaten ist die Dateigröße. Stellt man fest, daß die übertragenen Dateien zu überwiegendem Anteil klein sind, so kann man die Überlegungen zur Optimierung auf kleine Dateien begrenzen. Eine universelle durchschnittliche Dateigröße zu ermitteln ist jedoch schwierig. Zu unterschiedlich ist das Angebot von Server zu Server, zu weit gestreut das Verhalten des Benutzers und nicht zu vergessen zu dynamisch die Entwicklung des WWW als Basisplattform vielfältiger und immer neuer Multimediadienste. In diesem Sinne ist die Abschätzung 1995 von J. C. Mogul [71] auch nur als grobe Richtgröße für die heutige Situation zu betrachten. Er ermittelte in zwei Anwendungsszenarien Werte von 1770 Byte und 958 Byte als Median des WWW-Verkehrs. Im folgenden wird eine aktuelle Abschätzung getroffen, um einen Anhaltspunkt zu erhalten oder zu manifestieren, welche Dateigrößen in den kommenden Untersuchungen betrachet werden sollen.

Aus den Logdateien des WWW- und WWW-Cacheservers des FB Mathematik und Informatik wurden dazu die protokollierten Dateigrößen extrahiert. Die Daten des WWW-Cache stammen aus der Woche vom 20.3.1999, die Daten des WWW-Servers aus der Woche vom 4.4.1999. Die Anzahl der WWW-Zugriffe wurde aus technischen Gründen für die Auswertung auf 65000 begrenzt. Für diese beiden Zahlenreihen wurde der Median und der Durchschnitt für die Dateigrößen ermittelt. Der Median hat gegenüber dem Durchschnitt den Vorteil, daß einzelne große Dateien den Wert nicht so stark beeinflussen. Die Werte für Median und Durchschnitt sind in *Tabelle 17* dargestellt.

Server	Zugriffszahl	Gesamtvolumen	Median	Durchschnitt
WWW	65000	765 MB	2155 Byte	12351Byte
WWW-Cache	19583	255 MB	3586 Byte	13671Byte

Tabelle 17: Vergleich Median und Durchschnitt

Im Vergleich zum Durchschnitt ist der Median einer solchen Verteilung ist sicherlich eine treffendere Richtgröße für die folgenden Messungen, da bei der Ermittlung des Medians die Anzahl der Zugriffe eine stärkere Rolle spielt als die Dateigröße. Und gerade der Einzelzugriff an sich ist

das Ziel der Optimierung der Verbindungssteuerung. In der Histogramm-Grafik *Abbildung 35* sieht man sofort, daß die Majorität der Zugriffe sich bei den Zugriffen auf kleinere Dateien befindet. Für die Übersicht wurde die Anzahl der Zugriffe für beide Reihen auf den Wert 100 für die jeweilige maximale Zugriffszahl normiert. Bei diesem Histogramm wurden die Zugriffe in 100 Byte-Schritten zusammengefaßt.

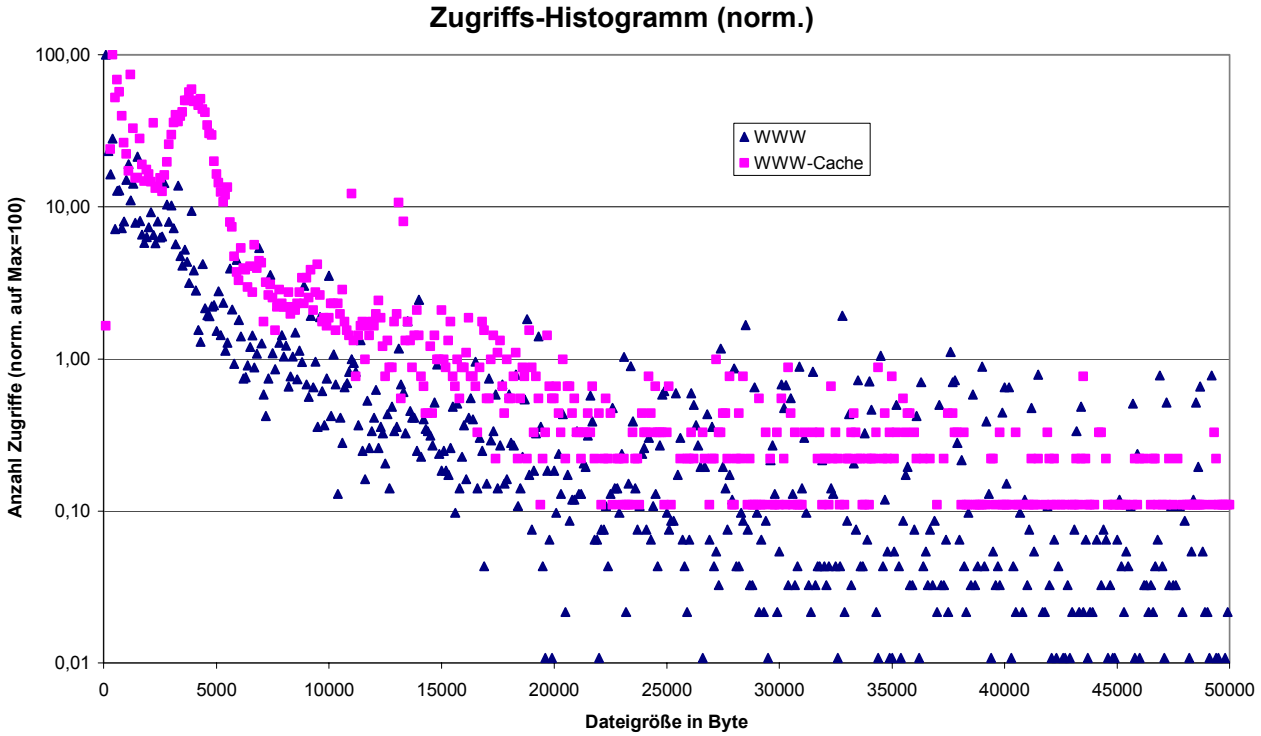

Abbildung 35: Histogramm WWW-Zugriffe (normiert)

Aus den vorgestellten Werten ergibt sich, daß 82,9% der WWW-Zugriffe und 87,2% der WWW-Cache-Zugriffe, gemittelt also ca. 85% der Web-Zugriffe einen Datentransfer von weniger als 10 KByte durchführen. Ungeachtet der Diskussion, welche Statistikgröße die Verteilung am besten beschreibt, läßt sich somit feststellen, daß Dateien unterhalb von 10 KByte den größten Anteil am WWW-Verkehr ausmachen und dementsprechend auch in Hinblick auf Optimierungsstrategien besondere Beachtung verdienen.

4.2 Vermittlungsschicht

Für die Vermittlungsschicht werden die Optimierungen IPv6 und IPComp diskutiert. Beim Design von IPv6 wurde darauf Wert gelegt, daß dieses Protokoll zukünftigen Entwicklungen gegenüber offen ist und im besonderen dem Wachstum des Internets standhalten kann. IPComp wurde explizit als Erweiterung von IPv4 und IPv6 zur Kompression der Daten konzipiert.

4.2.1 IPv4 und IPv6

In *Kapitel 2: Grundlagen* wurden bereits die Protokolle IPv4 und IPv6 beschrieben. An dieser Stelle werden die Unterschiede der Protokolle in Bezug auf mögliche Verbesserungen des IPv6 gegenüber dem IPv4 diskutiert.

In Bezug auf die Verbindungssteuerung lassen sich nur sehr schwer quantitative Aussagen treffen. Wie gut die Verbesserung in der Kommunikation ist, hängt im wesentlichen von der internen Verarbeitung des IPv6-Paketes, bzw. dessen Header ab. Die bisherigen qualitativen Überlegungen lassen zu wenige quantitative Schlüsse zu, um eine gesicherte Bewertung vornehmen zu können. Eine experimentelle Untersuchung ist mit einem hohen Aufwand an Hardware verbunden (mindestens zwei dedizierte Rechner plus Routersystem) und liefert nur Anhaltspunkte für das aufgebaute Laborsystem und hat zudem nur geringe Aussagekraft in Bezug auf „Real-Life"-Systeme von verschiedenen Routern. Dennoch soll an dieser Stelle eine grobe Abschätzung vorgenommen werden, um diese qualitativ eindeutige Verbesserung mit in die quantitative Bewertung einfließen zu lassen. Der IPv6 Header ist zwar mit 40 Bytes doppelt so groß wie der IPv4 Header, jedoch werden davon alleine 32 Bytes für die IP-Adressen benötigt. Die übrigen Optionen im IP-Header müssen vom Router System ausgelesen und ausgewertet werden. Dazu gehören Prioritäten und Flowlabel von IPv6 genauso wie Type of Service und Header checksum von IPv4. Der Ausgangspunkt der Abschätzung ist nun, daß die Auswertung und Validierung der IP-Optionen mehr Zeit erfordern als das Lesen der IP-Adresse. Daraus kann man nun ableiten, daß ein IP-Paket mit weniger Optionen auch schneller weitergeleitet werden kann. Genau dies trifft auf das IPv6-Paket zu, welches nur 8 Byte Header Optionen besitzt, im Gegensatz zum IPv4 Paket, welches 12 Byte Header Optionen beinhaltet (jeweils ohne IP-Adressen).

Bei der nun folgenden Abschätzung wird davon ausgegangen, daß die Verarbeitung der größeren IP-Adressen nur 50% mehr Zeit für eine viermal so große Adresse beansprucht, da im Gegensatz zu Prüfsummen, etc. keine aufwendige Verarbeitung dieser Information notwendig ist. Zudem soll der Zeitaufwand für die Bearbeitung eines IPv4 Paket Headers verteilt sein als: 20% für die IP-Adressen, 80% für den restlichen IP-Header. Beim IPv6 Paket hat man nun einen um 2/3 kleineren (Rest-)Header. Der Zeitaufwand für eine IPv6 Paket ergibt sich dann als: 20%*1,5 = 30% für die IP-Adressen und 80%*2/3 = 53,3% für den restlichen IP-Header, also insgesamt 83,3% im Vergleich zum IPv4 Header, oder im folgenden verwendet als aufgerundeter Faktor von 0,85.

Untersuchungen in Bezug auf die Datenrepräsentation können ebenfalls theoretisch, aber wesentlich präziser durchgeführt werden. Dazu werden nun IPv4- und IPv6-Pakete in Abhängigkeit von der Payload (Nutzlast der zu transportierenden Daten) verglichen. Damit kann zumindest eine Aussage getroffen werden, inwieweit die Umstellung von IPv4 auf IPv6 eine Veränderung im gesamten Transportvolumen des Internets bewirkt.

Im Vergleich der beiden Protokolle sollen keine weiteren IP-Optionen gesetzt sein. Daraus ergibt sich für das IPv4-Paket eine Headerlänge von:

- 4 Bytes Source-IP + 4 Bytes Dest-IP + 12 Bytes restlicher Header
  = 20 Bytes

für IP-Payloads, die kleiner als 65516 Bytes sind. Ist die Payload größer, so muß sie auf mehrere Pakete verteilt werden.

Das IPv6-Paket hat eine Länge von

- 16 Bytes Source-IP + 16 Bytes Dest-IP + 8 Bytes restlicher Header
  = 40 Bytes

für IP-Payloads, die kleiner als 65536 Bytes sind. Ansonsten ergibt sich mit der Jumbo-Payload Option:

- 16 Bytes Source-IP + 16 Bytes Dest-IP + 8 Bytes restlicher Header + 8 Bytes Jumbo-Option = 48 Bytes

Die entsprechenden Ergebnisse für die gesamte Paketlänge (in Bytes) sind in *Tabelle 18* aufgeführt.

		Header		
Protokoll	**Payload**	**IP + Rest**	**IP-Pakete**	**Gesamt-Länge (Byte)**
IPv4	beliebig	8 + 12	Payload DIV 65516 + 1	20*(Payload DIV 65516 + 1) + Payload
IPv6	0 – 65535	32 + 8	1	40 + Payload
IPv6	65536 – 4 GByte	32 + 16	1	48 + Payload

Tabelle 18: Vergleich Datenrepräsentation IPv4 und IPv6

Aus der Tabelle kann man bereits erkennen, daß erst ab einer Paketlänge von ca. 128 KByte das Protokoll IPv6 gegenüber dem IPv4 einen Vorteil bezüglich des Volumens hat. Bei wachsender Payload steigert sich die Volumendifferenz zwischen den Protokollen um lediglich 20 Bytes pro 64 KByte Payload.

4.2.2 IPComp

Wie schon in *Kapitel 3.3.1.2: IPComp* erläutert, komprimiert IPComp die Payload der IP-Pakete. Weil noch keine zugängliche Implementation des Protokolls vorliegt, wird eine Simulation durchgeführt. Ziel der Simulation ist nun, die Effektivität des Protokolls zu bestimmen. Dazu wird der Aufruf einer Webseite protokolliert und diese Daten nachträglich komprimiert. Der Vergleich mit den unkomprimierten Daten liefert den Grad der Kompression für diese Webseite, der als geeignete Abschätzung für die Effektivität dieser Optimierung angesehen werden kann.

Die verwendete Webseite heißt Microscape [111] und wurde vom W3C aus zwei häufig frequentierten Webseiten zusammengesetzt: Der Homepage von Microsoft und der von Netscape. Damit wurde gewährleistet, daß diese Testseite zumindest zu einem gewissen Maß repräsentativ für das Design und die technischen Inhalte einer kommerziellen Homepage ist. Die Seite besteht aus 42 GIF-Dateien (128287 Bytes) und einer HTML-Seite (42943 Bytes). Diese Seite wurde bereits mehrfach vom W3C für quantitative Untersuchungen an Server- und Clientsoftware, sowie an Protokollen verwendet und stellt damit einen gewissen Standard dar.

Die Durchführung im einzelnen: Zunächst wird mit Hilfe des Netzwerk-Sniffers tcpdump [55] der Aufruf der Microscape-Seite protokolliert. Beim Clientsystem kann ein beliebiger Browser verwendet werden. Hier wurde ein Netscape-Browser unter WindowsNT eingesetzt. Auch beim Serversystem kann ein beliebiger WWW-Server eingesetzt werden, hier war dies ein Apache

WWW-Server unter FreeBSD. Die Microscape Webseite wird nun vom Client aufgerufen und der WWW-Verkehr mit dem Netzwerksniffer protokolliert. Aus der Protokolldatei kann man nun die IP-Pakete extrahieren und in separate Dateien schreiben. Für die Simulation des IPComp-Protokolls muß nun an den IP-Header der IPComp-Header angehängt werden und die Payload des IP-Paketes, was einem vollständigen TCP-Paket entspricht, komprimiert werden.

Für die Realisierung dieser Anforderungen wurde ein Java-Programm verwendet, welches aus der tcpdump-Protokolldatei die IP-Pakete, TCP-Pakete und komprimierten TCP-Pakete extrahieren kann. Die Kompression wurde über die Klasse java.util.zip.GZIPOutputStream implementiert. Die Länge des resultierenden IPComp-Paketes ergibt sich direkt aus der Länge des komprimierten TCP-Paketes + 20 Bytes IP-Header + 4 Bytes IPComp-Header.

Die Summen der Längen aller 279 IP-Pakete sind in *Tabelle 19* eingetragen. Die Längenangaben sind alle in Byte angegeben.

