Universität Leipzig

Fakultät für Mathematik und Informatik

Institut für Informatik

Diplomarbeit

Thema:

Entwicklung und Untersuchung eines IP-Multicast-Systems zur Realisierung von Videokonferenzen auf Breitbandnetzen

ausgeführt bei

Prof. Spruth
Lehrstuhl für technische Informatik
Universität Leipzig

vorgelegt von

Nguyen Ky Giang
Matrikelnummer 7051797

Leipzig, den 1.3.1998

Erklärung an Eides Statt

Hiermit versichere ich, daß die vorliegende Arbeit ohne unzulässige Hilfe und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt wurde und die aus fremden Quellen direkt oder indirekt übernommenen Gedanken in der Arbeit als solche kenntlich gemacht sind.

(*Nguyen Ky Giang*)

Inhalt

Inhalt ***4***

Vorwort ***8***

Abkürzungen ***106***

Abbildungen ***15***

Tabellen ***9***

Gleichungen ***10***

1 Einleitung ***1***

2 Aspekte einer Videokonferenz und deren Anforderungen ***15***

Modell eines Videokonferenzsystems **15**

2.2 Übertragungsbandbreite **16**

2.3 Ende-zu-Ende Verzögerung **17**

2.4 Synchronisation der Datenströme **18**

2.5 Delay Jitter und Jitterkompensation **19**

3 Kommunikationssysteme ..... ***xxi***

3.1 TCP/IP ..... **xxi**

3.1.1 Architektur xxi

3.1.2 Internet Protokoll (IP) xxiii

3.1.3 Transmission Control Protocol (TCP) xxv

3.1.4 User Datagram Protocol (UDP) xxv

3.1.5 IP Multicast xxvi

3.1.6 Das Windows Socket Interface xxviii

3.1.7 Vergleich beider Protokolle UDP und TCP xxx

3.2 Transportnetzwerk ..... **xxxiii**

3.2.1 Die Leistungsgrenzen von Ethernet xxxiii

3.2.2 Token Ring und FDDI xxxiv

3.2.3 ATM: eine neue Technologie xxxvi

3.3 Asynchronous Transfer Mode (ATM) ..... **xxxvii**

3.3.1 Charakteristika von ATM xxxviii

3.3.2 Architektur von ATM-Netzwerken xl

3.3.3 IP über ATM xliv

3.3.4 ATM LAN-Emulation xliv

4 Multimedia ..... *l*

4.1 WAVE-Format für digitale Audiodaten ..... **l**

4.2 Kompressionsalgorithmen für Motion Video ..... **li**

4.2.1 Motion-JPEG li

4.2.2 MPEG ..... **Fehler! Textmarke nicht definiert.**

4.2.3 Vergleich beider Verfahren 60

4.3 Video-Overlay **61**

5 Implementierung ..... *lxiii*

5.1 Entwicklungsumgebung und Übersicht ..... **lxiii**

5.2 Das Programmierkonzept ..... **lxv**

5.3 Das Audiomodul ..... **lxvi**

5.3.1 Übersicht lxvi

5.3.2 Audioaufnahme lxvii

5.3.3 Audioformat lxviii

5.3.4 Audioübertragung lxix

5.3.5 Audioempfang lxix

5.3.6 Die Kompensierung der Übertragungsschwankungen lxx

5.3.7 Ausgleich von Paketverlusten lxxi

5.4 Das Videomodul ..... **lxxii**

5.4.1 Videobild-Aufnahme und Übertragung der komprimierten Frames lxxii

5.4.2 Dekompression und Wiedergabe der empfangenen Frames lxxvi

5.4.3 Synchronisation mit dem Audiomodul lxxviii

5.5 Das Netzwerkmodul ..... **lxxix**

5.5.1 Das Übertragungsprotokoll lxxxi

5.5.2 Fragmentierung/Reassemblierung der Datenpakete lxxxiii

5.5.3 Multicast lxxxvi

5.5.4 Multitasking lxxxviii

5.6 Das Benutzermodul ..... **xci**

5.7 Performance-Untersuchung ..... **xciii**

5.7.1 Kompressionsfaktor/Datenrate xciii

5.7.2 Verhältnis von Paketgröße zu Übertragungszeit xciv

5.7.3 Kompatibilität der eingesetzten Videokarten xcvi

5.7.4 Beschränkungen bezüglich der Betriebsysteme c

5.7.5 Kommunikationsmöglichkeiten c

5.7.6 Prozessorauslastung ci

5.7.7 Datenschutz ci
5.7.8 Management-Ebene und Zusatzfunktionen cii

6** Zusammenfassung* ***105

Literatur ***106***

Anhang ***110***

Abbildungen

Abbildung 1: Das Videokonferenzmodell. *15*
Abbildung 2: TCP/IP-Protokoll-Familie ..... *xxii*
Abbildung 3: WinSock und die zugrundeliegenden Protokolle ..... *xxix*
Abbildung 4: Effizient der Bandbreitenausnutzung bei Ethernet [NETZWERK92] . ***Fehler! Textmarke nicht definiert.***
Abbildung 5: Das Prinzip von ATM [ATM96] ..... *xxxviii*
Abbildung 6: B-ISDN-Protokollschichten ..... *xl*
Abbildung 7: UNI-Zellformat ..... *xli*
Abbildung 8: NNI-Zellformat ..... *xlii*
Abbildung 9: B-ISDN Protokoll-Referenzmodell ..... *xliii*
Abbildung 10: LAN-Emulation Configuration-Server (LECS) ..... *xlvii*
Abbildung 11: Der Verkehrfluß des LAN-Emulations-Dienstes [ATM96] ..... *xlviii*
Abbildung 12: Schema des JPEG-Encoders ..... *lii*
Abbildung 13: Zickzack-Abtastung des 8x8-Blocks bei DCT [MUL95] ..... *liv*
Abbildung 14: Unterschiedliche Quantisierungen (a) fein, (b) grob [MULT95] ..... *lv*
Abbildung 15: Arten der Einzelbilder in MPEG [STEIN94] *59*
Abbildung 16: Wirkungsweise des Video-Overlay *62*
Abbildung 17: Übersicht des Programmaufbaus ..... *lxv*
Abbildung 18: Audioaufnahme und -wiedergabe ..... *lxvi*
Abbildung 19: Multicast-Interface Netzwerk-Klasse ..... ***Fehler! Textmarke nicht definiert.***
Abbildung 20: UDP-Paket und Video-Frames ..... *lxxxii*
Abbildung 21: der Aufbau eines Sequenzkopfes. ..... *lxxxii*
Abbildung 22: Fragmentierung eines Frames ..... *lxxxiv*
Abbildung 23: Darstellung einer erfolgreichen Defragmentierung ..... ***Fehler! Textmarke nicht definiert.***
Abbildung 24: Darstellung einer nicht erfolgreichen Defragmentierung eines Frames , z.B. Nr. 20 ..... *lxxxv*
Abbildung 25: Arbeitsweise von RecvThread() ..... *lxxxviii*
Abbildung 26: Verhältnis Kompressionsfaktor zu Datenrate mit ¼ PAL-Bildqualität (M-JPEG Verfahren) ..... *xciii*
Abbildung 27: Verhältnis Kompressionsfaktor zu Datenrate bei PAL-Bildqualität (M-JPEG Verfahren) ..... *xciv*
Abbildung 28: Verhältnis Paketgröße zu Übertragungszeit ..... *xcv*

Tabellen

Tabelle 1: Übertragungsbandbreite und Anwendungen [ATM96] *17*
Tabelle 2: Auswirkung der Übertragungsverzögerung auf die Qualität von Audioübertragungen [ATM96] *18*
Tabelle 3: Zeitliche Anforderungen an die Intersynchronisation [MÜL94] *19*
Tabelle 4: Aufbau eines IP-Datagrammkopfes ..... *xxiii*
Tabelle 5 Aufbau eines UDP-Protokollkopfes ..... *xxvi*
Tabelle 6: Die Echzeitleistungsgrenzen von Ethernet, Token Ring und FDDI ..... *xxxvi*
Tabelle 7: Serviceklassen und AAL-Typen [ATM96] ..... *xliv*
Tabelle 8: Qualitäten der Audiodaten mit dem WAVE-Format ..... *l*
Tabelle 9: Vergleich verschiedener Eigenschaften von M-JPEG und MPEG *61*
Tabelle 10: optimale Warteschlangenlänge beim Empfänger in Abhängigkeit von der Audio-Abtastfrequenz bei der Aufnahme ..... ***Fehler! Textmarke nicht definiert.***
Tabelle 11: Synchronisation zwischen den Audio- und Videoströmen ..... ***Fehler! Textmarke nicht definiert.***
Tabelle 12: Kompatibilität der eingesetzten Videokarten DC30 von MIRO zu AVMaster und FPS60 von FAST ..... *xcix*
Tabelle 13: Verschiedene Optionen der Kommunikation ..... *ci*

Gleichungen

Gleichung 1: Umkodierung von RGB-Werte nach YUV-Werte ..... *lii*
Gleichung 2: Transformationsgleichung bei DCT ..... *liii*
Gleichung 3: gesamte Ende-zu-Ende Verzögerung (a) ..... *lxx*
Gleichung 4: gesamte Ende-zu-Ende Verzögerung (b) ..... *lxx*
Gleichung 5: Bestimung der Warteschlangeslänge ..... *lxxi*

Vorwort

1 Einleitung

Leistungsfähige Computer gehören heutzutage zum täglichen Leben und spielen besonders in der Datenkommunikation eine große Rolle. Gerade in den letzten Jahren ist es in diesem Bereich zu einer starken Leistungssteigerung gekommen. Durch eine schnelle Steigerung der Prozessorleistung, große Speicherkapazität und niedrige Kosten ist der PC nicht nur ein „Rechner“, sondern mehr ein Rechen-, Multimedia- und Kommunikationsgerät geworden. Die Weiterentwicklung der Datenkommunikation wird dank immer schnellerer und sicherer Computernetzwerke forciert. Die analoge Kommunikation (Telefon) wird in Zukunft durch digitale Kommunikation z.B. Videoconferencing ersetzt, wobei nicht nur das Hören, sondern das Mitsehen und der Datenaustausch zwischen mehreren Teilnehmern ermöglicht werden. Anspruch und Interesse an einer solchen Kommunikation werden heutzutage immer stärker. So existieren bereits verschiedene Konzepte und kommerzielle Produkte für Videokonferenz-Technik. Standardisierte Konzepte wie H.320 für Schmalband-Übertragung finden in der Praxis schon viele Anwendungen und wurde von verschiedenen Anbietern (SIEMENS,TELEKOM, …) realisiert. Diese können aber auf Grund der schmalen Übertragungsbandbreiten nicht für alle Belangen eine genügende Bildqualität bieten. Für Breitband-Übertragung sind bisher meist spezielle, nicht standardisierte Anwendungen entwickelt worden (z.B. U-Bahn Überwachungssystem von SIEMENS oder Videoconferencing over ATM im GMD-FOKUS-Institut). Solche Systeme sind jedoch teuer und finden deshalb nur in wenigen speziellen Bereichen Anwendung.

Am Institut für Informatik der Leipziger Universität wird an einem Projekt gearbeitet, welches eine Übertragung von Videosequenzen in PAL-Qualität auf Hochgeschwindigkeits-Netzen in Echzeit auf der Basis herkömmlicher Technik ermöglichen soll. Die vorliegende Arbeit soll mithelfen, die dabei auftretenden Probleme zu lösen und Entwicklungs- und Implementierungsvorschläge unterbreiten.

Die Kompressionstechnik für Videosequenzen wurde in den letzten Jahren beschleunigt entwickelt. Dadurch ist für den PC-Bereich kostengünstige Videohardware verfügbar, welche die Kompression bzw. Dekompression von Videodaten mit hoher Qualität in Echtzeit realisiert. Es ist daher möglich, verstärkt zu Anwendungen zu kommen, die kostengünstig neue Konzepte und Hardware nutzen. Es wird in völlig neue Qualitätsbereiche vorgestoßen - im Fall der Video (Konferenz-)-Technik z.B. PAL-Qualität erlaubt.

In der vorliegenden Arbeit wird ein Videokonferenz-System entwickelt und untersucht, welches durch den Einsatz vorhandener, handelsüblicher Videokarten und PC die Übertragung von Video-/Audioströmen mit hoher Qualität in Echtzeit von einem Sender zu mehreren Empfängern realisiert. Die in PAL-Qualität aufgenommenen Video- und Audiosignale werden komprimiert, über ein Netz zum Empfänger verschickt, dekodiert und mit möglichst kleiner Verzögerung dort dargestellt.

Dazu ist ein Netzwerk-Interface implementiert worden, das durch UDP/IP Protokolle einen transparenten Zugriff auf die darunterliegenden Netzwerke ermöglicht. Im Rahmen der Untersuchungen wurde eine Möglichkeit zur effektiven Verarbeitung der Video- und Audio-Datenströme gefunden und die Kompatibilität von Videoadaptern verschiedener Hersteller weitgehend hergestellt, um eine kostengünstige Kombination kommerzieller Hardware-Produkte zu ermöglichen. Die durchgeführten Untersuchungen belegen die Wahl des Herangehens und der verwendeten Parameter. Zusätzlich ist eine ansprechende Benutzerschnittstelle für das Windows System erstellt worden.

Die Erstellung der Arbeit gliederte sich in folgende Schritte:

- Einrichtung der für die Entwicklung notwendigen Systemumgebung in Hinsicht auf Hard- und Software
- Entwicklung des Videokonferenz-Systems
- Implementierung der Module und der gesamten Anwendung
- Untersuchung der Systemeigenschaften

Im Kapitel 2 werden die Eigenschaften von Videokonferenzen und die entsprechenden Anforderungen an die Hard- und Software erörtert.

Kapitel 3 behandelt die Grundlagen des TCP/IP-Protokolles und gibt eine kurze Einführung in ATM. Es wird insbesondere auch auf die Windows-Socketschnittstelle und IP-Multicast eingegangen. Anschließend zeigt ein Vergleich der Transportprotokolle UDP und TCP, daß UDP unter den gegebenen Bedingungen in Videokonferenzen besser geeignet ist.

In Kapitel 4 werden die verschiedenen Kompressionsverfahren und die Video-Overlay-Technik vorgestellt.

Die Entwicklung des Konferenz-Systems wird in Kapitel 5 dargestellt. Es wird die Implementierung auf einem PC-System aus 3 Geräten diskutiert. Die Eigenschaften werden untersucht, um die günstigsten Parameter des Systems zu finden.

Kapitel 6 enthält eine kurze Zusammenfassung der Ergebnisse und einen Ausblick auf weitere notwendige Aufgaben zur Verbesserung des Systems.

Im Anhang sind eine Tabelle über Inhalt und Aufbau des Audiodatenformats *.WAV und die wesentlichen Abschnitte des Programm-Quelltextes gegeben.

