

Universität Leipzig

Fakultät für Mathematik und Informatik

Institut für Informatik

Der Asynchronous Transfer Mode im LAN,
Performanceuntersuchungen und Vergleich mit Ethernet, Fast
Ethernet und Token Ring

Diplomarbeit

Leipzig, April 2001

vorgelegt von
André Staehler

1	EINLEITUNG.....	5
2	KOMMUNIKATION IN LOKALEN NETZWERKEN	7
2.1	Begriff LAN.....	7
2.2	Das OSI-Referenzmodell der ISO	8
2.2.1	Aufbau	9
2.2.1.1	Schicht 1 – Bitübertragungsschicht (Physical Layer)	9
2.2.1.2	Schicht 2 – Sicherungsschicht (Data Link Layer).....	9
2.2.1.3	Schicht 3 – Vermittlungsschicht (Network Layer)	10
2.2.1.4	Schicht 4 – Transportschicht (Transport Layer)	10
2.2.1.5	Schichten 5 bis 7 - Verbindungs-, Darstellungs- und Verarbeitungsschicht (Session, Presentation, Application Layer)	11
2.2.2	Datenübertragung nach dem OSI-Referenzmodell.....	11
2.3	LAN-Technologien.....	12
2.3.1	Ethernet.....	12
2.3.1.1	Verkabelungstypen für Ethernet	13
2.3.1.2	Frameformat nach IEEE 802.3	14
2.3.1.3	Leistungssteigerung bei Ethernet.....	15
2.3.1.4	Weiterentwicklungen Fast und Gigabit Ethernet	15
2.3.2	Token Ring	17
3	DER ASYNCHRONE TRANSFER MODUS.....	19
3.1	Referenzmodell	19
3.1.1	Die Physische Schicht	20
3.1.2	Die ATM-Schicht.....	21
3.1.3	Die ATM-Anpassungsschicht (AAL ATM Adaption Layer)	22
3.1.4	Die höheren Schichten	22
3.2	ATM im LAN	23
3.2.1	LLC-Einkapselung	23
3.2.2	Classical IP over ATM (CLIP)	25
3.2.3	LAN Emulation (LANE).....	26
3.2.3.1	Funktionseinheiten der LANE.....	27
4	LEISTUNGSMESSUNG.....	30
4.1	TCP	30
4.2	Die Testsoftware	31
4.2.1	Implementation	31
4.2.1.1	Die Socket Schnittstelle.....	31
4.2.2	Hardware.....	35
4.2.3	Vorbetrachtungen.....	36
4.2.3.1	Ethnet mit 10 Mbps	37

4.2.3.2	Ethernet mit 100 Mbps (100BaseTX)	39
4.2.3.3	ATM mit 25 Mbps.....	39
4.2.3.4	ATM mit 155 Mbps.....	41
5	ZUSAMMENFASSUNG UND AUSBLICK	42
6	VERZEICHNIS DER ABKÜRZUNGEN.....	43
7	LITERATURVERZEICHNIS	45
8	VERZEICHNIS DER ANLAGEN	47
8.1	Verbindungskabel zwischen den Ethernet- NICs	48
8.2	Verbindungskabel zw. IBM8282 und TurboWays 25 NIC	49
8.3	Quelltext der Haupt- Threads von NTester	51
8.4	Inhalt der beiliegenden CD-R.....	58
9	ERKLÄRUNG	59

1 Einleitung

In der Vergangenheit konnte man die Kommunikation in Netzwerken klar unterscheiden. Es gab jeweils eindeutig getrennte Netze zum Austausch von Sprache (wie das Telefonnetz) und Daten (z.B. Datex-P). Die Deregulierung des Telekommunikationsmarktes in den zurückliegenden Jahren leitete die fortschreitende Konvergenz von Sprach- und Datenübertragung ein. Energieversorger, Bahn- und andere Unternehmen, die traditionell über gut ausgebaute und weit reichende Kommunikationsnetze verfügen, drängen nun in den klassischen Telekommunikationsmarkt indem sie die verschiedensten Sprach- und Datenübertragungsdienste über ihre Infrastrukturen anbieten. Das führt perspektivisch dazu, dass es auch auf der Seite des Endkunden in Zukunft keine separaten Netze für die unterschiedlichen Dienste mehr geben wird.

In der Entwicklung der Computervernetzung lassen sich derzeit zwei klare Tendenzen erkennen. Das ist einerseits das quantitative Wachstum an Kommunikationsteilnehmern sowie die qualitative Entwicklung, womit das Entstehen immer neuer und leistungshungriger Dienste gemeint ist. Die Anzahl an vernetzten Computern ist in den vergangenen Jahren explosionsartig angestiegen. Als Beispiel sei die Entwicklung des Internet herangezogen (Abbildung 1). Die Zahlen beziehen sich auf die permanent an das Internet angeschlossenen Hosts im Zuständigkeitsbereich des RIPE¹.

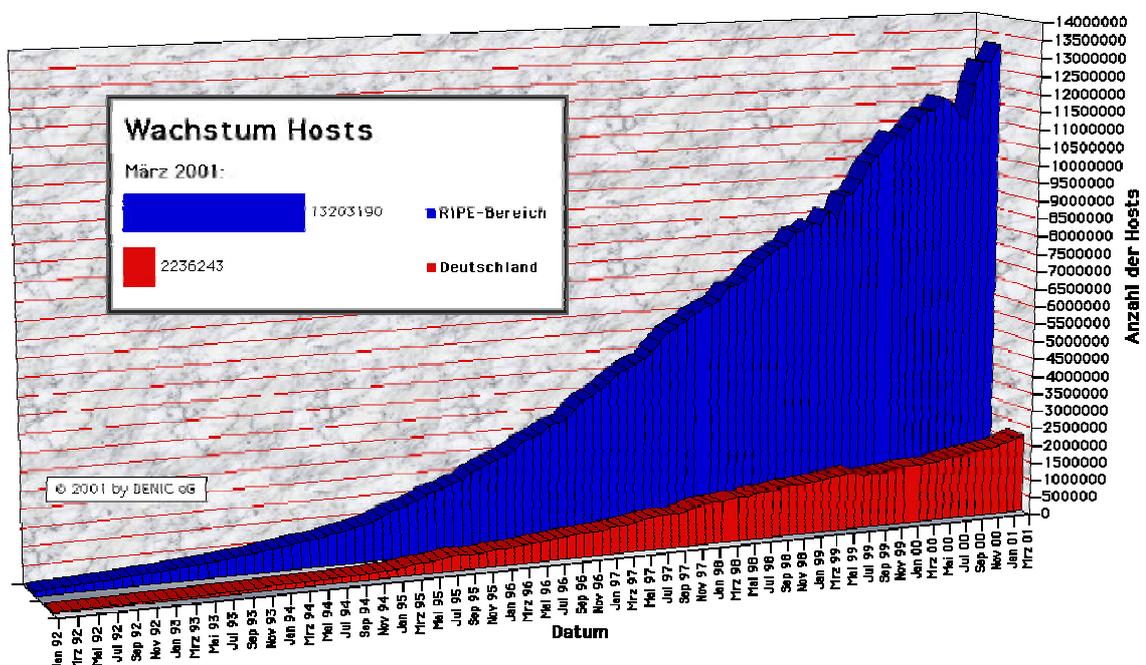


Abbildung 1 Anzahl der in Europa registrierten Internethosts.

¹ RIPE: Réseaux IP Européens, u.a. für die Vergabe von IP-Adressen in Europa zuständig.

Der zweite Aspekt, die qualitative Entwicklung, geht einher mit der enormen Leistungsentwicklung und dem Preisverfall der Personalcomputer in den letzten Jahren. So sind Anwendungsgebiete wie die computergestützte Telefonie (CTI²) und Audio- und Videobearbeitung erst mit entsprechend leistungsfähigen PCs möglich. Als Beispiel dafür sei die aktuelle Entwicklung im Bereich der Hörfunk- und Fernsehsender angeführt. Moderne Produktionshäuser wie der MDR arbeiten heute schon zum großen Teil mit PCs, Medienservern und Computernetzwerken zum Erstellen, Bearbeiten und Ausspielen ihrer Beiträge. Dafür sind sehr leistungsfähige und sichere Technologien erforderlich, die herkömmliche Netze nicht bieten können.

Eine solche Technologie stellt der Asynchrone Transfer Modus (ATM³) dar, der Hauptgegenstand dieser Arbeit ist. Es soll untersucht werden, in wie weit sich das auf dem Gebiet der Weitverkehrsnetze etablierte ATM für den Einsatz im LAN eignet.

Im folgenden Kapitel werden die Grundlagen für die Kommunikation in LANs erarbeitet. Das klassische Ethernet, seine Nachfolger Fast Ethernet und Gigabit Ethernet sowie Token Ring werden mit ihren Grundprinzipien dargestellt.

Kapitel 3 stellt den Asynchronen Transfer Modus ATM vor, erläutert seinen Ursprung und geht im Detail auf seinen Aufbau und das Referenzmodell ein. Dabei wird im Besonderen erläutert, welche Möglichkeiten es gibt, ATM im LAN einzusetzen.

Das vierte Kapitel umfasst die Beschreibung des Testprogramms und seiner Implementation sowie die Leistungsmessung im Modellnetzwerk und die Auswertung der dabei gewonnenen Ergebnisse.

Gegenstand des letzten Kapitels ist eine abschließende Gegenüberstellung von ATM und den klassischen LAN-Technologien.

² CTI: Computer-Telephony Integration, Verschmelzung von PC- und Telefonieanwendungen.

³ ATM: Asynchronous Transfer Mode.

2 Kommunikation in lokalen Netzwerken

2.1 Begriff LAN

Lokale Netzwerke dienen der Verbindung von Geräten der Rechentechnik (z.B. PCs, Drucker u.a.) mit dem Ziel, Daten und Informationen auszutauschen und Ressourcen gemeinsam nutzbar zu machen. Das Hauptmerkmal eines LAN ist – wie der Name schon sagt – seine räumliche Ausdehnung. Diese ist meist auf nur ein Gebäudeteil bzw. Gebäude begrenzt. Darüber hinaus lassen sie sich anhand ihrer Topologie und der verwendeten Übertragungstechnik von anderen Netzarten unterscheiden [10].

Bei den im Folgenden betrachteten LAN-Arten Ethernet und Token Ring handelt es sich um Broadcastnetze, das heißt eine auf das Übertragungsmedium gegebene Information kann prinzipiell von allen angeschlossenen Endgeräten empfangen werden. Die Endgeräte teilen sich das Medium, weshalb Verfahren notwendig sind, die die Zuteilung des Mediums an sendewillige Teilnehmer regeln. Diese Zuteilungsmechanismen sind Bestandteil des jeweiligen LAN-Standards und werden in den entsprechenden Kapiteln erläutert.

Um die hohe Komplexität der Kommunikation in modernen Netzwerken besser beherrschen zu können, wurde sie mit Hilfe verschiedener Modelle strukturiert. Basis dieser Modelle bilden einzelne Ebenen oder Schichten (Layer), welche sich in Bezeichnung, Inhalt und Funktion von Modell zu Modell unterscheiden. Das Grundprinzip besteht jedoch immer darin, dass jede der übereinander angeordneten Schichten (n) der nächsthöheren Schicht ($n+1$) über eine definierte Schnittstelle Dienste zur Verfügung stellt (Abbildung 2).

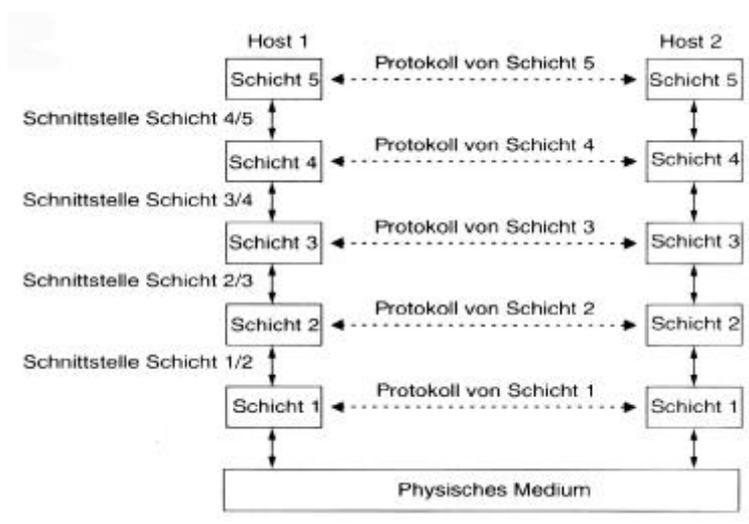


Abbildung 2 Schichten, Schnittstellen und Protokolle [10]

Zur Erfüllung ihrer Funktion greift sie auf die Dienste der unter ihr angeordneten Schicht ($n-1$) zurück. Dabei werden die Details der schichtinternen Funktionen nach außen hin verborgen. Die Schichten finden sich in der Realisierung der Netzwerkfunktionen (Hardware und/oder Software) der einzelnen Endgeräte wieder. Zum Grundprinzip gehört weiterhin, dass eine Schicht n eines Systems ausschließlich mit der korrespondierenden Schicht n des Kommunikationspartners in Verbindung tritt. Dazu benutzen sie das Protokoll der Schicht n .

Ein wichtiges Referenzmodell sei im folgenden Abschnitt etwas näher erläutert, da es die Grundlagen der Kommunikation in Netzwerken gut veranschaulicht.

2.2 Das OSI-Referenzmodell der ISO

In einem LAN sind im Normalfall offene Kommunikationssysteme miteinander verbunden. Diese sind dadurch charakterisiert, dass die Schnittstellen zu den angeschlossenen Endgeräten herstellerunabhängig und weitestgehend anwendungsneutral sind. Neben der Festlegung aller Schnittstelleneigenschaften bedeutet dies die Veröffentlichung aller Festlegungen und die Patentfreiheit der Normungen.

Offene Kommunikationssysteme sind die Voraussetzung für die Bereitstellung von Kommunikationsmöglichkeiten zwischen Geräten verschiedener Hersteller und Bauart und damit zur weltweiten Verbreitung eines Kommunikationssystems. In einem offenen Kommunikationssystem ist die Identifikation eines Endgerätes innerhalb des Netzes eindeutig. Die Zahl anschließbarer Endgeräte hat keine systembedingte Grenze.

Anfang der 80er Jahre stellte die ISO⁴ ein Modell zur Standardisierung solcher offenen Kommunikationssysteme vor, das Modell der Open Systems Interconnection (OSI). Dieses Modell hat Normungscharakter, darüber hinaus veranschaulicht es den Ablauf der Kommunikation in einem Netzwerk auf sehr verständliche Weise.

⁴ ISO: International Organisation for Standardization

2.2.1 Aufbau

Abbildung 3 zeigt das OSI-Referenzmodell in seinem Aufbau. Es besteht aus sieben Schichten, in denen die normierten Funktionen und Protokolle hierarchisch gegliedert sind.