Protokoll	IP	TCP	TCP comp	IP comp
Summe der Längen	209136	203556	176527	179479

Tabelle 19: IPComp Simulationsergebnisse

Der bereits aus der Tabelle ersichtliche relativ geringe Kompressionsgrad liegt an dem bereits oben erwähnten großen Anteil von komprimierten GIF-Dateien.

4.3 Transportschicht

In diesem Kapitel wird der Einfluß vom T/TCP auf Datenrepräsentation und Verbindungssteuerung untersucht. Die Untersuchungen können in Form von Messungen an einer Implementation des T/TCPs durchgeführt werden. Zusätzlich zu den Messungen am T/TCP werden parallel dazu die gleichen Messungen an den Protokollen TCP und UDP durchgeführt.

4.3.1 UDP, TCP und T/TCP

Die Optimierungsstrategien bezüglich der Verbindungssteuerung wurden in *Kapitel 3: Methoden der Optimierung* bereits erläutert. Zu dieser Optimierung in Form des Protokolls T/TCP können quantitative Messungen durchgeführt werden, da im Gegensatz zu anderen Strategien bereits eine Implementierung vorliegt.

Ziel ist es nun, eine quantitative Analyse der Verbindungssteuerung durchzuführen. Dazu werden Messungen an den Transportschichtprotokollen UDP, TCP und T/TCP vorgenommen. Die Meßgrößen werden nach den Optimierungsansätzen beim T/TCP festgelegt. Die bei den Messungen an den drei Transportschichtprotokollen erfassten Meßwerte können dann miteinander verglichen werden, um eine Aussage über die Wirkung der Optimierungsmethode zu treffen. Bedingt durch den Versuchsaufbau wurden hier die Pakete der Sicherungsschicht protokolliert, obwohl diese Protokollschicht in der restlichen Arbeit nicht mit in die Diskussion einbezogen wird. Dennoch sind diese Ergebnisse wertvoll und können mit in die Diskussion einbezogen werden, weil z.B. die Paketanzahl der Sicherungsschicht für kleine Anfragen und Antworten identisch mit der Paketanzahl der Transportschicht ist. Klein bedeutet in diesem Fall kleiner als die maximale Payload eines Ethernetpaketes, also <= 1500 Bytes.

4.3.1.1 Versuchsaufbau

Durch die Anforderung der Verwendung von T/TCP fiel die Wahl der Plattform recht leicht: SunOS wird nur in einer am Institut nicht mehr eingesetzten Version unterstützt, bei Linux liegt auch nur ein Kernelpatch für eine bestimmte Kernelversion vor, lediglich bei FreeBSD ist das T/TCP seit Jahren fester Bestandteil des Kernels. Zudem liegt FreeBSD (wie auch Linux) vollständig im Quelltext vor, sodaß es auch möglich ist, spezielle Funktionen oder Probleme nötigenfalls im Quelltext nachzuvollziehen. Somit fiel die Wahl auf FreeBSD, hier verwendet in der Version 2.2.8 mit den Kernelsourcen vom Stand Januar 1999. Das ist die letzte Version des 2.x-STABLE Zweiges in der FreeBSD Entwicklung.

Als Meßumgebung dienten drei Rechner: Ein Server, ein Client und ein Meßsystem. Die drei Rechner waren alle mit Fast Ethernet über eine Halbduplex-Verbindung an einen Hub angeschlossen. In dieser Collisiondomain waren auch weitere Rechner angeschlossen, die jedoch zum Zeitpunkt der Messung inaktiv waren. Damit ist gemeint, daß lediglich Kontrollpakete der verschiedenen Netzdienste sowie einige Broadcasts zu sehen waren, jedoch keine Netzanwendung mit größerer Belastung (NFS, FTP, WWW) lief. Daher ist davon auszugehen, daß die Meßergebnisse nicht signifikant von dem Optimalfall einer ungestörten Verbindung, bzw. einer Simulation einer solchen Verbindung abweichen. Collisiondomain bedeutet, daß alle Rechner, die an diesem Netzwerkstrang angeschlossen sind, im sogenannten „promiscous mode“ alle Pakete der anderen Rechner empfangen.

Als Client und Server wurden Pentium-Systeme verwendet, die Messungen wurden auf einem 486-PC durchgeführt. Die Wahl der Hardware beeinflußt bei dieser Art der Messungen nicht das Resultat, da keine CPU- oder IO-lastigen Versuche durchgeführt wurden. Client- und Server-Programme bedienen sich ausschließlich der Routinen des TCP/IP-Stacks des Betriebssystems, die nur geringe Anforderungen an die CPU-Leistung des Systems stellen. Auf den beteiligten Systemen liefen keine weiteren Anwendungsprogramme und die Last (load average über eine Minute) war vor und während der Messungen stets kleiner als 0,02.

Für die Simulation einer WWW-Verbindung wurde ein generisches Client-Server Beispielprogramm aus [99] in der Form modifiziert, daß die Verbindung einer WWW-Transaktion gleicht, wenn auch das Serverprogramm keine Standard-Browser bedienen kann. Der Grund dafür ist, daß die Programme bewußt einfach gehalten sind, um störende Einflüsse von nicht messrelevanten Programmteilen und die Verarbeitungszeiten minimal zu halten. Die Programmiersprache ist auf C/C++ festgelegt, weil nur dort die entsprechenden Routinen für T/TCP zur Verfügung stehen. Zur Messung der Pakete wurde das für alle UNIX-Plattformen verfügbare tcpdump [55] verwendet, dessen Ausgabe wurde mit tcpshow [43] weiterverarbeitet. Das Programm tcpshow wurde dazu unter anderem um die Ausgabe diverser Paketoptionen (für T/TCP, etc.) erweitert. Der Ablauf des Meßvorgangs wurde von diversen Shell-Scripten unter UNIX gesteuert. Die Meßergebnisse wurden in Textdateien geschrieben und mit Microsoft Excel unter Windows gegebenenfalls weiterverarbeitet, bzw. aufgearbeitet.

4.3.1.2 Durchführung

Bei der Messung wird eine Client-Server Transaktion durchgeführt und die dabei ausgetauschten Pakete zur späteren Auswertung aufgezeichnet.

Der Ablauf ist wie folgt: Beim Meßsystem wird die Protokollierung (tcpdump) gestartet. Dieses Programm protokolliert auf der Sicherungsschicht alle Pakete, die in der Collisiondomain ausgetauscht werden. Dann verbindet sich das Client-System zum Server und setzt die Anfrage „GET /file_0N HTTP/1.0“ (file_0N ist eine Datei der Größe N Bytes) ab. Der Server beantwortet die Anfrage mit der angeforderten Datei. Dann wird die Protokollierung über das Meßsystem gestoppt. Der binäre Dump der Protokolldatei wird in verschiedene lesbare Protokolldateien umgewandelt. Ein Analyseprogramm faßt eine Messung der Meßreihe in einer Zeile zusammen, mit den Informationen:

```
source=>destination prot suffix ident filesize pkts pktsize payload time
```

Dabei sind „`source=>destination`“ die Namen des Clients (source) und des Servers (destination), „`prot`“ das verwendete Protokoll (TCP, T/TCP oder UDP), „`suffix`“ der Namenssuffix der angforderten Datei, „`ident`“ ein Zähler für wiederholte Messungen der gleichen Art, „`filesize`“ die Dateilänge der Antwort, „`pkts`“ die Anzahl der Ethernetpakete, „`pktsize`“ die Gesamtlänge aller Ethernetpakete, „`payload`“ die Nutzlast der Transportschicht und „`time`“ die insgesamt benötigte Zeit.

Bei den Messungen wurde die Größe der Transferdatei („`filesize`“) bis 2 KByte in Schritten von 256 Byte und bis 8 KByte in Schritten von 512 Byte variiert. Die Messungen wurden für die Protokolle („`prot`“) TCP, T/TCP und UDP durchgeführt. Die Grafiken der Meßergebnisse sind in den folgenden Abschnitten vorgestellt.

Bemerkungen zu den Messungen

Die Messung mit dem Protokoll UDP mußte auf Messungen bis 7168 Byte beschränkt werden. Wurden bei der Antwort größere Datagramme versendet, so sind diese zwar beim Clientsystem korrekt eingetroffen, wurden jedoch vom Meßsystem nicht vollständig mitprotokolliert. Dies ist somit keine grundsätzliche Beschränkung bei einer Client-Server Kommunikation mit UDP, sondern ein Fehler in der Kombination der verwendete Programm.

Schwankungen in der Paketzahl und Ausführungszeit für gleiche Messungen sind bei der Grafiken zu erkennen. Sie sind auf Meßungenauigkeit der Systeme, Netzwerkverhalten (Kartentreiber, Kollisionen) und Systemarchitektur (Multiuser- Multitaskingsysteme) zurückzuführen.

4.3.1.3 Paketanzahl

Die Paketanzahl einer Transaktion ergibt sich aus der Anzahl der Pakete für Verbindungsaufbau, Anfrage, Antwort und Abbau der Verbindung. Zum Einfluß dieser Meßgröße auf eine Verbindung wurden beim vorgestellten Versuchsaufbau die Protokolle über UDP, TCP und T/TCP variiert und die Länge der Serverantworten sukzessiv erhöht. Gemessen wurde die Paketanzahl der Sicherungsschicht und die dazu benötigte Zeit, die Grafik ist in *Abbildung 36* zu sehen. Wie bereits in den vorangegangenen Kapiteln besprochen wurde, ergeben sich erheblich Unterschiede in der Paketanzahl für die unterschiedlichen Transportschicht Protokolle.

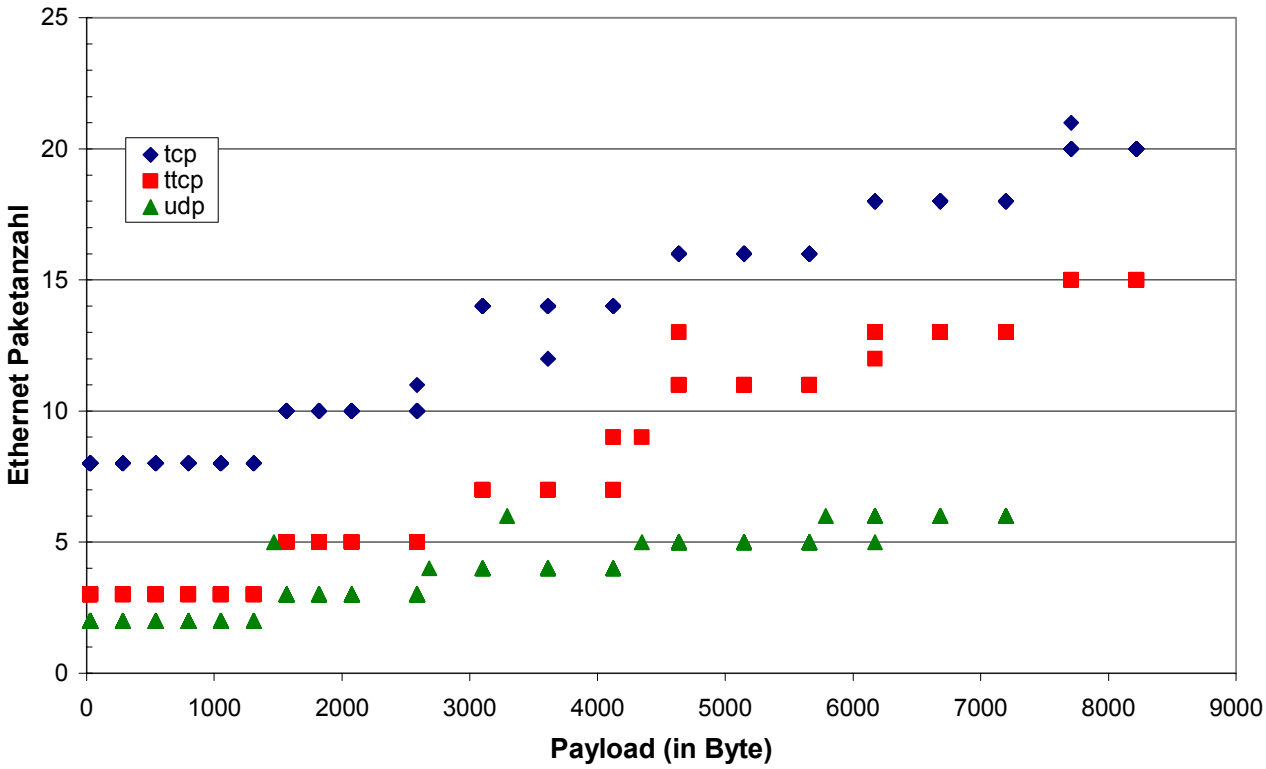

Abbildung 36: Ethernet Paketanzahl

Die minimale Paketanzahl ergibt sich für UDP, das T/TCP benötigt unabhängig von der Payload fünf Pakete weniger als das TCP.

4.3.1.4 Protokoll-Overhead

Eine weitere wichtige Größe für die Effektivität einer Verbindung ist der Protokoll-Overhead bei der Kommunikation. Mit dem Protokoll-Overhead sind alle Bytes gemeint, die nicht direkt zu der Anfrage und Antwort der Anwendungsschicht gehören. Zum Overhead wird somit nicht nur der Ethernet-Header, sondern auch der IP- und TCP-(bzw. T/TCP- /UDP-) Header dazugezählt. Je geringer der Anteil des Overheads ist, desto effektiver wird das Transportmedium genutzt. In *Abbildung 37* ist der Payloadanteil in Abhängigkeit von der Payload aufgetragen. Ein Payloadanteil von 1 entspricht einem Protokoll-Overhead von 0%, was definitionsgemäß nicht möglich ist. Es würde bedeuten, daß die Kommunikation ohne jegliches Protokoll stattfindet.

Aus der Abbildung ist zu erkennen, daß das T/TCP gegenüber dem TCP besonders bei kleinen Paketen (kleine payload) in der Summe der Pakete einer Transaktion bezüglich des Datenvolumens um mindestens 10% effektiver ist. Das UDP hat nochmals einen geringeren Overhead, ist aber keine echte Alternative zu TCP und T/TCP, da es lediglich einen verbindungslosen Dienst bereitstellt.

Mit der Größe der Payload steigt für alle Protokolle die Effektivität. Speziell das T/TCP nähert sich mit steigender Payload dem TCP. Auch daran läßt sich erkennen, daß sich die Optimierung des T/TCP hauptsächlich auf den Verbindungsaufbau und Abbau bezieht.