2 Aspekte einer Videokonferenz und deren Anforderungen

Dieses Kapitel stellt die Funktionsweise eines herkömmlichen Videokonferenzsystems vor. Es wird auf die bestimmenden wichtigen Eigenschaften eines Systems, die möglichen, erreichbaren Qualitäten sowie die daraus folgenden Anforderungen an das System eingegangen.

2.1 Modell eines Videokonferenzsystems

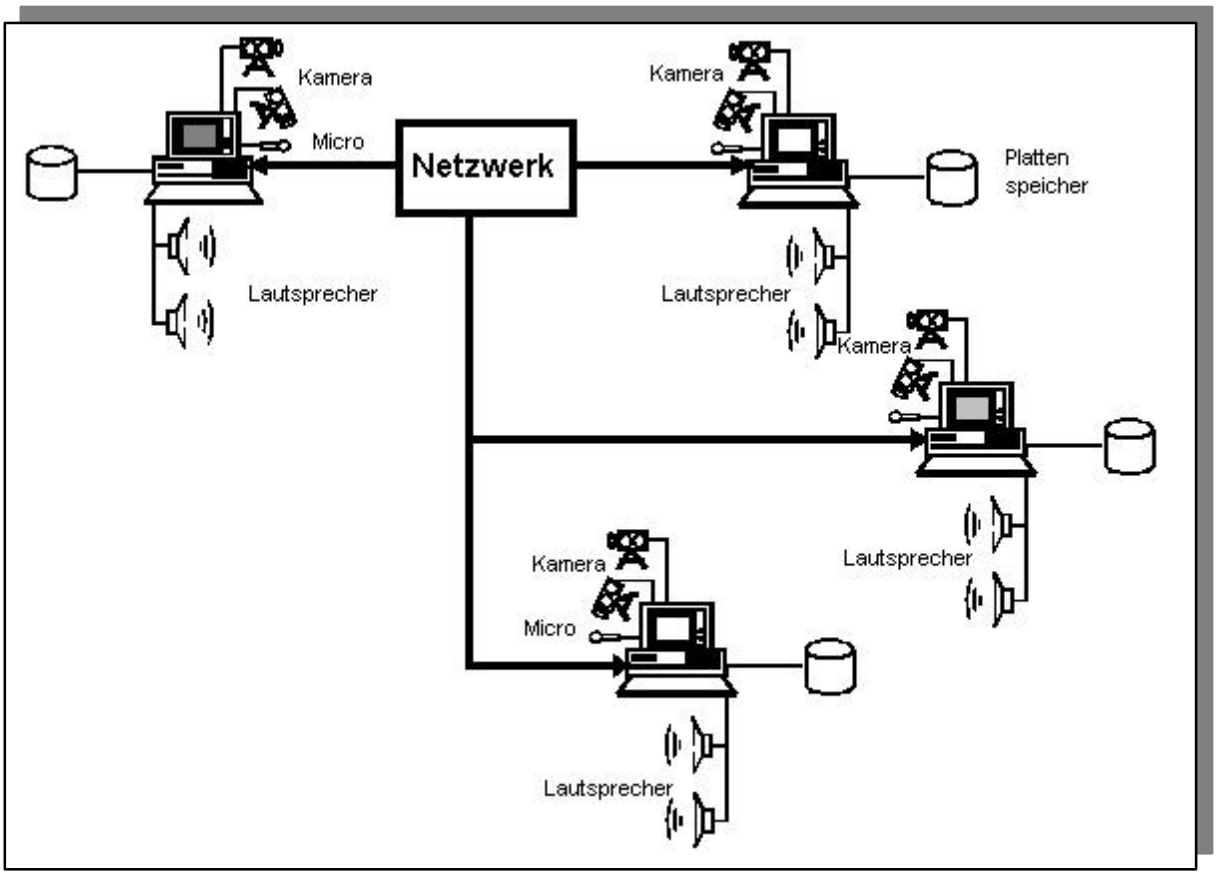


Abbildung 1: Das Videokonferenzmodell.

Abbildung 1 gibt einen groben Überblick über das Modell und die Funktionsweise eines Videokonferenzsystems. Die ganze Arbeitsweise besteht aus drei wesentlichen Schritten: Die Aufnahme der Video- und Audiodaten, die Übertragung und die Wiedergabe dieser Daten beim Empfänger. Die Videosignale werden zuerst von der Kamera aufgenommen. Anschließend werden die einzelnen Bilder komprimiert und vom Netzwerkadapter über das darunterliegende Netzwerk an mehrere Empfangstationen geschickt. Die übertragenen Bilder werden beim Empfänger entsprechend ihrer Komprimierung dekodiert und wiedergegeben. Analog werden Audiodaten aufgenommen und übertragen. Der

Empfänger leitet dann die notwendigen Maßnahmen für eine möglichst gute Synchronisation bei der Wiedergabe der Audio- und Videoströme ein.

Für Videokonferenzsysteme existieren zur Zeit verschiedene Konzepte, die sich in Funktionalität und Qualität stark voneinander unterscheiden. Die Qualität einer Videokonferenz wird durch folgende Aspekte bestimmt:

- die Qualität der Videobilder (klein oder hochauflösend)
- die Qualität der Audio-Kommunikation (Verständlichkeit).

Die meisten der zur Zeit verfügbaren Videokonferen-Systeme bieten Übertrag-ungen auf Schmalband-Netzen (ISDN[1]-2 Kanäle) und damit mindere Bildqualitäten. Die Funktionalität einer Videokonferenz wie der Datenaustausch, Whiteboard[2], Application Sharing[3] oder die Konferenzkontrolle sind von System zu System unterschiedlich implementiert und können nach Bedarf einbezogen werden.

Es werden in den folgenden Abschnitten die Eigenschaften von Videokonferenzen und deren Anforderungen an das Konferenzsystem besprochen. Dabei sind drei wichtige Qualitätsparameter bei der Übertragung von Multimedia-Applikationen:

- die erforderliche Bandbreite des Transportnetzes,
- die gesamte Ende-zu-Ende Verzögerung und
- die Synchronisation von Datenströmen.

Es wird außerdem auf eine wichtige Eigenschaft, die Schwankung der Übertragungsverzögerung, welche eine große Rolle bei der Wiedergabe spielt, eingegangen.

2.2 Übertragungsbandbreite

Der Bandbreitenbedarf für heutige Multimedia-Applikationen variiert stark, von wenigen Kbit/s (CD-Lexikon ca. 30 Kbit/s) über 10 Mbit/s (Video – kleine Auflösung) bis zu 900 Mbit/s (HDTV[4]). Ein hoher Bedarf geht auf eine große zu übertragenden Datenmenge zurück. Ein Bildinhalt in PAL-Qualität besteht aus 720 * 576 Bildpunkten, wobei jeder Bildpunkt aus 3 Byte (24 Bits) Farbinformationen besteht. Es müssen pro Bild ca. 9,5 Mbit übertragen werden. Im Fall einer 64-Kbit/s-Leitung dauert die Übertragung ca. 150 Sekunden, im Falle von 155 Mbit/s nur 0,06 Sekunden. Um 25 Bilder dieser Größe in einer

[1] ISDN Intergrated Service Digital Network

[2] eine weiße Tafel, auf die alle Teilnehmer schreiben können

[3] ermöglicht den Konferenzteilnehmern das Zusammenarbeiten an Dokumente

[4] High Definition Television - sehr hohe Fernsehqualität

Sekunde übertragen zu können, muß die Bandbreite mindestens 270 Mbit/s betragen.

Informationsdienst	Übertragungsbandbreite	Typische Anwendung
Bildübertragung	1 Mbit/s	Monochrome Bilder
	1-10 Mbit/s	Farbbilder
	10-100 Mbit/s	Hochauflösende Farbbilder
Video/Multimedia Konferenzen	1 Mbit/s	Bildausschnitt: sprechende Köpfe
	1-10 Mbit/s	Kleiner Bildausschnitt hohe Qualität
	10-100 Mbit/s	Großer Bildschirm hohe Qualität

Tabelle 1: Übertragungsbandbreite und Anwendungen [ATM96]

Die Anforderung an die Übertragungsbandbreite im Fall von Videokonferenzen mit hohen Bildqualitäten ist daher sehr groß. Der Bandbreitenbedarf konnte aber auf Grund von hocheffizienten Kompressionsverfahren sowie erhöhter Rechenleistung der Netzknoten in den letzten Jahren deutlich reduziert werden. Tabelle 1 gibt einen Überblick über Anwendungen und ihre jeweiligen Anforderungen an die Übertragungsbandbreite.

2.3 Ende-zu-Ende-Verzögerung

Verzögerung pro Richtung	Auswirkung auf die Kommunikation
> 600 ms	Keine Kommunikation möglich
600 ms	Kaum zusammenhängende Kommunikation möglich

250 ms	Die Verzögerung wirkt stark störend, der Gesprächsstil muß angepaßt werden
150 ms	Akzeptabel
100 ms	Die Verzögerung ist nicht wahrnehmbar, wenn der Hörer den Sprechenden nur über das Netz und nicht direkt hört
50 ms	Keine Verzögerung wahrnehmbar

Tabelle 2: Auswirkung der Übertragungsverzögerung auf die Qualität von Audioübertragungen [ATM96]

Der zweite wichtige Parameter bei der Übertragung von Multimedia Applikationen ist die gesamte Ende-zu-Ende-Verzögerung. Diese besteht hauptsächlich aus folgenden Teilzeiten: die Aufnahmezeit beim Sender, die Übertragungszeit und die Wartezeit bis zur Wiedergabe beim Empfänger. Tabelle 2 gibt eine Übersicht über die gesamte Verzögerung und deren Auswirkung auf die Kommunikation im Fall von Audioübertragungen.

Wie in der Tabelle dargestellt, ist für Videokonferenzen eine maximale Ende-zu-Ende Verzögerung von 150 ms des Audiostromes akzeptabel. Dieser Wert dient als Grundlage für die zu erreichende Minimierung der gesamten Ende-zu-Ende-Verzögerung.

2.4 Synchronisation der Datenströme

Datenstrom 1	Datenstrom 2	Anwendungsfall	Anforderung
Video	Audio	Lippensynchronisation	+/- 80 ms
	Animation	Voneinander abhängig	+/- 120 ms
	Einzelbild, Text	Überlagert	+/- 240 ms

		nicht überlagert	+/- 500 ms
Audio	Audio	Eng gekoppelt (Stereo)	+/- 10 ms
		lose gekoppelt	+/- 120 ms
		Lose gekoppelt (Hintergrundmusik)	+/- 500 ms
	Einzelbild	eng gekoppelt (Musik und Noten)	+/- 5 ms
		Lose gekoppelt (Diashow)	+/- 500 ms
	Text	Textanmerkungen zum Ton	+/- 240 ms
	Zeiger	Ton bezüglich der Zeigerbewegung	+/- 500 ms

Tabelle 3: Zeitliche Anforderungen an die Intersynchronisation [MÜL94]

Der dritte wichtige Aspekt bei der Übertragung von Audio- und Videodaten ist die Synchronisation beider Ströme. Die Anforderungen an die Synchronisation hängen dabei stark vom Anwendungsfall ab. Die in Tabelle 3 angegebenen Richtwerte dienen als Grundlage für die zu erreichende Synchronisationsgüte.

Für den Fall einer Videokonferenz darf die zeitliche Anforderung an die Intersynchronisation beider Video- und Audioströme bei maximal 80 ms liegen.

2.5 Delay Jitter[5] und Jitterkompensation

Ein wichtiges Element für die fehlerfreie Wiedergabe der Video- bzw. Audiodaten ist die Funktion der Taktrückgewinnung. Bei einem idealen Übertragungssystem werden die Pakete den Empfänger in regelmäßigen Abständen erreichen. Man spricht in diesem Zusammenhang von einer isochronen Übertragung. Tatsächlich schwanken aber die Laufzeiten der einzelnen Pakete. Diese Schwankungen

[5] Schwankungen bei der Übertragungsverzögerung

werden *Delay Jitter* genannt. Die Ursachen für diese Schwankungen liegen sowohl im Rechner als auch im Netz. Außerdem sind die verwendeten herkömmlichen Übertragungsprotokolle nicht speziell für eine isochrone Übertragung eingerichtet, sodaß diese nicht unmittelbar unterstützt wird.

Aufgabe der Taktrückgewinnung ist es, die Abspielgeschwindigkeit beim Empfänger und die Aufnahmerate beim Sender (gemessen in Bildern pro Sekunde FPS[6]) zu synchronisieren. Gerät die Abspielgeschwindigkeit bei hohen Delay Jitter außer Takt, d.h. entstehen Unterbrechungen in der Abspielfolge, dann mindert sich die Qualität der Echzeit-Übertragung. Die Taktrückgewinnung wird durch die Jitterkompensation erreicht, die wiederum durch eine Warteschlange nach dem FIFO[7]-Prinzip realisiert werden kann. Die Warteschlange muß genügend viele Pakete zwischenspeichern können, um den Jitter zu spät eintreffender Pakete auszugleichen. Eine große Warteschlange hat aber wiederum den Nachteil, daß sich die gesamte Ende-zu-Ende Verzögerung um die Wartezeit in der Warteschlange erhöht, die wie bereits im Abschnitt 2.3 besprochen, nicht größer als 150 ms sein darf.

Es ist deshalb die Aufgabe der Anwendung, eine Balance zwischen Echzeitanforderung und Qualität der Abspielfolge zu halten. Die Größe der Warteschlange muß den darunterliegenden Netzwerken und der gewünschten Übertragungsqualität angepaßt werden.

[6] engl. frames per second

[7] First In First Out

3 Kommunikationssysteme

3.1 TCP/IP

Dieser Abschnitt verschafft einen kurzen Überblick über Struktur und Anwendungsgebiete von TCP/IP

3.1.1 Architektur

Die Kommunikation erfolgt über verschiedene Schichten, für die jeweils Kommunikationsprotokolle definiert werden. Damit wird der riesige Problembereich in überschaubare und leichter verständliche Teilaspekte gegliedert. Das Modell, das aus der Kombination der definierten Schichten resultiert, wird häufig *Protokollstapel* genannt [TCP94], [NETZ92].

TCP/IP ist im Gegensatz zum ISO-OSI-Modell, das 7 Schichten definiert, in 5 Schichten geteilt. Abbildung 2 zeigt, welche Informationseinheiten den einzelnen Schichten zugeordnet sind.

Die Schichten 1 und 2 des Modells sind in den RFCs zu TCP/IP eigentlich nicht definiert, weil mit TCP/IP ein von den physikalischen Medien unabhängiges Protokoll entwickelt werden sollte. TCP/IP kann in jedem beliebigen Netz, das Bits mit einer hinreichend kleinen Bit-Fehlerrate von einem Ort zu einem anderen befördert, arbeiten. Die Bit-Fehlerrate soll so klein wie möglich sein, auf jeden Fall kleiner als 1 zu 500.000 (End-zu-End-Übertragungen), damit die Fehlerrate bei der Übertragung von Standard-IP-Datagrammen mit 586 Oktetts in einem WAN unter 10% bleibt.