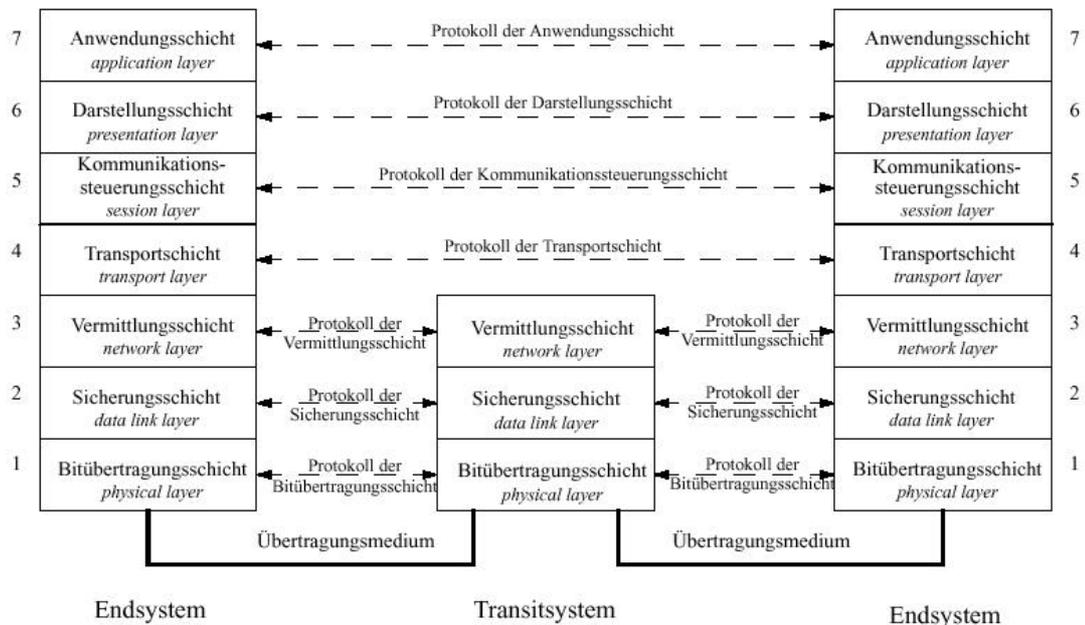


Abbildung 3 Das OSI-Referenzmodell der ISO

2.2.1.1 Schicht 1 – Bitübertragungsschicht (Physical Layer)

Die Bitübertragungsschicht beinhaltet die physikalische Übertragung der einzelnen, „rohen“ Bits über das Medium. Hierbei werden die elektrischen und mechanischen Eigenschaften für die Schnittstelle zwischen Endsystem und Übertragungsmedium festgelegt. Auch die Regeln nach denen auf das Übertragungsmedium zugegriffen wird, sowie dessen physikalische Eigenschaften sind Bestandteil der Bitübertragungsschicht.

Für die Schicht 2 stellt sie folgende Dienste zur Verfügung:

- Ungesicherte Systemverbindungen zwischen Schicht-2-Instanzen im End- und Transitsystem.
- Völlig transparente Übertragung der Bits, so wie sie der Schicht übergeben werden.

2.2.1.2 Schicht 2 – Sicherungsschicht (Data Link Layer)

Die Aufgaben der Sicherungsschicht bestehen in der Verwaltung des ankommenden bzw. abgehenden Datenstroms. Dieser wird dazu in Übertragungsrahmen (Frames) oder auch Pakete eingeteilt, die vom Sender sequentiell an den Empfänger übertragen werden. Eine Steuerung des Informationsflusses ist dann möglich, wenn Quittierungsmechanismen im Schicht-2-Protokoll vorgesehen sind, wodurch zum Beispiel eine Datenüberflutung des Empfängers durch

den Sender verhindert werden kann. Des Weiteren besitzt die Sicherungsschicht Kontrollmechanismen mit denen sie Bitfehler erkennen und durch erneute Übertragungsanforderung – soweit möglich – beheben kann.

Dienste für Schicht 3:

- Auf- und Abbau von gesicherten Systemverbindungen.
- Übertragung von Daten der Schicht 3.
- Meldung nicht behebbarer Fehler an Schicht 3.

2.2.1.3 Schicht 3 – Vermittlungsschicht (Network Layer)

Die Vermittlungsschicht ist für die Ausführung sämtlicher vermittlungstechnischer Funktionen zuständig. Dazu zählen der Verbindungsauf- und -abbau sowie die Wegewahl. Letztere beinhaltet die Entscheidung zwischen mehreren möglichen Wegen aufgrund vorher festgelegter Kriterien (z.B. Geschwindigkeit oder Kosten). Eine genaue Kenntnis der Netzstruktur ist dazu unbedingt erforderlich.

Während die Sicherungsschicht nur für die Übertragung zwischen zwei unmittelbar benachbarten Systemen (z.B. End- und Transitsystem) zuständig war, überwacht die Vermittlungsschicht die gesamte Verbindung vom Sender zum Empfänger („Ende-zu-Ende“). Eine Endsystemverbindung besteht also aus der Kopplung mehrerer gesicherter Systemverbindungen der Schicht 2.

Dienste für Schicht 4:

- Auf- und Abbau von Endsystemverbindungen zweier Schicht-4-Instanzen.
- Übertragung der Schicht-4-Daten.
- Vereinbarung möglicher Dienstgüte- oder Leistungsmerkmale.
- Meldung nicht behebbarer Fehler an Schicht 4.
- Rücksetzen einer Endsystemverbindung im Fehlerfall.

2.2.1.4 Schicht 4 – Transportschicht (Transport Layer)

Die Transportschicht bildet die Grenze zwischen den unteren, transportorientierten Schichten, die sich in erster Linie mit Netzwerkaufgaben befassen, und den anwendungsorientierten Schichten. Das der Übertragung zugrunde liegende Netz spielt dabei keine Rolle mehr und ist für die Schicht 4 transparent, sie kennt nur die Adressen des Quell- und Zielsystems.

Dienste für Schicht 5:

- Auf- und Abbau von Transportverbindungen zwischen zwei durch ihre Transportadressen repräsentierten Dienste.
- Bereitstellung qualitativ unterschiedlicher Verbindungen bezüglich Durchsatz, Verzögerung, Restfehlerrate, Verfügbarkeit und Kosten.
- Transportverbindungen unterschiedlicher Priorität.

2.2.1.5 Schichten 5 bis 7 - Verbindungs-, Darstellungs- und Verarbeitungsschicht (Session, Presentation, Application Layer)

Diese drei Schichten beinhalten anwendungsorientierte Dienste und Protokolle, weshalb sie hier zusammengefasst erwähnt werden sollen. So enthalten sie z.B. Dialogsteuerung, Datenkodierung und -darstellung, Terminalemulationen u.s.w.

2.2.2 Datenübertragung nach dem OSI-Referenzmodell

Zwei Prozesse in den Endgeräten wollen miteinander kommunizieren (Abbildung 4). Der Sendeprozess übergibt seine Daten – die Nutzdaten – an die oberste Schicht. Diese reicht sie bearbeitet und ergänzt um einen eventuell nötigen schicht- bzw. protokollspezifischen Vorspann (Header) an die nächstuntere Schicht weiter. Auch diese stellt den übergebenen Daten nach der Bearbeitung wieder falls erforderlich einen Header voran und reicht sie „nach unten“ weiter. Dabei kann eine Schicht die ihr übergebenen Daten nicht in Header und Nutzdaten unterscheiden. Erreichen die Daten die Bitübertragungsschicht im Sendesystem, werden sie zum Empfängersystem übertragen, wo sie die Schichten in der umgekehrten Reihenfolge, also von unten nach oben, durchlaufen. In jeder Schicht werden dabei die Protokollheader nacheinander wieder entfernt, sodass dem Empfängerprozess schließlich die ursprünglichen Nutzdaten übergeben werden.

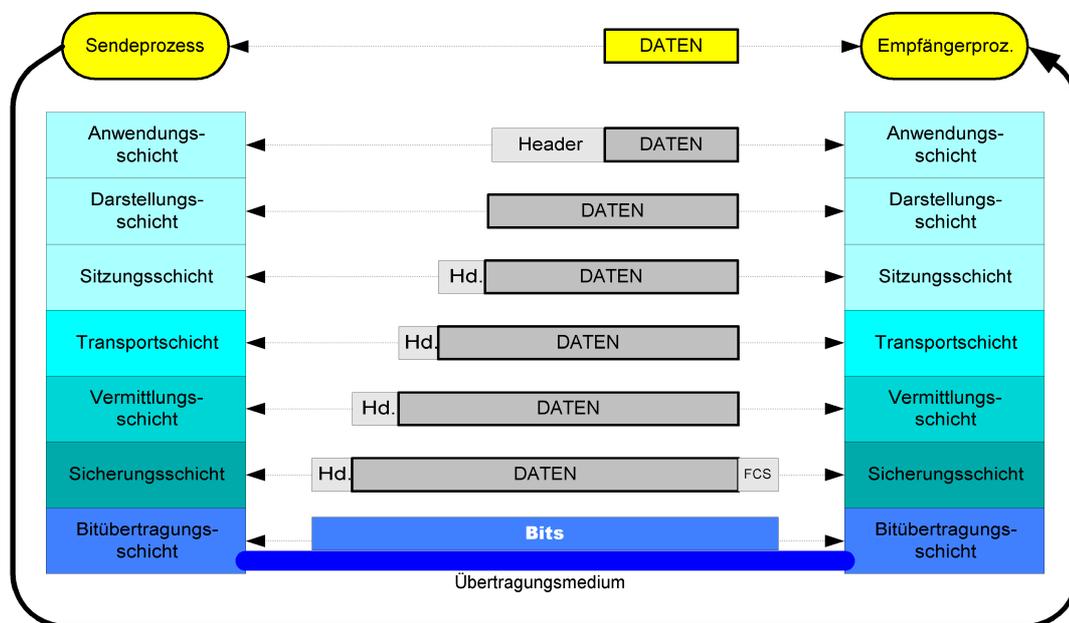


Abbildung 4 Datenübertragung nach ISO-OSI

2.3 LAN-Technologien

Lokale Netzwerke basieren in der Regel auf dem Prinzip des gemeinsam genutzten Übertragungsmediums („shared medium“). Eine auf diesem Medium übertragene Information kann theoretisch von jedem angeschlossenen System gelesen werden unabhängig davon, ob es an der Übertragung beteiligt ist oder nicht. Daraus resultiert die Notwendigkeit, in den zu übertragenden Daten Adressinformationen unter zu bringen. Anhand dieser Adressinformationen entscheiden die Systeme im Netzwerk, ob die Daten für sie bestimmt sind oder nicht. Dieses Prinzip wird sowohl von Ethernet und seinen Nachfolgern als auch von Token Ring genutzt.

Shared Media LANs nutzen zur Datenübertragung die Paketvermittlung. Dabei werden die Daten auf Schicht 2 des OSI-Referenzmodells in Rahmen („Frames“) eingebettet, deren Format von der verwendeten Technologie abhängt.

2.3.1 Ethernet

Der weitaus größte Teil heute installierter LANs arbeitet auf Basis der Ethernet-Technologie, welche als Standard 802.3 von der IEEE⁵ verankert wurde. Dieser Standard beschreibt lokale Netzwerke mit konkurrierendem Zugriff der Endsysteme auf ein gemeinsames Übertragungsmedium. Die Zuteilung des Medium auf sendewillige Stationen wird mit dem CSMA/CD⁶-Verfahren geregelt:

Jede Station, die Daten senden möchte, muss vor Sendebeginn das Medium abhören („Listen before Talk“). Wenn sie dabei feststellt, dass das Medium bereits belegt ist (Carrier Sense), muss sie warten bis es frei wird, andernfalls kann sie sofort beginnen zu senden. Auch während des Sendens überwacht sie den Übertragungskanal („Listen while Talk“). Durch die begrenzte Ausbreitungsgeschwindigkeit des Signals auf dem Medium kann es passieren, dass zwei genügend weit auseinander liegende Stationen zu einem Zeitpunkt ein freies Medium erkennen und gleichzeitig beginnen zu senden. Es kommt zu einer Kollision, beide Sendesignale überlagern sich und die Datenrahmen auf dem Medium werden unbrauchbar. Die Stationen erkennen die Kollision, weil die auf dem Medium befindlichen Daten nicht mit den gesendeten überein stimmen. Beide brechen die Übertragung ab und wiederholen nach einer zufälligen Wartezeit den Sendeversuch bis zu zehnmal, bevor sie mit einem Übertragungsfehler abbrechen. Dieser Mechanismus wird als binärer exponentieller Backoff-Algorithmus ausführlich in [10] behandelt.

⁵ IEEE: Institute of Electrical and Electronics Engineers

⁶ CSMA/CD Carrier Sense Multiple Access with Collision Detection

Bei CSMA/CD handelt es sich um ein nicht deterministisches Zugriffsverfahren, da nicht garantiert ist, dass und nach welcher Zeit eine Datenübertragung erfolgt. Die Effizienz eines solchen Netzes hängt neben der Bandbreite (bei Ethernet sind es 10Mbps⁷) und der Rahmengröße der Daten auch von der Anzahl der an dem Übertragungsmedium angeschlossenen Stationen ab (Abbildung 5). In der Praxis liegt der maximale Durchsatz von Ethernet durch die Kollisionen bei 40 bis 50% des Nenndurchsatzes.

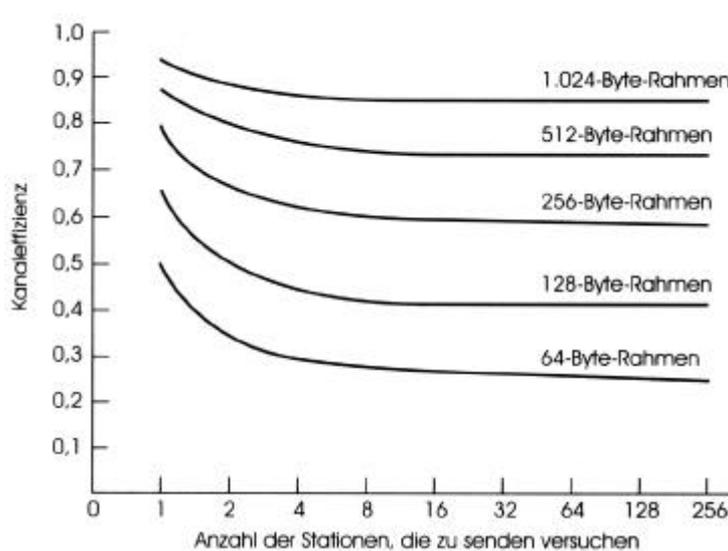


Abbildung 5 Effizienz von Standard Ethernet

2.3.1.1 Verkabelungstypen für Ethernet

Für das 10Mbps- Ethernet sind vier verschiedene Kabelarten definiert, deren Eigenschaften u.a. die räumliche Ausdehnung und die maximale Anzahl anschließbarer Stationen in einem Ethernet-Netzwerk bestimmen (Tabelle 1).

Ethernet-Standard	Kabel	max. Segmentlänge	Stationen pro Segment (max.)
10Base5	Koaxial 50Ohm	500 m	100
10Base2	Koaxial 50 Ohm dünn	~200 m	30
10BaseT	Twisted Pair	100 m	1024
10BaseF	LWL	2 km	1024

Tabelle 1 Kabeltypen für Standard Ethernet

⁷ Mbps: Megabits per second, $1 \cdot 10^6$ Bits pro Sekunde

Das ursprüngliche Koaxialkabel wird heute für neu zu errichtende Netzwerke nicht mehr verwendet. Es veranschaulicht aber das Prinzip des Shared medium sehr gut, weil die Stationen auch physisch an einen Kabelstrang angeschlossen werden. Es hat aber Nachteile wie sehr schlechte Verlegbarkeit (beim „Yellow Cable“ des 10Base5) bzw. hohe Störanfälligkeit im Betrieb (beim „Thin Coax“ des 10Base2). Die Entwicklung von Ethernet über Twisted Pair Kabel wurde dadurch begünstigt, dass in modernen Gebäuden eine vorhandene Infrastruktur in Form einer Telefonverkabelung nach Kategorie 3 oder 5 besonders preisgünstig genutzt werden kann. Twisted Pair Kabel stellen physisch nur Punkt zu Punkt- Verbindungen zwischen den Geräten her, das gemeinsame Medium wird hier von einem elektrischen Verteiler, dem so genannten Hub, repräsentiert, der die Kabel sternförmig verbindet. Mit dem Einsatz von Hubs gehen eine Reihe von Vorteilen einher, angefangen bei leichter Strukturierung der Netzwerkverkabelung durch die Abbildung des logischen Busnetzes auf einen physischen Stern bzw. Baum, bis hin zur Minderung der Störanfälligkeit durch automatische Abschaltung fehlerhafter Kabelsegmente durch den Hub. Nach diesem Muster funktionieren auch die Lichtwellenleiter. Ihr Einsatz ist mit höheren Kosten für Material und Verlegung verbunden. Dafür haben sie aber auch einige Vorzüge: sie sind elektrisch störfest, weitgehend abhörsicher und wegen der höheren Übertragungsbreiten zukunftssicherer.