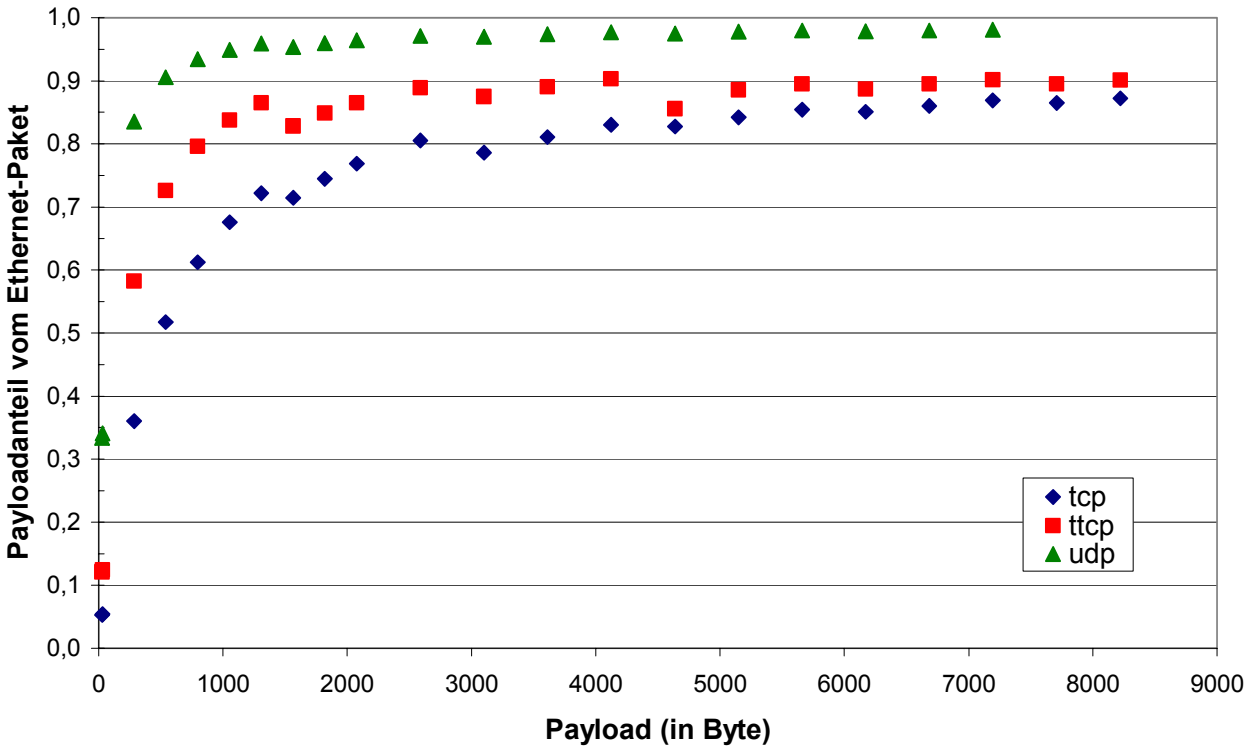

Abbildung 37: Payloadanteil der Ethernetpakete

4.3.1.5 Transaktionszeit

Die Transaktionszeit ist die Zeit für eine vollständige Transaktion, d.h die Zeit für alle dafür notwendigen Pakete. Dementsprechend ist das beim Server verwendete TIME_WAIT Intervall dabei nicht eingeschlossen. Zeitmessungen sind zwar aufgrund der Hardwareabhängigkeit grundsätzlich mit Vorsicht zu genießen, jedoch bleibt stets die Aussage des Verhältnisses zwischen den Protokollen bestehen.

Eine Besonderheit, deren Ursache im Verlauf der Arbeit nicht festgestellt werden konnte, sind die Transaktionszeiten für eine Null-Byte Antwort. Sie liegen entgegen dem Trend bei einem Vielfachen der Transaktionszeiten einer Ein-Byte Antwort.

Eine weitere Auffälligkeit ist der Sprung der T/TCP Transaktionszeit zwischen der Payload 4096 Byte und 4608 Byte (x-Achse). Die Ursache dafür liegt in der Ethernet-Paketanzahl: Es findet dort ebenfalls ein Sprung in der Paketanzahl statt, wie man auch aus *Abbildung 36* erkennen kann. Es werden für 4608 Byte 11 Pakete statt 7 Pakete für 4096 Byte versendet, was durch zusätzliche RTTs den Anstieg der Transaktionszeit erklärt.

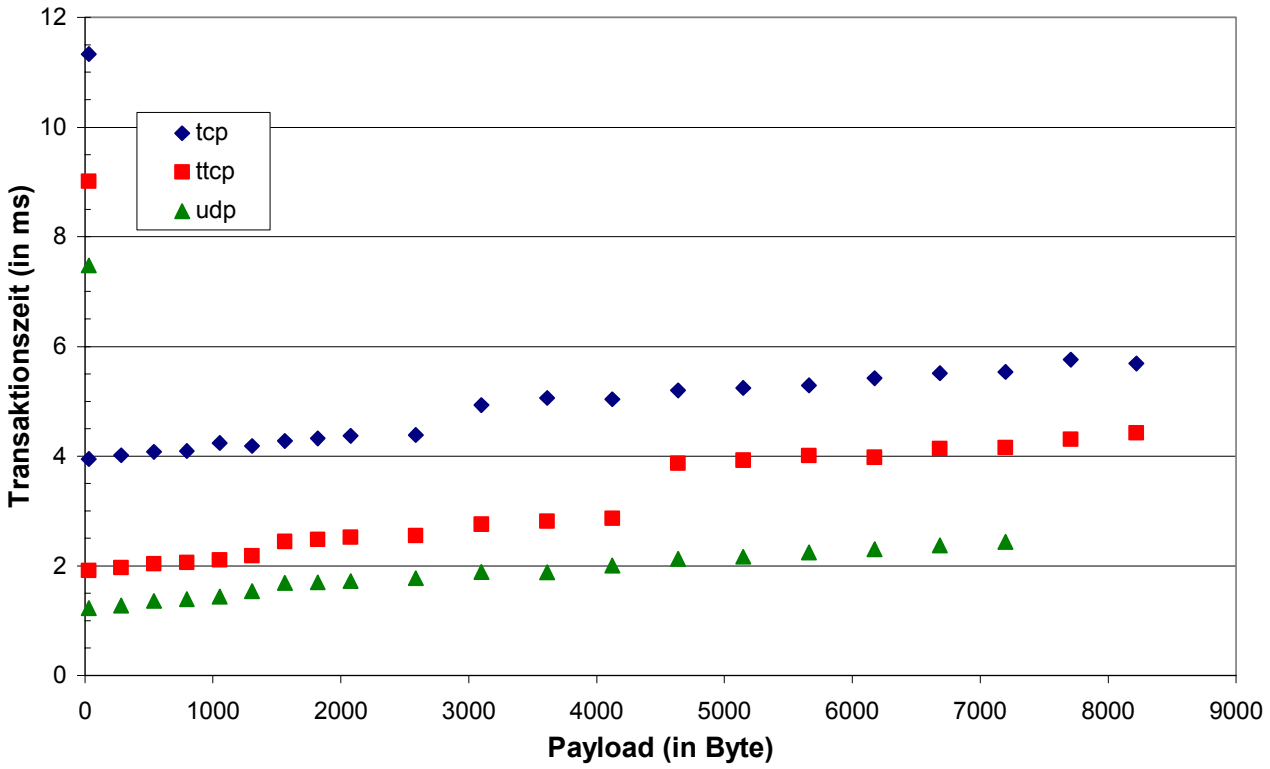

Abbildung 38: Transaktionszeit

Für kleine Transaktionen (geringe Payload) verhält sich das T/TCP ähnlich performant wie das UDP, nähert sich dann aber dem TCP für größere Payloads an.

4.4 Anwendungsschicht

In der Anwendungsschicht werden im folgenden die Ergebnisse von vier Optimierungsmethoden vorgestellt. Das HTTP Pipelining hängt im Praxiseinsatz von den persistenten Verbindungen ab. Theoretisch kann man zwar auch Pipelining über mehrere unabhängige Verbindungen realisieren, was aber den Optimierungsgewinn deutlich verringert und auch der grundsätzlichen Idee des Pipelinings widerspricht. Die dritte Methode ist das HTTP Caching und die vierte das HTTP Content-Encoding.

4.4.1 HTTP Pipelining und Persistente Verbindungen

In einer vom W3C unterstützten Studie [77], wurde u.a. der Effekt von persistenten Verbindungen und Pipelining bei HTTP/1.1 mit der ursprünglichen Implementation des HTTP/1.0 verglichen.

Für diese Untersuchung wurde die Microscape-Webseite verwendet. Bezüglich der Verbesserungen des HTTP/1.1 muß die Diskussion in *Kapitel 3.2.3: Anwendungsschicht* ein wenig relativiert werden, denn einige Nachteile kann das HTTP/1.0 durch die Verwendung von mehreren gleichzeitigen Verbindungen abschwächen. Dementsprechend überrascht zeigte man sich auch im oben genannten Artikel beim Vergleich von HTTP/1.0 mit HTTP/1.1 (Pipelining, Persistent). Das erste Resultat der Messungen (*Tabelle 20*) zeigte zwar eine deutliche Reduktion der Paketanzahl bei der Verwendung von persistenten Verbindungen und zusätzlichem Pipelining, die benötigte

Zeit für alle Transaktionen war jedoch beim HTTP/1.0 mit sechs gleichzeitigen Verbindungen geringer.

	HTTP/1.0	HTTP/1.1 Persistent	HTTP/1.1 Pipeline
Max. gleichzeitige Verbindungen	6	1	1
Anzahl verwendeter sockets	40	1	1
Pakete vom Client zum Server	226	70	25
Pakete vom Server zum Client	271	153	58
Pakete insgesamt	497	223	83
Zeit insgesamt (in sec)	1,85	4,13	3,02

Tabelle 20: Vergleich HTTP-Versionen [77]

Es stellte sich heraus, daß sich das Zusammenspiel aus TCP Nagle Algorithmus und dem Puffer für das Pipelining negativ auf die Performanz auswirkt. Der TCP Nagle Algorithmus sorgt dafür, daß die Daten, die von der Anwendungsschicht an die Transportschicht übergeben werden, nicht sofort, sondern erst nach 200 ms gesendet werden, wenn das zu sendende Paket kleiner als $1/2 * MTU$ (MTU= Maximum Transmission Unit, maximale Payload der Sicherungsschicht) ist. Die Details des Algorithmuses variieren je nach Implementation, bewirkt jedoch stets sinngemäß, daß keine kleinen Pakete in rascher Folge versendet werden, sondern die Daten in einem Zeitfenster von ca. 200 ms gesammelt werden. Eben dieses Puffer-Verhalten kollidierte mit dem Pipelining Puffer der HTTP/1.1 Implementation. Durch das Setzen der Socket-Option TCP_NODELAY, welche den Nagle Algorithmus abschaltet, und weiteren Modifikationen an Client und Server zeigte sich dann eine bessere zeitliche Performanz für das HTTP/1.1 Pipelining (*Tabelle 21*).

	HTTP/1.0	HTTP/1.1 Persistent	HTTP/1.1 Pipeline
Pakete insgesamt	510,2	281	181,8
Transfervolumen (in Bytes)	216289	191843	191551
Zeit insgesamt (in sec)	0,97	1,25	0,68

Tabelle 21: Vergleich HTTP-Versionen, optimiert [77]

Insgesamt wurden nur geringe Modifikationen an Client und Server vorgenommen, so daß man gesichert feststellen kann, daß der Schritt von HTTP/1.0 zu HTTP/1.1 mit Pipelining in jeder Beziehung ein Erfolg war.

4.4.2 HTTP-Caching

WWW-Cacheserver protokollieren ihre Aktivität in Logdateien. Diese können später über Programme ausgewertet werden. An dieser Stelle werden die Statistiken zweier Cacheserver präsentiert: Einmal der des Fachbereichs Mathematik und Informatik der Universität Marburg [23] und einmal der von der University of Melbourne (Australien). Die Daten des Cacheservers der Universität von Melbourne [36] wurden hinzugezogen, weil sie das Cacheverhalten für eine andere Größenordnung repräsentieren. Die Anwendungsszenarien sind nicht unmittelbar vergleichbar, da der Cacheserver in Melbourne eine ganze Universität bedient und der in Marburg nur einen Fachbereich. Dennoch sind beides durchaus realistische Anwendungsszenarien.

Im August 1999 wurden im Fachbereich Mathematik und Informatik in Marburg ca. 10 Tausend Requests pro Woche bearbeitet, in Melbourne ca. 10 Millionen Requests pro Woche. Die zeitliche

Entwicklung der Nutzung der Cacheserver ist in *Abbildung 39* und *Abbildung 40* zu sehen. Die Grafiken wurden von dem an der Universität von Melbourne entwickelten Auswertungsprogramm „pwebstats“ generiert. Das Programm liest die Logdateien von WWW- oder Cache- oder FTP-Servern ein und erzeugt wöchentliche Statistiken, die auch in der zeitlichen Entwicklung dargestellt werden können.

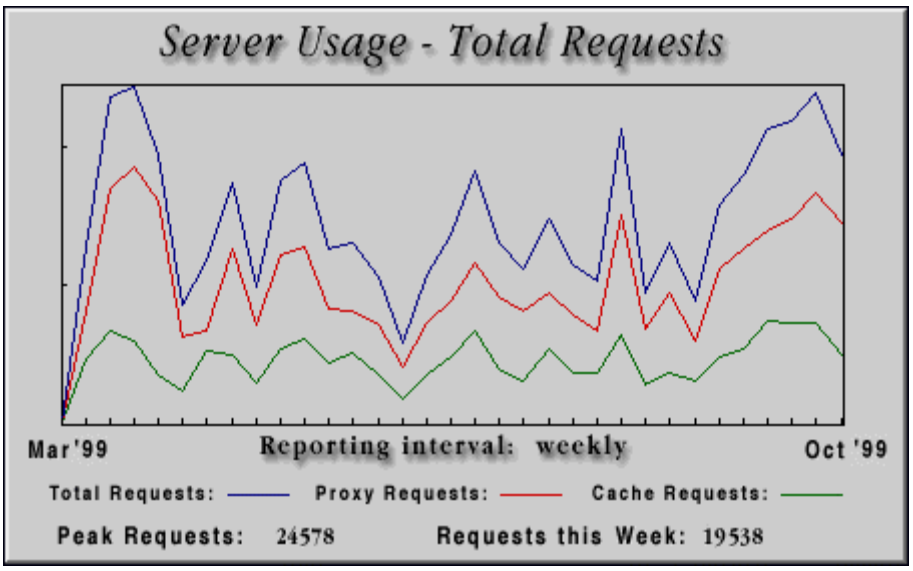

Abbildung 39: Cache Requests FB 12 der Uni Marburg

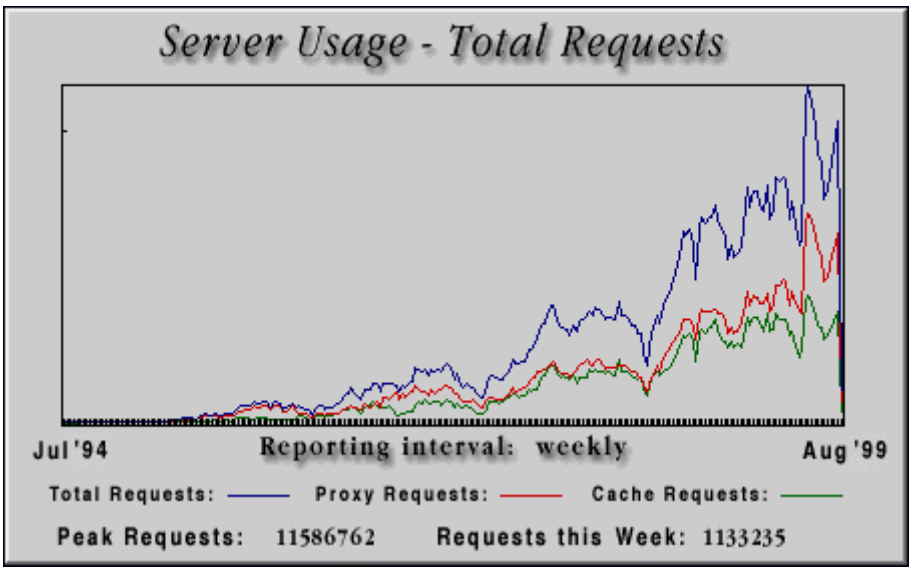

Abbildung 40: Cache Requests Uni Melbourne

Zusätzlich wurde die Hitrate-Bytes in Abhängigkeit des transferierten Volumens in *Abbildung 41* dargestellt. Von der Universität Melbourne lagen die Daten über einen Zeitraum von 248 Wochen ab Juli 1994 vor, vom FB 12 der Universität Marburg über 32 Wochen ab März 1999. Die Abbildungen sollen nicht dem unmittelbaren Vergleich dienen, sondern nur einen Eindruck über die zeitliche Entwicklung und Schwankung der Servernutzung vermitteln.