Schicht 3 ist die Internet-Protokoll-Schicht (IP-Schicht). Diese Schicht bietet einen einfachen Datagramm-Dienst, und zwar eine Beförderung der IP-Daten ohne Garantie der Zustellung. In dieser Schicht sind noch zwei weitere Protokolle angesiedelt:

- ARP (Adress Resolution Protocol – *Protokoll für Adressenauflösung*) und

- RARP (Reserve Adress Resolution Protocol - *Protokoll für umgekehrte Adressenauflösung*).

Mit diesen Protokollen werden die IP-Adressen den entsprechenden physikalischen Adressen der Netzmedien zugeordnet. In der Schicht 3 arbeitet ein weiterer Dienst namens ICMP (Internet Control Message Protocol – *Protokoll für Kontrollnachrichten im Internet*). ICMP meldet Probleme, die bei der Übertragung von Datagrammen auftreten und ermöglicht den Systemen, auf den aktuellen Zustand des Netzes zu reagieren.

ISO Schichten Modell	**TCP/IP Protokolle**	
7 Applikation	Applikationen Protokolle	*Telnet* *FTP* *Rlogin* *SMTP* *Socket Lib.*
6 Präsentation		
5 Sitzung		
4 Transport	Transport Protokolle	*TCP* *UDP*
3 Netzwerk	Internetwork Protokolle	*EGP, RIP* *IP* *ICMP* *ARP* *RARP*
2 logische Verbindung	Network Access Protokolle	*Ethernet* *Tokenring* *FDDI* *ATM*
1 physikalische Verbindung		

Abbildung 2: TCP/IP-Protokoll-Familie

Schicht 4 ist die Transportschicht, in der sich zwei Protokolle befinden: UDP (User Datagram Protocol - *Protokoll für Benutzerdatenpakete*) und TCP

(Transmission Control Protocol – *Protokoll für eine kontrollierte Übertragung*). UDP ist für verbindungslose und TCP für verbindungsorientierte Übertragung gedacht. Näheres über die Protokolle der Transportschicht wird in Abschnitten 3.1.3, 3.1.4 und 3.1.7 beschrieben.

Schicht 5 ist die Anwendungsschicht. Beispiele für Dienste der Anwendungsschicht sind TELNET (Terminalverbindung), E-Mail (elektronische Post), das SMTP (Simple Mail Transfer Protocol - *Einfaches Protokoll für Nachrichtenaustausch*) und das FTP (File Transfer Protocol - *Protokoll für Dateiübertragung*).

3.1.2 Internet Protokoll (IP)

IP stellt die Grundlage für die Übermittlung der TCP/IP Protokolle höherer Schichten über beliebige physikalische Netze dar. Die Hauptaufgabe der Vermittlungschicht ist die Routen-Wahl (Leitwegbestimmung), d.h., Daten an den richtigen Rechner im richtigen physikalischen Netz weiterzuleiten.

Version	IHL	TOS (Diensttyp)	Gesamtlänge	
Kennung			Flags	Fragment-Offset
Zeitangabe		Protokoll	Kopfprüfsumme	
Absender-IP-Adresse				
Empfänger-IP-Adresse				
Optionen				Füllzeichen
Daten				
…				

Tabelle 4: Aufbau eines IP-Datagrammkopfes

Die IP-Schicht arbeitet verbindungslos, d.h., es gibt keine feste Verbindung zwischen zwei Kommunikationsknoten. Jedes IP-Datagramm enthält alle Informationen, die für eine korrekte Weiterleitung erforderlich sind. In dieser

Schicht findet keinerlei Fehlerkorrektur statt. Es gibt keine Garantie sowie keine Bestätigung für den korrekten Empfang eines Datagramms. Der Aufbau eines IP-Datagrammskopfes wird in Tabelle 4 illustriert.

Da die IP-Schicht verbindungslos arbeitet, ist zwischen zwei Kommunikationsknoten keine spezielle Route (Leitweg) definiert. Datagramme können daher zwischen zwei Knoten unterschiedliche Routen nehmen. Das ist ein Grund dafür, daß sie nicht in der Reihenfolge ankommen müssen, wie sie gesendet wurden. Dies ermöglicht aber anderseits Flexibilität, da bei Netzausfall die Daten ohne große Vermittlungsverzögerung leicht umgelenkt werden können.

Die IP-Basisfunktionen sind für die Adressierung und Fragmentierung von Datagrammen und die Ermittlung des passenden Diensttyps zuständig. In der IP-Schicht sind 3 Protokolle definiert: ARP, RARP und ICMP.

- *ICMP* wird von IP für den Transport von Fehlernachrichten verwendet. Es transportiert diese in IP-Datagrammen zwischen den Knoten. Die ICMP-Fehlermeldungen werden von Stationen gesendet, die auf ein Problem bei der Übertragung gestoßen sind, und zwar an diejenigen Stationen, deren Datagramme dieses Problem verursacht haben.

- *ARP* und *RARP* sind am unteren Rand der IP-Schicht angesiedelt. Da sie IP nicht verwenden, werden sie von der Datensicherungschicht als unabhängige Protokolle anerkannt. Sie sind für die Zuordnung von IP-Adressen in physikalische Adressen und umgekehrt zuständig. Dazu benutzen sie eine Tabelle (ARP-Cache), in der die Einträge der IP-Adressen und der korrespondierenden Hardwareadressen enthalten sind. Fehlt z.B. zu einer IP-Adresse die passende physikalische Adresse, wird diese durch ein spezielles Rundsende-Datagramm (Broadcast) ermittelt. Stimmt die im Rundruf erfragte IP-Adresse mit der einer an das Netz angeschlossenen Maschine überein, schickt diese Maschine als Antwort ihre physikalische Adresse zurück.

3.1.3 Transmission Control Protocol (TCP)

TCP ist ein verbindungsorientiertes Protokoll, das die Zuverlässigkeit von IP erhöht. Bevor eine Kommunikation zwischen zwei Knoten stattfinden kann, muß eine Verbindung aufgebaut werden. Für die Übertragung eines Datenstromes entstehen für die kommunizierenden Anwendungen scheinbar permanente, exklusive Verbindungen, die nach der Übertragung wieder abgebaut werden müssen. TCP übernimmt die Aufgabe der Segmentierung von großen Datenblöcken in sogenannte TCP-Segmente, die anschließend in IP-Datagrammen zur Übertragung gekapselt werden. Es entsteht bei der Übertragung ein geordneter Bitstrom, d.h., der Empfänger bekommt die exakte Bytesequenz, wie sie vom Sender geschickt wurde.

TCP bietet folgende Funktionen zur Erhöhung der Zuverlässigkeit:

- Fehlererkennung und Fehlerkorrektur
- Flußkontrolle
- Entfernung doppelter Segmente
- Neuordnung der Übertragungssegmente

Diese Funktionalität wurde in TCP durch folgende Mechanismen implementiert:

- die Verwendung von Sequenznummern zur Kennzeichnung der Daten
- eine positive Bestätigung beim Empfang jedes richtig angekommenen Segments
- das wiederholte Senden von Segmenten, für die innerhalb eines bestimmten Zeitraums keine Empfangsbestätigung eingegangen ist.

3.1.4 User Datagram Protocol (UDP)

UDP erweitert die grundlegenden IP-Datagramm-Dienste nicht. Es leitet die Daten von der IP-Schicht an die entsprechenden Dienste der Anwendungschicht weiter. Dies geschieht durch die Aufnahme der Felder Absender-Port und Empfänger-Port im UDP-Datagrammkopf, siehe Tabelle 5.

Absender-Port	Empfänger-Port
Länge	UDP-Prüfsumme
Daten	

Tabelle 5 Aufbau eines UDP-Protokollkopfes

Die UDP-Prüfsumme ist zur Sicherstellung der Integrität der von UDP übermittelten Daten implementiert. Zu dieser Summe werden nicht nur UDP-Felder, sondern auch der sogenannte Pseudo-Protokollkopf, der auf bestimmten Feldern der IP-Schicht basiert, berücksichtigt.

UDP arbeitet nicht verbindungsorientiert. Der Sender schickt Daten an Zielrechner, ohne auf Rückmeldung des Empfängers zu achten.

Man kann die Funktionalität von UDP wie folgend auffassen: UDP bietet

- einen verbindungslosen Übertragungsdienst
- keine Steuerung des Datenflusses, keine Kontrollmechanismen
- keine gesicherte Datenzustellung
- keine Fehlerbehebung, keine Zeitüberwachung

Daher arbeitet UDP sehr effektiv mit wenig Overhead sowie mit minimalen Protokollmechanismen.

3.1.5 IP Multicast

3.1.5.1 Vorstellung

Es gibt drei fundamentale Typen von IP-Adressen: unicast, broadcast und multicast. Eine Unicast-Adresse ist für die Übertragung eines Datenpaketes an eine bestimme Zieladresse vorgesehen. Eine Broadcast-Adresse wird für die Übertragung eines Datenpaketes an alle Stationen des gesamten Subnetzs benutzt. Eine Multicast-Adresse wird dagegen verwendet, um ein Datenpaket an

mehrere bestimmte Stationen übertragen zu können, die zu einer Multicast-Gruppe gehören, sich aber in verschiedenen Subnetzen befinden können.

Multicast ist nicht verbindungsorientiert, d.h., es gibt keine Garantie, daß die Multicast-Pakete bei allen Zielstationen der Multicast-Gruppe und in geordneter Reihenfolge ankommen. Der einzige Unterschied zwischen einem Unicast-IP-Paket und einem Multicast-IP-Paket ist die Repräsentation einer Multicast-Gruppen-Adresse im IP-Paket-Kopf als Zieladresse.

3.1.5.2 Multicast-Gruppe

Jede beliebige Station kann jeder Zeit in eine Multicast-Gruppe eintreten oder daraus austreten. Es gibt keine Einschränkung über die Anzahl der Mitglieder einer Gruppe sowie deren physikalischen Standort. Eine Station kann je nach verfügbarem Adreßraum und technischen Gegebenheiten zu mehreren Gruppen gehören. Jede Station kann Datenpakete zu Mitgliedern einer beliebigen Multicast-Gruppe schicken und braucht selbst nicht Mitglied dieser Gruppe zu sein.

3.1.5.3 Multicast-Adresse

Ein Sender benutzt eine spezielle IP-Adresse, um Datenpakete an alle Mitglieder einer Multicast-Gruppe zu schicken. Die IP-Adressen zwischen 224.0.0.0 und 239.255.255.255 sind für Multicast definiert.

Der Internet Assigned Numbers Authority (IANA – Behörde für die Vergabe von Internet-Adressen) hat eine Liste von reservierten IP-Multicast-Adressen festgelegt. Die Adresse 224.0.0.0 ist reserviert und darf nicht benutzt werden. Die Adressen von 224.0.0.1 bis 224.0.0.255 sind für Routing Protokolle reserviert. Multicast-Router übermitteln keine Pakete mit solchen IP-Multicast-Adressen. Die verbleibenden Adressen von 224.0.1.0 bis 239.255.255.255 können für unterschiedliche Multicast-Anwendungen benutzt werden.

3.1.5.4 Senden und Empfangen von Multicast-Paketen

Zum Verschicken von Multicast-Paketen braucht die Sendestation nur die IP-Multicast-Adresse als IP-Zieladresse zu benutzen. Der Empfänger teilt dagegen seiner IP-Schicht mit, daß er Datenpakete mit einer bestimmten Multicast-Adresse empfangen will, indem er als Mitglied in diese Multicast-Gruppe eintritt.

Ein Multicast-Paket wird mit einem TTL-Parameter (time-to-live – *Lebenszeit*) geschickt. Multicast-Pakete mit einem TTL-Parameter von 0 werden nicht über einen Router hinweg übermittelt. Multicast-Pakete mit einem TTL-Parameter größer als 1 werden an mehr als ein Subnetz übermittelt, wenn mehrere Multicast-Router mit dem Subnetz verbunden sind. An jedem Subnetz wird der TTL-Parameter um 1 vermindert; beim Erreichen von Null wird das Paket verworfen.

Jeder Multicast-Router muß Protokolle implementieren, um Mitglieder der Multicast-Gruppen im verbundenen Subnetz verwalten und Multicast-Pakete übermitteln zu können. Es existiert zur Zeit eine ganze Reihe von implementierten Router-Protokollen und Übermittlungsalgorithen wie das Internet Group Management Protocol (IGMP), Distance Vector Multicast Routing Protocol (DVMRP), Multicast Extension to Open Shortest Path First (MOSPF) und Protocol-Independent Multicast (PIM) [IPMUL97].

3.1.6 Das Windows Socket Interface

Im Jahr 1981 wurden Sockets als Schnittstelle von Berkeley Software Distribution für die Kommunikation zwischen lokalen Prozessen im 4.1c-BSD-System auf VAX-Anlagen eingeführt. 1986 erweiterte die Unix-Version 4.3 BSD die Socket-Schnittstelle für den Einsatz über TCP/IP und XNS-Protokollen. Diese Erweiterung ermöglicht zwei Prozessen die Kommunikation, egal ob sie lokal in einem Rechner laufen oder über das Netz verbunden sind.