2.3.1.2 Frameformat nach IEEE 802.3

Der Aufbau eines Ethernetrahmens ist in Abbildung 6 dargestellt. Am Anfang steht eine Präambel, deren Kodierung es dem Empfänger ermöglicht, sich auf den Sendetakt zu synchronisieren. Das darauf folgende Oktett markiert den Beginn des Frames.

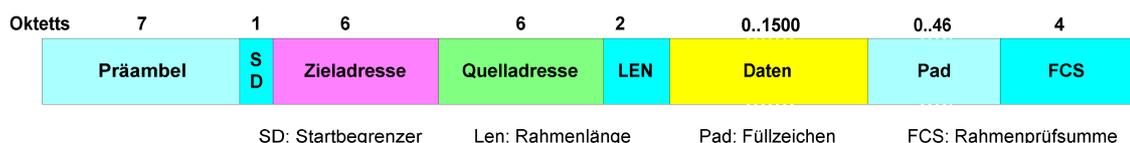


Abbildung 6 Das Ethernet- Rahmenformat

Es folgen die Adressen der Ziel- und der sendenden Station mit einer Länge von jeweils 6 Oktetts. Das MSB⁸ der Zieladresse teilt die möglichen Adressen in normale (0) und Gruppenadressen (1). Ein Frame mit einer Gruppenadresse als Ziel wird von mehreren Stationen empfangen (Multicast). Ein Sonderfall ist die Zieladresse deren Bits alle auf 1 gesetzt sind. Sie ist die Broadcastadresse, das heißt ein Frame an diese Adresse wird von allen Stationen im Netz empfangen. Das nächste Bit legt fest ob die restlichen 46 Bit eine lokale oder globale Adressen repräsentieren. Globale Adressen werden weltweit

⁸ MSB: Most Significant Bit, höchstwertiges Bit.

eindeutig von der IEEE vergeben und sind fester Bestandteil der Hardware des Netzwerkinterfaces.

Das Längenfeld enthält die Anzahl der folgenden Nutzdaten-Oktetts. Für die Kollisionserkennung innerhalb des CSMA/CD ist eine Mindestlänge des Frames von 64 Oktetts ab Ende des Startbegrenzers erforderlich, weshalb bei entsprechend wenig Nutzdaten mit Hilfe des Padfeldes auf die Mindestlänge aufgefüllt werden muss. Das letzte Feld ist die Frame Check Sequence. Sie enthält einen 32 Bit langen Hashcode über die Daten, mit dessen Hilfe Bitfehler bei der Übertragung erkannt werden können.

2.3.1.3 Leistungssteigerung bei Ethernet

Mit wachsender Anzahl an Endsystemen nimmt die Leistungsfähigkeit des Ethernet ab (siehe Abbildung 5). Genügt die Bandbreite von 10Mbps jedoch den Anforderungen, so ist ein erster Schritt zur Leistungssteigerung die Unterteilung des Netzwerks in mehrere Segmente mit Hilfe von Switches. Switches sind Vermittlungsgeräte, die –anders als Hubs– das Ethernet in Kollisionsdomänen spalten indem sie auf ISO-Schicht 2 die Adressinformationen eines Frames nutzen, um ihn direkt vom Quell- in das Zielsegment zu transportieren. Dabei gelangt er nicht in die anderen am selben Switch angeschlossenen Segmente, was dort den Verkehr verringert und so die Effizienz erhöht. Diese direkte Vermittlung beherrscht ein guter Switch an mehreren –im Idealfall allen– Portpaaren gleichzeitig und mit Leitungsgeschwindigkeit, das heißt ein Ethernetswitch mit 16 Ports könnte 8 parallele Rahmenweiterleitungen mit jeweils 10 Mbps realisieren. Als weiterer Vorteil ist zu nennen, dass der Switch auch Kollisionen und fehlerhafte Rahmen filtert. Da er für seine Funktion die MAC-Adressen auswerten und speichern muss, ist schnelle und relativ teure Hardware im Switch nötig. Dafür ist aber nur an den zentralen Stellen im Netz eine Umrüstung erforderlich, die Netzwerkadapter in den Endstationen können weiter genutzt werden.

Verfolgt man diese Strategie konsequent weiter, so sind letztlich alle Hubs eines Netzwerks durch Switches ersetzt, an deren Ports nur jeweils eine Station angeschlossen ist (Mikrosegmentierung). Bei entsprechend leistungsfähigen Switches kann so jede Station die volle Bandbreite des Netzes kollisionsfrei nutzen.

2.3.1.4 Weiterentwicklungen Fast und Gigabit Ethernet

Da aber auch die Mikrosegmentierung die Bandbreite des Ethernet nicht erhöht, wurden andere Wege gesucht, die Netzwerke leistungsfähiger zu machen. Das führte zur Entwicklung von Fast Ethernet und Gigabit Ethernet. Wegen der Verbreitung von Ethernet entschied sich der Normungsausschuss 803.2 des IEEE dafür, die Technik des Ethernet beizubehalten und nur die Datenrate zu erhöhen. Die erste Stufe wurde 1995 als Fast Ethernet 802.3u standardisiert. Fast Ethernet übernimmt vom Ethernet Paketformat, Schnittstellen, Prozeduren und Regeln, erhöht die Geschwindigkeit auf dem Medium aber um den Faktor 10 auf 100 Mbps. Die Verwendbarkeit von

Koaxialkabeln fällt bei Fast Ethernet weg, da man für die Kollisionserkennung deren maximale Länge um den Faktor 10 kürzen müsste. Folglich sind nur Twisted Pair Kabel der Kategorien 3 und 5 und Lichtwellenleiter einsetzbar. Das stellt aber keinen Nachteil dar, weil heute bei Neuerrichtung von Netzwerken bzw. Gebäuden im Allgemeinen eine strukturierte Verkabelung nach DIN EN 50173 eingesetzt wird, welche auf diesen Kabeltypen basiert [17]. Eine Umstellung eines bestehenden Ethernet auf Fast Ethernet hat damit im schlimmsten Fall einen Wechsel der Hubs, Switches, Netzwerkadapter und der Verkabelung zur Folge.

Gigabit Ethernet nach IEEE 802.3z ist die nächste Stufe der Ethernetentwicklung, da auch 100 Mbps in manchen Anwendungsfällen einen Engpass darstellen können. Auch hier handelt es sich im Wesentlichen um die Erhöhung der Mediengeschwindigkeit um den Faktor 10 auf 1000 Mbps oder 1 Gbps⁹. In Tabelle 2 sind die heute gebräuchlichen Varianten zusammengefasst. Es sei noch angemerkt, dass heute schon eine weitere Stufe, das 10 Gigabit Ethernet, in Entwicklung ist.

Ethernet-Standard	Kabel	max. Segmentlänge
100BaseT4	Twisted Pair Kat. 3	100 m
100BaseTX	Twisted Pair Kat. 5	100 m
100BaseFX	LWL	2 km
1000BaseSX	LWL MMF ¹⁰ , kurzwelliges Licht	550 m
1000BaseLX	LWL SMF ¹¹ , langwelliges Licht	5 km
1000BaseT	Twisted Pair Kat. 5, alle 4 Paare	100 m

Tabelle 2 Nachfolger des Standard Ethernet

⁹ Gbps: Gigabits per second, Gigabits pro Sekunde, 1 Gbps = 1 · 10⁹ bps

¹⁰ MMF: Multimode Fiber, Mehrmoden-LWL mit Gradientenindex 50/125µ bzw. 62,5/125µ

¹¹ SMF: Singlemode Fiber, Einmoden-LWL 5-9/100µ

2.3.2 Token Ring

Token Ring ist eine Shared Media LAN-Technologie, die ein deterministisches Zugriffsverfahren auf das Medium einführt. Grundlage sind eine Reihe von Punkt-zu-Punkt-Verbindungen zwischen den Endgeräten, welche durch die Verbindung der ersten mit der letzten Station einen Ring bilden. Diese Technik wurde von der IEEE als LAN-Standard 802.5 ausgearbeitet.

Die Stationen sind am Ring gleichberechtigt. Das Senderecht wird mit Hilfe einer einzelnen 24 Bits langen Sendemarke, dem Token vergeben. Im Leerlauf kreist dieses Token permanent im Ring, wobei es von jeder Station der Reihe nach eingelesen und um eine Bitzeit verzögert wieder auf den Ring gegeben wird. Dabei wird das Signal regeneriert, wodurch bei Token Ring keine Repeater erforderlich sind. Möchte eine Station Daten senden, so muss sie zunächst warten, bis das Token bei ihr angekommen ist. Daraufhin wandelt sie es durch Setzen eines Bits im AC-Feld (siehe Abbildung 7) in einen Datenrahmen und komplettiert ihn durch Anhängen von Adressen, Nutzdaten, Prüfsumme und Endfelder.

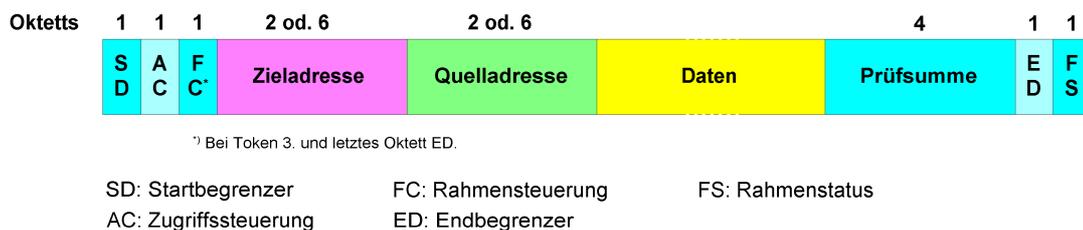


Abbildung 7 Token- und Rahmenformat nach IEEE802.5

Erreicht der Frame den Adressaten liest ihn dieser, wie alle Zwischenstationen auch, bitweise ein und gibt ihn wieder auf den Ring. Der Empfang wird mit Hilfe zweier Bits im Rahmenstatusfeld am Ende des Frames quittiert. Nachdem der Frame den Ring einmal umkreist hat, gelangt er wieder beim Sender an, welcher ihn vom Ring entfernen muss. Anhand der zwei Bits im Rahmenstatus kann der Sender erkennen, ob die Zielstation erreichbar war und ob sie den Frame akzeptiert hat. Am Ende der Übertragung muss der Sender wieder ein Token auf den Ring geben. Damit nicht eine Station ununterbrochen senden kann, indem es einen Datenframe nach dem anderen auf den Ring schickt, bekommt jede Station nur eine bestimmte Zeit (typ. 10ms) für die Übertragung ihrer Daten. Ist diese Zeit abgelaufen, muss sie wieder ein Token generieren, unabhängig davon, ob sie noch Daten zu versenden hat oder nicht. Auf diese Weise rotiert auch das Senderecht im Ring. Davon unabhängig existiert auch noch ein Mechanismus zur Priorisierung des Senderechts. Da Token Ring aber heute für neue

Netzwerke keine große Bedeutung mehr hat, sei an diese Stelle auf die Literatur verwiesen [2, 7 und 10].

Token Ring ist für die Bandbreiten 4 Mbps und 16 Mbps standardisiert.

3 Der Asynchrone Transfer Modus

Ende der 80er Jahre wurde damit begonnen, die Architektur für ein internationales Hochgeschwindigkeits- WAN¹² zu entwickeln, welches in der Lage sein sollte, die Leistungsanforderungen der heutigen Sprach-, Daten- und Fernnetzwerke zu erfüllen sowie für zukünftige Entwicklungen genügend Reserven zur Verfügung zu stellen. Die Standardisierung dieses Breitband-ISDN (im Folgenden B-ISDN) genannten Netzwerks wurde 1990 von der ITU begonnen. Basistechnologie für das B-ISDN ist die Übertragung von ATM-Zellen, die in Rahmen der SDH¹³ eingebettet werden. Nachdem seit Mitte der 90er Jahre weltweit öffentliche Datendienste auf ATM- Basis verfügbar sind, wurde ATM aufgrund seiner Eigenschaften auch für den Einsatz in LANs adaptiert.

Wegen seiner Herkunft im WAN- Bereich unterscheidet sich ATM grundlegend von den bisher beschriebenen Technologien der lokalen Netze. Werden bei Ethernet und Token Ring Pakete variabler Größe ausgetauscht, arbeitet ATM mit Paketen fester Länge, den Zellen. ATM ist anders als die oben behandelten Broadcastnetze eine verbindungsorientierte Kommunikationssart, das heißt bevor Daten übertragen werden können, muss eine dedizierte Verbindung zwischen Sender und Empfänger aufgebaut werden. Es existiert hier also kein shared medium, statt dessen sind Vermittlungseinrichtungen, prinzipiell ähnlich den Switches bei der Mikrosegmentierung bei Ethernet, notwendig.

3.1 Referenzmodell

Wie für Ethernet und Token Ring existiert auch für ATM ein Referenzmodell, das B-ISDN-Referenzmodell (Abbildung 8).

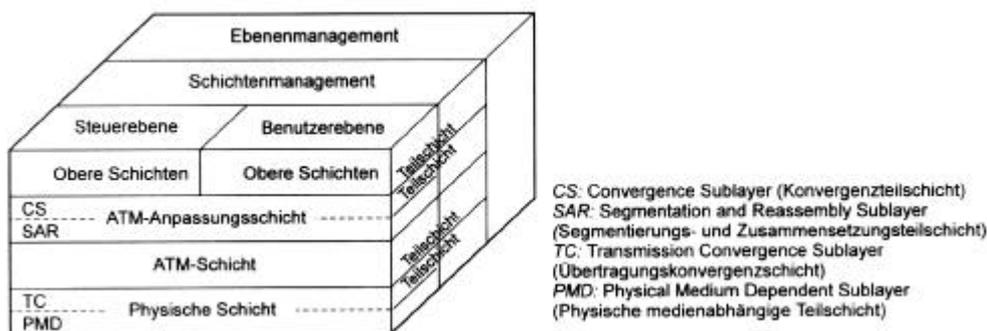


Abbildung 8 Das B-ISDN-Referenzmodell

¹² WAN: Wide Area Network, Weitverkehrsnetz.

ATM besteht aus einer Reihe aufeinander aufbauender Protokollschichten, woraus ein im Vergleich mit dem ISO- OSI- Referenzmodell relativ komplexes Referenzmodell resultiert. Notwendig sind die vielen Schichten für die Abbildung der verschiedensten Netzdienste auf den Zelltransportmechanismus des ATM. Grob unterteilt sich das Modell in drei Ebenen und vier Schichten. In der ersten Ebene liegen die Steuer- und die Benutzerebene, dahinter befinden sich zwei Managementebenen für das Schichten- und das Ebenenmanagement. In der Benutzerebene ist der Nutzdatenfluss zwischen den ATM- Kommunikationspartnern symbolisiert, die Steuerebene beinhaltet alle Funktionen für den Auf- und Abbau sowie die Überwachung der Verbindungen [3].

3.1.1 Die Physische Schicht

Die Physische Schicht ist in die Teilschichten Übertragungsanpassung (TC, Transmission Convergence) und die physikalische Medienschiicht (PMD, Physical Medium Dependent) unterteilt. TC erhält von der übergeordneten ATM-Schicht Zellen übergeben, die es in die Übertragungsrahmen des jeweiligen Transportmediums einbetten muss (beispielsweise in SDH- Rahmen). Dazu werden die Zellen kodiert und mit einer Prüfsumme (HEC siehe Abb. 9) versehen.

Die PMD-Schicht des ATM unterstützt ein großes Spektrum an Übertragungsmedien. Während für Weitverkehrsnetze SMF-LWL- und Koaxialkabel der entsprechenden ITU-Standards verwendet werden, wurden für ATM im LAN preisgünstigere Medien definiert, welche Übertragungsraten von 25 Mbps bis 2,4 Gbps bieten. Für die Leistungsmessungen (Kapitel 4) kommen die in Tabelle 3 genannten Medien zum Einsatz.

Datenrate	Kabeltyp	max. Länge
25,92 Mbps	Twisted Pair, Kat. 5	100 m
155 Mbps	Twisted Pair, Kat. 5	150 m
155 Mbps	MMF-LWL 50/125 μ	2 km

Tabelle 3 In Kapitel 4 verwendete Übertragungsmedien

¹³ SDH: Synchroner Digitale Hierarchie.