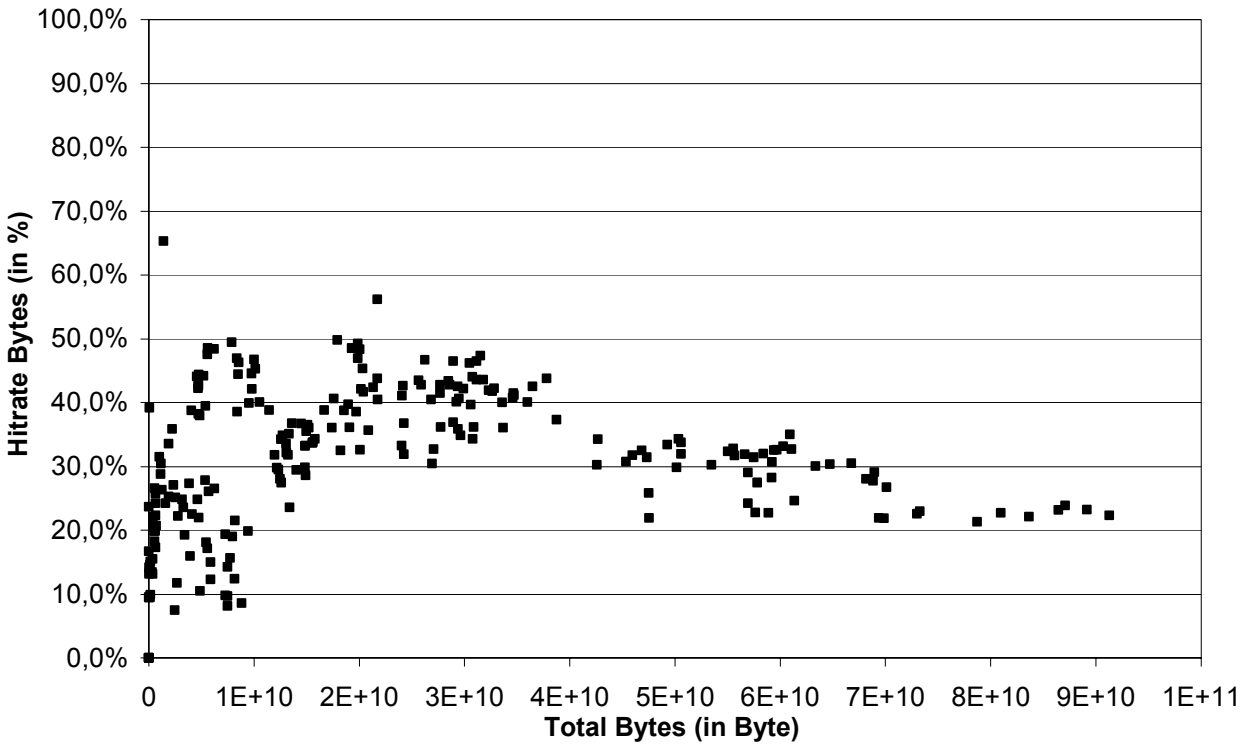

Abbildung 41: Melbourne Hitrate Bytes (Bytes)

In *Tabelle 22* sind alle Durchschnittswerte (wöchentliche Werte, gemittelt über die zur Verfügung stehenden Daten: Marburg: 32 Wochen, Melbourne: 248 Wochen) eingetragen.

Standort	Requests Total	Bytes Total	Requests Hitrate	Bytes Hitrate
Marburg, FB12	**15131**	**251644219**	**31,1%**	**16,6%**
Melbourne	**2755121**	**27943953731**	**36,2%**	**31,0%**

Tabelle 22: Durchschnittswerte Cacheserver Marburg und Melbourne

Erstaunlicherweise liegen die Hitraten für Requests mit 31% und 36% sehr nahe beieinander, obwohl die Nutzung der Server im Durchschnitt zwei Größenordnungen auseinander liegen. Bei der Hitrate der Bytes (Volumen der Requests) wird der Unterschied zwischen den Cacheservern deutlicher. Jedoch liegt auch bei geringer Nutzung des Cacheservers (FB12, Marburg) das eingesparte Volumen der transferierten Daten bei 17%. Sicherlich lassen sich bei konsequenter Nutzung auch bei kleinen Abteilungen oder Instituten mit wenigen hundert Nutzern Einsparungen von weit mehr als 20% erreichen, wie dies auch in Melbourne (*Abbildung 41*) der Fall ist. In einem bestimmten Nutzungsbereich liegt die Ersparnis bei beachtlichen 40% und erreicht im Spitzenfall sogar 50%. Welche Parameter zu einer so hohen Einsparung führen, kann an dieser Stelle mangels Konfigurationsdaten nicht diskutiert werden. Jedoch läßt schon die Abbildung erahnen, daß die Anzahl der Requests und damit verbunden das Volumen (TotalBytes) ein bestimmender Faktor ist. Weitere Faktoren sind zweifellos die Cachegröße, Anzahl der Clients, die Anbindung des Servers an Internet und Clients und nicht zu vergessen die Hardwarekonfiguration des Cacheservers. Zudem wird man bei einer weitergehenden Analyse, die auch das Clientverhalten einschließt, auch auf andere Aspekte eingehen müssen, wie z.B. das Benutzerverhalten.

4.4.3 HTTP Content-Encoding

Für die Kodierung der HTTP-Payload, auch HTTP-Content genannt, können nach der HTTP/1.1 Spezifikation die Kompressionsverfahren Gzip, Compress und Deflate verwendet werden.

Für die Messung der Kompression von WWW-Inhalten wurde die Microscape-Seite [111] vom W3C verwendet, die auch in der Untersuchung des W3Cs über persistente Verbindungen und Pipelining beim HTTP (*Kapitel 4.4.1: HTTP Pipelining und Persistente Verbindungen*) benutzt wurde. Die 43 Dateien der Seite wurden einzeln mit dem gzip-Programm komprimiert und die Summe der Dateilängen vorher und nach der Kompression ermittelt (*Tabelle 23*).

Dateilängen, unkomprimiert	**Dateilängen, komprimiert**
171230 Byte	136241 Byte

Tabelle 23: HTTP Content-Encoding

Im folgenden Kapitel werden diese Ergebnisse diskutiert und bewertet.

5 Ergebnisse und Diskussion

In diesem Kapitel werden die Ergebnisse der Messungen, Simulationen aus dem letzten Kapitel und von Publikationen dargestellt und diskutiert. Nach der Einordnung der Optimierungen in den letzten Kapiteln findet nun eine quantitative Bewertung dieser Strategien statt.

Dazu wird für jede Optimierungsstrategie ein Faktor berechnet, der die Veränderung von Datenrepräsentation und Verbindungssteuerung als eine Zahl bzw. Formel darstellt. Damit wird es möglich, Abhängigkeiten der Optimierungen zu erkennen und quantitativ zu benennen. Dieser Faktor *OFaktor* ist so definiert (*Formel 1*), daß er das Verhältnis von den optimierten Daten *AusgabeDaten* zu den Eingabedaten *EingabeDaten* wiedergibt. Entsprechend der vorgenommenen Klassifikation (*Kapitel 3.1: Klassifikation von Protokolloptimierungen*) gibt es für jede Optimierungsmethode zwei Faktoren. Einen für den Kommunikationsaspekt der Verbindungssteuerung (Zeit) und einen für die Repräsentations der Daten (Volumen).

$$OFaktor_{Kommunikationsaspekt} = \frac{AusgabeDaten_{Kommunikationsaspekt}}{EingabeDaten_{Kommunikationsaspekt}}$$

$$Kommunikationsaspekt \in \{Verbindungssteuerung, Datenrepräsentation\}$$

Formel 1: Definition Optimierungsfaktor OFaktor $_{Kommunikationsaspekt}$

Der Faktor ist auf eins normiert für den Norm-Protokollstack mit IPv4, TCP und HTTP/1.0. Die beteiligten Protokolle sollen folgende Eigenschaften haben: Das IPv4 soll keine zusätzlichen Optionen im Header haben, woraus sich eine IP-Headerlänge von 20 Bytes ergibt. Das gleiche gilt auch für den TCP-Header, der damit ebenfalls eine Länge von 20 Bytes hat. Dieser Norm-Protokollstack ist zwar nicht in jedem Detail zeitgemäß (HTTP/1.1 hat sich mittlerweile etabliert und auch im TCP ist z.B. die TIMESTAMP-Option weit verbreitet), definiert aber einen Protokollstack, an dem man die Entwicklung der letzten Jahre gut dokumentieren kann und der in dieser Ausprägung als robustes Modell über Jahre hinaus die Basis des Internets gebildet hat.

Mit diesem Modell ist es möglich einen neuen Protokollstack zu bewerten, wenn man die entsprechenden Faktoren der neuen Protokolle oder Protokolländerungen bestimmt hat. Über die Produktbildung der einzelnen Faktoren erhält man eine quantitative Aussage über Volumen (Datenrepräsentation) und zeitlichen Verhalten (Verbindungssteuerung) der Protokollsuite.

Wie schon in *Kapitel 4: Messungen und Simulationen* erwähnt, ist die Qualität der quantitativen Aussagen unterschiedlich. Die diesbezüglichen Hinweise im genannten Kapitel werden im folgenden durch eine Bewertung der Aussagekraft der Ergebnisse (kurz: AK) in den Formeln und Tabellen vermerkt. Dazu wird jeweils ein Symbol einer dreistufige Bewertung der Aussagekraft zu den quantitativen Ergebnissen in der Form „AK: Symbol“ angefügt. Die Zuordnung der Symbole zu der Bewertung der Aussagekraft ist in *Tabelle 24* eingetragen.

Aussagekraft	Bewertung der Ergebnisse	Symbol
Gut	Exakt oder gute Abschätzung, geringe Abhängigkeit von Leistungsparameter	+
Befriedigend	Befriedigende Genauigkeit oder Abschätzung, Abhängigkeit von Leistungsparameter	O
Eingeschränkt	Eingeschränkte Aussagekraft, subjektive Abschätzung oder starke Abhängigkeit von Leistungsparameter	-

Tabelle 24: Bewertung der Aussagekraft

5.1 Vermittlungsschicht

In *Kapitel 4.2.1: IPv4 und IPv6* wurden die Protokolle IPv4 und IPv6 miteinander verglichen. Nur die Ergebnisse der Betrachtung der Datenrepräsentation können als gesichert bezeichnet werden, da eine quantitative Analyse der Verbindungssteuerung mit großen Unsicherheiten belegt ist. Die folgende Auswertung der Messungen beschränkt sich auf Pakete, die kleiner als 64 KByte sind, da diese die Majorität im Internetverkehr repräsentieren.

Bezüglich der Datenrepräsentation fügt das IPv4 der Payload der Transportschicht (für den hier diskutierten Protokoll-Stack sind das TCP-Pakete) einen 20 Bytes langen Header zu. Wie schon in *Kapitel 4.2.1: IPv4 und IPv6* diskutiert, beträgt damit die Länge der IPv4-Pakete $IPv4head + Payload = 20 + Payload$. Für IPv6-Pakete ergibt sich analog eine Länge von $IPv6head + Payload = 40 + Payload$. Daraus berechnet sich ein $OFaktor$ von:

$$OFaktor_{Datenrepräsentation}(IPv6) = 1 + \frac{20}{20 + Payload}; AK : +$$

Formel 2: OFaktor $_{Datenrepräsentation}$ (IPv6)

Für kleine Pakete verdoppelt sich das Volumen, jedoch beträgt die Volumenvergrößerung für Pakete im KB-Bereich nur wenige Prozent.

Bezüglich der Verbindungssteuerung wurde in *Kapitel 4.2.1: IPv4 und IPv6* über die Auswirkung der Verwendung von IPv6 diskutiert und folgender Faktor über eine Abschätzung bestimmt:

$$OFaktor_{Verbindungssteuerung}(IPv6) = 0{,}85; AK : -$$

Formel 3: OFaktor $_{Verbindungssteuerung}$ (IPv6)

Für das IPComp-Protokoll wurde mangels einer Implementation eine Simulation durchgeführt. Das Ergebnis ist in *Tabelle 19* eingetragen und ergibt für den Optimierungsfaktor:

$$OFaktor_{Datenrepräsentation}(IPComp) = \frac{179479}{209136} = 0{,}86; AK : O$$

Formel 4: OFaktor $_{Datenrepräsentation}$ (IPComp)

Obwohl die TCP-Paketgröße bei der Microscape Webseite relativ klein ist, wurde dennoch eine Kompression auf 86% der ursprünglichen Paketgröße erreicht. Zudem besteht ein großer Teil der

übertragenen Daten aus GIF-Dateien, die ja selbst schon komprimiert vorliegen. Daher kann man erwarten, daß dieses Protokoll in anderen Szenarien wesentlich bessere Ergebnisse erzielt. Zudem läßt sich die Effektivität von IPComp durch die Verwendung von effizienteren Kompressionsalgorithmen, wie z.B. dem BWT-Algorithmus [94] noch weiter steigern. Der Einsatz von alternativen Kompressionsalgorithmen ist ja von IPComp explizit vorgesehen.

5.2 Transportschicht

Eine der beiden Optimierungen, die in dieser Arbeit diskutiert werden, ist das T/TCP. Das Resultat der Verwendung von T/TCP ist zum Teil bereits im Methodenteil (*Kapitel 3.2.2: Transportschicht*) erläutert worden. Ausgehend von dieser theoretischen Betrachtung zusammen mit den Messungen aus *Kapitel 4.3.1: UDP, TCP und T/TCP* werden nun die Optimierungsfaktoren berechnet.

Der Ausgangspunkt für die folgenden Überlegungen soll eine typische Transaktion sein, die aus einem Request-Response Paar besteht. Wie schon vorher diskutiert, kann man davon ausgehen, daß die Majorität der Anfragen und Antworten in der Länge kleiner als 64 KByte ist.

In *Kapitel 3.2: Optimierung der Verbindungssteuerung* wurden bereits theoretisch die Protokolle TCP, T/TCP und UDP für eine HTTP-Transaktion verglichen und in *Abbildung 26* gegenübergestellt. Die resultierende Anzahl der Pakete in der Transportschicht ist in *Tabelle 25* aufgetragen. Die Anzahl der Pakete in der Sicherungsschicht ist vom verwendeten Sicherungsschichtprotokoll abhängig und kann daher nicht unmittelbar angegeben werden. Diese stark hardwareabhängigen Einflüsse sollen, wie schon in der Einleitung bemerkt, in dieser Arbeit auch nicht im Detail diskutiert werden.