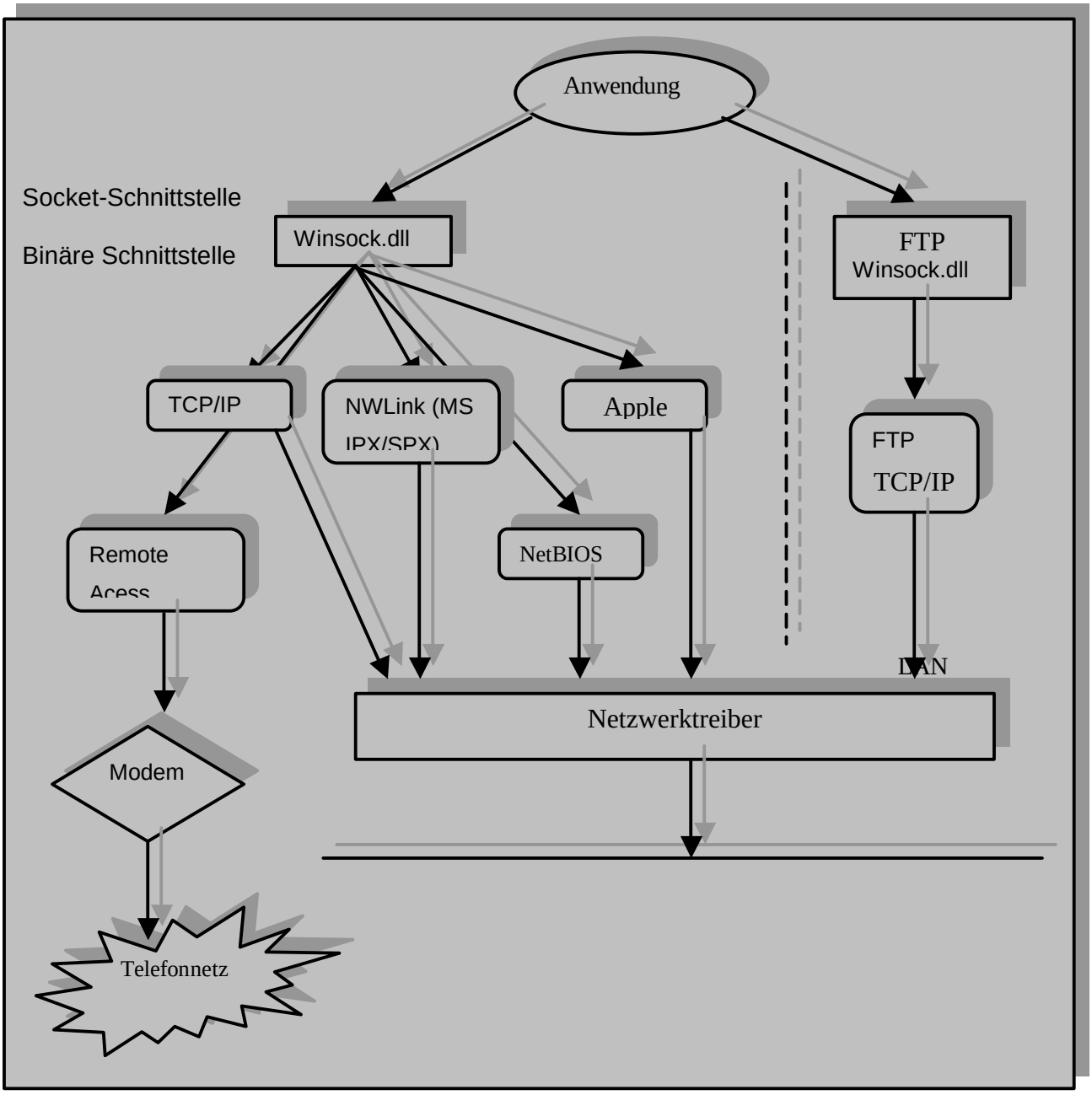


Abbildung 3: WinSock und die zugrundeliegenden Protokolle

Für Windows wurde ein standardisiertes Socketinterface für Windows 3.x und NT entwickelt: WinSock (Abkürzung für Windows Socket). Es vereinheitlicht die Anwendungsprogrammierschnittstellen und definiert eine binäre Schnittstelle, die für Entwickler standardisiert ist. Die Spezifikationen definieren Bibliotheksaufrufe und die damit verknüpfte Semantik, die von den Netzwerksoftware-Herstellern implementiert und von den Anwendungsprogrammierern benutzt wird. Dadurch

können Anwendungen, die für WinSock geschrieben wurden, mit den Bibliotheken verschiedener Hersteller und mit den Protokollen beliebiger WinSock-kompatibler Hersteller zusammenarbeiten. Abbildung 3 gibt einen Überblick über die Relation zwischen WinSock und den zugrundeliegenden Protokollen.

Ein vollständig verbundener Socket repräsentiert einen logischen Kommunikationskanal zwischen zwei Prozessen. Er kann bidirektional sein und ist entweder Stream- oder Datagrammorientiert, je nach dem, ob das zugrundeliegende Protokoll einen verbindungsorientierten oder –losen Dienst bietet. Ein verbundener Socket kann folgendermaßen beschrieben werden:
{Protokoll, lokale Adresse, lokaler Prozeß, externe Adresse, externer Prozeß}

Sowohl der lokale als auch der externe Prozeß müssen ein gemeinsames Netzwerkprotokoll (z.B. TCP für verbindungsorientierte Interprozeßkommunikation) wählen. Beide Prozesse benötigen Netzwerkadressen, durch die sie sich eindeutig im Netz identifizieren lassen. Außerdem brauchen sie eindeutige Namen (oder IDs), damit sie von allen anderen Prozessen auf der selben Maschine unterschieden werden können. Um eine Verbindung aufzubauen, erzeugt jeder Prozeß zunächst seinen eigenen Anteil an der Socket-Infrastruktur: die sogenannten Endpunkte mit dem Inhalt *{Protokoll, lokale Adresse, lokaler Prozeß}*.

Sobald beide Prozesse ihre Endpunkte generiert haben, können sie eine oder mehrere Socket-Funktionen aufrufen, um die Endpunkte miteinander zu verbinden. Das ergibt einen verbundenen Socket. Dann können die verbundenen Prozesse über Socket-Funktionen wie send() und recv() Daten senden und empfangen. Am Ende treten die Prozesse in Verbindung und schließen ihre Socket-Endpunkte, indem sie wiederum Socket-Funktionen verwenden.

3.1.7 Vergleich der beiden Protokolle UDP und TCP

Bei der Übertragung unter Verwendung der TCP/IP-Software über das Windows-Socket-Interface kann der Entwickler auf die dort bereits implementierten Funktionalitäten zurückgreifen. Der Vergleich zwischen den beiden Transportprotokollen TCP und UDP soll zeigen, daß der Einsatz von UDP für den Transport von Multimedia-Daten besser geeignet ist als der Einsatz von TCP. Die Wahl von UDP als Transportprotokoll erfolgt aus folgenden Gründen:

- Wegen der zeitlichen Eigenschaften von Video und Audio ist eine Wiederholung von verlorenen oder fehlerhaft übertragenen Datagrammen ausgeschlossen. Wiederholte Übertragungen von Daten bei fehlerhaften oder nicht in Reihenfolge empfangenen Daten, wie im Fall von TCP, sind sehr zeitaufwendig. Die resultierende Verzögerung führt zu großen Delay Jitter (siehe Abschnitt 2.5 Delay Jitter) und zu störenden Verzögerungen bei der Ausgabe. Ein einfaches Aussetzen des fehlerhaften Frames und statt dessen das wiederholte Anzeigen seines Vorgängers in der Sequenz würde genügen, um weniger Störungen in Datenfluß und Bildqualität zu bringen als eine lange Pause in der Abspielfolge. Zu spät eintreffende oder fehlerhafte Frames sind unbrauchbar und sollten einfach verworfen werden – eine Forderung, der eine Anwendung auf Basis von TCP nicht nachkommen kann.

- UDP hat keine Flußkontrolle und ist deshalb gut für Videoübertragungen in Echtzeit geeignet. Die Flußkontrolle spielt bei der Videoübertragung eine untergeordnete Rolle. Für den Empfänger ist es bei der Ausgabe gleichgültig, wie viele Frames durch Überlaufen der internen Puffer verloren gehen, solange zur Anzeige weiterhin Frames zur Verfügung stehen. Um eine konstante, minimale Ende-zu-Ende-Verzögerung zu gewährleisten, muß der Sender Daten immer mit konstanter Rate schicken. Bei Überlastung des Empfängers werden noch nicht angezeigte Frames aus der Warteschlange entfernt, um Platz für die neu angekommenen Frames zu schaffen. Das wirkt positiv auf das Geschehen im Fall des Videoconferencing - es kann damit ein „flüssiger“ Bildstrom erreicht werden.

- M-JPEG hat die Eigenschaft, komprimierte Frames mit unterschiedlicher Länge zu erzeugen. Der Stream-Mode von TCP ist dafür weniger geeignet, weil der TCP-Empfänger nur in den seltensten Fällen innerhalb eines Leseschrittes exakt die gesendete Datenmenge (nämlich genau ein komprimiertes Frame) empfängt. Bedingt durch den stream-mode werden in der Regel nur Bruchstücke (Fragmente) des komprimierten Frames gelesen. Der Datenstrom kennt keine Grenzen, die die einzelnen Frames voneinander trennen können. Dies ist besser bei der Übertragung mit UDP, wo immer Pakete geschickt und empfangen werden. Ein einzelnes Frame kann dann zusammen mit weiteren bestimmten Informationen in ein Paket eingepackt und verschickt werden.

- Multicast kann problemlos auf UDP implementiert werden.

Der Einsatz von UDP als Transportprotokoll hat auch Nachteile gegenüber TCP.

- Nachteile durch den Einsatz von UDP zur Übertragung von zeitabhängigen Medien wie Video oder Audio können sich in großen Netzwerken ergeben, wenn mehrere mögliche Routen zwischen Sender und Empfänger existieren. Für die abgeschickten Datagramme kann deshalb nicht garantiert werden, daß sie auch in den Sende-Reihenfolge ankommen. Das kann zu einer gestörten Ausgabe führen. Das Anwendungsprogramm muß sich deshalb zusätzlich um diese Flußkontrolle kümmern. In diesem Fall bedarf es einer zusätzlichen Logik, die ähnlich wie bei TCP, auf Sequenznummern basieren könnte.

Im Fall des Einsatzes von ATM als Transportnetzwerk treten diese Probleme selten auf, da seine Architektur mit verbindungsorientierter Kommunikation durch Virtuelle Pfade (VP) und Virtuelle Kanäle (VC) eine Übertragung über verschiedene Router eingeschränken kann. Der Einfluß von Cache-Speichern kann aber trotzdem die zeitliche Reihenfolge der Zellen durcheinander bringen.

3.2 Transportnetzwerk

Netzwerke gewinnen in der Wirtschaft zunehmend an Bedeutung. Die Übertragungsgeschwindigkeit bzw. der Datendurchsatz sind in den vergangenen Jahren um einige Größenordnung gestiegen. Sowohl bei der Kommunikationssoftware als auch bei den Hardwarekomponenten für das Netz entstanden herstellerunabhängige, allgemein verfügbare Lösungen, die der internationalen Normung folgten. Lokale Datennetze sind heute ein fester Bestandteil jedes Kommunikationskonzeptes geworden. Sie ermöglichen die Integration von Großrechnern, Terminals und PC zu einem gemeinsamen Netzwerk, in dem die bereitstehende Rechnerleistung auf mehrere Knoten verteilt sein und prinzipiell von allen Endgeräten in Anspruch genommen werden können. Traditionelle Netzwerke wie Ethernet, Tokenring oder FDDI (Fibre Distributed Data Interface) sind als lokale Netze weltweit verbreitet und haben durch ihre Popularität eine sehr wichtige Rolle in heutigen LAN´s. Die folgenden Abschnitte stellen deren Architektur und deren Auswirkungen auf die Übertragung von Echtzeit-Multimedia-Anwendungen vor.

3.2.1 Die Leistungsgrenzen von Ethernet

Das Ethernet wurde Ende der siebziger Jahre von der Firmengruppe Digital Equipment Corp., Intel und Xerox (DIX) auf den Markt gebracht. Es ist heute das weltweit am häufigsten installierte lokale Datennetz und zeichnet sich durch einfache Handhabung, hohe Flexibilität sowie durch seine Vielseitigkeit aus [NETZ92],[ATM96].

Der Ethernet-(CSMA/CD) Mechanismus basiert auf einem nicht-deterministischen Verfahren. Die Übertragungsmethode ist Halb-Duplex, das

bedeutet, daß das Senden und Empfangen von Daten nicht gleichzeitig stattfinden kann. Sendet eine Station, so müssen sich andere Stationen im Empfangsmodus befinden. Senden zwei Stationen gleichzeitig, so kollidieren die Datenpakete. Die beiden Sendesignale überlagern sich und können nicht mehr eindeutig empfangen (interpretiert) werden. Die an einer Kollision beteiligten Stationen müssen eine durch einen Zufallsgenerator bestimmte Zeit warten, bis sie wieder auf Sendung gehen können.

Die Wahrscheinlichkeit einer Kollision wächst daher mit zunehmender Anzahl der Netzteilnehmer und steigender Länge des betreffenden Netzsegments. In nur wenig belasteten Netzwerken funktioniert der CSMA/CD-Algorithmus außerordentlich gut. Ab einer Netzlast von ca. 40% bei mehr als 10 aktiven Netzteilnehmern nimmt die Ausnutzung der zur Verfügung stehende Bandbreite drastisch ab. Auch die Paketlänge spielt eine große Rolle für die Leistungsfähigkeit von Ethernet: Je kürzer übertragene Datenpakete sind, desto geringer ist die Leistungsfähigkeit von Ethernet und desto schlechter ist die Ausnutzung der zur Verfügung stehenden Bandbreite.

Die typischen Betriebscharakteristika von Ethernet können wie folgt zusammengefaßt werden:

- nicht vorhersehbare Übertragungsverzögerung
- geringe Effizienz bei kleinen Paketgrößen
- geringe Bandbreiteneffizienz bei mehr als vier Stationen (< 50%)

Ethernet ist daher wegen dieser Eigenschaften für die Übertragung von Echzeit-Multimedia-Anwendungen ungeeignet.

3.2.2 Token Ring und FDDI

Der Token Ring wurde 1972 entwickelt und Anfang der achtziger Jahre von der Firma IBM im Rahmen des IBM-Verkabelungs-Systems (IVS) auf dem Markt eingeführt. Die internationale Standardisierung erfolgte bei IEEE 802.5 erst Mitte der achtziger Jahre. Ein Ring stellt eigentlich kein Rundsende-Medium dar,

sondern eher eine Ansammlung von Punkt-zu-Punkt-Verbindungen, die zufällig die Form eines Kreises hat. Punkt-zu-Punkt-Verbindungen stellen eine gut durchdachte und anwendungsreife Technologie dar und laufen auf verdrillten Doppelkabeln, Koaxialkabeln oder Lichtwellenleitern [NETZ92].

Das Token-Ring-Verfahren ist im Gegensatz zu Ethernet derteministisch, d.h. jeder Station im Ring wird per Definition die Gelegenheit gegeben, in regelmäßigen Abständen Daten zu senden. Dafür wird eine spezielle Bitsequenz benutzt, ein sogenanntes Token, dessen Besitz zum Versenden von Daten berechtigt. Das Senderecht (Token) wird von Station zu Station weitergegeben, sodaß für jede Station eine bestimmte Zeit zur Verfügung steht, in der die Daten auf das Netz gesendet werden können. Sendet keine Station, so kreist das Token unbenutzt im Ring. Möchte eine Station Daten senden, so wartet sie das Eintreffen des Token ab und beginnt unmittelbar danach, damit. Dabei wird das (Frei-) Token zu einem (Belegt-) Token modifiziert und die zu sendenden Daten hinter ihm angehängt. Das Datenpaket wird von allen Stationen gelesen. Erkennt die Empfängerstation eines Datenpakets ihre eigene Adresse als Zieladresse, so kopiert sie es nicht nur in ihren Empfangsspeicher, sondern versieht es mit einem Bestätigungsflag und sendet das so modifizierte Datenpaket zur ursprünglichen Sendestation weiter. Diese wertet die Statusinformation aus, nimmt das Paket vom Netz und gibt ein neues (Frei-) Token auf das Netz. Ein sofortiges Aussenden von Daten ist nicht möglich.