3.1.2 Die ATM-Schicht

Vollständig entkoppelt von der darunter liegenden Physischen Schicht, hat die ATM-Schicht die Aufgabe, die ihr von der Anpassungsschicht übergebenen Daten zu ihrem Ziel zu transportieren. Dazu bildet sie die ATM-Zellen, deren Format in Abbildung 9 dargestellt ist.

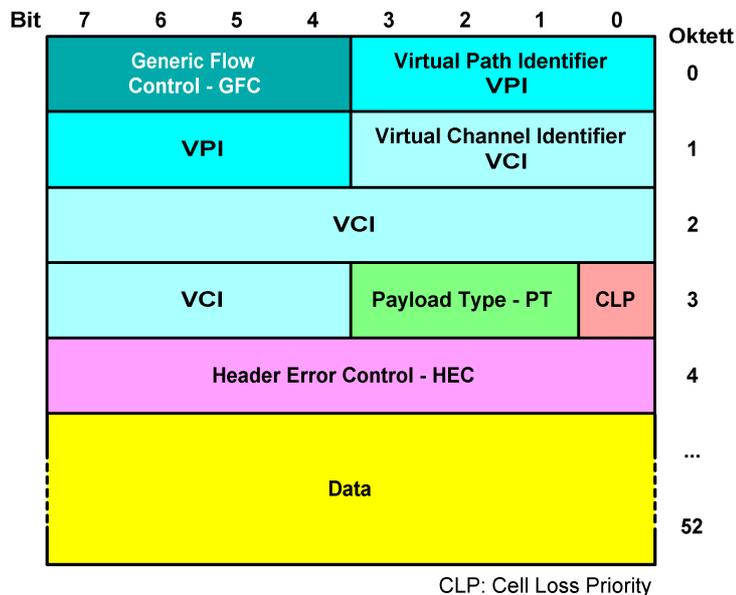


Abbildung 9 Aufbau einer ATM-Zelle

Eine ATM-Zelle ist 53 Oktetts lang, die ersten fünf Oktetts bilden den Zellkopf (Header), die restlichen 48 Oktetts stehen für die Nutzdaten zur Verfügung. Da ATM verbindungsorientiert arbeitet, d.h. der Weg der Zellen nach Aufbau einer Verbindung bekannt ist, sind keine Adressen im Sinne der 802.x – MAC- Schicht nötig. Statt dessen enthält der Zellheader nur die Nummer (VPI/VCI) der Verbindung. Diese verwenden die an der Übertragung beteiligten Vermittlungsknoten zur Zuordnung der Zellen verschiedener Verbindungen –welche im selben Zellstrom gemultiplext vorliegen können– zu den geschalteten Transportwegen. Um die Vermittlung zu vereinfachen können mehrere virtuelle Kanäle zu virtuellen Pfaden zusammen gefasst werden. Der ATM-Schicht obliegt die Bildung der ersten vier Oktetts des Zellheaders, die Prüfsumme wird von der TC- Teilschicht der Physischen Schicht eingefügt.

Im Feld GFC befinden sich 4 Bits zur Flusskontrolle, mit den drei PT- Bits können verschiedene Zellinhalte unterschieden werden. Mit dem CLP-Bit können Zellen dahingehend priorisiert werden, dass im Falle einer Netzüberlastung oder Verkehrsvertragsverletzung zuerst die mit CLP markierten Zellen verworfen werden.

3.1.3 Die ATM-Anpassungsschicht (AAL ATM Adaption Layer)

Die Anpassungsschicht ist wieder in zwei Teilschichten untergliedert, die Konvergenzteilschicht (CS) und die Segmentier- und Reassemblier- Teilschicht (SAR). Letztere teilt beim Senden die Daten der höheren Schichten in Portionen von 48 Bytes auf und übergibt sie an die ATM-Schicht. Beim Empfang fügt sie die von „unten“ kommenden 48-Byte-Fragmente wieder zusammen und leitet sie den höheren Schichten zu.

Die Konvergenzteilschicht übernimmt die Anpassung der ATM- Zellübertragung an die Spezifika der übergeordneten Anwendungen. Da die Anwendungscharakteristiken sehr weit gefächert sein können, werden von AAL vier verschiedene Serviceklassen definiert: AAL1, AAL2, AAL3/4 und AAL5, die jeweils verschiedene Anwendungsklassen berücksichtigen. Die Zuordnung der Service- zu den Anwendungsklassen ist in Tabelle 4 dargestellt.

	Klasse A	Klasse B	Klasse C	Klasse D
Zeitkompensation	erforderlich		nicht erforderlich	
Bitrate	konstant	variabel		
Verb.-modus	verbindungsorientiert			verbindungslos
Beispiel	Circuit Emulation	Video	Datentransfer, verb.-orient.	Datentransfer, verbindungslos
AAL-Typ	AAL1	AAL2	AAL3, 5	AAL4

Tabelle 4 Serviceklassen und AAL-Typen [3]

Der mit Abstand verbreitetste AAL-Typ ist wegen seiner vergleichsweise einfachen Implementierung AAL5.

3.1.4 Die höheren Schichten

Die höheren Schichten bilden die auf AAL aufsetzenden Netztransportdienste. Dazu gehören die in den nächsten Abschnitten beschriebenen Dienste LAN Emulation und Classical IP, aber auch Audio-/Videodienste, Telefonie u.v.m.

3.2 ATM im LAN

Um den schrittweisen Übergang von herkömmlichen LAN-Technologien zu ATM zu ermöglichen, ist eine Anpassung der herkömmlichen LAN-Verfahren und -Protokolle an ATM nötig. ATM kann dazu die etablierten Schnittstellen emulieren. Für die darüber auf ATM aufsetzenden LAN-Dienste ist das ATM-Netz transparent, eine Anwendung die Ethernet- Frames generiert merkt also nicht, dass diese nicht über ein Ethernet transportiert werden. Dazu existieren sowohl Möglichkeiten auf ISO-Schicht 2 (MAC), als auch auf Schicht 3 (Vermittlung):

- LLC-Einkapselung,
- Classical IP over ATM,
- LAN Emulation.

Die im Folgenden beschriebenen Verfahren basieren auf AAL5, der Serviceklasse für verbindungsorientierte und verbindungslose Datenübertragung ohne Zeitkompensation zwischen Sender und Empfänger.

3.2.1 LLC-Einkapselung

LLC-Einkapselung ist der erste spezifizierte Mechanismus zur LAN-Kopplung mittels ATM. Er ist in den RFCs¹⁴ 2225 und 2684 der IETF¹⁵ niedergelegt. Es werden zwei Methoden unterschieden:

- LLC-Einkapselung und
- VC-basiertes Multiplexen.

LLC-Einkapselung überträgt alle LAN-Pakete durch Kapselung in AAL5-PDUs innerhalb einer einzigen virtuellen ATM-Verbindung (Abbildung 10). Dieses Verfahren ist hauptsächlich für einfache ATM-Netze gedacht, welche nur permanente virtuelle Verbindungen (PVCs) unterstützen.

¹⁴ RFC: Request for Comments, Normierungsvorschläge der IETF

¹⁵ IETF: Internet Engineering Task Force, Normungsgremium für Internetentwicklungen.

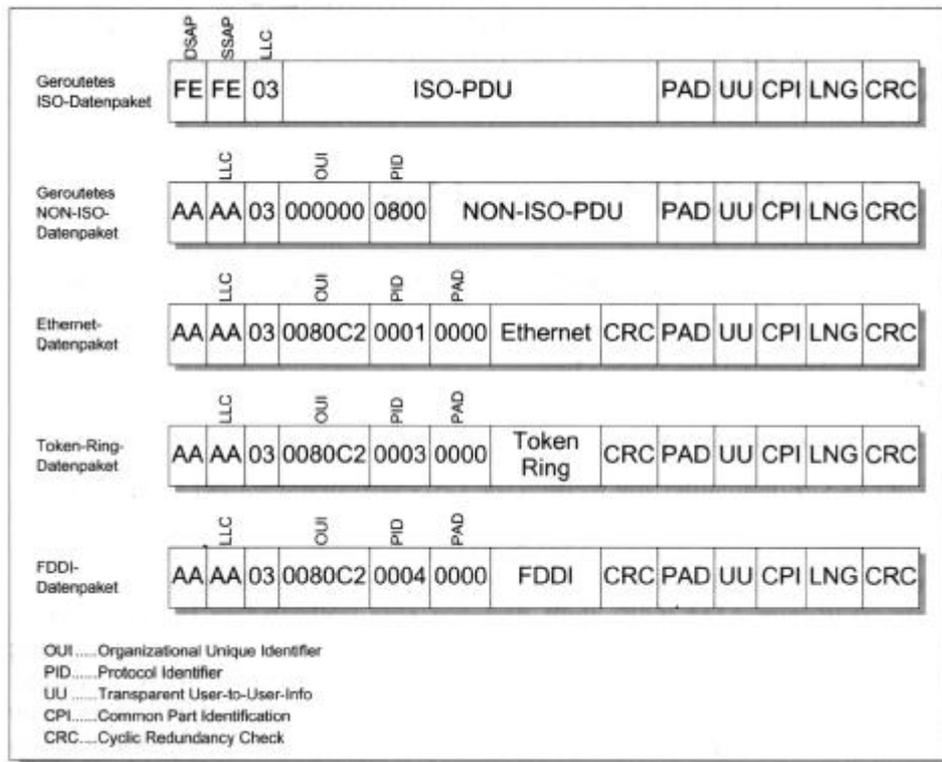


Abbildung 11 LLC-Einkapselung nach RFC 2684 (1483)

Beim VC-basierten Multiplexen (Abbildung 11) dagegen wird für jedes zu übertragende Protokoll eine eigene virtuelle Verbindung (VC) aufgebaut. Weil jetzt die LLC-Header nicht mehr übertragen werden müssen ist VC-basiertes Multiplexen effizienter als LLC-Einkapselung.

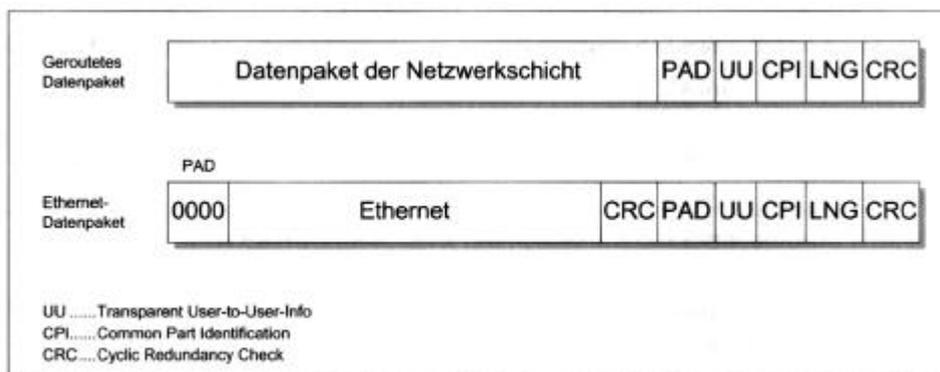


Abbildung 10 VC-basiertes Multiplexen

3.2.2 Classical IP over ATM (CLIP)

Bei Classical IP over ATM handelt es sich, im Gegensatz zur gerade beschriebenen LLC- Kapselung, um die vollständige Implementierung des Internetprotokolls IP in ATM. Da IP ein Protokoll der ISO-Schicht 3 (Vermittlung) ist, können mittels CLIP keine herkömmlichen LANs auf MAC-Ebene mit ATM verbunden werden. Alle Stationen müssen an das ATM-Netz angeschlossen werden. Die IP- eigenen Mechanismen zur Adressauflösung mittels [Reverse] Address Resolution Protocol (ARP/RARP) werden innerhalb des ATM von ATMARP- Servern emuliert. Dieser löst in jedem logischen IP-Subnetz (LIS) die IP- Adressen zu ATM- Stationsadressen und umgekehrt auf. Dazu stellt er die Funktionen ATMARP und InATMARP (Inverse ATMARP) zur Verfügung.

Sowohl die ATMARP-Pakete als auch die eigentlichen IP- Datenpakete werden wieder mittels LLC-Einkapselung innerhalb AAL5 übertragen (Abbildung 12).

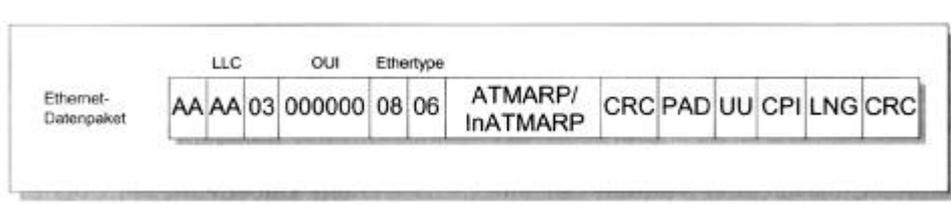


Abbildung 12 ATMARP und Inverse ATMARP

Folgende Regeln sind beim Einsatz von CLIP zu beachten [3]:

- CLIP unterstützt keine Broadcasts.
- Alle IP- Arbeitsstationen müssen direkt an das ATM- Netz angeschlossen sein.
- Alle Stationen innerhalb eines LIS haben dieselbe IP- Subnetzadresse.
- Zwischen verschiedenen logischen Subnetzen kann nur über Router kommuniziert werden.
- Alle Stationen müssen Adressauflösung nach ATMARP beherrschen.
- Jede IP- Station eines LIS muss mit jeder anderen IP- Station kommunizieren können.
- Die Adressauflösung muss sowohl für PVCs als auch für SVCs funktionieren.
- Die MTU¹⁶ bei CLIP beträgt standardmäßig 9180 Bytes.

¹⁶ MTU: Maximum Transfer Unit, maximale übertragbare Paketgröße.

3.2.3 LAN Emulation (LANE)

Das flexibelste Verfahren, ATM-Netze mit bestehenden LANs zu verschmelzen, ist die LAN Emulation. Sie stellt eine vollständige Emulation der OSI-Schicht 2 (MAC) des LANs dar, mit der Folge, dass alle herkömmlichen Anwendungen ohne Änderung über ATM weiterbetrieben werden können.

Dabei muss die LAN Emulation eine Reihe von Problemen überwinden, die in den verschiedenen Ansätzen von LAN-Technologien einerseits und ATM andererseits begründet liegen. Das größte Problem ist die Diskrepanz der Kommunikationsarten. ATM arbeitet ausschließlich verbindungsorientiert, die traditionellen LANs dagegen verbindungslos. Verbindungsorientierte Dienste arbeiten in LANs auf höheren Protokollebenen (wie z.B. TCP). Darüber hinaus unterstützen LANs durch ihr ursprüngliches shared medium-Konzept Broadcasts, d.h. alle Stationen im LAN können mittels einer speziellen MAC-Adresse mit ein und demselben Paket angesprochen werden. Daraus resultieren auch verschiedene LAN-Protokolle, welche für ihre korrekte Funktion auf diese Broadcasts angewiesen sind (z.B. DHCP mit BOOTP). Auch ARP nutzt zur Adressauflösung Broadcastpakete. Das steht im krassen Widerspruch zur verbindungsorientierten Arbeitsweise des ATM. Da hier zwischen jedem Paar von Kommunikationspartnern eine eigene Verbindung hergestellt wird, bedeutet das, dass ein Broadcast über je eine dedizierte Verbindung vom Sender zu jeder ATM-Station im emulierten LAN nachgebildet werden muss. Das kann einen beträchtlichen Ressourcenbedarf nach sich ziehen, der im schlimmsten Fall zu einer Überlastung des ATM-Netzes führen kann.

LAN Emulation (LANE) ist vom ATM Forum standardisiert und in [19] und [20] ausführlich beschrieben. LANE existiert für IEEE 802.3 Ethernet, IEEE 802.5 Token Ring und FDDI. Die Einbettung der MAC-Frames in die ATM-Zellen erfolgt wie in Abbildung 13 illustriert. Für ATM-Stationen innerhalb eines emulierten LANs stehen die QoS-Funktionen des ATM zur Verfügung.