Protokoll der Transportschicht	Paketanzahl für eine Transaktion
TCP	9
T/TCP	3
UDP	2

Tabelle 25: Paketanzahl in der Transportschicht für Transaktion

Für eine Aussage bezüglich der Datenrepräsentation in der Transportschicht betrachten wir die Transportschicht-Pakete in Abhängigkeit von der Payload, die in diesem Fall aus den Daten der Anwendungsschicht besteht.

Beim TCP gehen wir von dem minimalen TCP-Paket aus, welches aus der Payload und dem minimalen Header (ohne zusätzlichen Header-Optionen) besteht. Für eine Transaktion werden neun Pakete benötigt, jedoch nur zwei haben eine zusätzliche Payload (Request und Response). Somit ergibt sich ein Transaktions-Volumen von $Payload + 9 * TCPhead$.

Für das T/TCP sieht es ähnlich aus, nur muß man hier noch zusätzlich die Header-Optionen für die Connection-Count berücksichtigen, wie auch in *Abbildung 21* zu sehen ist (es wird davon ausgegangen, daß bereits eine Verbindung bestanden hatte und somit das erstmalige Three-Way Handshake entfallen kann). Es wird dreimal die Header-Option CC und einmal CCECHO verwendet. Es werden nur drei Pakete ausgetauscht, daher beträgt das Transaktionsvolumen $3 * TCPhead + Payload + 3 * OptCC + OptCCECHO$.

Somit ergibt sich nach einer Umformung als Faktor:

$$OFaktor_{Datenrepräsentation}(TTCP) = 1 + \frac{3 * OptCC + OptCCECHO - 6 * TCPhead}{Payload + 9 * TCPhead}; AK :+$$

Formel 5: OFaktor $_{Datenrepräsentation}$ (T/TCP), formal

Den Faktor kann man auch etwas genauer benennen. Mit den Größen $OptCC = 6$, $OptCCECHO = 6$ und $TCPhead = 20$ ergibt sich konkret:

$$OFaktor_{Datenrepräsentation}(TTCP) = 1 - \frac{96}{Payload + 180}; AK :+$$

Formel 6: OFaktor $_{Datenrepräsentation}$ (T/TCP)

Daraus erkennt man sofort, daß das Transaktionsvolumen in jedem Fall reduziert wird, im Extremfall bei kleinen Paketen bis auf die Hälfte im Vergleich zu einer normalen TCP-Verbindung.

Auch in der Verbindungssteuerung ergeben sich durch die Verwendung von T/TCP einige Verbesserungen. Neben der Reduktion der TIME_WAIT-Zeit (Wartezeit, bis der Port wieder freigegeben wird), bringt der Wegfall des Three-Way Handshakes eine entscheidende Beschleunigung von Transaktionen. Denn wie in *Abbildung 26* zu sehen ist, beträgt die Transaktionszeit (vom Client aus betrachtet) für eine TCP-Transaktion $2 * RTT + SPT$. Diese Zeit wird mit dem T/TCP auf $RTT + SPT$ reduziert und ergibt den Faktor:

$$OFaktor_{Verbindungssteuerung}(TTCP) = 1 - \frac{RTT}{2 * RTT + SPT}; AK :+$$

Formel 7: OFaktor $_{Verbindungssteuerung}$ (T/TCP)

Somit kann man davon ausgehen, daß bei schnellen Serversystemen (kleine SPT) sich die Transaktionszeit auf die Hälfte reduziert. Diese theoretische Betrachtung spiegelt sich auch in den vorgenommenen Messungen wieder. In *Abbildung 38* sieht man, daß bei kleinen Paketen die Transaktionszeit halbiert wird.

Eine weitere Optimierung in der Transportschicht, wenn auch älteren Datums, ist die TCP/IP-Header Kompression. Der Autor v. Jacobson nennt in der Protokollbeschreibung [108] selbst Werte über die Effizienz dieser Optimierung. Die von ihm beschriebenen Werte sind Kompressionsfaktoren (Kehrwert des hier verwendeten *OFaktors*) und lauten für die Payload 1, Payload mit TCP/IP-Header 0,83 und Payload mit TCP/IP-Header Kompression 0,98. Daraus ergibt sich als $OFaktor$ für die Datenrepräsentation:

$$OFaktor_{Datenrepräsentation}(TCPIPHeaderComp) = 0{,}85 ;\text{AK:-}$$

Formel 8: OFaktor $_{Datenrepräsentation}$ (TCP/IP-Header Kompression), Literatur

Dieser Faktor scheint jedoch für den allgemeinen Fall etwas hochgegriffen zu sein. Geht man von einer Aussage aus der Protokollbeschreibung [108] aus, so kann man allgemein von einer TCP/IP-Header Länge *HeaderCompressed* von 5 Bytes ausgehen, der sogar bis auf 3 Bytes schrumpfen kann. Setzt man diese Werte an, so erhält man im Vergleich zum Norm-Protokollstack mit einer TCP/IP-Header Länge *TCPIPheader* von 40 Bytes einen Optimierungsfaktor von:

$$OFaktor_{Datenrepräsentation}(TCPIPHeaderComp) = \frac{Payload + HeaderCompressed}{Payload + TCPIPheader} = 1 - \frac{35}{Payload + 40}$$

$$AK : O$$

Formel 9: OFaktor $_{Datenrepräsentation}$ (TCP/IP-Header Kompression)

Um mit *Formel 9* einen Wert von 0,85 zu erreichen, muß man davon ausgehen, daß die Payload nur 193 Bytes groß ist. Im Durchschnitt ist jedoch von einer größeren Payload auszugehen.

5.3 Anwendungsschicht

Die Messungen des W3Cs bezüglich des Nutzens von persistenten Verbindungen im HTTP/1.1 wurde in *Kapitel 4.4.1: HTTP Pipelining und Persistente Verbindungen* dargestellt und in *Tabelle 21* aufgeführt. Die Maßnahme, nur eine anstatt von vier Verbindungen zu verwenden, resultiert in einer Einsparung im Datenvolumen. Es werden nur noch 191843 Bytes anstatt von 216289 Bytes für den Transfer verwendet. Daraus ergibt sich für den Optimierungsfaktor:

$$OFaktor_{Datenrepräsentation}(PersistenteVerbindungen) = \frac{191843}{216289} = 0{,}87;\ AK : O$$

Formel 10: OFaktor $_{Datenrepräsentation}$ (PersistenteVerbindungen)

Die Verbindungssteuerung ist allerdings die primäre Motivation für die persistenten Verbindungen gewesen. Um so erstaunlicher wirkt das Ergebnis aus *Tabelle 21* auf den ersten Blick. Statt 0.97 Sekunden für den Transfer der Microscape-Seite benötigt man bei persistenten Verbindungen 1,25 Sekunden.

$$OFaktor_{Verbindungssteuerung}(PersistenteVerbindungen) = \frac{1{,}25}{0{,}97} = 1{,}29;\ AK : O$$

Formel 11: OFaktor $_{Verbindungssteuerung}$ (PersistenteVerbindungen) ein Clientsystem

Dieses vermeintlich schlechte Ergebnis relativiert sich allerdings dadurch, daß nur eine anstatt von vier Verbindungen verwendet wird. Damit kann der Webserver viermal soviele Verbindungen offenhalten. Würde man diesen Effekt mit in die Berechnung einbeziehen, so ergibt sich für den Transfer an vier Clients unter Verwendung von maximal vier gleichzeitigen Verbindungen für HTTP/1.0 eine Transferzeit von $4 * 0{,}97 = 3{,}88$ sec, HTTP/1.1 mit persistenten Verbindungen kann die vier Clientsysteme innerhalb von $1{,}25$ sec bedienen. Unter Berücksichtigung der Verbindungsanzahl ergibt sich dann ein Optimierungsfaktor von:

$$OFaktor_{Verbindungssteuerung}(PersistenteVerbindungen) = \frac{1{,}25}{3{,}88} = 0{,}32;\ AK : O$$

Formel 12: OFaktor $_{Verbindungssteuerung}$ (PersistenteVerbindungen)

Einen weiteren Performanzgewinn kann man mit dem HTTP Pipelining (*Kapitel 3.2.3.2: HTTP Pipelining*) erreichen. Pipelining setzt für die eigene Effektivität persistente Verbindungen voraus. Da sich die Requests and Responses beim Pipelining selbst nicht ändern, sondern nur die Reihen-

folge und Methodik der Verarbeitung, wundert es nicht, daß sich nach *Tabelle 21* nur eine minimale Veränderung bezüglich der Datenrepräsentation gegenüber den persistenten Verbindungen ergibt:

$$OFaktor_{Datenrepräsentation}(Pipelining) = \frac{191551}{216289} = 0{,}89; AK : O$$

Formel 13: OFaktor $_{Datenrepräsentation}$ (Pipelining)

Der Nutzen hinsichtlich der Verbindungssteuerung ist da schon wesentlich auffälliger:

$$OFaktor_{Verbindungssteuerung}(Pipelining) = \frac{0{,}68}{0{,}97} = 0{,}70; AK : O$$

Formel 14: OFaktor $_{Verbindungssteuerung}$ (Pipelining) ein Clientsystem

Berücksichtigt man auch hier die Anzahl der Verbindungen, so erhält man einen Optimierungsfaktor von:

$$OFaktor_{Verbindungssteuerung}(Pipelining) = \frac{0{,}68}{3{,}88} = 0{,}18; AK : O$$

Formel 15: OFaktor $_{Verbindungssteuerung}$ (Pipelining)

Damit ist ein optimiertes HTTP/1.1 Client-Server System ca. fünf mal so schnell wie ein HTTP/1.0 System.

Die Kompression der HTTP-Inhalte (HTTP Content-Encoding) wurde nicht nur in dieser Arbeit untersucht, sondern auch vom W3C [77]. In der Untersuchung vom W3C wurde allerdings nur eine HTML-Seite übertragen und die resultierende Zeit und das Volumen gemessen. In dieser Untersuchung wurde eine einzelne HTTP-Anfrage für die HTML-Seite über eine Modemverbindung gestellt und die Anzahl der übertragenen Pakete und die benötigte Zeit gemessen. Diese Messung ist insoweit sehr interessant, als sie aufzeigt, daß die Volumenreduktion auch unmittelbar zu einer zeitlichen Reduktion der Verbindungsdauer in der gleichen Größenordnung führt. Als Optimierungsfaktoren ausgedrückt ergibt sich dabei folgendes Ergebnis:

$$OFaktor_{Datenrepräsentation}(HTTPContentEncoding) = \frac{21}{67} = 0{,}31; AK : O$$

Formel 16: OFaktor $_{Datenrepräsentation}$(HTTP Content-Encoding), Paketzahl, W3C HTML

$$OFaktor_{Verbindungssteuerung}(HTTPContentEncoding) = \frac{4{,}35}{12{,}21} = 0{,}36; AK : O$$

Formel 17: OFaktor $_{Verbindungssteuerung}$ (HTTP Content-Encoding), W3C HTML

Dieses Ergebnis wurde auch von einer weiteren Studie des W3C untermauert [78], in welcher der Einfluß von Groß- und Kleinschreibung der HTML-Tags untersucht wurde. Selbst im ungünstigsten Fall (gemischte Groß- und Kleinschreibung) wurden mit dem deflate-Algorithmus ein Optimierungsfaktor von 0,35 erzielt.

Um diese Ergebnisse für ein typisches WWW-Scenario anzupassen, wurde in *Kapitel 4.4.3: HTTP Content-Encoding* eine datenrepräsentationsbezogene Messung der HTTP Content-Encoding mit

der W3C Microscape Webseite durchgeführt. Für diese Web-Seite (inklusive Grafiken) ergab sich ein Optimierungsfaktor für die Datenrepräsentation von:

$$OFaktor_{Datenrepräsentation}(HTTPContentEncoding) = \frac{136241}{171230} = 0{,}80; AK : O$$

Formel 18: OFaktor $_{Datenrepräsentation}$ (HTTP Content-Encoding)

Dieses Ergebnis hat schon eher als die beiden vorhergehenden einen gewissen Allgemeinheitsanspruch, jedoch ist gerade bei der Kompression der Inhalt der übertragenen Daten entscheidend. Dadurch ist dieser Faktor mit einer Unsicherheit behaftet.

Das HTTP-Caching ist mittlerweile stark verbreitet, nicht zuletzt deshalb, weil man es ohne Eingriff in den Protokollstack implementieren kann. Man muß lediglich beim Clientsystem einen Eintrag vornehmen, um alle WWW-Zugriffe über einen Cacheserver zu leiten. Trotz der großen Anzahl von Cacheservern lassen sich nur schwer allgemeine quantitative Aussagen über die Effektivität solcher Systeme machen. Die Gründe dafür sind die unterschiedliche Szenarien und Konfigurationen. Nicht nur die Benutzerzahlen und Konfigurationsparameter der Cacheserver unterscheiden sich stark, sondern auch die Netztopologien, in denen die Systeme eingebettet sind. So gibt es einzelne autonome Cacheserver und solche, die in bis zu dreistufigen Cacheserver Hierarchien eingebunden sind. Zudem müssen Cacheserver nicht zwangsweise effektiv sein, denn wenn die Benutzerinteressen und Profile stark voneinander abweichen, werden vom Cacheserver nur wenige Cache-Hits erzeugt. Damit entartet das Proxy-Cache System zu einem Proxyserver. Dennoch lassen sich pauschale Aussagen über den Grad der Optimierung treffen, die im folgenden an der Beispielen der Universität von Melbourne und des FB Mathematik und Informatik der Philipps-Universität Marburg hergeleitet werden sollen. Eine Aussage zur Verbindungssteuerung soll an dieser Stelle nicht getroffen werden, da die zeitliche Komponente zu stark an die individuelle Topologie des jeweiligen Serversystems gebunden ist. Zudem besteht hier noch eine starke Hardware-Abhängigkeit vom Serversystem. So trat beim Caches Server Projekt des DFN-Vereins zeitweise ein massive Überlastung des Systems auf, welche zu indiskutablen Transferzeiten geführt hat. Man kann lediglich die Aussage treffen, daß im Falle eines Cache-Hits die Antwortzeit auf einen Bruchteil schrumpft, während bei einem Cache-Miss die Antwortzeit in Abhängigkeit von der Topologie auf ein Mehrfaches einer Direktverbindung ansteigen kann. Im Gegensatz zur Verbindungssteuerung liegen für die Datenrepräsentation konkrete Messergebnisse vor, zum einen von der Uni Melbourne und zum anderen vom FB12 der Uni Marburg. Gemessen wurden in beiden Fällen die Anzahl von Bytes, die insgesamt an die Clientsysteme geliefert wurden und die Hitraten für die Requests und die transportierten Bytes. Obwohl die beiden Serversysteme in der Anzahl der Requests um zwei Größenordnungen auseinanderlagen, lagen die Hitraten für die Bytes nicht allzu weit voneinander entfernt. In Marburg wurde eine Volumen-Hitrate von 16,6% erzielt und in Melbourne eine Hitrate von 31,0%. Der allgemeine Anwendungsfall wird sicher eher dem Einsatzfall von Melbourne entsprechen, aber dennoch soll das Mittel der beiden Ergebnisse als Abschätzung für den Optimierungsfaktor des HTTP-Cachings verwendet werden.

$$OFaktor_{Datenrepräsentation}(HTTPCaching) = \frac{0{,}83 + 0{,}69}{2} = 0{,}76; AK : -$$

Formel 19: OFaktor $_{Datenrepräsentation}$ (HTTP Caching)

5.4 Diskussion und Zusammenführung der Ergebnisse

In den *Kapiteln 5.1, 5.2* und *5.3* wurden die Ergebnisse aus Messung, Simulation und Literatur zusammengetragen und in Form von Optimierungsfaktoren dargestellt. Diese Optimierungsfaktoren ermöglichen es, die Optimierung einer WWW-Verbindung quantitativ zu bestimmen, ohne Messungen durchführen zu müssen. Dazu werden die Optimierungsfaktoren einer Kategorie (Datenrepräsentation oder Verbindungssteuerung) der verwendeten Optimierungen multipliziert. Das Resultat ist ein Wert der angibt, um welchen Faktor sich die Größe der Kategorie (Datenrepräsentation: Volumen, Verbindungssteuerung: Zeit) gegenüber dem Standard Protokollstack (IPv4, TCP, HTTP/1.0) unterscheidet. Somit kann man vor dem Einsatz solcher Optimierungsstrategien bereits den Nutzen abschätzen.