Durch das (oftmals) unnötige Kreisen des Tokens im Netz geht jedoch beim Token Ring viel Bandbreite verloren. Es wurde daher mit FDDI eine neue Token-Technik, das sogenannte Early Token Release verwendet. Nach dem Versenden eines Datenpaketes wird sofort wieder ein neues Frei-Token generiert. Dadurch kann mehr als ein Datenpaket auf dem Netz kreisen.

Da ausschließlich jene Station, die das Token besitzt, senden darf, kommt es im Token Ring sowie FDDI niemals zu Kollisionen. Aus diesem Grund können diese Netzwerke die Bandbreite besser als Ethernet ausnutzen. Erst bei einer Netzlast

von mehr als 80% verdoppelt sich die Wartezeit auf das Token und ein Abfall der Netzleistung wird bemerkbar.

Trotzdem sind Token Ring und FDDI wegen ihrer Übertragungsmethode nicht für Echzeit-Multimedia-Anwendungen geeignet: Der Standard für die Token-Holding-Time (die maximale Zeit, in der eine Station senden darf) ist im Fall des Token Ring 10 ms, im Fall von FDDI 4 ms. Befinden sich nun beispielsweise 30 Netzknoten im Ring, so ist die maximale Zeit, die eine Station bis zur Sendung warten muß, im Token Ring 300 ms und im FDDI Ring 120 ms, also um ein Vielfaches höher als der für den Betrieb von Multimedia-Anwendungen höchste zulässige Wert von etwa 10 ms. Tabelle 6 faßt diese Beschränkungen zusammen.

Netzwerktopologie	**Maximale Übertragungsverzögerung (30 Stationen)**
Ethernet	Nicht vorhersagbar
Token Ring	300 ms
FDDI	120 ms

Tabelle 6: Die Echtzeitleistungsgrenzen von Ethernet, Token Ring und FDDI

3.2.3 ATM: eine neue Technologie

Die drastische Entwicklung der EDV-Systeme hat zur Folge, daß eine immer größere Übertragungsbandbreite für Computernetzwerke erforderlich wird. Die traditionellen Computernetzwerke über Ethernet und Tokenring mit einer Übertragungsbandbreite von 10 bzw. 16 MBit/s, die außerdem nicht für die echtzeit-orientierte Übertragung multimedialer Daten geeignet sind, sind nicht mehr ausreichend. Eine neue Technologie der Datenkommunikation, die die traditionelle Datennetzwerke ablösen kann, ist ATM.

ATM ist ein weltweiter Standard, der unabhängig vom Typ der Endsysteme und von deren Art der Information (Daten, Audio, Video) einen universellen Informationsaustausch ermöglicht. Die Architektur von ATM (53 Bytes-Zellenstruktur) ermöglicht den Aufbau von massiv parallelen Kommunikationskomponenten und damit die Realisierung von Hochgeschwindigkeitsnetzwerken mit Übertragungsbandbreiten im Gigabitbereich. Mit Hilfe dieser neuen Netzwerke ist es möglich, die großen Datenmengen, die von modernen Anwendungen erzeugt werden, kostengünstig und in Echtzeit zu übertragen (interaktives Fernsehen, Videokonferenzen, Telepräsenz, Videomail, Inter-LAN-Kommunikation, etc.).

Darüber hinaus ist ATM gleichermaßen für den lokalen und für den Weitverkehrsbereich geeignet. Durch die Fähigkeit von ATM, die traditionellen LAN- und WAN-Architekturen auch zu emulieren (LAN-Emulation, Frame Relay, etc.), wird eine sanfte Migration von der heutigen EDV-Infrastruktur hin zu auf ATM basierenden Hochgeschwindigkeitstechnologien gewährleistet [ATM94], [ATM96], [ATM97].

3.3 Asynchronous Transfer Mode (ATM)

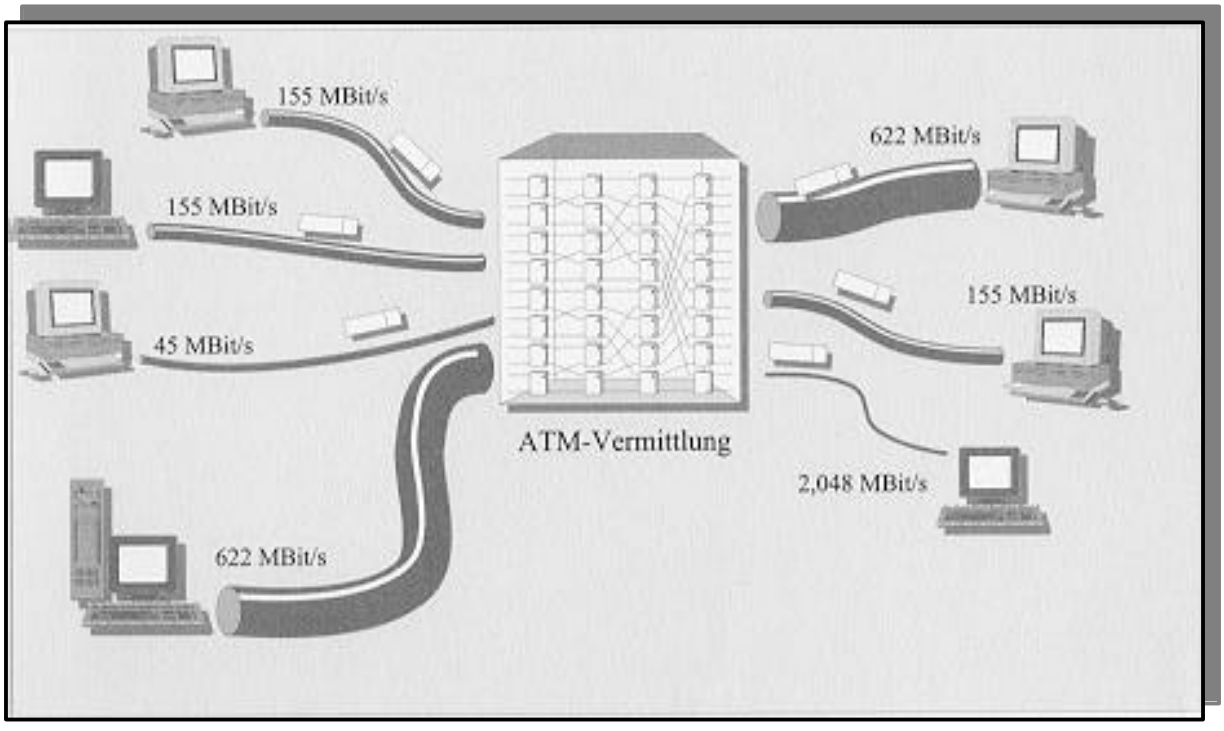


Abbildung 4: Das Prinzip von ATM [ATM96]

ATM ist eine Datenübertragungstechnik, die zur Familie der zellenvermittelnden Systeme (Cell Relay) gehört. Im Gegensatz zu paketvermittelnden Systemen, in denen Datenpakete variabler Länge über eine Leitungsschnittstelle gemultiplext werden, ist die Länge der Cell Relay Datenpakete fest. ATM benutzt zum Datentransport Zellen mit der festen Länge von 53 Bytes. Zellen lassen sich effizient und schnell durch Zellenvermittlungseinheiten vermitteln (Abbildung 4).

Da alle Zellen gleich lang sind, können 000000000alle Zellen, die zur selben Zeit an der Eingangsport einer ATM-Vermittlungseinheit anstehen, im Takt gleichzeitig an die gewünschten Ausgangsports vermittelt werden.

3.3.1 Charakteristika von ATM

- *ATM ist verbindungsorientiert*
  Vor der Übertragung muß eine virtuelle Verbindung zwischen Sender und Empfänger hergestellt werden. Danach werden die Daten in geordneter Reihenfolge gesendet und empfangen.

- *ATM bietet keine gesicherte Übertragung*
  ATM garantiert im Gegensatz zu anderen verbindungsorientierten Datenübertragungsdiensten wie z.B. X.25 nicht für die Korrektheit sowie Vollständigkeit der empfangenen Daten. Es findet kein wiederholtes Senden von fehlerhaft empfangenen Daten statt. Diese Funktionalität muß, falls gewünscht, in einer höheren Protokollschicht implementiert werden. Das erlaubt einerseits eine optimale Anpassung an den geforderten Dienst, anderseits bedeutet es einen Mehraufwand bei der Entwicklung.

- *ATM bietet keine Flußkontrolle*

Eine für diesen Mechanismus notwendige Logik (wie bei TCP) wäre für die von ATM geforderte Durchsatzrate zu komplex. Es existiert daher aus Performancegründen keine Flußkontrolle innerhalb des Netzwerkes. Es werden allerdings die einströmenden Datenmengen an den Endgeräten gemessen und mit den zuvor spezifizierten Parametern verglichen bzw. überwacht.

- *Übertragungsbandbreite wird von ATM nach Bedarf verteilt*
  Ein ATM-Netzwerk hat auf Grund der Verwendung des asynchronen Zeit-Multiplexing mit fester Paketlänge die Möglichkeit, die Verteilung der vorhandenen Übertragungsbandbreite flexibel zu gestalten. Einer sendenden Station wird je nach Bedarf mehr der weniger Bandbreite zugeordnet, die sie vor der Kommunikation durch einen bestimmten Parameter reserviert hat. Die gesamte Bandbreite teilt sich auf alle Benutzer auf und ermöglicht eine effiziente Auslastung. Damit ist es mit ATM möglich, Dienste mit sehr unterschiedlichem Bedarf an Übertragungsbandbreite wie:
    - Anwendungen mit stark variierenden Bitraten
    - Echzeit-Applikationen
    - Feste Bitraten und
    - Zeitunkritische Applikationen

  mit hoher Effizienz zu realisieren.

- *Starke Skalierbarkeit und Modularität*
  Es gibt keine eindeutige Festlegung, auf welchen physikalischen Medien ATM-Zellen mit welcher Geschwindigkeit übertragen werden sollen. Außerdem können ATM-basierende Netzwerke durch neue Benutzer erweitert werden, ohne daß die den bisherigen Teilnehmer zur Verfügung stehende Bandbreite eingeschränkt wird. In die zwischen den Teilnehmern vermittelnde ATM-Schalteinheit (ATM-Switch) werden lediglich weitere Anschlußmodule mit Bandbreiten von beispielsweise 155 Mbit/s eingesetzt. Jedes Modul stellt nun seinem Teilnehmer diese Bandbreite zur Verfügung, sofern das Bus-System (Backplane) das erlaubt.

- *Vereinfachte LAN-WAN-Kopplung*
  Bei Einsatz von ATM sowohl im WAN als auch im LAN wird der Aufwand, beide miteinander zu verbinden, erheblich reduziert. Anstelle komplexer Protokollkonversionen sind nur noch Modifikationen der Zellen-Adreßfelder nötig.

3.3.2 Architektur von ATM-Netzwerken

3.3.2.1 Das B-ISDN-Schichtenmodell

In Anlehnung an das OSI-Schichtenmodell wurde die logische B-ISDN-Netzwerkarchitektur aus vier voneinander unabhängigen Kommunikationschichten zusammengesetzt (siehe Abbildung **5**).

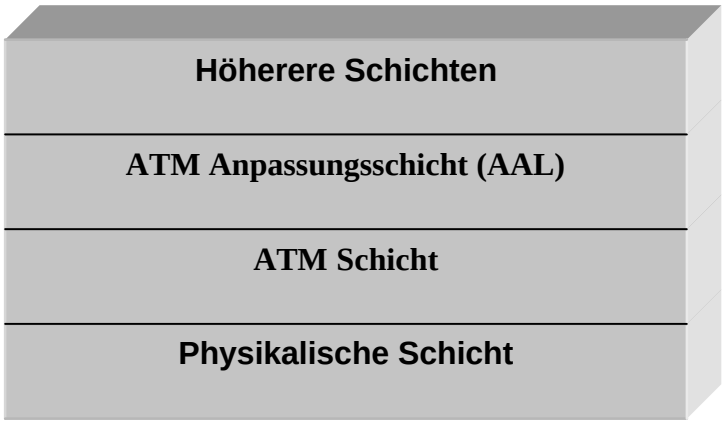


Abbildung 5: B-ISDN-Protokollschichten

3.3.2.2 Die physikalische Schicht

Im ATM-Referenzmodell wird kein Medium für die Übertragung festgelegt. Möglich sind z.B. Lichtwellenleiter, Koaxialkabel, Twisted Pair oder andere Medien.

Die physikalische Schicht sorgt dafür, daß die ihr übergeordnete ATM-Schicht unabhängig von der Art und der Geschwindigkeit des verwendeten Übertragungsmechanismus ist. Ihre Aufgabe besteht darin, ATM-Zellen auf das jeweilige Übertragungsmedium anzupassen und zu übertragen. Entscheidend dafür, ob ein physikalisches Medium als Übertragungsmedium von ATM-Zellen verwendet werden kann, ist allein, ob dafür eine Definition der Teilschicht „Übertragungsanpassung“ (Transmission Convergence) vorhanden ist.

Es gibt drei Arten von Übertragungsanpassungen:

- direkte Übertragung von Zellen
- Zellenanpassung auf Übertragungsrahmen
- Übertragung durch PLCP (Physical Layer Convergence Procedure), eine Übertragungsanpassung, die für die Übertragung von Datenpaketen in Metropolitan Area Network (IEEE-802.6) auf PDH-Leitungen definiert wurde.

3.3.2.3 Die ATM Schicht

Auf dieser Ebene werden ATM-Zellen transparent für die höhere Schicht zu einem kontinuierlichen Zellenstrom zusammengeführt (Multiplexing), Verbindungen zwischen Netzwerkknoten auf- und abgebaut sowie der Verkehrfluß überwacht. Die Funktionen der ATM-Schicht sind völlig unabhängig von der darunter liegenden physikalischen Schicht.