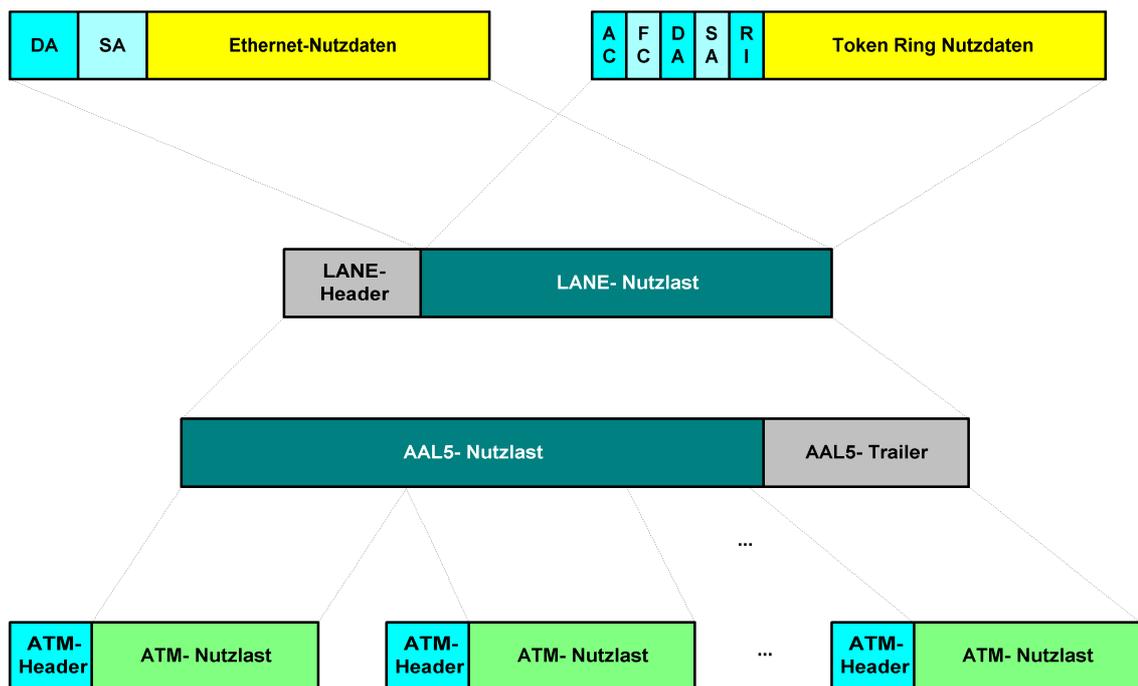


Abbildung 13 Bildung der LANE- Datenpakete

3.2.3.1 Funktionseinheiten der LANE

Zur Emulation von LANs sind folgende, auf AAL5 aufsetzende Funktionseinheiten erforderlich:

- LAN Emulation Client (LEC),
- LAN Emulation Server (LES),
- LAN Emulation Configuration Server (LECS) und
- Broadcast and Unknown Server (BUS).

Die Serverfunktionen können einzeln oder kombiniert in einem dedizierten Gerät (PC, Server) oder innerhalb der Vermittlungseinheiten (ATM-Switch, ATM-LAN- Brücken o.ä.) realisiert werden.

Der LEC ist Bestandteil aller an der LAN Emulation teilnehmenden Endstationen und meist als Software realisiert. Er stellt zusammen mit dem ATM- Interface den Anwendungen ein Standard- LAN- MAC- Interface zur Verfügung. Er führt neben den notwendigen Steuerfunktionen auch den eigentlichen Datentransfer über ATM durch.

Im Rahmen der Steuerung des emulierten LAN (ELAN) übernimmt der LES die Funktionen der LEC- Registrierung und der Auflösung von MAC- zu ATM- Adressen.

Der LECS verwaltet die Zugehörigkeit der LECs zu verschiedenen ELANs. Dazu wird von ihm eine Konfigurationsdatenbank gepflegt, aus der für jeden LEC hervorgeht, welchen ELANs er angehört. Daraus folgt, dass ein LEC gleichzeitig Mitglied mehrerer ELANs sein kann.

Dem BUS obliegt die schon weiter oben angesprochene Emulation von Broadcast- und Multicast- Verkehr innerhalb des ATM. Die LECs senden ihre Broad-/Multicast- Pakete an den BUS, der sie sequentiell an die LECs der adressierten Gruppe weiterleitet.

Die Kommunikation zwischen den Funktionseinheiten der LANE läuft in folgenden 5 Phasen ab (Abbildung 14):

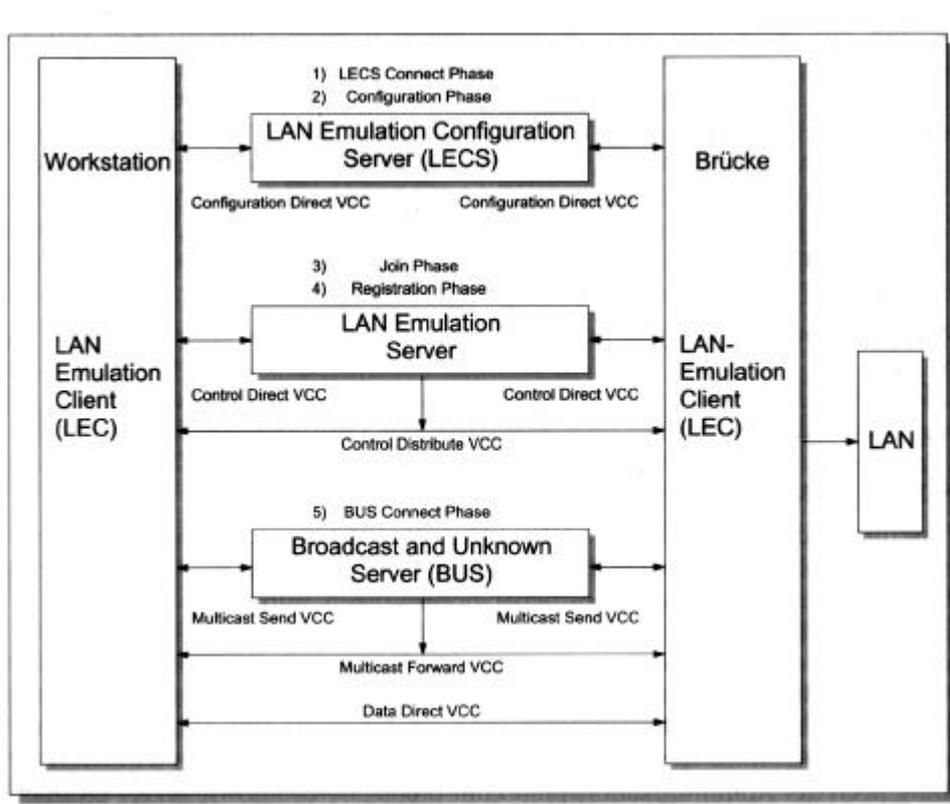


Abbildung 14 Kommunikationsphasen der LAN Emulation

1, 2) Wird dem ELAN einer neuer LEC hinzugefügt, so muss dieser über eine direkte Konfigurationsverbindung (Config. Direct VCC) Kontakt mit dem LECS aufnehmen. Der LEC registriert sich beim LECS für die gewünschten ELANs und handelt mit diesem die verschiedenen Parameter der LANE (Adressen, ELAN- Name, MTU) aus. Der LECS fügt die entsprechenden Daten in seine Konfigurationsdatenbank ein.

3, 4) Danach wird vom LEC eine Steuerverbindung (Control Direct VCC) zum LES aufgebaut, über die er die restlichen, notwendigen Informationen –wie LECID und LAN- Typ– zur LANE- Teilnahme bekommt.

5) Abschließend baut der LEC den Multicast Send VCC zum BUS auf, über den der LEC seinen Broad-/Multicast- Verkehr sendet. Nachdem der BUS seinerseits einen Multicast Forward VCC zum LEC aufgebaut hat, ist die Anmeldung des neuen LEC am emulierten LAN abgeschlossen. Über den Multicast Forward VCC empfängt der LEC Broad-/Multicast- Pakete vom BUS, die von den anderen LECs im ELAN stammen.

Der Datenaustausch findet von jetzt an entweder zwischen zwei LECs direkt, oder zwischen LEC und BUS statt. Einzig zur Auflösung einer ihm unbekanntem MAC-Adresse in eine ATM- Adresse muss der LEC mittels einer LE-ARP- Nachricht den LES kontaktieren.

4 Leistungsmessung

Bei den folgenden Performanceuntersuchungen steht das Verhalten der verschiedenen Netzwerktypen aus Sicht der im Netzwerk eingesetzten Applikationen im Mittelpunkt. Aus diesem Grund sollte ein Protokoll zum Einsatz kommen, das einerseits auf allen Netzwerktechnologien verfügbar ist und andererseits von einer Vielzahl weitverbreiteter Anwendungen benutzt wird. Als Ausgangspunkt wurde das größte bekannte, heterogene Netzwerk herangezogen, das Internet. Den größten Teil des Netzwerkverkehrs im Internet verursachen die Dienste bzw. Protokolle:

- Elektronische Post (E-Mail),
- Dateiaustausch per FTP (**F**ile **T**ransfer **P**rotocol) und
- WWW (**W**orld **W**ide **W**eb) per HTTP (**H**ypertext **T**ransfer **P**rotocol).

Alle diese Dienste benutzen als grundlegendes Transportprotokoll das zur TCP/IP-Familie gehörende Transmission Control Protocol (TCP).

4.1 TCP

TCP basiert auf dem in [1] ausführlich dargestellten Internetprotokoll IP und erweitert dieses um zusätzliche Funktionalitäten. TCP beinhaltet Mechanismen zur Vermeidung der Probleme, die bei unzuverlässigen Transportprotokollen wie z.B. UDP auftreten können. Diese Probleme sind verlorene oder zerstörte Pakete, in falscher Reihenfolge übertragene Pakete und mehrfach zugestellte Pakete (Duplikate). Es stellt somit aufsetzend auf das verbindungslose IP, einen verlässlichen, verbindungsorientierten Service für Datenpakete in Computernetzwerken zur Verfügung.

4.2 Die Testsoftware

Die Software für die Durchsatzmessungen besteht aus einem ausführbaren Programm für Microsoft Windows NT (NTTester.exe). Es wurde mit der Entwicklungsumgebung Visual Studio in Visual C++ erstellt. Dabei kam die Klassenbibliothek Microsoft Foundation Classes (MFC) zum Einsatz, welche die Programmierung einer einheitlichen Benutzeroberfläche unter Windows wesentlich vereinfacht [6].

Funktionell besteht NTTester aus zwei Hauptteilen, einem Server und einem Client zur Datenübertragung. Darüber hinaus beinhaltet das Programm noch Funktionen zum Überprüfen der Verbindung und Informationen über den Protokollstack von Windows.

4.2.1 Implementation

4.2.1.1 Die Socket Schnittstelle

Die bekannteste Schnittstelle zwischen den TCP/IP- Protokollen und der Kommunikationssoftware ist das Socket-Interface. Da die Protokollsoftware als ein Bestandteil des Betriebssystems angesehen wird, stellt das Socketmodell somit eine Schnittstelle zwischen Anwendungsprogrammen und Betriebssystem dar, auch API (Application Programming Interface) genannt. Das Socketmodell stammt aus der Welt der UNIX-Netzwerke, wo auch das Internet seinen Ursprung hat. Die erste Socket-Implementierung erfolgte in der Version 4.2 des BSD-Unix, welches 1983 erschien. Seitdem erlangte dieses Modell eine breite Akzeptanz, es ist für eine Vielzahl von Betriebssystemen verfügbar. Für Microsoft Windows sind die Socketfunktionen im WinSock API enthalten [11].

Ein Socket stellt in diesem Modell die Abstraktion für einen Endpunkt der Kommunikation dar. Das Socket-API beinhaltet alle notwendigen Funktionen, die die kommunizierenden Prozesse der Anwendungsprogramme benötigen. Die in NTTester verwendeten API-Funktionen sollen im Folgenden kurz aufgeführt werden. Dabei werden die Namen des WinSock API zugrundegelegt, welche sich von den Pendanten des Berkeley-UNIX geringfügig unterscheiden. Für eine detaillierte Beschreibung mit allen Parametern und Rückgabewerten wird auf [11] verwiesen.

socket()	Erzeugt einen Kommunikations-Endpunkt und gibt einen Sockethandle zurück.
bind()	Verbindet einen Socket mit einer definierten lokalen Adresse. Wird von Serverprozessen benutzt, um auf einem bestimmten Port zu arbeiten.
listen()	Wartet an dem übergebenen Socket auf eingehende Verbindungswünsche von Kommunikationspartnern.
Accept()*	Eine anliegende Verbindung wird von einem neuen Socket, den die Funktion als Rückgabe liefert, übernommen. Der ursprüngliche Socket wird wieder „frei“, d.h. es geht wieder in den listen-Modus über. Wird von Serverprozessen nach listen() aufgerufen.
connect()*	Mit dem übergebenen Socket wird eine Verbindung aufgebaut. Wird von Clientprozess aufgerufen.
send()*	Sendet Daten über einen verbundenen Socket.
recv()*	Empfängt Daten von einem (nicht) verbundenen Socket.
select()*	Dient der Ermittlung des Status eines oder mehrerer Sockets.
closesocket()*	Schließt den übergebenen Socket.
WSAStartup()	Initialisiert die WinSock-Bibliothek von MS Windows.
WSACleanup()	Beendet die Arbeit mit der WinSock-Bibliothek.

Die mit einem Stern gekennzeichneten Funktionen besitzen eine Eigenschaft, der bei der Programmierung besonderes Augenmerk gewidmet werden muss, sie blockieren. Dieses Thema wird im Abschnitt 4.2.1.1.2 beschrieben.

4.2.1.1.1 Ablauf der Kommunikation über das Socket-API

Die Abbildung 8 zeigt die prinzipielle zeitliche Abfolge der API-Aufrufe, die für eine Kommunikation zweier Prozesse nach dem Client/Server-Modell notwendig sind. Dabei

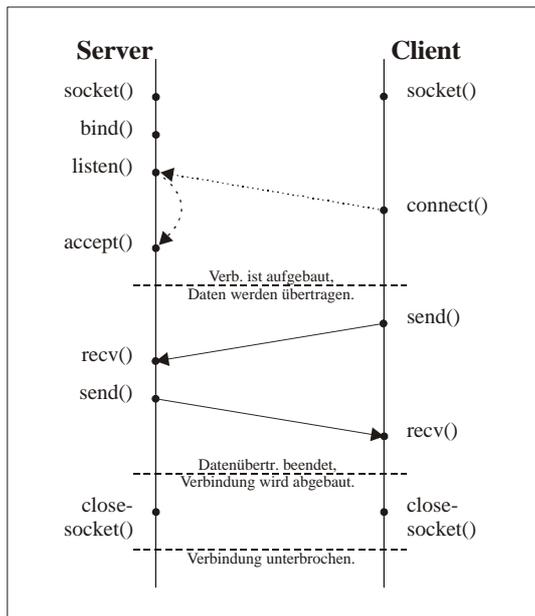


Abbildung 15

stellen die senkrecht verlaufenden Linien – von oben nach unten gesehen – die fortschreitende Zeit dar.

Zunächst muss jeder kommunikationswillige Prozess einen Socket öffnen (Funktion „socket()“). Danach wird der serverseitige Socket an einen definierten Port gebunden („bind()“), das heißt der Server stellt seinen Dienst unter einer bestimmten Adresse zur Verfügung. Mit dem Aufruf der Funktion „listen()“ geht der Server in den Wartezustand und hört den Socket auf ankommende Verbindungswünsche ab.

Einen solchen initiiert der Client, indem er die Funktion „connect()“ unter Angabe der Adresse des Servers (Hostadresse *und* Portnummer) aufruft. Sind ihm statt der Adressen nur die Namen des Servers bzw. des gewünschten Dienstes bekannt, muss der Client die Adressen über entsprechende Funktionen des Socket-API ermitteln. Diese Funktionen sind der Übersicht halber nicht in der Skizze berücksichtigt und in [11] beschrieben.