Zu beachten ist dabei allerdings, daß die Faktoren voneinander abhängig sein können und von unterschiedlicher Qualität sind. So muß man davon ausgehen, daß eine Kompression in der Anwendungsschicht dazu führt, daß die Kompression in der Vermittlungsschicht nicht den Optimierungsfaktor der hier gegebenen Abschätzung erreicht, weil dann die Daten, die in der Vermittlungsschicht vorliegen, nun bereits komprimiert sind. Dieses Problem taucht allerdings grundsätzlich in der Kategorie des Datenvolumens auf, wenn der Optimierungsfaktor durch eine experimentelle Abschätzung bestimmt wurde. Die Qualität einer solchen Abschätzung kann nicht an die einer Bestimmung nach Protokollspezifikation heranreichen, da der Kompressionsgrad von unbekannten Daten nicht ohne Kenntnis der Daten bestimmt werden kann.

Nach der Bestimmung der Optimierungsfaktoren kann nun die Klassifikationstafel *Tabelle 16* aus *Kapitel 3.4* für die untersuchten Optimierungen benannt werden (*Tabelle 26*).

Schicht	Optimierung	Verb.-Steuerung	Datenrepräsentation
Vermittlungsschicht	IPv6	$0{,}85$;AK:-	$1+\frac{20}{20+Payload}$;AK:+
	IPComp		$0{,}86$;AK:O
Transportschicht	T/TCP	$1-\frac{RTT}{2*RTT+SPT}$;AK:+	$1-\frac{96}{Payload+180}$;AK:+
	TCP/IP Header comp.		$1-\frac{35}{Payload+40}$;AK:O
Anwendungsschicht	Persistente Verb.	$0{,}32$;AK:O	$0{,}87$;AK:O
	Pipelining	$0{,}18$;AK:O	$0{,}89$;AK:O
	Caching		$0{,}76$;AK:-
	Content-Encoding		$0{,}80$;AK:O

Tabelle 26: Optimierungsfaktoren, formal

Man kann bei der Betrachtung aber noch einen Schritt weiter gehen und die Optimierungsfaktoren für eine typische WWW-Verbindung berechnen. Dazu müssen nur die Werte für RTT, SPT und $Payload$ in die *Tabelle 26* eingesetzt werden.

Ausgehend von der Diskussion in *Kapitel 4.1* kann man in der Anwendungsschicht einen Median von 2155 Byte annehmen. Dies ist auch die Payload der Transportschicht. Daraus berechnet sich die Payload der Vermittlungsschicht (TCP-Header 20 Bytes) mit 2175 Bytes. Geht man von einer durchschnittlichen WAN-Verbindung aus, so kann man mit einer RTT von 100 ms rechnen. Auch die SPT liegt im allgemeinen in dieser Größenordnung und soll auch mit 100 ms in die Berechnung mit einfließen. Daraus ergibt sich die *Tabelle 27*, die eine auf WWW-Kommunikation zugeschnittene Bewertung der besprochenen Optimierungsmethoden darstellt.

Schicht	Optimierung	Verb.-Steuerung	Datenrepräsentation
Vermittlungsschicht	IPv6	0,85 ;AK:-	1,01;AK:+
	IPComp		0,86 ;AK:O
Transportschicht	T/TCP	0,67 ;AK:+	0,96 ;AK:+
	TCP/IP Header comp.		0,98 ;AK:O
Anwendungsschicht	Persistente Verb.	0,32 ;AK:O	0,87 ;AK:O
	Pipelining	0,18 ;AK:O	0,89 ;AK:O
	Caching		0,76 ;AK:-
	Content-Encoding		0,80 ;AK:O

Tabelle 27: Optimierungsfaktoren, WWW-Verkehr

Geht man von *Tabelle 27* aus, um den Optimierungsfaktor für die optimale Kombination der einzelnen Optimierungsmethoden zu finden, so sieht man, daß die Kombination von IPv6, T/TCP und Pipelining den optimalen Gewinn für die Verbindungssteuerung ergibt. Der Faktor für die persistenten Verbindungen darf nicht mit berücksichtigt werden, weil das HTTP-Pipelining bereits die persistenten Verbindungen mit nutzt. Als Produkt ergibt sich somit:

$$OFaktor_{Verbindungssteuerung} = 0{,}85 * 0{,}67 * 0{,}18 = 0{,}10$$

Formel 20: Produkt der OFaktor $_{Verbindungssteuerung}$ (OptimierungsMethode)

Nach dem gleichen Schema ergibt sich für die Datenrepräsentation die Kombination von IPv6, IPComp, T/TCP, Pipelining, Caching und Content-Encoding. Bei der kombinierten Verwendung von IPComp und Content-Encoding muß man aus den oben angesprochenen Gründen das Minimum der Optimierungsfaktoren von IPComp und Content-Encoding bei der Produktbildung verwenden. Damit ergibt sich als Produkt der Optimierungsfaktoren für die Datenrepräsentation:

$$OFaktor_{Datenrepräsentation=} 1{,}01 * 0{,}96 * 0{,}89 * 0{,}76 * Min(0{,}86;0{,}80) = 0{,}53$$

Formel 21: Produkt der OFaktor $_{Datenrepräsentation}$ (OptimierungsMethode)

Unter Einsatz der Optimierungsmethoden

- IPv6
- IPComp
- T/TCP
- HTTP Pipelining

- HTTP Caching
- HTTP Content-Encoding

kann somit gegenüber dem Standard Protokollstack eine Beschleunigung der WWW-Kommunikation um den Faktor Zehn erreicht werden bei gleichzeitiger Halbierung des übertragenen Volumens.

6 Zusammenfassung

Nach einem kurzen Überblick über die Arbeit werden die konkreten Aussagen zusammengefaßt. Zum Schluß wird ein Ausblick auf weiteren Arbeiten in diesem Kontext gegeben

.

6.1 Überblick

Das Internet befindet sich in einem exponentiellen Wachstum. Seit März 1995 ist der WWW-Verkehr der dominante Anteil des Verkehrs im Internet. Mit dieser Motivation wurde die WWW-Kommunikation in dieser Arbeit genauer betrachtet.

An der WWW-Kommunikation sind mehrere Protokollschichten beteiligt. Diese Protokollschichten wurden im Laufe der Entwicklung um Erweiterungen bzw. Optimierungen ergänzt. Im Grundlagenteil wird die Basis der Kommunikation erarbeitet. Daran schließt sich im Methodenteil die Erläuterung der Optimierungen an. Im darauffolgenden experimentellen Teil werden Messungen und Simulationen zu den Optimierungen durchgeführt. Die Ergebnisse dieser Messungen werden im Ergebnisteil zusammen mit Ergebnissen aus der Literatur diskutiert und bewertet.

6.2 Einblick

Im Methodenteil wird zunächst ein Klassifikationsmodell zur Einordnung von Optimierungen eingeführt. Dieses Klassifikationsmodell teilt die Optimierungen anhand des OSI-Modells in Schichten ein und orthogonal dazu in die Klassen der Verbindungssteuerung und der Datenrepräsentation. Unter der Verbindungssteuerung sind alle Aspekte zusammengefaßt, die sich auf das zeitliche Verhalten des Protokolls auswirken. Dazu gehört der Verbindungsaufbau, -abbau und das zeitliche Verhalten beim Datenaustausch. Die Klasse der Datenrepräsentation faßt alle Aspekte zusammen, die mit der Datenstruktur und dem Datenvolumen zu tun haben.

Während dieses Kapitels werden Optimierungen vorgestellt, die im Laufe der Entwicklung des Internets vorgestellt bzw. bereits implementiert wurden. Das Spektrum des Entwicklungsstandes dieser Optimierungen ist weit gefächert und erstreckt sich von langjährig im Einsatz befindlichen Optimierungen bis hin zu Optimierungen, die erst als RFC vorliegen und noch nicht implementiert wurden. Die Optimierungen bezüglich der Verbindungssteuerung sind IPv6 in der Vermittlungsschicht, T/TCP in der Transportschicht und HTTP Persistente Verbindungen sowie HTTP Pipelinening in der Anwendungsschicht. Bezüglich der Datenrepräsentation wurden die Optimierungen IPv6 und IPComp in der Vermittlungsschicht, T/TCP und TCP/IP Header Compression in der Transportschicht sowie HTTP Caching und HTTP Content Encoding in der Anwendungsschicht betrachtet.

Im Kapitel zu Messungen und Simulationen wird eine theoretische Betrachtung am IPv6 durchgeführt, eine Simulation des IPComp und jeweils Messungen zu UDP, TCP und T/TCP, HTTP Caching und HTTP Content-Encoding.

Im Ergebnis-Kapitel wurden die Ergebnisse aus Messungen, Simulationen und Literatur zusammengetragen und für jede Optimierung eine quantitative Bewertung abgegeben, die über einem spezifischen sogenannten Optimierungsfaktor benannt wurde. Diese Optimierungsfaktoren repräsentieren das Verhältnis der Optimierungen gegenüber einem konservativen Protokollstack. Diese Faktoren können nun in die Klassifikationstafel eingetragen werden und liefern einen quantitativen Überblick über die diskutierten Optimierungen. Um die Abhängigkeiten der Optimierungs-

faktoren von Verbindungsparametern zu eliminieren und eine konkrete Aussage zum Zusammenspiel der Optimierungen zu erhalten, wurde eine mit Einschränkungen repräsentative WWW-Kommunikation angenommen und das Ergebnis der Kombination der Optimierungen berechnet. Dabei ergab sich, daß das Volumen auf die Hälfte reduziert werden kann bei gleichzeitiger Verminderung der Transferzeit auf ein Zehntel.

In dieser Arbeit wurde eine Klassifikation und quantitative Bewertung von Optimierungsmethoden für den WWW-Verkehr im Internet erarbeitet. Damit wird ein Hilfsmittel zur Verfügung gestellt, aktuelle und zukünftige Optimierungen und Protokollerweiterungen einzuordnen und quantitativ zu bewerten.

6.3 Ausblick

Noch ausstehende Arbeiten sind in jedem Fall die Bewertung des IPv6 in Bezug auf die Verbindungssteuerung.

Zudem wäre es denkbar, das Modell auf die Sicherungsschicht auszuweiten und die Klassifikation um z.B. die Paketanzahl zu erweitern. Eine weitere Differenzierung der Anwendungsschicht in die Schichten 5-7 des OSI-Modells wäre ebenso denkbar.

Eine sinnvolle formale Erweiterung wäre auch die Darstellung der Optimierungen als Matrizen. Damit könnte die WWW-Kommunikation als Matrix-Operation dargestellt werden, bei der die Optimierungen als Operatoren auf Eingabevektoren wirken.

Zudem sollte das Beispielszenario des W3C (Microscape-Webseite) überdacht werden und gegebenenfalls durch alternative Szenarien ergänzt werden. Überhaupt wäre eine weitere Analyse der Webinhalte wünschenswert, denn so könnten die unsicheren Ergebnisse der kompressionsbezogenen Optimierungen gefestigt werden.

7 Anhang

Im Anhang befinden sich die Verzeichnisse für Abkürzungen, Abbildungen, Formeln, Tabellen und Literatur.

7.1 Abkürzungsverzeichnis

6BONE	Testnetzwerk für IPv6
ACK	TCP Flag: Acknowledgement number ist korrekt
ACL	Access Control Lists, Zugriffliste für zu sichernde Objekte
ARPANET	Advanced Research Projects Agency NET
ASCII	American Standard Code for Information Interchange, Zeichensatzstandard
ATM	Asynchronous Transfer Mode, Hochgeschwindigkeits-Netzwerkprotokoll mit Echtzeiteigenschaften
BSD	UNIX-Typ (Berkeley Software Design)
B-WiN	Breitband-Wissenschaftsnetz
CC	T/TCP Option: Connection Count
CCECHO	T/TCP Option: Connection Count Echo
CCNEW	T/TCP Option: Connection Count New
CERN	Conseil Européen pour la Recherche Nucléaire, offizieller Name: European Organization for Nuclear Research
CPT	Client Processing Time, Zeit, die der Client zur Bearbeitung der Aufgabe benötigt
CSMA/CD	Carrier sense multiple access with collision detection
DFN	Deutsches Forschungsnetz
DNS	Domain Name System
DOS	Denial of Service
EBNF	Extended Backus-Naur Form, Syntax Beschreibungssprache
FDDI	Fiber-Distributed Data Interface
FIN	TCP Flag: FIN – Sender hat Datenübertragung abgeschlossen
FTP	File Transfer Protocol
GIF	Graphics Interchange Format, Dateiformat für Pixel-Grafiken
G-WiN	Gigabit-Wissenschaftsnetz
HTML	Hypertext Markup Language, Hypertext Beschreibungssprache
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
ICP	Internet Cache Protocol

IEEE	Institute of Electrical and Electronics Engineers, Inc.
IETF	Internet Engineering Task Force, Institution zur Koordinierung von Internet Protokollentwicklungen
IGMP	Internet Group Management Protocol
InterNIC	Internet Network Information Center, geführt von Network Solutions und verschiedenen Firmen in Kooperation mit der U.S. Regierung
IPCA	IPComp Association
IPComp	IP Payload Compression Protocol
IPng	Internet Protokoll new generation, Arbeitstitel für IPv6
IPv4	Internet Protokoll Version 4
IPv6	Internet Protokoll Version 6
IRC	Internet Relay Chat
IRTP	Internet Reliable Transaction Protocol
ISAKMP	Internet Security Association and Key Management Protocol, Protokoll zum Aufbau einer gesicherten Verbindung zwischen zwei Hosts im Internet
ISDN	Integrated Services Digital Network
ISO	International Standards Organisation
JPEG	Joint Photographic Experts Group, Kompressionsalgorithmus für Bilder
LAN	Local Area Network
LZS	STAC Lempel-Ziv Standard Compression
LZW	Lempel Ziv Welch Compression
MILNET	Militärische Teil des ARPANETs
MIME	Multipurpose Internet Mail Extensions, Erweiterung der Spezifikation für Internet Mails
MPEG	Moving Picture Experts Group, MPEG1-MPEG7 Kompressionsalgorithmen für Audio/Video
MSS	TCP Option: Maximum Segment Size
NCSA	National Center for Supercomputing Applications (University of Illinois)
NFS	Network File System
NIC	Network Information Centre
NNTP	Network News Transfer Protocol
NPL	National Physical Laboratory, United Kingdom's national standards laboratory
NSFNET	NSF gefördertes Backbone Projekt, 1987 - 1995
OSI	Open System Interconnect
PPP	Point to Point Protocol