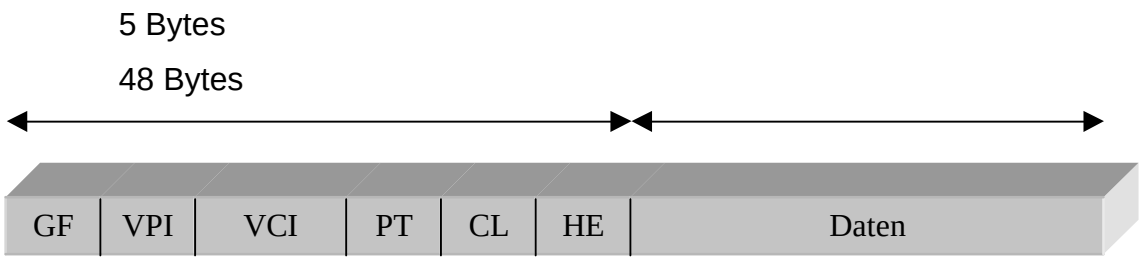


Abbildung 6: UNI-Zellformat

5 Bytes

48 Bytes

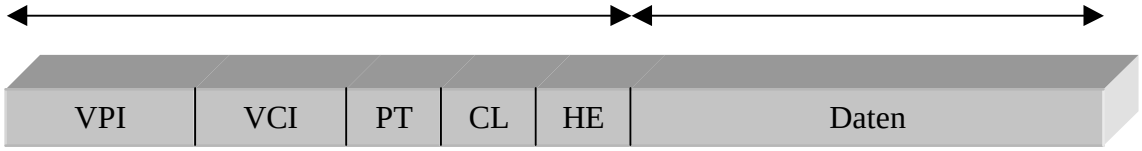


Abbildung 7: NNI-Zellformat

Die Informationseinheiten der ATM-Schicht sind 53 Bytes große Zellen und werden wie folgt definiert.

ATM-Zellen bestehen aus einem 5 Bytes großen Zellkopf und 48 Bytes Nutzdaten. Der Zellkopf besitzt bestimmte Informationen, die die Zelle einer bestimmten Verbindung zuordnet. Er besteht aus 4 Bits zur Datenflußkontrolle (Generic Flow Control - GFC), 20 Bits zur Pfad- und Kanalidentifikation (VPI, VCI), 3 Bits zur Bestimmung der Nutzlastidentifikation (Payload-Type PT) und 1 Bit zum Setzen der Zellenverlustpriorität (Cell Lost Priority CLP).

Es gibt Unterschiede zwischen ATM-Zellen für Netzwerk-Netzwerk-Kommunikation (NNI) und ATM-Zellen für Benutzer-Netzwerk-Kommunikation (UNI). NNI-Zellen haben kein Feld für die Datenflußkontrolle und dagegen einen größeren Bereich für die Werte zur Pfadidentifikation (VPI). In NNI-Zellen beträgt dieser Wert dann 12 Bits und ermöglicht die Verwaltung von 4096 Pfaden (in UNI-Zellen nur 8 Bits, also 256 Pfade), siehe
Abbildung 6 und Abbildung 7.

Die Hauptaufgaben der ATM-Schicht sind:

- Multiplexen von ATM-Zellen verschiedener Verbindungen
- Erzeugen und Auswerten der ATM-Zellköpfe
- Festlegen und Auswerten der VPI und VCI
- Kennzeichnen von speziellen Zellen
- Generierung der Kopf-Prüfsumme (Header Error Control HEC)
- Vermitteln von ATM-Zellen
- Verkehrsüberwachung

3.3.2.4 Die ATM Anpassungschicht

Die ATM-Anpassungschicht ist die dritte Schicht im B-ISDN Protokoll-Referenzmodell, wie in der Abbildung 8 illustriert ist. Die ATM-Anpassungschicht (AAL) hat die Funktion, die Datenstrukturen der höheren Anwendungschicht auf die Zellenstruktur der ATM-Schicht abzubilden sowie die dazugehörenden Steuer- und Managementfunktionen zur Verfügung zu stellen.

Es wurden 6 AAL-Typen für verschiedene Dienste definiert

- AAL-0 für vom Benutzer definierte Dienste
- AAL-1 für Dienste mit konstanten Bitsraten
- AAL-2 für Dienste mit Variablen Bitsraten
- AAL-3 für die verbindungsorientierte Übertragung von Datenpaketen
- AAL-5 ist eine simplifizierte Version von AAL-3.
- AAL-4 für die nicht verbindungsorientierte Übertragung von Datenpaketen

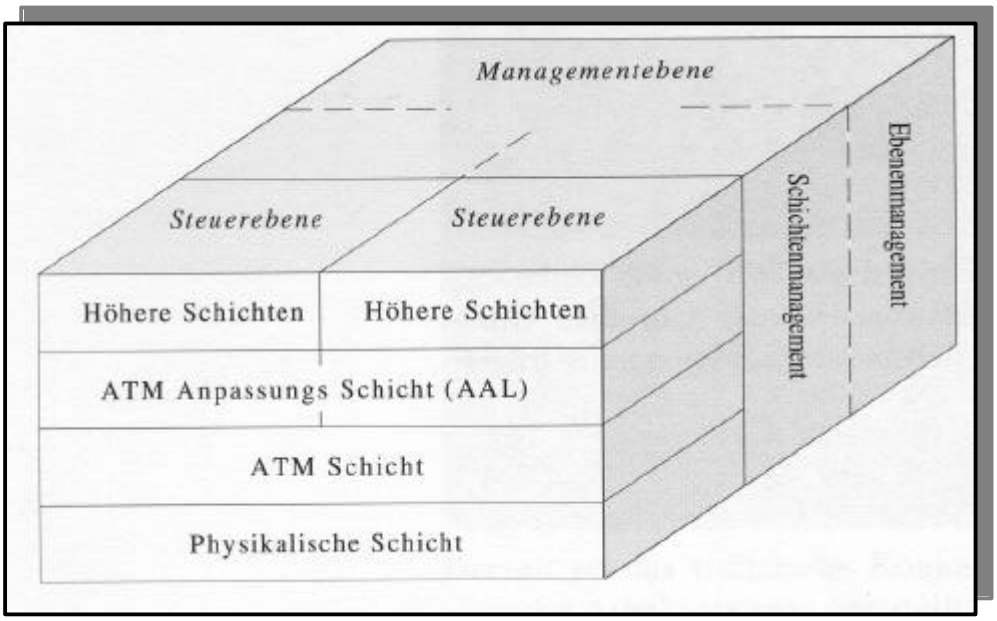


Abbildung 8: B-ISDN Protokoll-Referenzmodell

Es wurde aber sehr bald erkannt, daß keine Notwendigkeit für die Unterscheidung zwischen verbindungsorientierter und nicht verbindungsorientierter Datenübertragung im Rahmen der AAL-Schicht besteht und AAL-3 und AAL-4 wurden zu AAL-3/4 zusammengefaßt. (Tabelle 7)

	Klasse A	Klasse B	Klasse C	Klasse D
Zeitkompensation	Erforderlich		nicht erforderlich	
Bit Rate	Konstant	Variabel		
Verbindungs Modul	Verbindungsorientiert			verbindungslos
Beispiel	Circuit Emulation	Bewegtbild-Video	verbindungs-orientierter Datentransfer	verbindungs-loser Datentransfer
AAL Typ	Typ 1	Typ 2	Typ 3, 5	Typ 4

Tabelle 7: Serviceklassen und AAL-Typen [ATM96]

3.3.3 IP über ATM

RFC 1577 (Request For Commend – *Empfehlungsgesuch*) [RFC 1577] spezifiziert eine vollständige Internet-Protokoll-Implementation für ATM. Die Adressenzuordnung, wie IP sie im Ethernet mit Hilfe von ARP und RARP realisiert, wird dabei entsprechend durch eine ATMARP bzw. eine InATMARP Funktion realisiert. Die Zuordnungstabelle für ATMARP bzw. InATMARP ist in einem ARP-Server gespeichert, der in jedem logischen IP-Netzwerk (LIS) vorhanden sein muß. Die ARP-Clients sind selbst verantwortlich, ihre eigene IP- und ATM-Adresse bei dem ARP-Server zu registrieren. Die gewünschte IP- bzw. ATM-Adresse des Zielsystems wird beim ARP-Server abgefragt.

Die Einträge der ATMARP-Tabelle sind einem Alterungsprozeß unterworfen. Die ATMARP-Einträge der Clients sind höchstens 15 Minuten, des Servers mindestens 20 Minuten lang gültig.

3.3.4 ATM LAN-Emulation

Die aus Sicht der LAN-Applikationen universellste Methode, ATM-Netzwerke effizient in bestehende, herkömmliche LAN-Strukturen integrieren zu können, besteht in einer vollständigen Emulation der LAN-MAC-Schicht. Aus Sicht der LAN-Software verhält sich der LAN-Emulationsdienst dabei genauso wie ein traditioneller MAC-LAN-Treiber.

3.3.4.1 LAN-ATM-Integration

Das zentrale Problem bei der LAN-ATM-Integration besteht nicht in den unterschiedlichen Übertragungsgeschwindigkeiten oder inkompatiblen Paket- bzw. Zellenformaten, sondern darin, daß ATM auf verbindungsorientierten Kommunikationsbeziehungen basiert. Alle traditionellen LAN-MAC-Protokolle arbeiten dagegen nicht verbindungsorientiert.

Da die ATM-Technik im Gegensatz zu Token Ring, Ethernet oder FDDI über dedizierte Kanäle arbeitet, mußte eine Methode gefunden werden, Broacast-Informationen über ein ATM-Netzwerk übermitteln zu können. Bei einem Übergang zwischen dem LAN und einem ATM-Netz müssen die traditionellen LAN-Datenpakete in ATM-Zellen umgesetzt werden. Bei diesem Übergang werden Token Ring-, Ethernet- oder FDDI-Pakete in 48 Byte lange Zellen aufgeteilt und mit einem 5 Byte langen ATM-spezifischen Header versehen.

Die LAN-Emulation des ATM-Forums wird durch vier auf AAL-5 aufsetzende ATM-Dienstmodule realisiert:

- LE-Client (LEC)
- LE-Server (LES)
- LE-Emulation-Configuration-Server (LECS)
- Broadcast und Unknown-Server (BUS)

3.3.4.2 LAN Emulation Client (LEC)

Die LAN-Emulation-Client-Software führt alle notwendigen Steuerfunktionen und die eigentliche Datenübertragung über das ATM-Interface durch. Sie stellt für die darüberliegende Schicht eine Standard-LAN-MAC-Schnittstelle zur Verfügung.
Ein LEC besteht aus folgenden Komponenten:

- Systemhardware (PC, Workstation, Brücken, …)
- ATM-Interface
- Standard LAN-Software (TCP/IP, Novell, …)
- LAN-Emulation-Client-Software, die die Datenübertragung über ATM-Interface sowie alle notwendigen Steuerfunktionen beinhaltet.

3.3.4.3 LAN Emulation Server (LES)

Der LES hat die Aufgabe der Steuerung der emulierten ATM-LAN. Er enthält die Zuordnungstabelle der LEC-Adressen. Jeder an einem emulierten LAN teilnehmende LEC meldet dem LES seine LAN-MAC- sowie seine ATM-Adresse. Bevor ein LAN-Datenpaket gesendet wird, fragt der LEC beim LES nach der ATM-Adresse der Zielstation. Diese wird per Zuordnungstabelle gesucht. Ist diese dort aber nicht vorhanden, muß die Adreßzuordnung mit Hilfe eines BUS durchgeführt werden.

3.3.4.4 LAN Emulation-Configuration-Server (LECS)

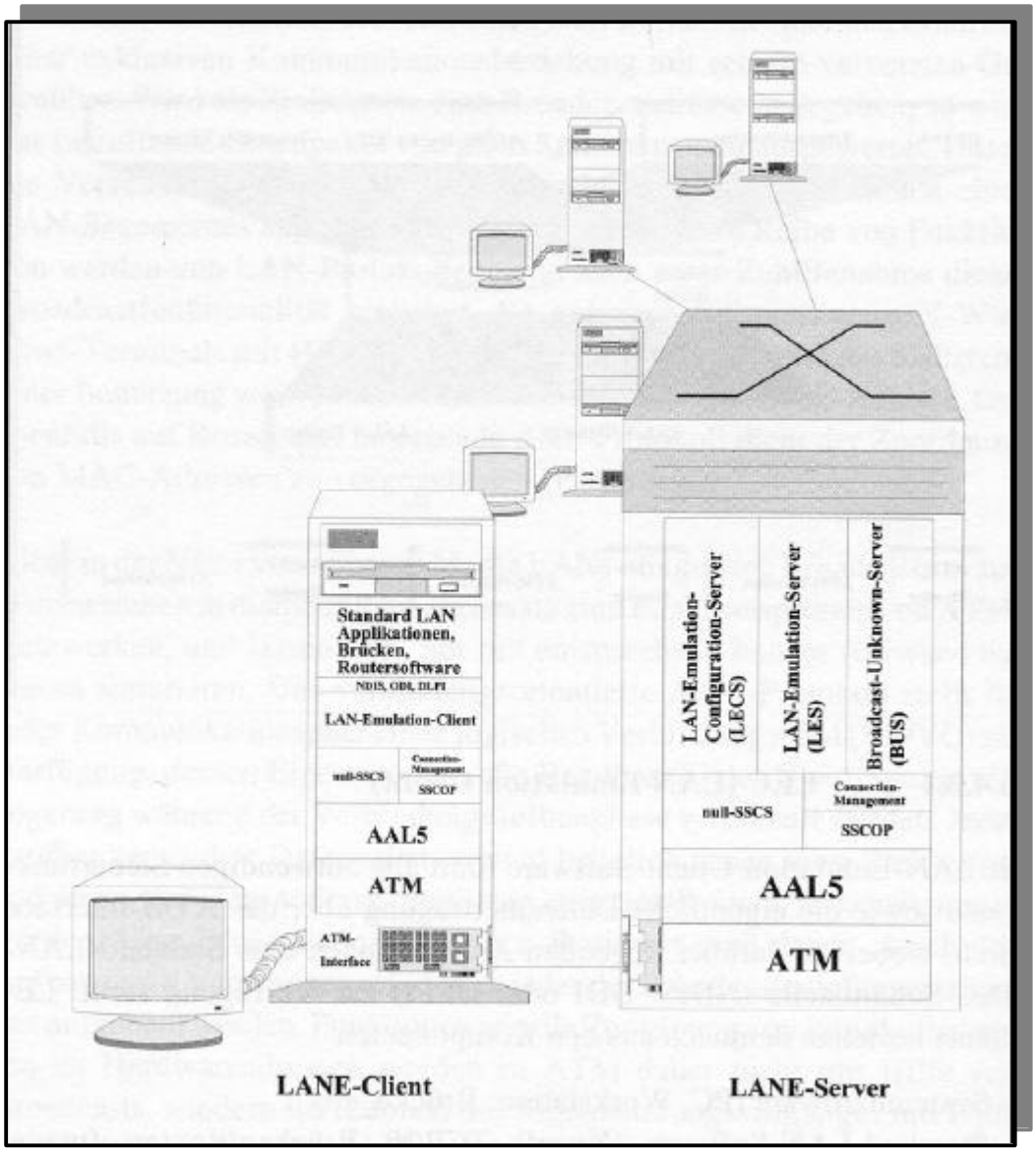


Abbildung 9: LAN-Emulation Konfiguration

Der LECS verwaltet die LEC´s unterschiedlicher LAN´s. Er hält eine Konfigurationsdatenbank mit Konfigurationsinformationen des LAN, mit deren Hilfe die Zugehörigkeit von LEC´s zu unterschiedlichen emulierten LAN´s bestimmt werden kann. Ein LEC kann gleichzeitig zu mehreren unterschiedlichen emulierten LAN´s gehören (siehe Abbildung 9).