Daraufhin beendet „listen()“ innerhalb des Serverprozesses den Wartezustand und der Server bestätigt den Verbindungswunsch des Clients mit Aufruf von „accept()“. Zu diesem Zeitpunkt ist eine bidirektionale Verbindung aufgebaut und die beiden Prozesse können beliebige Daten austauschen.

Mit Aufruf von „closesocket()“ wird die Verbindung abgebaut und die Systemressourcen wieder freigegeben.

4.2.1.1.2 Implementierungsdetails

Für die objektorientierte Programmierung des Testprogramms wurden die beschriebenen Socketfunktionen in einer neuen Klasse „CBlockingSocket“ gekapselt. Anders als die in der MFC enthaltene Klasse CSocket, geht die neue Klasse mit der 32-Bit-Umgebung von Windows NT konform und enthält keine 16-Bit-Funktionen [6].

Wie bereits im Abschnitt 4.2.1.1 erwähnt, unterliegen einige der Socketfunktionen einer Besonderheit: ihr Aufruf kann die Ausführung des aufrufenden Prozesses blockieren. Dies würde immer genau dann passieren, wenn die jeweilige Funktion nicht sofort mit einem Ergebnis beendet werden kann. Wird zum Beispiel die Funktion „listen()“ aufgerufen, würde der Server solange blockieren bis ein Client eine Verbindung zu ihm aufbaut. In einem Programm mit grafischer Benutzeroberfläche wie Windows bedeutet das, dass während dieser Zeit keine Eingaben vom Anwender mehr akzeptiert werden. Das Programm wäre nicht mehr bedienbar!

Um diese Einschränkung zu umgehen, wurde von der Multithreading-Fähigkeit von Windows NT Gebrauch gemacht. Diese ermöglicht es, ein Programm in mehrere Ablaufstränge –sogenannte Threads– aufzuteilen, welche quasi parallel ausgeführt werden können. Wird nun ein solcher Thread blockiert, können alle anderen trotzdem weiter von der CPU des Computers abgearbeitet werden, solange sie nicht selbst blockiert sind.

Im Fall des Programms NTester wurden die beiden Hauptaufgaben in jeweils einem eigenen Arbeits-Thread implementiert, einem Server- und einem Clientthread. Diese befinden sich als die Funktionen ServerThreadProc() und ClientThreadProc() in der Datei Global.cpp (siehe Anhang). Durch diese Trennung vereinfacht sich die Programmierung, da innerhalb der Arbeits-Threads keine Benutzeraktionen behandelt werden müssen. Außerdem bleibt die grafische Oberfläche bedienbar und Eingaben werden ohne Verzögerung abgearbeitet.

4.2.2 Hardware

Für die Durchführung der Durchsatzmessungen fanden zwei PCs Anwendung, deren Grundausstattung sich folgendermaßen darstellt:

Ausstattungsmerkmal	Testserver	Testclient
Prozessor	Intel Pentium P54C	Intel Pentium P54C
Chipsatz	Intel 430VX	Intel 430VX
Arbeitstaktfrequenz	133 MHz	133 MHz
Arbeitsspeicher	64 Mbyte, FPM	64 Mbyte, FPM
Bussysteme für Adapterkarten	PCI (33 MHz), ISA (8 MHz)	PCI (33 MHz), ISA (8 MHz)
Betriebssystem	Microsoft Windows NT4.0 Server	Microsoft Windows NT4.0 Workstation

Tabelle 5 Hardwareausstattung der Test- PCs

Beide übernehmen zum Test die Rolle der Arbeitsstationen an den Enden der zu testenden Übertragungsstrecke. Für die verschiedenen Medien- bzw. Netzwerktypen werden die Testrechner mit jeweils einer Netzwerk-Interfacekarte ausgestattet, deren Auswahl die Tabelle 6 zeigt. Die Details können den entsprechenden Abschnitten entnommen werden.

Netzwerk- / Kabeltyp	PC- Bussystem	
	ISA	PCI
Ethernet, 10 Mbps, 10Base2	NDC 4113	---
Ethernet, 10 Mbps, 10BaseT	NDC 4113, 3Com 3C509B TP	---
Ethernet, 100 Mbps, 100BaseTX	---	Accton EN1207Tx
ATM, 25 Mbps, Twisted Pair	---	IBM TurboWays 25 (ATM-Forum kompat.)
ATM, 155 Mbps, MMF LWL, Twisted Pair	---	ForeRunner 200

Tabelle 6

4.2.3 Vorbetrachtungen

Vor der Durchführung der Tests mit den realen Netzwerken sollen zunächst einige Vorbetrachtungen zur verwendeten Technik angestellt werden. Um die Leistungsfähigkeit der PCs abschätzen zu können, wurde als erster Test ein NTester-Durchlauf auf dem Loopback- Interface des WindowsNT- IP- Stacks gestartet. Das Loopback- Interface ist Bestandteil der IP- Protokollsoftware des Betriebssystems und stellt den Anwendungen die IP- Adresse 127.0.0.1 zur Verfügung. Auf diese Adresse kann die Anwendung zugreifen, ohne dass dadurch Datenpakete auf dem physischen Netzwerk- Interface (der Netzwerkkarte) erzeugt werden. Die IP- Pakete werden nur vom IP- Stack derart verarbeitet, dass sowohl der Server als auch ein oder mehrere Clients auf derselben Maschine laufen können, ohne dass ein Netzwerkkabel angeschlossen ist. Die dabei erreichten Ergebnisse (Abbildung 16) bieten einen Anhaltspunkt für die Abschätzung der maximal erzielbaren Datendurchsatzraten.

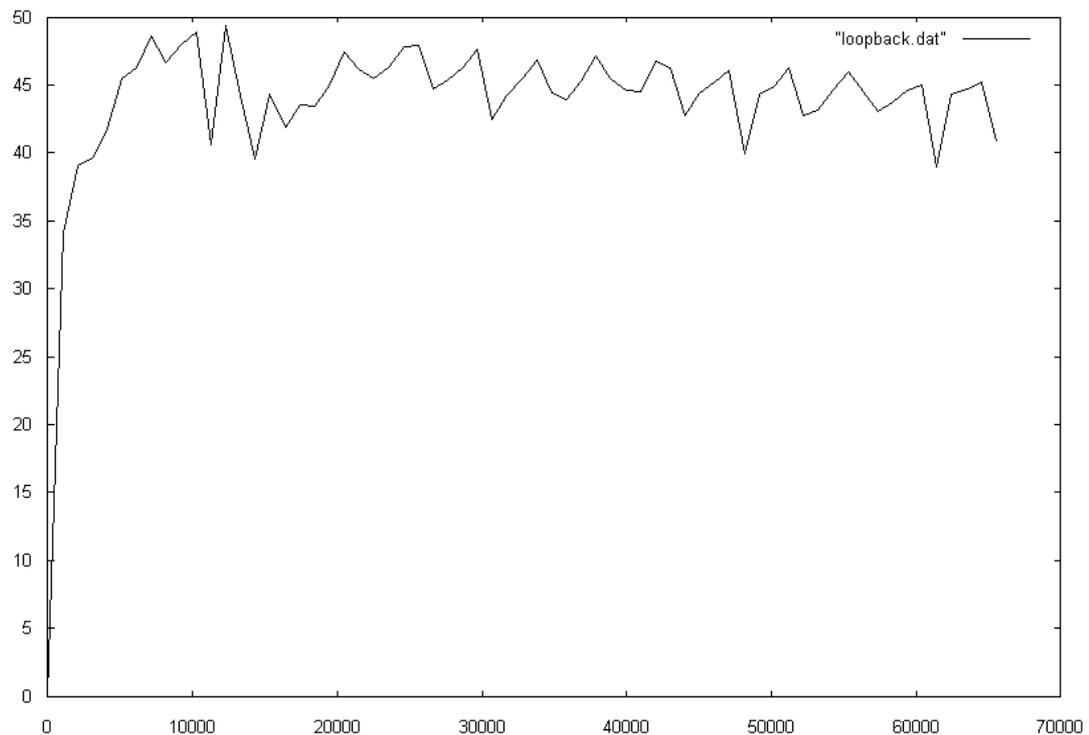


Abbildung 16 Testdurchlauf auf dem Loopback- Interface

Bei einem Testdurchlauf werden zwischen zwei Instanzen des Testprogramms Datenblöcke unterschiedlicher Länge ausgetauscht und die für jede Blockgröße die benötigte Zeit gemessen. Eine Instanz arbeitet als Server, die andere als Client. Die pro Blockgröße übertragene Datenmenge ist konstant und betrug bei allen Tests 5 Megabyte. Jede Datenübertragung wurde fünfmal durchgeführt und die gemessenen Zeiten arithmetisch gemittelt.

Der aus Abbildung 16 erkennbare maximale Durchsatz von ca. 50 Mbps lässt darauf schließen, dass die verwendeten PCs auf einem realen Netzwerk einen Datendurchsatz von Faktor Zwei bis Drei des hier ermittelten Wertes. Die Begründung dafür ist, dass in diesem Test beide Instanzen des Testprogramms auf derselben Arbeitsstation liefen. Für das Betriebssystem bedeutet dies ein ständiges Wechseln der aktiven Anwendung, was neben Ressourcen auch Zeit kostet.

4.2.3.1 Ethernet mit 10 Mbps

Für die Tests mit Ethernet standen zwei Typen von Netzwerkkarten mit ISA-PC-Bus zur Verfügung. Zum einen handelte es sich um zwei Karten der Firma 3Com vom Typ Etherlink III mit Anschluss für Twisted Pair Kabel. Diese wurden nach dem Einbau in die PCs mit dem nach Abschnitt 8.1 angefertigten Cross-Over- Kabel verbunden. Die anderen beiden Adapter waren sogenannte NE2000-kompatible Netzwerkkarten der Firma NDC, ihre genaue Bezeichnung lautet NDC 4113. Sie sind als Combo-Adapter

mit Anschlüssen für Koaxial- und Twisted Pair Kabel sowie mit einer AUI-Buchse für 10Base5- Dropkabel ausgestattet. Sie wurden mit einem Koaxial- Buskabel unter Verwendung je eines T- Abzweigstückes und je eines Abschlusswiderstandes (50 Ohm) miteinander verbunden. Das AT-Bus Interface der PCs arbeitet mit ca. 8 MHz und gestattet in der verwendeten Betriebsart „Programmed I/O“ eine Datenrate von etwas mehr als 40 Mbps, ist also für ein Netzwerk mit 10 Mbps genügend schnell und liegt weit unter den gemessenen und geschätzten Durchsätzen aus dem letzten Abschnitt.

Abbildung 17 zeigt zusammengefasst die gemessenen Datendurchsätze.

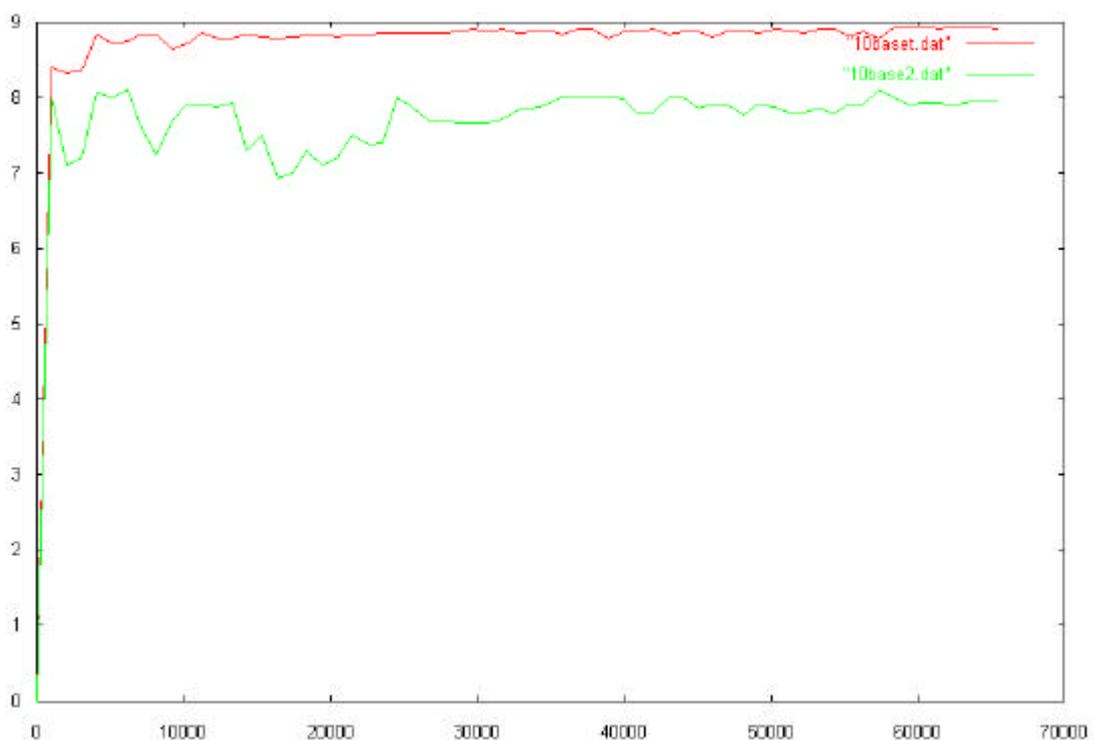


Abbildung 17 Testdurchlauf mit 10Base2 und 10 BaseT

Die Abbildung lässt erkennen, dass die erzielten Durchsatzraten recht nah an das theoretische Maximum heran reichen. Das liegt daran, dass hier keine weiteren Stationen im Netz vorhanden waren, und folglich auch keine Kollisionen aufgetreten sind. Dennoch liegen die beiden Kurven deutlich auseinander. Der Grund dafür liegt erfahrungsgemäß im Unterschied der eingesetzten Netzwerkkarten. Während die 10Base2- NICs Nonameprodukte sind, sind die Karten der Firma 3Com für die Performance der eigenentwickelten Chips bekannt. Daraus könnte man aber auch auf Unterschiede in der Qualität der mitgelieferten Treibersoftware schließen.

4.2.3.2 Ethernet mit 100 Mbps (100BaseTX)

Hierfür kamen zwei Netzwerkadapter der Firma Accton zum Einsatz, die mit dem gleichen Cross-Over- Kabel verbunden waren, wie im letzten Abschnitt. Erwartet werden hier, entsprechend der Schätzungen und der Messung bei 10BaseT, Werte um die 80 Mbps. Abbildung 18 zeigt das Ergebnis der Messung.

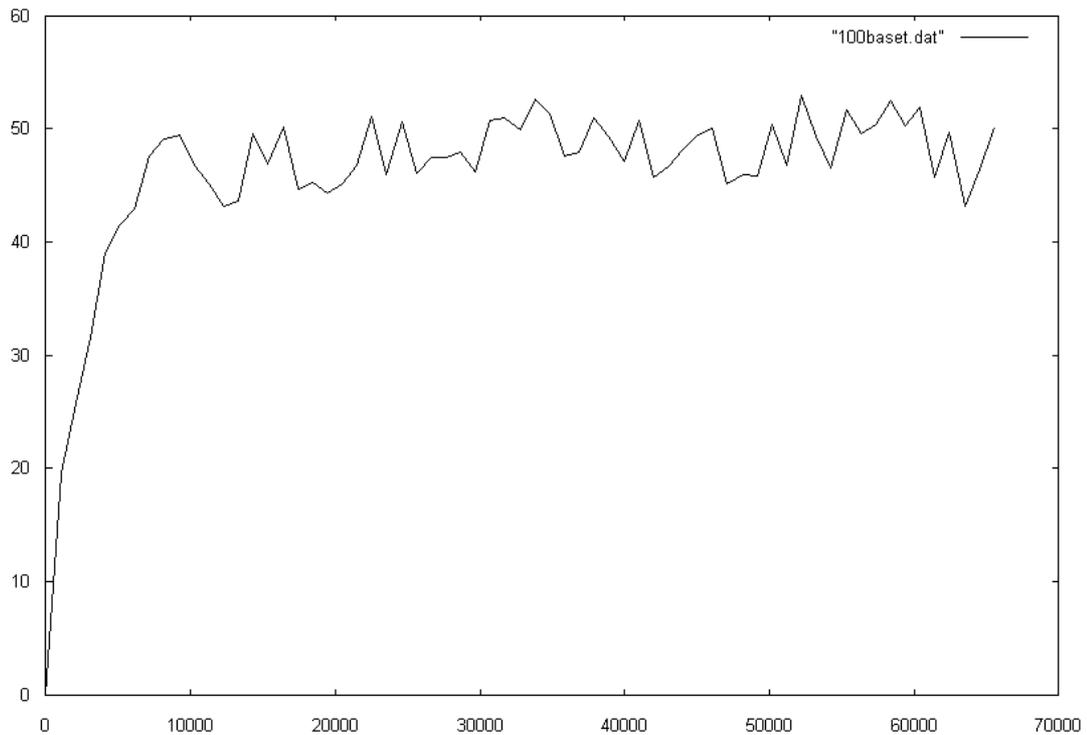


Abbildung 18 Durchsatzmessung mit 100BaseT

Das Messergebnis liegt hier deutlich unter den Erwartungen, ließ sich aber auch durch wiederholte Messdurchläufe nicht wiederlegen.