PSH	TCP Flag: PUSH – Empfänger soll Daten schnell an Anwendung geben
QoS	Quality of Service
RIPE	Réseaux IP Européens (Network Coordination Centre)
RST	TCP Flag: RESET - Verbindung zurücksetzen
RTT	Round Trip Time, Zeit vom Sender zum Empfanger und wieder zurück
SLIP	Serial Line IP
SMTP	Simple Mail Transfer Protocol
SPT	Server Processing Time, Zeit, die der Server zur Bearbeitung der Aufgabe benötigt
SYN	TCP Flag: SYNC – Sequenznummer soll synchronisiert werden
T/TCP	TCP for Transactions
TCP	Transmission Control Protocol
TFTP	Tiny File Transfer Protocol
TOS	IPv4 Header Option: Type Of Service
TTL	Time To Live
UCLA	University of California, Los Angeles
UDP	User Datagram Protocol
URG	TCP Flag: URGENT - Urgent pointer ist korrekt
USENET	Virtuelles Netzwerk zur Distribution von News im Internet aus dem Jahre 1979
VPN	Virtual Private Network
W3C	World Wide Web Consortium
WAN	Wide Area Network
WWW	World Wide Web
XHTML	Extensible HyperText Markup Language, Eine Formulierung von HTML 4 in XML
XML	Extensible Markup Language, Beschreibungssprache für Dokumente mit strukturierten Informationen

7.2 Abbildungsverzeichnis

Abbildung 1: Zeichnung des Netzwerks im Dezember 1969 von Alex McKenzie vom BBN [1].... 3
Abbildung 2: ARPANET Oktober 1980 [3] 4
Abbildung 3: Entwicklung Hostcount Marburg, Europa, Welt 6
Abbildung 4: Monatliches Transfervolumen Marburg in MByte 7
Abbildung 5: Protokollanteile des NSFNET Verkehrs [63] 8
Abbildung 6: Kommunikation über Router 13
Abbildung 7: Hierarchie der physikalischen Struktur (Auszug, Stand 1998)..... 16

Abbildung 8: Hierarchie der DNS Struktur (Auszug, Stand 1998) 17
Abbildung 9: Transfervolumen des NSFNet Backbone (Stand 12/1994)..... 18
Abbildung 10: Topologie des ISP UUNET in USA (Stand 06/2000) [107]..... 19
Abbildung 11: Karte von verschiedenen Informationsquellen in der Domain .ch 20
Abbildung 12: Karte der Newsgruppe comp.ai.neural-nets..... 20
Abbildung 13: Skizze des ersten Ethernetsystems von Dr. Robert M. Metcalfe, 1976 [65] 22
Abbildung 14: Ethernet Packetformat: Ethernet II und IEEE 802.3 [16]..... 22
Abbildung 15: ATM Reference Modell [14] 24
Abbildung 16: Kommunikationskanäle in den Schichten der TCP/IP Protokollfamilie [102]..... 26
Abbildung 17: IPv6 Header und optionale Erweiterungsheader 29
Abbildung 18: 6BONE konnektierte Netze, Stand August 2000 [44]..... 30
Abbildung 19: UDP Einkapselung..... 31
Abbildung 20: TCP Einkapselung 32
Abbildung 21: Verbindungsaufbau T/TCP 36
Abbildung 22: HTTP Client-Server Beispiel mit zwei Servern 38
Abbildung 23: HTTP Client-Server Beispiel mit zwei Servern (Verbindungsdiagramm)..... 39
Abbildung 24: Netzwerk-Topologien 43
Abbildung 25: Ethernet Verbindungsstecker..... 44
Abbildung 26: Transaktionen in der Transportschicht 56
Abbildung 27: IPComp-Header 59
Abbildung 28: TCP/IP-Datagramm mit konstanten Header-Feldern [108]..... 61
Abbildung 29: TCP/IP-Datagramm mit komprimiertem Header [108]..... 62
Abbildung 30: WWW-Cacheserver Topologie, einfach..... 64
Abbildung 31: Squid-2 caching algorithm [8] 65
Abbildung 32: Karte der DFN Cacheserver [83] 66
Abbildung 33: DFN Cacheserver Topologie, 3-stufig [37] 67
Abbildung 34: DFN Cacheserver Topologie, 2-stufig [37] 68
Abbildung 35: Histogramm WWW-Zugriffe (normiert) 72
Abbildung 36: Ethernet Paketanzahl..... 78
Abbildung 37: Payloadanteil der Ethernetpakete..... 79
Abbildung 38: Transaktionszeit 80
Abbildung 39: Cache Requests FB 12 der Uni Marburg 82
Abbildung 40: Cache Requests Uni Melbourne 82
Abbildung 41: Melbourne Hitrate Bytes (Bytes)..... 83

7.3 Formelverzeichnis

Formel 1: Definition Optimierungsfaktor OFaktor $_{Kommunikationsaspekt}$ 85
Formel 2: OFaktor $_{Datenrepräsentation}$ (IPv6) 86
Formel 3: OFaktor $_{Verbindungssteuerung}$ (IPv6) 86
Formel 4: OFaktor $_{Datenrepräsentation}$ (IPComp) 86
Formel 5: OFaktor $_{Datenrepräsentation}$ (T/TCP), formal 88
Formel 6: OFaktor $_{Datenrepräsentation}$ (T/TCP) 88
Formel 7: OFaktor $_{Verbindungssteuerung}$ (T/TCP) 88
Formel 8: OFaktor $_{Datenrepräsentation}$ (TCP/IP-Header Kompression), Literatur 88
Formel 9: OFaktor $_{Datenrepräsentation}$ (TCP/IP-Header Kompression) 89
Formel 10: OFaktor $_{Datenrepräsentation}$ (PersistenteVerbindungen) 89
Formel 11: OFaktor $_{Verbindungssteuerung}$ (PersistenteVerbindungen) ein Clientsystem 89
Formel 12: OFaktor $_{Verbindungssteuerung}$ (PersistenteVerbindungen) 89
Formel 13: OFaktor $_{Datenrepräsentation}$ (Pipelining) 90
Formel 14: OFaktor $_{Verbindungssteuerung}$ (Pipelining) ein Clientsystem 90
Formel 15: OFaktor $_{Verbindungssteuerung}$ (Pipelining) 90
Formel 16: OFaktor $_{Datenrepräsentation}$(HTTP Content-Encoding), Paketzahl, W3C HTML 90
Formel 17: OFaktor $_{Verbindungssteuerung}$ (HTTP Content-Encoding), W3C HTML 90
Formel 18: OFaktor $_{Datenrepräsentation}$ (HTTP Content-Encoding) 91
Formel 19: OFaktor $_{Datenrepräsentation}$ (HTTP Caching) 91
Formel 20: Produkt der OFaktor $_{Verbindungssteuerung}$ (OptimierungsMethode) 93
Formel 21: Produkt der OFaktor $_{Datenrepräsentation}$ (OptimierungsMethode) 93

7.4 Tabellenverzeichnis

Tabelle 1: Abschätzung Hostcount Marburg, Europa, Welt (Stand: Mitte 2000) 6
Tabelle 2: Abschätzung monatliches Transfervolumen Marburg 7
Tabelle 3: OSI-Schichtenmodell 11
Tabelle 4: Einordnung verschiedener Protokollfamilien in das OSI-Modell [74] 14
Tabelle 5: Aufbau Ethernetpaket 23
Tabelle 6: ATM-Dienstklassen und zugehörige AAL-Schichten [89] 25
Tabelle 7: IPv4 Paket 27
Tabelle 8: IPv6-Paket Header 29

Tabelle 9: UDP Paket 31
Tabelle 10: Protokolleigenschaften von IP, UDP, TCP [99] 32
Tabelle 11: TCP Paket 33
Tabelle 12: TCP Optionen 35
Tabelle 13: Protokolle der Anwendungsschicht 37
Tabelle 14: Status-Codes des HTTP-Servers 41
Tabelle 15: Klassifikationstafel 53
Tabelle 16: Klassifikationstafel Optimierungsansätze 69
Tabelle 17: Vergleich Median und Durchschnitt 71
Tabelle 18: Vergleich Datenrepräsentation IPv4 und IPv6 74
Tabelle 19: IPComp Simulationsergebnisse 75
Tabelle 20: Vergleich HTTP-Versionen [77] 81
Tabelle 21: Vergleich HTTP-Versionen, optimiert [77] 81
Tabelle 22: Durchschnittswerte Cacheserver Marburg und Melbourne 83
Tabelle 23: HTTP Content-Encoding 84
Tabelle 24: Bewertung der Aussagekraft 86
Tabelle 25: Paketanzahl in der Transportschicht für Transaktion 87
Tabelle 26: Optimierungsfaktoren, formal 92
Tabelle 27: Optimierungsfaktoren, WWW-Verkehr 93

7.5 Literatur und Verweise

[1] "BBN Technologies." http://www.gte.com/AboutGTE/gto/bbnt/, 19-3-1999.

[2] "IPv6 Implementations." http://www.ipv6.org/impl/index.html, 4-3-1999.

[3] ACM SIGCOMM "Selected ARPANET Maps." ACM SIGCOMM, http://www.acm.org/sigcomm/ccr/archive/1990/oct90/, 1990.

[4] Baran, Paul "Publications in the On Distributed Communications Series." http://www.rand.org/publications/RM/baran.list.html, 19-3-1964.

[5] Berners-Lee, Tim "Information Management: A Proposal." http://www.w3.org/History/1989/proposal.html, 1-3-1989.

[6] Berners-Lee, Tim "HyperText Transfer Protocol Design Issues." http://www.w3.org/Protocols/DesignIssues.html, 1-1-1991.

[7] Berners-Lee, Tim, Fielding, Roy T., and Nielsen, Henrik Frystyk "RFC1945: Hypertext Transfer Protocol -- HTTP/1.0." ftp://ftp.isi.edu/in-notes/rfc1945.txt, 1996.

[8] Bertold, Kolics "Squid-1.1 and Squid-1.NOVM caching algorithm." http://squid.nlanr.net/Squid/FAQ/refresh-flowchart.gif, 9-9-1999.

[9] Borman, David A. "RFC2147: TCP and UDP over IPv6 Jumbograms." ftp://ftp.isi.edu/in-notes/rfc2147.txt, 1-5-1997.

[10] Braden, Bob "RFC1379: Extending TCP for Transactions -- Concepts." ftp://ftp.isi.edu/in-notes/rfc1379.txt, 1992.

[11] Braden, Bob "RFC1644: T/TCP -- TCP Extensions for Transactions." ftp://ftp.isi.edu/in-notes/rfc1644.txt, 1994.

[12] Braden, R. "RFC955: Towards a Transport Service for Transaction Processing Applications." , 1985.

[13] Bush, Vannevar "As We May Think." http://www.isg.sfu.ca/~duchier/misc/vbush/vbush-all.shtml, 1-7-1945.

[14] CISCO "Asynchronous Transfer Mode (ATM) Switching." http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/atm.htm, 1999.

[15] CISCO "Bridging and Switching Basics." http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/bridging.htm, 1999.

[16] CISCO "Cisco Connection Documentation CD." http://www.cisco.com/univercd/home/home.htm, 1999.

[17] CISCO "Token Ring/IEEE 802.5." http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/55031.pdf, 1999.

[18] Crocker, David H. "RFC822: STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES." ftp://ftp.isi.edu/in-notes/rfc822.txt, 13-8-1982.

[19] Davies, Donald "Donald Davies." http://www.npl.co.uk/npl/online/html/about/100yrs/famous/name1.htm, 1-1-1924.

[20] Deering, Stephen E. and Hinden, Robert M. "RFC1883: Internet Protocol, Version 6 (IPv6) Specification." ftp://ftp.isi.edu/in-notes/rfc1883.txt, 1995.

[21] Deering, Stephen E. and Hinden, Robert M. "RFC2460: Internet Protocol, Version 6 (IPv6) Specification." ftp://ftp.isi.edu/in-notes/rfc2460.txt, 1-12-1998.

[22] Deutsch, L. Peter "RFC 1951: DEFLATE Compressed Data Format Specification version 1.3." ftp://ftp.isi.edu/in-notes/rfc1951.txt, 1-5-1996.

[23] Dippel, Oliver "Server Usage: FB Mathematik u. Informatik WWW-Cache Server." http://www.mathematik.uni-marburg.de/~dippel/system/www-cache/g-index.html, 9-9-1999.

[24] Fielding, Roy T., Gettys, Jim, Mogul, Jeffrey C., Nielsen, Henrik Frystyk, and Berners-Lee, Tim "RFC2068: Hypertext Transfer Protocol -- HTTP/1.1." ftp://ftp.isi.edu/in-notes/rfc2068.txt, 1997.

[25] FreeBSD Core Team "The FreeBSD Project." http://www.freebsd.org/, 1992.

[26] Freed, Ned and Borenstein, Nathaniel S. "RFC1341: MIME (Multipurpose Internet Mail Extensions)." ftp://ftp.isi.edu/in-notes/rfc2045.txt, 1-6-1992.

[27] Freed, Ned and Borenstein, Nathaniel S. "RFC2045: MIME (Multipurpose Internet Mail Extensions) Part One: Format of Internet Message Bodies." ftp://ftp.isi.edu/in-notes/rfc2045.txt, 1-11-1996.

[28] Freed, Ned and Borenstein, Nathaniel S. "RFC2046: MIME (Multipurpose Internet Mail Extensions) Part Two: Media Types." ftp://ftp.isi.edu/in-notes/rfc2046.txt, 1-11-1996.

[29] Freed, Ned and Borenstein, Nathaniel S. "RFC2049: MIME (Multipurpose Internet Mail Extensions) Part Five: Conformance Criteria and Examples." ftp://ftp.isi.edu/in-notes/rfc2049.txt, 1-11-1996.

[30] Freed, Ned, Klensin, John, and Postel, Jon "RFC2048: MIME (Multipurpose Internet Mail Extensions) Part Four: Registration Procedures." ftp://ftp.isi.edu/in-notes/rfc2048.txt, 1-11-1996.

[31] Friend, Robert and Monsour, Robert "RFC2395: IP Payload Compression Using LZS." ftp://ftp.isi.edu/in-notes/rfc2395.txt, 1-12-1998.

[32] Friend, Robert and Simpson, William Allen "RFC 1974: PPP Stac LZS Compression Protocol." ftp://ftp.isi.edu/in-notes/rfc1974.txt, 1-8-1996.

[33] Gailly, Jean-loup and Adler, Mark "GZIP." http://www.gzip.org/, 14-6-2000.

[34] Girardin, Luc "Cyberspace geography visualization." http://heiwww.unige.ch/girardin/cgv/report/, 15-10-1995.

[35] Görke, Winfried "Proseminar Redundanz (fehlerkorrigierende und datenkomprimierende Codes)." http://goethe.ira.uka.de/redundanz/, 11-2-1999.

[36] Gleeson, Martin "The University of Melbourne Caching Proxy Server." http://www.unimelb.edu.au/proxy-usage/, 9-9-1999.