3.3.4.5 Broadcast und Unknown Server (BUS)

Wie bereits beim LES genannt wurde, ist der BUS-Server zur Vermittlung aller Broadcast- und Multicast-Datenpakete von LE-Clients verantwortlich. Dazu gehören

- alle Datenpakete mit Broadcast- bzw. Multicast-Adresse
- Datenpakete, deren MAC-Adresse und korrespondierende ATM-Adresse nicht durch LES zugeordnet werden konnte
- Explorer-Datenpakete des Source-Routing-Mechanismus, die zur Ermittlung der optimalen Routen dienen.

Die vom BUS-Server empfangenen Datenpakete werden sequentiell an die betreffenden Gruppen von LEC´s übertragen.

3.3.4.6 Der Verkehrfluß des LAN-Emulations-Dienstes

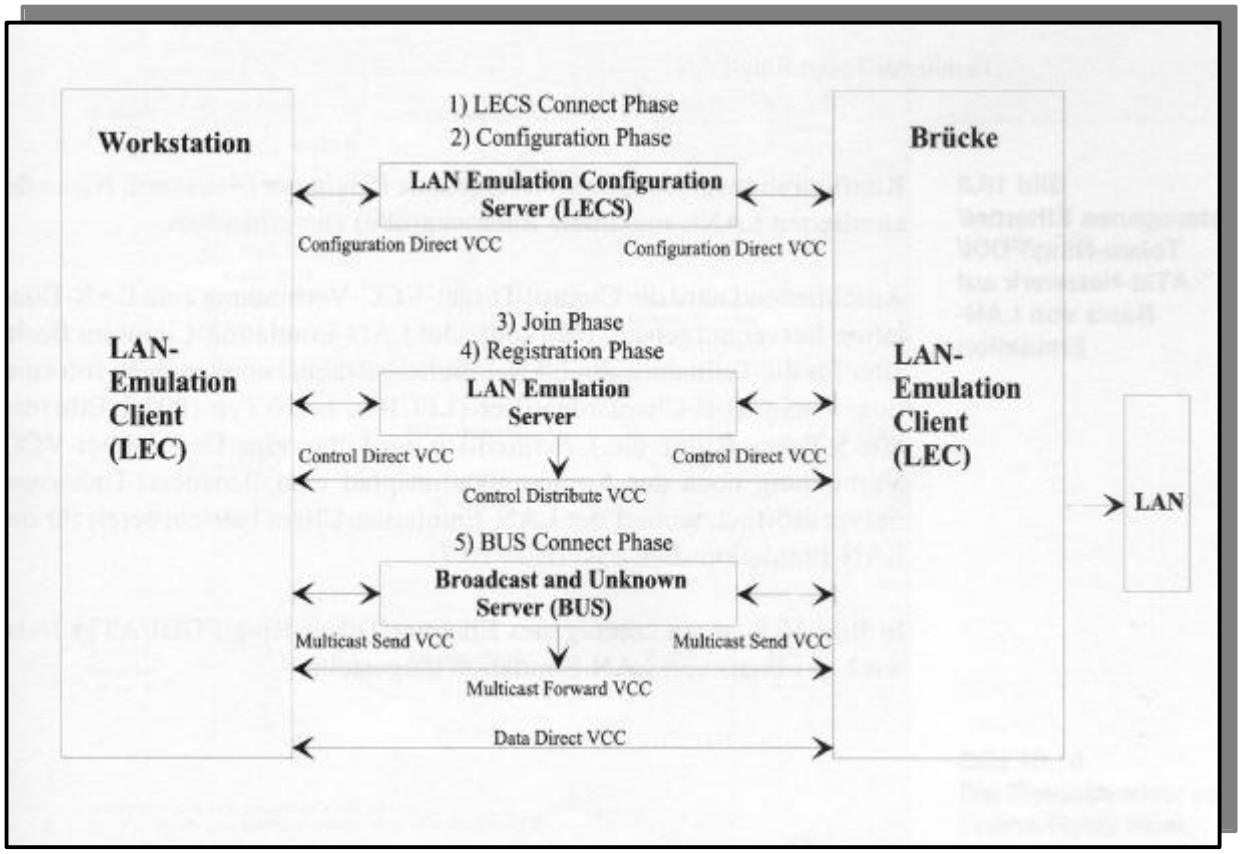


Abbildung 10: Der Verkehrfluß des LAN-Emulations-Dienstes [ATM96]

Die Kommunikation und Operationen zwischen den einzelnen LAN-Emulation-Dienstmodulen erfolgen über Steuerverbindungen (Control VCCs) und Datenverbindungen (Data VCCs). Steuerverbindungen koppeln LAN-Emulation-Clients mit LAN-Emulation-Servern sowie LAN-Emulation-Configuration-Servern. Zwischen LAN-Emulation-Clients untereinander sowie zwischen LAN-Emulation-Clients und Broadcast-Unknown-Servern erfolgt die Kommunikation über virtuelle Datenverbindungen (Data VCCs). Wird ein neuer LAN-Emulation-Client einem emulierten LAN hinzugefügt, so baut der LEC zunächst eine Control-Direct-VCC-Verbindung zum LE-Configuration-Server auf, um sich für ein bestimmtes „Emuliertes LAN“ zu registrieren, und unter Benutzung des LE-Konfigurationsprotokolles verschiedene Parameter auszuhandeln (Adressen, Name des emulierten LANs, maximale Rahmengröße).

Anschließend wird die Control-Direkt-VCC-Verbindung zum LAN-Emulation-Server aufgebaut. Damit ist der LAN- Emulation-Client im Besitz aller für die Teilnahme am LAN-Emulation-Dienst notwendigen Informationen (LE-Client Identifier (LECID), LAN-Typ (802.3 Ethernet, 802.5 Token Ring), etc.). Schließlich wird über eine Data-Direct-VCC-Verbindung noch der Kommunikationspfad zum Broadcast-Unknown-Server geöffnet, worauf der LAN-Emulation-Client betriebsbereit für den LAN-Emulation-Dienst ist (siehe Abbildung 10).

4 Multimedia

4.1 WAVE-Format für digitale Audiodaten

Für den Windows-Standard hat sich hauptsächlich das WAVE-Format durchgesetzt. Die Anordnung der Daten gestaltet sich einfach: Nach dem Header (meist aus 44 Bytes bestehend, ausführlich über den Header des WAVE-Formates siehe Anhang A) schließt sich direkt die Datenbytes an. Bei Stereodateien alternieren ständig die Daten für den linken und rechten Kanal, wobei mit dem linken Kanal begonnen wird. Allerdings enthält diese Methode keinerlei Kompression. Durch diese direkte Datenstruktur gestattet das WAVE-Format eine einfache Möglichkeit der Wiedergabe.

Tabelle 8 gibt einen Überblick über die verschiedenen Qualitäten, mit denen die Audiosignale im WAVE-Format aufgenommen und wiedergeben werden können.

Formate	Frequenz	Type	Bits/Sample	Qualitäten
WAVE-PCM (Puls Code Mode)	8000	Stereo	8 oder 16	Telefonqualität
		Mono		
	11025	Stereo	8 oder 16	Telefonqualität
		Mono		
	22050	Stereo	8 oder 16	Radioqualität
		Mono		
	44100	Stereo	8 oder 16	CD-Qualität
		Mono		

Tabelle 8: Qualitäten der Audiodaten mit dem WAVE-Format

Für weitergehende Information zu diesem Thema, der Audiodigitalisierung, sei an dieser Stelle auf einschlägige Literatur [MULT95] hingewiesen.

4.2 Kompressionsalgorithmen für Motion Video

Bei der Übertragung oder Speicherung von Motion-Video ist es auf Grund des immensen Datenvolumens sinnvoll, dieses je nach verfügbaren Übertragungsbandbreiten zu komprimieren. Dabei lassen sich folgende Anforderungen an Kompressionsverfahren stellen [STEIN94]:

- Unabhängigkeit von Bildgröße, Farben, Inhalt, Bildwiederholfrequenz
- In Verbindung mit Echtzeitanwendungen soll die Kompressions- bzw. Dekompressionsverzögerung nicht größer als 40 ms sein.
- Die Komplexität des Verfahrens darf Softwarelösungen nicht ausschließen.
- Der Qualitätsverlust des komprimierten und anschließend dekomprimierten Bildes sollte gering sein.
- Standardtreue, um Interoperabilität auch in einer heterogenen Umgebung zu gewährleisten.

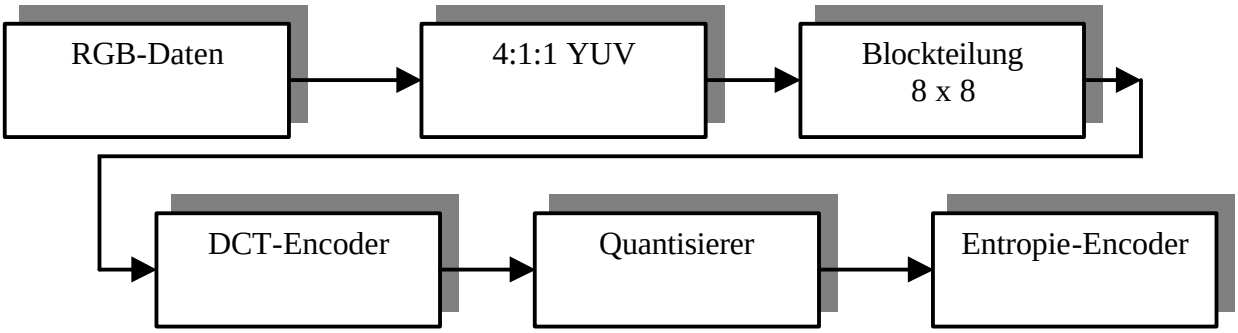
Außerdem müssen die Kompressionsverfahren verschiedene Kompressionsstärken ermöglichen und dadurch je nach verfügbaren Übertragungsmöglichkeiten entsprechende Datenraten eingestellt werden können.

Zur Zeit gibt es zwei sehr verbreitete standardisierte Verfahren, die in der Praxis schon viel angewendet wurden – Motion-JPEG und MPEG.

4.2.1 Motion-JPEG

Motion-JPEG, kurz M-JPEG, basiert auf dem internationalen Standard JPEG (Joint Photographik Experts Group) und ist eine der leistungsfähigen Kompressionsmethoden. Die Kompression bzw. Dekompression wird in mehrere Schritte geteilt. Abbildung 11 zeigt einen Überblick über die Umwandlung in das JPEG-Datenformat. Die Rückwandlung geschieht ganz analog, nur in umgekehrter Richtung. Die einzelnen Arbeitsschritte werden heutzutage bereits

über spezielle Prozessoren hardwaremäßig in Echtzeit bewältigt [MULT95],



[STEIN94], [VIDEO96].

Abbildung 11: Schema des JPEG-Encoders

4.2.1.1 Umkodierung von RGB-Werten in YUV-Werte

RGB-Werte (Rot-Grün-Blau) der Pixel werden zunächst in YUV-Werte nach folgenden Gleichungen umgewandelt:

$$Y = 0.299\,R + 0.587\,G + 0.144\,B$$
$$U = -0.1687\,R - 0.3313\,G + 0.5\,B$$
$$V = 0.5\,R - 0.4187\,G - 0.0813\,B$$

Gleichung 1: Umkodierung von RGB-Werten in YUV-Werte

Das YUV-Format ist im Videobereich sehr verbreitet. Dabei beschreibt Y den Helligkeitseindruck und U und V beinhalten die Farbwerte des Bildes. Dieser Vorgang, der für sich betrachtet noch keinerlei Datenreduktion beinhaltet, bildet jedoch die Basis für eine erste Reduktion. Diese beruht auf der Tatsache, daß das menschliche Auge auf örtliche Helligkeitsänderungen sehr viel empfindlicher reagiert als auf Farbänderungen. Für ein Bildelement von 2 x 2 Pixeln erfolgt nur noch eine Abtastung bzw. Wertangabe jeweils für U und V. Das Abtastverhältnis zwischen Luminanz und beiden Chrominanzkanälen ist also 4:1:1. Für dieses Feld von 4 Pixeln sind also nicht mehr 12 Werte (wie für RGB nötig ist), sondern

nur noch 4+1+1=6 Werte relevant. Die Farbkodierung geschieht nach den Mittelwerten der vier Pixel.

Nach diesem Schritt findet bereits eine Farbreduktion von 50% statt. Die sich ergebenden Werte werden dann in Blöcke zu 8 x 8 Pixeln zusammengefaßt.

4.2.1.2 Diskrete Kosinus Transformation (DCT)

Es gibt verschiedene Technik zur Transformation von einer in eine andere mathematisch günstigere Form wie z.B. die Karhunen-Loeve-Transformation (KLT), die Diskrete Fourier Transformation (DFT) und die Diskrete Kosinus Transformation (DCT). Im Bereich der Bildkodierung wird meisten die DCT eingesetzt.

Jeder Datenblock (8 x 8 Pixel) wird zunächst einer zweidimensionalen DCT unterworfen. Die Werte für Y, U und V werden in einen Bereich zwischen –128 und +127 transformiert.

Die Transformationsgleichung lautet:

$$H(i,j) = \frac{1}{4} C(i) C(j) \left[\sum_{x=0}^{7} \sum_{y=0}^{7} A(x,y) \cdot \cos \frac{(2x+1) i \pi}{16} \cdot \cos \frac{(2y+1) j \pi}{16} \right]$$

$$\text{mit } C(i), C(j) = \begin{cases} 1/\sqrt{2} & \text{für } i, j = 0 \\ \text{sonst } 1 \end{cases}$$

Gleichung 2: Transformationsgleichung bei DCT

Das Abtasten des 8x8 Datenblockes geschieht in Zickzacklinien nach Abbildung 12 und zwar für alle Kanäle Y, U und V, die mit dem Pixelwert A(x,y) korrespondieren. Es ergeben sich entsprechend 64 Werte für H, wobei diese jetzt allerdings Frequenzkomponenten darstellen.

Bei der DCT entsteht aus der Zeitfunktion des Bildes eine Frequenzfunktion. Prinzipiell erhält man zwar ebenfalls wieder 64 Werte, allerdings weisen die Komponenten niedriger Indizes bei den meisten Bildinformationen hauptsächlich einen von Null verschiedenen Wert auf. Die DCT bewirkt, daß große, langwellige Bilddetails in Form von informationstragenden Koeffizienten erhalten bleiben und

sich in der oberen linken Ecke des Datenblockes konzentrieren. Die für die feinen und kurzwelligen, vom menschlichen Auge nur schwer zu erkennenden Details verantwortlichen Koeffizienten werden zu Null und belegen den unteren rechten Bereich.