4.2.3.3 ATM mit 25 Mbps

Hier kommen als Verbindungen der Testrechner mit dem Netzwerk Twisted Pair- Kabel nach Kategorie 5 zum Einsatz.

Das Netzwerk bestand zum Test aus dem modularen zentralen Vermittlungsknoten vom Typ IBM 8260, über den der gesamte Netzverkehr vermittelt wurde. Die für die einzelnen Betriebsarten notwendigen Funktionseinheiten –ATMARP- Server, LES, LECS, und BUS– werden von einem eigenen Switchmodul des Typs IBM 8210 „Multi Switched Services“ breit gestellt.

Die beiden Test- PCs wurden über einen 25 Mbps-Konzentrator- Einschub mit 12 RJ-45- Anschlüssen mit dem Switch verbunden. Ursprünglich war für weitere Tests mit 25 Mbps ein separater Konzentration vom Typ IBM 8282 mit 6 RJ-45- Anschlüssen und

einem 100 Mbps- LWL- Uplink zum Switch vorhanden. Dieser produzierte während der Messungen aber so viele Fehler und Übertragungsabbrüche, dass keine aussagekräftigen Ergebnisse erzielt werden können. Es ließ sich auch nicht umkonfigurieren, da der administrative Zugang zum Gerätemanagement versperrt war.

Es verblieben also die Messungen am Switchmodul, die in den Betriebsarten Classical IP over ATM, LAN Emulation mit Ethernet und LAN Emulation Token Ring durchgeführt wurden.

Da die Nenndatenrate von 25 Mbps auch hier unter den bereits erreichten Durchsätzen liegt, müssten sich realistische Messungen ergeben. Abbildung 19 zeigt die Zusammenfassung:

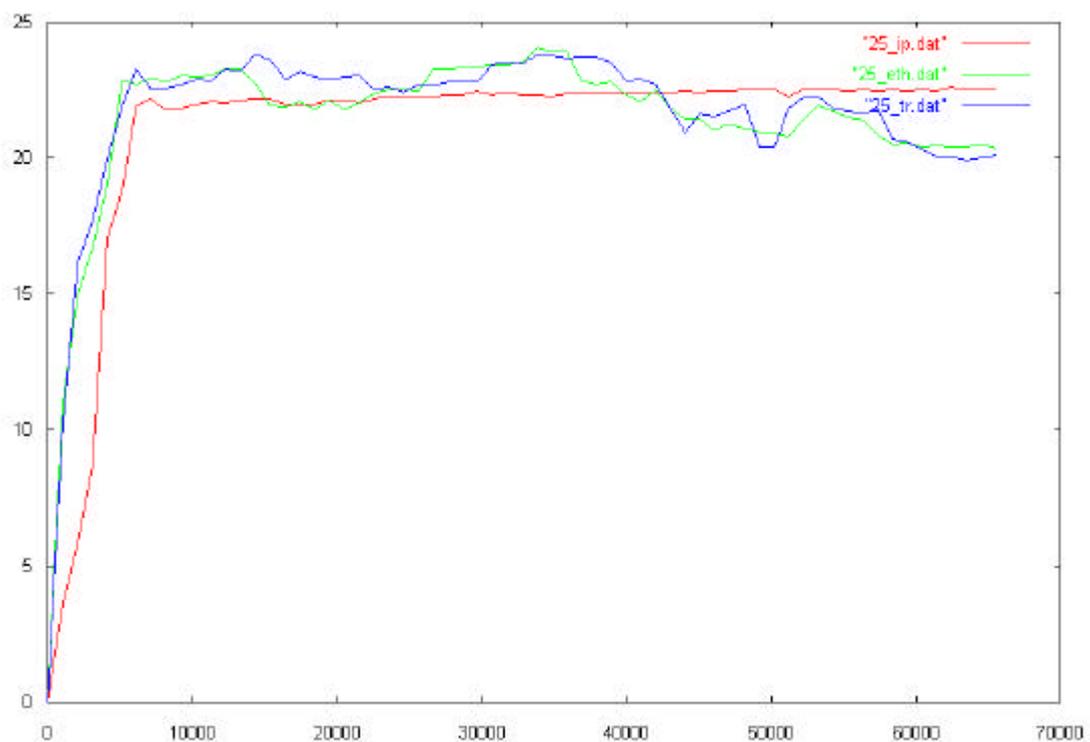


Abbildung 19 ATM- Messungen bei 25 Mbps

Hier decken sich die Messergebnisse wieder mit den Erwartungen. Erkennbar ist, dass die Datenrate von CLIP einen homogenen Verlauf über die Blockgröße aufweist als die beiden LANE- Messungen. Alle Werte liegen im Mittel bei etwa 22 bis 23 Mbps, was ca. 90 Prozent der Nenndatenrate entspricht. Durch die Charakteristik des ATM ist auch nicht zu erwarten, dass diese Leistung bei wachsender Stationszahl einbricht. Einzig große Mengen Broad- und Multicastverkehr können die LAN Emulation überfordern.

4.2.3.4 ATM mit 155 Mbps

Bis auf ein anderes Konzentratormodul mit 4 jeweils 155 Mbps schnellen RJ-45-Anschlüssen unterscheidet sich der Testaufbau hier nicht von dem des letzten Abschnitts.

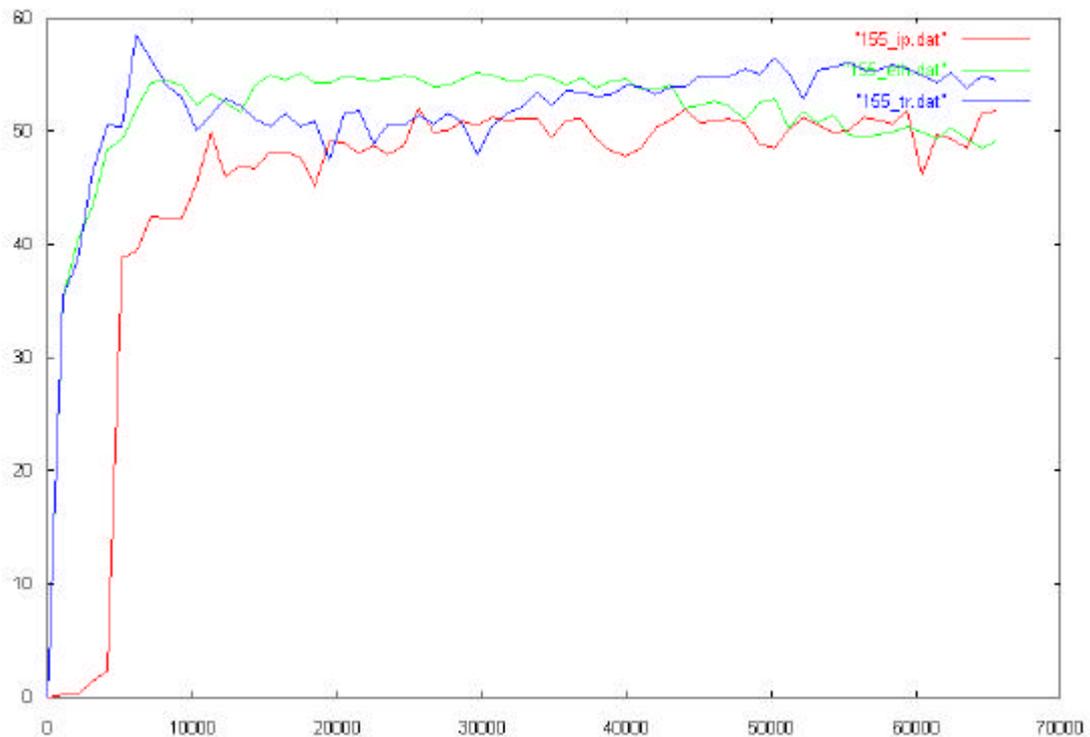


Abbildung 20 ATM- Messungen mit 155Mbps

Abbildung 20 zeigt –ähnlich wie die Messung in Abschnitt 4.2.3.2– ein Ergebnis, dass nicht das Potenzial des Netzwerkes widerspiegelt. Vielmehr muss daraus geschlossen werden, dass die Leistung der eingesetzten PCs in Verbindung mit der verwendeten Software nicht genügt, die Lastgrenzen der Netzumgebungen mit mehr als 60 Mbps zu erreichen. Die Kurven in den Abbildungen und 18 und 20 stellen daher die Leistungsgrenzen der Testumgebung dar.

5 Zusammenfassung und Ausblick

Das Gros der installierten LANs basiert heute auf Ethernet und auch die meisten in der nahen Zukunft realisierten LANs werden auf diese Basis bauen, wenn auch als Fast Ethernet, Gigabit Ethernet oder demnächst auch auf dem bereits absehbaren 10-Gigabit Ethernet. Dazu sind die vorhandenen Erfahrungen mit Ethernet einfach zu umfangreich. Auch haben zurzeit längst nicht alle Anwender einen so großen Bandbreitenbedarf, wie die Marketingabteilungen der Netzwerkhersteller es gerne glauben machen.

Auf eine Technologie umzurüsten, die auf völlig anderen Prinzipien beruht als die schon lange eingesetzten bedeutet neben den erheblichen Investitionen für ATM auch eine aufwändige Weiterbildung des betreuenden Personals. Die Ethernet- Abkömmlinge benötigen keine Umschulungen in solchem Ausmaß, sie erweitern ja bereits Bekanntes. Durch den bereits existierenden Massenmarkt für Ethernetkomponenten kompensieren die Hersteller über große abgesetzte Stückzahlen schnell die eingesetzten Entwicklungskosten und können so die Preise senken und den Markt noch weiter vergrößern.

Bei ATM sieht es dagegen derzeit so aus, dass sich immer mehr Hersteller aus dem ATM- Geschäft zurückziehen, auch IBM stellt heute keine ATM- Kernkomponenten mehr her. Die verbleibenden großen Hersteller konzentrieren sich auf den Telekommunikationsmarkt, der auch weiterhin Hauptanwender der ATM- Technologie. Dieser Trend wird durch die Entwicklung des neuen UMTS- Standards, der auf ATM basiert, noch verstärkt.

Die Tatsache, dass Ethernet nach wie vor keine so konsequente Priorisierung des Datenverkehrs und so differenzierte Dienstegüten bietet wie ATM, wird teilweise dadurch wettgemacht, dass man bei Ethernet bisher einfach die Datenrate um eine Größenordnung erhöht hat. Doch auch hier bietet ATM mit seinen differenzierten und stabilen Dienstegüten eine bessere Skalierbarkeit.

So wird ATM –abseits von WAN- Anwendungen– im LAN- Bereich wohl weiterhin ein Nischenprodukt bleiben. Hauptanwendung hier ist eher der Backbone von Hochleistungsnetzwerken, welche aber im Bereich der Arbeitsstationen weiterhin größtenteils auf Ethernet- Technolgien beruhen werden.

Ausnahmen bilden hier nur Spezialanwendungen, die auf die QoS- Merkmale und Echtzeitfähigkeit der Netzwerke angewiesen sind. Aus den Erfahrungen aus seiner Arbeit bei einem Systemhaus der Rundfunkbranche kann der Autor hier vor allem die großen öffentlichen Sendeanstalten. Sie vertrauen –zumindest im Rahmen des kritischen Sendeausspiels– heute zum großen Teil auf ATM- Hochleistungsnetzwerke.

6 Verzeichnis der Abkürzungen

AAL	ATM Adaption Layer
ATM	Asynchronous Transfer Mode
BUS	Broadcast and Unknown Server
CPCS	Common Part Convergence Sublayer
ELAN	Emulated LAN
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
ILMI	Integrated Local Management Interface
IP	Internet Protocol
ISO	International Standards Organisation
ITU	International Telecommunications Union
LAN	Local Area Network
LE	LAN Emulation
LE_ARP	LAN Emulation Address Resolution Protocol
LEC	LAN Emulation Client
LECS	LAN Emulation Configuration Server
LES	LAN Emulation Server
LLC	Logical Link Control
LUNI	LAN Emulation User-Network Interface
MAC	Medium Access Control
MPoA	Multiprotocol over ATM
MSB	Most Significant Bit
MTU	Maximum Transmission Unit

NHRP	Next Hop Resolution Protocol
NIC	Network Interface Card
OSI	Open Systems Interconnection
PDU	Protocol Data Unit
QoS	Quality of Service
RFC	Request For Comment
SAAL	Signalling AAL
SDH	Synchrone Digitale Hierarchie
SDU	Service Data Unit
SSCS	Service Specific Convergence Sublayer
SVC	Switched Virtual Connection
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UNI	User-Network Interface
VCC	Virtual Channel Connection
VPC	Virtual Path Connection
VCI	Virtual Channel Identifier
VPI	Virtual Path Identifier
WAN	Wide Area Network

7 Literaturverzeichnis

- [1] Comer, Douglas E.:
„Internetworking with TCP/IP, Volume I: Principles, Protocols, and Architecture“,
Prentice Hall, 1995
- [2] Kyas, Othmar:
„Fast Ethernet · Fast Token Ring · Segment Switching“,
Datacom Buchverlag GmbH, 1995“
- [3] Kyas, Othmar:
„ATM-Netzwerke: Aufbau – Funktion - Performance“,
Datacom Buchverlag GmbH, 1996
- [4] Borowka, Petra:
„Brücken und Router: Wege zum strukturierten Netzwerk“,
Datacom Buchverlag GmbH, 1995
- [5] Fischer, S.; Müller, W.:
„Netzwerkprogrammierung unter LINUX und UNIX“,
Reihe UNIX[®] easy, Carl Hanser Verlag, 1996
- [6] Kruglinski, David J.:
„Inside Visual C++ Version 5“, Microsoft Press, 1997
- [7] Proebster, Walter E.:
„Rechnernetze: Technik, Protokolle, Systeme, Anwendungen“,
R. Oldenbourg Verlag, 1998
- [8] Schill, A. u.a.:
„ATM-Netze in der Praxis“,
„Einsatzszenarien, Produktkategorien, aktuelle Standards“,
Addison-Wesley, 1997
- [9] Sinha, Alok K.:
„Network Programming in Windows NT“, Addison Wesley, 1996
- [10] Tanenbaum, Andrew S.:
„Computernetzwerke“, Prentice Hall, 1997
- [11] Windows Sockets 2 API,
„An Interface for Transparent Network Programming under
Microsoft Windows“, Revision 2.2.2, Microsoft, 1997
- [12] Gillhuber, A.:
„Kurze Wege durchs Netz, Multi-Protocol over ATM und IP-Switching“,
Zeitschrift „c’t“, Heft 11/1997, Seite 334ff
- [13] Kuri, J.:
„Fahrplanauskunft Netze, Orientierungshilfen im Netzwerkmarkt“,

Zeitschrift „c't“, Heft 16/1997, Seite 146ff

- [14] <http://www.atmforum.com>
- [15] <http://www.ibm.com/networking>
- [16] Handbuch zur ATM-Netzwerk-Interfacekarte IBM TurboWays 25
- [17] Deutsche Norm DIN EN 50173,
„Anwendungsneutrale Verkabelungssysteme“,
Deutsches Institut für Normung e.V., November 1995
- [18] Rech, Jörg:
“Glas statt Kupfer, Neue Glasfaser-Techniken machen Lichtleiter bis zum
Desktop-PC realistisch“,
Zeitschrift „c't“, Heft 10/2000, Seite 242ff
- [19] ATM Forum:
LAN Emulation over ATM, Version 1,
The ATM Forum Technical Committee, Januar 1995
- [20] Finn, N. u.a.:
LAN Emulation over ATM, Version 2 – LUNI Specification,
The ATM Forum Technical Committee, Juli 1997
- [21] Alexander, C. u.a.:
Multi-Protocol over ATM, Version 1,
The ATM Forum Technical Committee, Juli 1997
- [22] ATM Forum:
Integrated Local Management Interface (ILMI) Specification, Version 4
The ATM Forum Technical Committee, September 1996
- [23] Amsden, P. u.a.:
ATM User-Network Interface (UNI) Signalling Specification, Version 4.0,
The ATM Forum Technical Committee, Juli 1996
- [24] Internet Engineering Task Force, <http://www.ietf.org/>

8 Verzeichnis der Anlagen

8.1 Verbindungskabel zwischen den Ethernet- NICs

8.2 Verbindungskabel zw. IBM8282 und TurboWays 25 NIC

8.3 Quelltexte der Haupt- Threads von NTester

8.4 Inhalt der beiliegenden CD-R

8.1 Verbindungskabel zwischen den Ethernet- NICs

Für die direkte Kopplung zweier Ethernetstationen über den Twisted Pair- Anschluss (RJ-45- Buchse) ist ein Spezialkabel nötig, welches den Ausgang der einen Station mit dem Eingang der anderen verbindet und umgekehrt (siehe Tabelle 7). Dieses Kabel wurden vom Autor angefertigt. Dafür wurden geschirmte Steckverbinder „RJ45“ sowie flexibles Kabel „S/UTP“ mit 4 verdrehten Aderpaaren und einem Gesamtschirm nach Kategorie 5 bzw. Klasse D verwendet.