[37] Grimm, Christian, Vöckler, Jens-Sönke, and Pralle, Helmut "Konzeption einer Cache-Server-Infrastruktur auf dem Wissenschaftsnetz, Abschlußbericht Projekt Teil I." http://www.cache.dfn.de/DFN-Cache/Veroeffentlichungen/Abschlussbericht_I.pdf, 1-8-1998.

[38] Gumm, Heinz-Peter and Manfred Sommer. Einführung in die Informatik. München: Oldenbourg, 1998.

[39] Hinden, Bop "IP Next Generation (IPng)." http://playground.sun.com/pub/ipng/html/ipng-main.html, 20-12-1999.

[40] Hooper, Tracey "DESIRE: Development of a European Service for Information on Research and Education." http://www.desire.org/, 1-6-1999.

[41] HRZ Uni Marburg "Entwicklung der Internet-Domain Uni-Marburg.DE." http://www.uni-marburg.de/hrz/komm/domainza.html, 18-3-1999.

[42] Huffman, D. A. "A Method for the Construction of Minimum-Redundancy Codes." Proceedings of the Institute of Radio Engineers 1 Jan, 1952: 1098-1101.

[43] I.T.NetworkX "TCPSHOW Analysis tool for tcpdump." ftp://ftp.networx.ie/pub/unix/net/tcpshow/, 18-3-1999.

[44] IETF IPng "6BONE." http://www.6bone.net/, 6-3-2000.

[45] Information Sciences Institute "RFC793: TRANSMISSION CONTROL PROTOCOL PROTOCOL SPECIFICATION." ftp://ftp.isi.edu/in-notes/rfc793.txt, 1981.

[46] IRTF-RD "Harvest Information Discovery and Access Project." http://www.tardis.ed.ac.uk/harvest/, 1995.

[47] ISC "Internet Domain Survey, January 2000." http://www.isc.org/ds/WWW-200001/report.html, 18-3-1999.

[48] ISC "About ISC." http://www.isc.org/ISC/, 1-1-2000.

[49] ISC "Internet Domain Survey Background." http://www.isc.org/ds/new-survey.html, 1-1-2000.

[50] Jentschke, Tobias "WWW-Server Statistik." http://www.mathematik.uni-marburg.de/~jentschk/servstat/servstat.htm, 18-3-1999.

[51] Kohonen, Teuvo "WEBSOM - Self-Organizing Maps for Internet Exploration." http://websom.hut.fi/websom/, 15-10-1995.

[52] Krol, Ed and Hoffman, Ellen "RFC1462: FYI on "What is the Internet?"." ftp://ftp.isi.edu/in-notes/rfc1462.txt, 1993.

[53] Lafon, Yves and Gettys, Jim "HTTP Activity Statement." http://www.w3.org/Protocols/Activity.html, 27-10-1999.

[54] Lancaster University Computing Department "UK IPv6 Resource Centre." http://www.cs-ipv6.lancs.ac.uk/, 12-3-2000.

[55] Lawrence Berkeley National Laboratory "LBNL's Network Research Group." http://www-nrg.ee.lbl.gov/nrg.html, 18-3-1999.

[56] Lempel, A. and J. Ziv. "A Universal Algorithm for Sequential Data Compression." IEEE Transactions On Information Theory 1 May, 1977a

[57] Lempel, A. and J. Ziv. "compression of individual sequences via variable rate coding." IEEE Transactions On Information Theory 1 Sep, 1978b

[58] Luotonen, Ari, Nielsen, Henrik Frystyk, and Berners-Lee, Tim "W3C: CERN httpd." http://www.w3.org/Daemon/, 28-9-1999.

[59] Luotonen, Ari, Nielsen, Henrik Frystyk, and Berners-Lee, Tim "W3C: Jigsaw - The W3C's Web Server." http://www.w3.org/Jigsaw/, 28-9-1999.

[60] Markatos, Evangelos P. and Chronaki, Catherine E. "A Top-10 Approach to Prefetching the Web." http://archvlsi.ics.forth.gr/html_papers/INET98_prefetch/paper.html, 1-5-1998.

[61] Maughan, Douglas, Schneider, Mark, Schertler, Mark, and Turner, Jeff "RFC2408: Internet Security Association and Key Management Protocol (ISAKMP)." ftp://ftp.isi.edu/in-notes/rfc2408.txt, 1-11-1998.

[62] Melve, Ingrid "DESIRE: Review of Caching Technologies & New Opportunities." http://www.desire.org/html/research/deliverables/D4.1/, 1-6-1999.

[63] Merit Network "The NSFNET Backbone Project." http://www.merit.edu/merit/archive/nsfnet/, 19-3-1999.

[64] Messer, James "Ethernet Network Questions and Answers." ftp://rtfm.mit.edu/pub/usenet-by-hierarchy/comp/dcom/lans/ethernet/comp.dcom.lans.ethernet_FAQ, 2-4-1999.

[65] Metcalfe, Robert M. "A drawing of the first Ethernet system by Bob Metcalfe." http://www.ots.utexas.edu/ethernet/, 1976.

[66] Metcalfe, Robert M., et al. Ethernet Patent: Multipoint data communication system with collision detection. US4063220. Dec 13, 1977.

[67] MIDS "Internet Resource Discovery Services on NSFNET by Bytes." http://www.mids.org/growth/internet/html/bportsl.html, 18-3-1999.

[68] MIDS "Matrix Information and Directory Services, Inc. (MIDS)." http://www.mids.org/who.html, 18-3-1999.

[69] Miller, Trudy "RFC938: Internet Reliable Transaction Protocol." ftp://ftp.isi.edu/in-notes/rfc938.txt, 1985.

[70] Mockapetris, P. "RFC1034: DOMAIN NAMES - CONCEPTS AND FACILITIES." ftp://ftp.isi.edu/in-notes/rfc1034.txt, 1987.

[71] Mogul, Jeffrey C. "The Case for Persistent-Connection HTTP." http://www.research.digital.com/wrl/techreports/abstracts/95.4.html, 1995.

[72] Moore, Keith "RFC2047: MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text." ftp://ftp.isi.edu/in-notes/rfc2047.txt, 1-11-1996.

[73] NCSA "Visualization Study of the NSFNET." http://www.ncsa.uiuc.edu/SCMS/DigLib/text/technology/Visualization-Study-NSFNET-Cox.html, 14-7-1992.

[74] Network Engineering "Protocol Stacks in Relationship to the OSI Model." http://www.neteng.com/osimodel.htm, 29-7-1997.

[75] Network Wizards "About Network Wizards." http://www.nw.com/nw/WWW/whoarewe.html, 18-3-1999.

[76] Network Wizards "Domain Survey Definitions." http://www.nw.com/zone/WWW/defs.html, 18-3-1999.

[77] Nielsen, Henrik Frystyk, Gettys, Jim, Baird-Smith, Anselm, Prud'hommeaux, Eric, Lie, Hakon Wium, and Lilley, Chris "Network Performance Effects of HTTP/1.1, CSS1, and PNG." http://www.w3.org/Protocols/HTTP/Performance/Pipeline.html, 24-6-1997.

[78] Nielsen, Henrik Frystyk and Prud'hommeaux, Eric "Simple Test of Compressing HTML Using ZLib." http://www.w3.org/Protocols/HTTP/Performance/Compression/HTMLCanon.html, 9-8-1997.

[79] Pereira, Roy "RFC2394: IP Payload Compression Using DEFLATE." ftp://ftp.isi.edu/in-notes/rfc2394.txt, 1-12-1998.

[80] Piper, Derell "RFC2407: The Internet IP Security Domain of Interpretation for ISAKMP." ftp://ftp.isi.edu/in-notes/rfc2407.txt, 1-11-1998.

[81] Postel, Jon "RFC768: User Datagram Protocol." ftp://ftp.isi.edu/in-notes/rfc768.txt, 1980.

[82] Postel, Jon "RFC801: NCP/TCP TRANSITION PLAN." ftp://ftp.isi.edu/in-notes/rfc801.txt, 1981.

[83] Pralle, Helmut "DFN-Cache-Service." http://www.cache.dfn.de/, 1-5-1998.

[84] Quick Connect "The Genius behind HyperCard: Bill Atkinson." http://www.savetz.com/ku/ku/quick_genius_behind_hypercard_bill_atkinson_the_november_1987.html, 1-11-1987.

[85] Ragget, Dave and et.a "XHTML™ 1.0: The Extensible HyperText Markup Language." http://www.w3.org/TR/xhtml1/, 26-1-2000.

[86] Ragget, Dave, Jacobs, Ian, and Ishikawa, Masayasu "HyperText Markup Language Home-page." http://www.w3.org/MarkUp/, 15-3-2000.

[87] Ragget, Dave, Le Hors, Arnaud, and Jacobs, Ian "HTML 4.01 Specification." http://www.w3.org/TR/html401, 24-12-1999.

[88] Reynolds, Joyce K. and Postel, Jon "RFC1340: ASSIGNED NUMBERS." ftp://ftp.isi.edu/in-notes/rfc1340.txt, 1992.

[89] Riggert, Wolfgang. ATM - Technik und Einfürung. bhv Verlags GmbH, 1998.

[90] RIPE NCC "RIPE Region Hostcount - Monthly hostcount progression: Table." http://www.ripe.net/ripencc/pub-services/stats/hostcount.html#overview, 18-3-1999.

[91] RobT "RobTex Viking Server." http://www.robtex.com/viking/, 26-8-2000.

[92] Salomon, David. Data Compression: The Complete Reference. 1997.

[93] Salus, Peter H. "20 Jahre und kein bißchen weise." iX 95 A.D.: 148

[94] Seward, Julian "bzip2 and libbzip2." ftp://sourceware.cygnus.com/pub/bzip2/docs/manual_toc.html, 31-3-2000.

[95] Shacham, Abraham, Monsour, Robert, Pereira, Roy, and Thomas, Matt "IETF: IP Payload Compression Protocol (IPCOMP)." ftp://ftp.ietf.org/ietf/ippcp/, 1-7-1997.

[96] Shacham, Abraham, Monsour, Robert, Pereira, Roy, and Thomas, Matt "RFC2393: IP Payload Compression Protocol (IPCOMP)." ftp://ftp.isi.edu/in-notes/rfc2393.txt, 1-12-1998.

[97] Stacey, Mark "Implementation of T/TCP for Linux." http://www.csn.ul.ie/~heathclf/fyp/, 24-4-1997.

[98] Sterling, Bruce. "Short History of the Internet." THE MAGAZINE OF FANTASY AND SCIENCE FICTION 1 Feb, 1993

[99] Stevens, W. Richard. Programmieren von UNIX-Netzen. Wien: Hanser, 1992.

[100] Stevens, W. Richard. Aufwand Source HTTP-Client/Server in: TCP/IP Illustrated, Volume 3. 3 ed. Addison-Wesley Publishing Comany, Inc., 1994d.

[101] Stevens, W. Richard. T/TCP Backward Compatibility in: TCP/IP Illustrated, Volume 3. 3 ed. Addison-Wesley Publishing Comany, Inc., 1994c.

[102] Stevens, W. Richard. TCP/IP Illustrated, Volume 1. 1 ed. Addison-Wesley Publishing Comany, Inc., 1994b.

[103] Stevens, W. Richard. TCP/IP Illustrated, Volume 3. 3 ed. Addison-Wesley Publishing Comany, Inc., 1994a.

[104] Sun Microsystems "Sun Microsystems." http://www.sun.com/, 1999.

[105] Tanenbaum, Andrew S. Computer-Netzwerke. Wolframs Fachverlag, 1990.

[106] transtec AG "Netzwerke LAN." http://www.transtec.de/adb/WWW_cat.main.plp/sid=036F78E7F0202F7M?Z/D/D/R/gellan, 1999.

[107] UUNET "The UUNET U.S. Backbone." http://www.uu.net/lang.en/network/usa.html, 1999.

[108] Van Jacobson "RFC1144: Compressing TCP/IP Headers for Low-Speed Serial Links." ftp://ftp.isi.edu/in-notes/rfc1144.txt, 1-2-1990.

[109] Viagénie Inc "Freenet6." http://www.freenet6.net/, 3-3-2000.

[110] W3C "Press Release: World Wide Web Consortium Supports HTTP/1.1 Reaching IETF Draft Standard." http://www.w3.org/1999/07/HTTP-PressRelease, 7-7-1999.

[111] W3C and Nielsen, Henrik Frystyk "W3C Microscape Test Web Site." http://www.w3.org/Protocols/HTTP/Performance/microscape/, 7-7-1999.

[112] Wessels, Duane "SQUID Frequently Asked Questions: How does Squid work?" http://www.squid-cache.org/Doc/FAQ/FAQ-12.html, 20-9-1999.

[113] Wessels, Duane "Squid Web Proxy Cache." http://www.squid-cache.org/, 9-9-1999.

[114] Wessels, Duane and Claffy, Kimberly "Information Resource Cache (IRCACHE): A Distributed Testbed for National Information Provisioning." http://ircache.nlanr.net/Cache/, 14-3-1998.

[115] Whitehead, E. James Jr. and Makoto, Murata "RFC2376: XML Media Types." ftp://ftp.isi.edu/in-notes/rfc2376.txt, 1-7-1998.

Danksagung

Mein Dank gilt vor allem meinen Eltern, die mir diesen Lebensweg ermöglicht und mich stets unterstützt haben. Zudem möchte ich meinem Bruder herzlichst für die vielfältige Hilfestellungen in allen Lebenslagen danken.

Meinem Doktorvater Prof. Dr. M. Sommer danke ich für die Betreuung der Arbeit und gute Zusammenarbeit während meiner universitären Beschäftigung. Auch Dr. Gasiorowski sowie Herrn Kreile danke ich für die wertvollen Tips und Erklärungen während der langjährigen Zusammenarbeit. Schließlich möchte ich noch Prof. Seeger für die finalen Bemerkungen, sowie allen meinen Kollegen und Kolleginnen (damit soll Frau Dinklage explizit eingeschlossen sein) für die nicht nur fachlichen Gespräche und Anregungen in heiterer Runde danken.

Lebenslauf

Name, Vorname	Dippel, Oliver
Geburtsort, -datum	Marburg, 9.3.1967
Schule	Grundschule Mornshausen, August 1973 - Juni 1977 Gymnasium Elisabethschule Marburg, August 1977 - Juni 1986
Schulabschluß	Allgemeine Hochschulreife (5. Juni 1986)
Hochschule	Diplom Physik Philipps-Universität Marburg, 13 Sem., Aug. 1986 - Mai 1993
Hochschulabschluß	Diplom Physiker
Beschäftigungsverhältnisse	Wissenschaftliche Hilfskraft Philipps-Universität Marburg, Juni 1993 - Aug. 1995 Wissenschaftlicher Mitarbeiter Philipps-Universität Marburg, Sept. 1995 - Aug. 2000

Erklärung

Ich versichere, daß ich meine Dissertation

Quantitative Untersuchungen im Internet - Vorschläge zur Optimierung von Internet-Protokollen

selbstständig ohne unerlaubte Hilfe angefertigt und mich dabei keiner anderen als der von mir ausdrücklich bezeichneten Quellen und Hilfen bedient habe.

Die Dissertation wurde in der jetzigen oder einer ähnlichen Form noch bei keiner anderen Hochschule eingereicht und hat noch keinen sonstigen Prüfungszwecken gedient.

_____ _____

(Ort, Datum) (Unterschrift mit Vor- und Zuname)