Ethernetadapter Sender "RJ-45"			Ethernetadapter Empfänger "RJ-45"	
Signal	PIN		PIN	Signal
Receive-	1	-	2	Transmit-
Transmit+	2	-	7	Receive+
Transmit-	3	-	8	Receive-
Receive+	6	-	1	Transmit+

Tabelle 7 Belegung des Crosslink- Kabels

8.2 Verbindungskabel zw. IBM8282 und TurboWays 25 NIC

Zur Verbindung der ATM-Adapterkarten mit dem Konzentrador sind auf Grund unterschiedlicher Signalbelegung der Anschlüsse spezielle Kabel nötig. Die verschiedenen Signalbelegungen resultieren daraus, dass die Firma IBM vor der Definition einer einheitlichen Anschlussbelegung durch das ATM-Forum die Belegungen aus dem Standard IEEE 802.5 für Token Ring übernahm. Zur Unterscheidung sind die Anschlüsse nach IEEE 802.5 mit einem grünen Punkt versehen („green dot“), während die ATM-Forum-konformen mit einem orangefarbenen Punkt („orange dot“) gekennzeichnet sind.

Die Kabel wurden vom Autor nach folgendem Anschlusschema angefertigt. Dafür wurden geschirmte Steckverbinder „RJ45“ sowie flexibles Kabel „S/UTP“ mit 4 verdrehten Aderpaaren und einem Gesamtschirm nach Kategorie 5 bzw. Klasse D verwendet.

IBM 8282 "green dot"			IBM TurboWays 25 "orange dot"	
Signal	PIN		PIN	Signal
Receive-	3	-	2	Transmit-
Transmit+	4	-	7	Receive+
Transmit-	5	-	8	Receive-
Receive+	6	-	1	Transmit+

Tabelle 8 Pinbelegung der ATM-Anschlüsse nach [16]

8.3 Quelltext der Haupt- Threads von NTester

```
////////////////////////////////////
// Serverthread
// Wartet auf eingehende Verbindung, erledigt Handshake und beantwortet
// Testdaten
////////////////////////////////////

UINT ServerThreadProc(LPVOID pParam)
{
    CSockAddr saClient; // SockAddr des Client
    CBlockSock sockConnect; // Socket, der die Verbindung uebernimmt
    char recvBuf[_MAXBUFFER_]; // Puffer fuer empfangene Bytes
    char pingREQ[] = _PINGREQ_; // Statusmeldungen fuer Handshake
    char pingACK[] = _PINGACK_;
    char testREQ[] = _TESTREQ_;
    char testACK[] = _TESTACK_;
    UINT pSize; // momentane Nachrichtengroesse
    ULONG BytesToRecv, recvBytes, pNum;
    CString logStr; // fuer Logmeldungen (Ausgabefenster)

    try { // Fehler werden mit eigener Exception abgefangen (s.u.)
        if(!sockListen.Accept(sockConnect, saClient)) // Socket wurde von Ansicht oder Applikation geschlossen
        {
            bIsServer = FALSE;
            return 0;
        }
        else if (bLogSockFkt) LogMessage(pParam, "Server: Socket.Accept().", TRUE); // neue Verbindung steht...

        // WorkerThread starten -> mehrere Verbindungen parallel
        AfxBeginThread(ServerThreadProc, pParam, THREAD_PRIORITY_HIGHEST);

        // Anforderung vom Client lesen
        sockConnect.Receive(recvBuf, _MAXBUFFER_, _CONNTIMEOUT_);
        if (bLogSockFkt) LogMessage(pParam, "Server: Socket.Receive().", TRUE);

        if ( memcmp(recvBuf, pingREQ, strlen(pingREQ)) == 0 ) // Ping Request
        {
            if (bLogConn)
            {
                logStr.Format("PING von %s:%u.", saClient.DottedDecimal(), saClient.Port());
                LogMessage(pParam, (char*) LPCTSTR(logStr), TRUE);
            }
            sockConnect.Write(pingACK, strlen(pingACK), _CONNTIMEOUT_);
            if (bLogSockFkt) LogMessage(pParam, "Server: Socket.Write().", TRUE);
        }
    }
}
```

```

sockConnect.Close();
if (bLogSockFkt) LogMessage(pParam, "Server: Socket.Close().", TRUE);
}
else if ( memcmp(recvBuf, testREQ, 4) == 0 )
{
    // Test Request
    // Testparameter lesen
    pSize = atoi(recvBuf+5);
    pNum = atoi(recvBuf+11);
    BytesToRecv = pNum * pSize;

    if (bLogConn)
    {
        logStr.Format("- Erwarte\t%u Bytes von %s:%u", BytesToRecv, saClient.DottedDecimal(), saClient.Port());
        LogMessage(pParam, (char*) LPCTSTR(logStr), TRUE);
    }

    sockConnect.Write(testACK, strlen(testACK), _CONNTIMEOUT_); // mit OK bestaetigen
    if (bLogSockFkt) LogMessage(pParam, "Server: Socket.Write().", TRUE);

    recvBytes = 0;
    while (recvBytes != BytesToRecv) { // auf Daten warten
        recvBytes += sockConnect.Receive(recvBuf, pSize, _CONNTIMEOUT_);
        if (bLogSockFkt) LogMessage(pParam, "Server: Socket.Receive().", TRUE);
    }

    if (bLogConn)
    {
        logStr.Format("+\t%u Bytes von %s:%u empfangen.", recvBytes, saClient.DottedDecimal(), saClient.Port());
        LogMessage(pParam, (char*) LPCTSTR(logStr), TRUE);
    }

    sockConnect.Write(testACK, strlen(testACK), _CONNTIMEOUT_); // mit OK bestaetigen
    if (bLogSockFkt) LogMessage(pParam, "Server: Socket.Write().", TRUE);

    sockConnect.Close();
    if (bLogSockFkt) LogMessage(pParam, "Server: Socket.Close().", TRUE);
}
else
{
    // Verbindungswunsch kommt nicht von NTester-Client -> Fehlermeldung.
    // Ausgabe auf 100 Zeichen begrenzen
    recvBuf[101] = 0;
    logStr.Format("!!!\tAnforderung von %s:%u nicht bekannt:\r\n%s.", saClient.DottedDecimal(), saClient.Port(), recvBuf);
    LogMessage(pParam, (char*) LPCTSTR(logStr), TRUE);
    sockConnect.Close();
    if (bLogSockFkt) LogMessage(pParam, "Server: Socket.Close().", TRUE);
}
} // try Fehler innerhalb des try-Blocks behandeln (Ausgabe im Fenster).

```

```
        catch(CBlockSockException* pe)
        {
            LogBlockSockException(pParam, "Server:", pe);
            pe->Delete();
        }
        return 0;
    } // ServerThreadProc // und Ende
```

```

////////////////////////////////////
// Clientthread
// Initiiert den Datenaustausch (Handshake) und wickelt Test-Dateneubertragung
// entsprechend den Einstellungen ab.
////////////////////////////////////

UINT ClientThreadProc(LPVOID pParam)
{
    CBlockSock sockTest;                // Socket fuer Test
    CSockAddr saServer;                 // Adressdaten des Servers
    char sendBuf[_MAXBUFFER_];         // Sendepuffer
    char recvBuf[20];                 // Puffer fuer Rueckmeldungen vom Server
    UINT recvBytes;
    CString Request, logStr;
    DWORD _starttime, _endtime;       // Start-, End- Zeitmarken
    double _runtime, _rate;           // Laufzeit, -Rate
    double _average = 0;              // kumulierter Mittelwert -Rate

    UINT _vol = Volume * 1048576;     // Multiplikator fuer _M_ in MByte
    UINT _curmsgsize = MsgInit;       // momentane Nachrichtengroesse
    UINT _msgcount;                   // Anzahl notwendiger Nachrichten
    CStdioFile f;                     // Exportdatei fuer Ergebnisse

    memset( sendBuf, '*', _MAXBUFFER_ ); // Sendepuffer fuehlen
    bIsClient = TRUE;

    // Exportdatei anlegen bzw. leeren
    if (f.Open(strExportFile, CFile::modeCreate | CFile::modeWrite)) f.Close();
    else AfxMessageBox("Exportdatei kann nicht geöffnet werden.", MB_OK | MB_ICONSTOP);

    LogMessage(pParam, "Client- Thread gestartet.\r\n", TRUE);

    try
    {
        // Init der Adressdaten des Servers (ggf. Domainnamen auflösen)
        if (bIsNumAdr) saServer = CSockAddr(strServerAdr, (USHORT) Port);
        else
        {
            saServer = CBlockSock::GetHostByName(strServerAdr, Port);
            if (bLogSockFkt) LogMessage(pParam, "Client: GetHostByName().", TRUE);
        }

        while (_curmsgsize <= MsgStop) // Schleife fuer Nachrichtengroesse
        {
            _msgcount = _vol / _curmsgsize; // Wieviele Nachrichten werden benoetigt?
            Request.Format("TEST_%05d_%09d", _curmsgsize, _msgcount);
        }
    }
}

```

```

// Infozeile ausgeben
if (bLogConn)
{
    logStr.Format(" %u Bytes:", _curmsgsize);
    LogMessage(pParam, (char*) LPCTSTR(logStr), TRUE);
}

_average = 0;

// Schleife fuer Wiederholungen zur Mittelwertbildung
for (UINT _curloop = 1; _curloop <= Loops; ++_curloop)
{
    if (bLogConn)
    {
        logStr.Format(" Durchgang %u von %u:", _curloop, Loops);
        LogMessage(pParam, (char*) LPCTSTR(logStr), TRUE);
    }

    sockTest.Create(); // Socket oeffnen
    if (bLogSockFkt) LogMessage(pParam, "Client: Socket.Create().", TRUE);

    sockTest.Connect(saServer);
    if (bLogSockFkt) LogMessage(pParam, "Client: Socket.Connect().", TRUE);

    sockTest.Write( Request, strlen(Request), _CONNTIMEOUT_ );
    if (bLogSockFkt) LogMessage(pParam, "Client: Socket.Write().", TRUE);

    recvBytes = sockTest.Receive( recvBuf, sizeof(recvBuf), _CONNTIMEOUT_ );
    if (bLogSockFkt) LogMessage(pParam, "Client: Socket.Receive().", TRUE);
    if (bLogConn)
    {
        logStr.Format("Client: %d Bytes empfangen: %s", recvBytes, recvBuf);
        LogMessage(pParam, (char*) LPCTSTR(logStr), TRUE);
    }

    if ( strcmp(recvBuf, _TESTSTACK_) == 0 )
    {
        if (bLogConn) LogMessage(pParam, "Client: Server sendete OK.", TRUE);

        // los gehts...
        _starttime = GetTickCount();
        for (UINT _count=1; _count<=_msgcount; ++_count)
        { // Schleife f3r einzelne Nachrichten
            sockTest.Write( sendBuf, _curmsgsize, _CONNTIMEOUT_ );
            if (bLogSockFkt) LogMessage(pParam, "Client: Socket.Write().", TRUE);
        }
        _endtime = GetTickCount();
    }
}

```

```

Sleep(1000); // kurze Pause zur „Beruhigung“ des Netzwerks

recvBytes = sockTest.Receive( recvBuf, sizeof(recvBuf), _CONNTIMEOUT_ );
if (bLogSockFkt) LogMessage(pParam, "Client: Socket.Receive().", TRUE);
if (bLogConn)
{
    logStr.Format("Client: %d Bytes empfangen: %s", recvBytes, recvBuf);
    LogMessage(pParam, (char*) LPCTSTR(logStr), TRUE);
}
if ( strcmp(recvBuf, _TESTACK_) == 0 )
    if (bLogConn) LogMessage(pParam, "Client: Server sendete OK.", TRUE);

_runtime = double(_endtime - _starttime); // Laufzeit in Millisekunden

// Umrechnung in Megabits per second (Mbps)
// 1 Mbps = 1.000.000 bps !!!
// ( _curmsgsize * _msgcount * 8 * 1000 ) / ( 1000 * 1000 * _runtime )
// kann gekürzt werden...
if (_runtime > 0)
{
    _rate = double(_curmsgsize)/125;
    _rate *= _msgcount;
    _rate /= _runtime;
}
else _rate = 0;
_average += _rate / Loops; // Mittelwert kumulieren

if (bLogConn)
{
    logStr.Format("%5.3f ms\t= %3.2f Mbps.", _runtime, _rate);
    LogMessage(pParam, (char*) LPCTSTR(logStr), TRUE);
}
} // if
else LogMessage(pParam, "Client: Server ist nicht bereit.", TRUE);

sockTest.Close();
if (bLogSockFkt) LogMessage(pParam, "Client: Socket.Close().", TRUE);

} // for

// Ergebnis ausgeben (Bildschirm und Datei)
if (bLogResults)
{
    logStr.Format("Nachrichtengröße: %u Bytes: %3.3f Mbps.", _curmsgsize, _average);
    LogMessage(pParam, (char*) LPCTSTR(logStr), TRUE);
}

```

```

// Testergebnis in Datei schreiben
    if (f.Open(strExportFile, CFile::modeWrite))
    {
        f.SeekToEnd();
        logStr.Format("%u, %3.3f\r\n", _curmsgsize, _average);
        f.WriteString((char*) LPCTSTR(logStr));
        f.Close();
    }
    else AfxMessageBox("Exportdatei kann nicht geöffnet werden.", MB_OK | MB_ICONSTOP);

    // Weiter mit nächster Nachrichtengroesse
    _curmsgsize += MsgDelta;
} // while
} // try
catch (CBlockSockException* e)
{
    LogBlockSockException(pParam, "Client:", e);
    e->Delete();
}

bIsClient = FALSE;
LogMessage(pParam, "Test beendet.\r\n", TRUE);
LogMessage(pParam, "Client- Thread beendet.", TRUE);

return 0; // und fertig.
} // ClientThreadProc

```

8.4 Inhalt der beiliegenden CD-R

Auf der CD-R sind alle Quelldateien des NTester und der Text der vorliegenden Arbeit im Format Postscript und PDF enthalten. Weiterhin sind Dokumente der verschiedenen Standardisierungsgremien, welche online im Internet verfügbar sind, beigefügt.

9 Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Leipzig, 21. April 2001

André Staehler