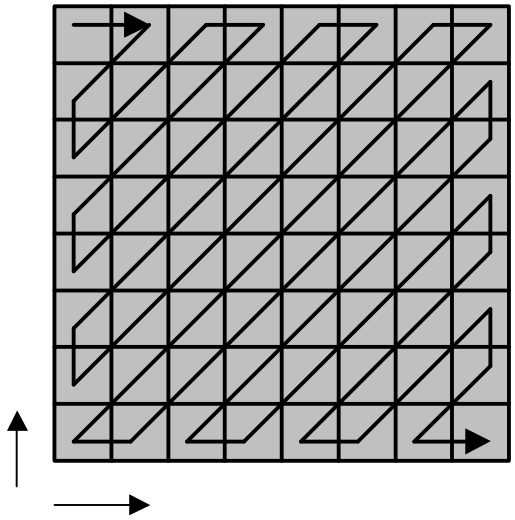


Abbildung 12: Zickzack-Abtastung des 8x8-Blocks bei DCT

Diese Verteilung ist bei jedem Bild anders. Die DCT trennt lediglich Wichtiges von weniger Wichtigem, ohne dabei (abgesehen von der Rechenungenauigkeit) Informationen zu zerstören. Sie ist umkehrbar, d.h. sie bewirkt keinen Informationsverlust und keine Datenverdichtung. Zur Rekonstruktion eines transformierten Bildes wird die Inverse DCT (IDCT) benutzt.

Abbildung 14 Illustriert ein einfaches Beispiel für die Kodierung eines 8x8 Pixel-Blockes.

4.2.1.3 Quantisierung

Die Unterteilung der sich ergebenden Werte in diskrete Stufen nennt man Quantisierung. Der sich später einstellende Kompressionsfaktor hängt maßgeblich mit der Feinheit der Unterteilung zusammen, wie dies auch aus Abbildung 13 hervorgeht. Die Gleichung für die Quantisierung ist die Funktion

round(*Wert***,** *Stufen*)[8], eine Funktion, in die der vom Nutzer festgelegte Kompressionsfaktor eingeht:

H*(i,j) = **round** (H(i,j), Stufen) z.B. 6 = **round**(19,3)

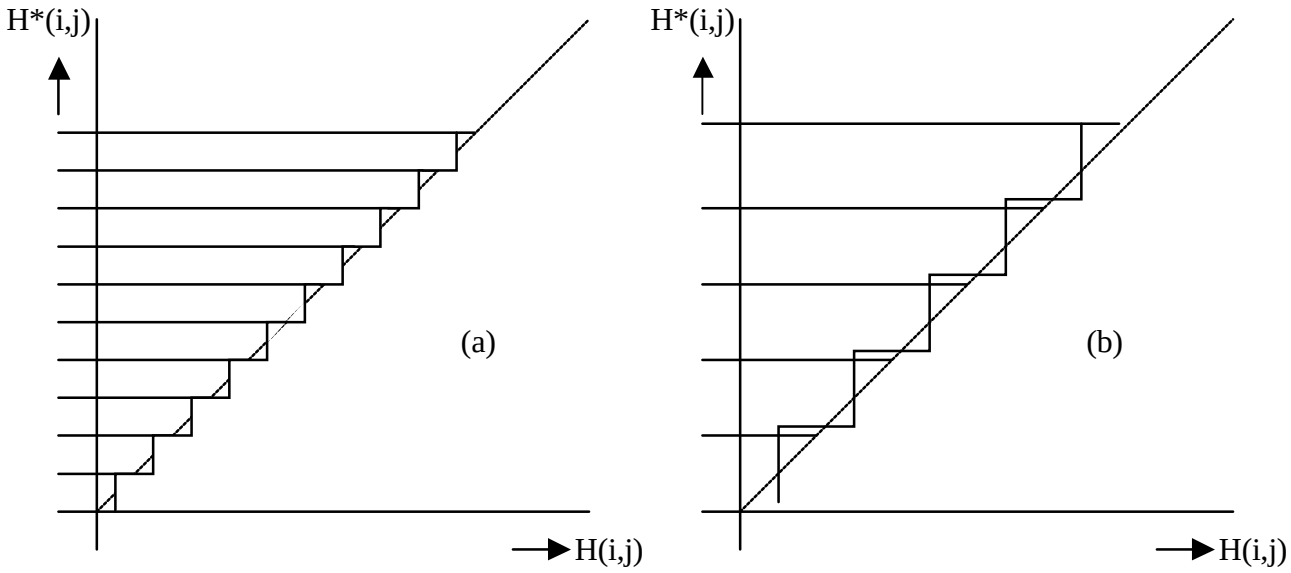


Abbildung 13: Unterschiedliche Quantisierungen (a) fein, (b) grob [MULT95]

So werden die in Abbildung 14 (b) aus dem Originalblock erzeugten DCT-Koeffizienten H(i,j) in die quantisierten Koeffizienten H*(i,j) umgewandelt. Der erste Koeffizient A(0,0) wird mit der Stufe 8, alle andere Koeffizienten werden mit der Stufe 6 quantisiert. Nur die ersten drei Felder bleiben dabei von Null verschieden: 684 -> 86, 19 ->3, -37 -> -6, alle andere sind Null, Abbildung 14(c).

Jeder DCT-Koeffizient H(i,j) wird mit einer Schrittweite quantisiert, d.h., jeder der 64 Werte eines 8x8 Blockes kann separat eingestellt werden und bietet somit der Anwendung die Möglichkeit, bestimmte Frequenzen stärker zu wichten als andere. Durch die Wahl entsprechend großer Quantisierungsstufen wird erreicht, daß bevorzugt Information vernichtet wird, die das Auge nicht wahrnimmt, in dem

[8] **Round** rundet das Ergebnis der Division in eine ganze Zahl

die hochfrequenten Bereiche – verantwortlich für die Detailsdarstellung – weggelassen werden..

Die Anzahl der diskreten Stufen ergibt die jeweilige Informationsbreite. Durch die Quantisierung entsteht ein Datenverlust, der um so größer ist, je größer die Stufen sind. Die Quantisierungsschrittweiten werden in Tabellen im Header der JPEG-Datei abgelegt. JPEG definiert keine Vorgaben für die Quantisierungstabellen. Es muß aber bei der Dequantisierung die gleiche Schrittweite wie bei der Quantisierung verwendet werden.

Abbildung 14: Beispiel über die Kodierung eines 8x8 Blockes
(a) Original Block (8*8*8=512 Bits) A(x,y); (b) DCT-Koeffizienten H(i,j); (c) quantisierter Koeffizient H*(i,j), erster Koeffizient mit Stufe 8, alle anderen 6; (d) Lauflängenkodierung (e) inverser quantisierter Koeffizient – inverse H*(i,j) (f) der rekonstruierte Block

75	76	77	78	79	80	81	82
77	78	79	80	81	82	83	84
79	80	81	82	83	84	85	86
81	82	83	84	85	86	87	88
83	84	85	86	87	88	89	90
85	86	87	88	89	90	91	92
87	88	89	90	91	92	93	94
89	90	91	92	93	94	95	96

(a)

684	19	1	2	0	-1	0	-1
-37	0	-1	0	0	0	0	-1
0	0	0	0	0	0	0	0
-4	-1	-1	-1	-1	0	-1	-1
0	0	0	0	0	0	0	0
-2	0	0	-1	0	-1	0	-1
0	0	0	0	-1	-1	-1	-1
-1	-1	-1	0	-1	0	-1	0

(b)

86	3	0	0	0	0	0	0

-6	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(c)

76	76	77	79	80	81	82	83
77	77	78	80	81	82	83	84
79	79	80	81	83	84	85	86
81	82	83	84	85	87	88	88
84	84	85	87	88	89	90	91
86	87	88	89	91	92	93	93
88	89	90	91	92	94	95	95
89	90	91	92	93	95	96	96

(f)

688	-21	0	0	0	0	0	0
-39	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(e)

Lauflängenkodierung

0	86	01010110
0	-3	001011
0	-6	001000011
	EOB	10

Totale Kodelänge=25 Bits

Kompressionsfaktor 20

EOB – Blockende

(d)

Jede quantisierte Matrix wird durch Zickzack-Sequenz von oben links nach unten rechts abgetastet. Die Koeffizienten werden in dieser Folge sequentiell betrachtet. Durch dieses Vorgehen wird eine Vorsortierung erreicht, d.h. die durch Quantisierung meist zu Null gewordenen hochfrequenten DCT-Werte reihen sich nach den informations-tragenden Koeffizienten der oberen linken Ecke ein.

Die Lauflängenkodierung, z.B. Abbildung 14(d) ersetzt im Prinzip jede Sequenz von gleichen Werten durch deren Längenangabe und ihren Wert selbst. Ein spezielles Flag markiert dann die Kodierung. Jede solche Sequenz wird dann durch einen drei Byte großen Ausdruck ersetzt. Zehn Nullen (0000000000) in einer Folge würden beispielsweise durch den Ausdruck 0#10 ersetzt werden, mit dem Zeichen (#) als Flag. Dieses Verfahren ist effizient ab einer Länge von mindestens vier gleichen Werten.

Wenn man im Beispiel aus Abbildung 14 umgekehrt aus der kodierten Matrix wieder das Bild rekonstruiert, erhält man zunächst die rekonstruierten Werte H*(i,j), vgl. Abbildung 14(e), aus denen sich mit Hilfe der IDCT das rekonstruierte Bild ergibt. Diese ist durch die Koeffizienten A*(i,j), vgl. Abbildung 14(f) charakterisiert, die den entsprechenden Farbwert des Bildelementes darstellen. Vergleicht man das Original (a) mit dem rekonstruierten Bild (f) läßt sich trotzt hoher Kompression relativ hohe Ähnlichkeit feststellen; die Koeffizienten unterscheiden sich um jeweils maximal +/- 2 Einheiten. Als Farbwert[9] ist dies mit dem menschlichen Auge nur schwer zu unterscheiden.

4.2.1.5 Huffman-Kodierung

Ein weiterer Schritt bei der Kompression basiert auf der Tatsache, daß die quantisierten Werte mit sehr unterschiedlichen Häufigkeiten vorkommen. Durch die Huffman-Kodierung werden statistisch häufig vorkommende längere Bitfolgen durch kürzeren Code ersetzt, die dann in Tabellen abgelegt sind. Nur selten vorkommende Werte erhalten die selbe Kodierung. Diese Tabellen werden für die

[9] genauer: Wert der Farbintensität einer der drei Grundfarben.

Dekompression unbedingt gebraucht und befinden sich deshalb neben den Quantisierungstabellen im Header der JPEG-Datei.

4.2.2 MPEG

MPEG (Motion Picture Experts Group) berücksichtigt im Gegensatz zu JPEG die zeitlichen Beziehungen aufeinander folgender Bilder, stützt sich aber in hohem Maße auf das im vorigen Abschnitt besprochene JPEG. Bewegt sich ein Bildteil, so werden die Verschiebungen der Bereiche einfach durch einen Bewegungsfaktor beschrieben, das ganze Bild selbst braucht nicht neu kodiert zu werden.

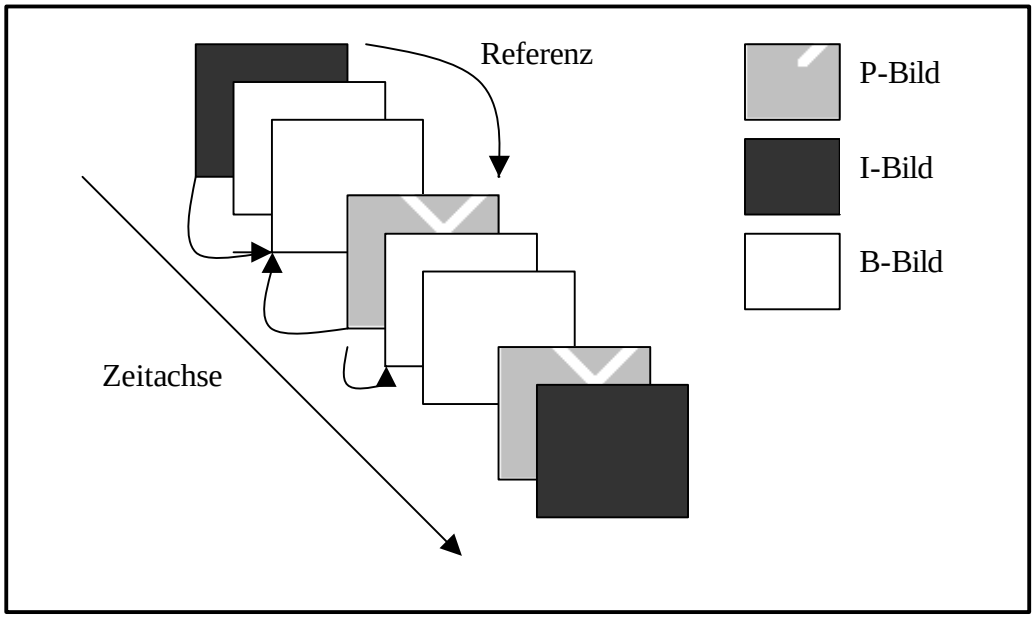


Abbildung 15: Arten der Einzelbilder in MPEG [STEIN94]

Bei der MPEG-Kompression unterscheidet man vier Arten der Kodierung eines Bildes. Grundsätzlich erfolgt im Gegensatz zu JPEG die Aufteilung des Bildes in 16x16 Datenblöcke. Auch hier kommt die diskrete Kosinus-Transformation zur Anwendung, allerdings nur bei komplett neu zu kodierenden sogenannten I-

Bildern[10] (Intraframe Kodierung). I-Bilder werden ohne zusätzliche Informationen anderer Einzelbilder kodiert. P-Bilder[11] (Voraus-Bild) benötigen zur Kodierung und Dekodierung Informationen vorangegangener I- oder P-Bilder. Zur Dekodierung ist hier die Dekompression des vorherigen I-Bildes und eventuell noch dazwischenliegender P-Bilder nötig. B-Bilder[12] benötigen zur Kodierung und Dekodierung Informationen vorangegangener und später auftretender I- oder P-Bilder. Die höchste Kompressionsrate ist durch die Kodierung von B-Bildern erreichbar. D-Bilder[13] sind intraframekodiert und werden für einen schnellen Vorlauf verwendet (siehe Abbildung Abbildung 15).

Für weitere Informationen zu MPEG sei an dieser Stelle auf einschlägige Literatur [STEIN94],[MULT95] zu diesem Thema hingewiesen.

4.2.3 Vergleich beider Verfahren

Der größte Komprimierungsfaktor bei MPEG für Motion Video beträgt ca. 1:240, bei M-JPEG ca. 1:100. Allerdings sind hohe Kompressionsfaktoren nur auf Kosten der Bildqualität zu erzielen. Für eine akzeptable Bildqualität muß bei M-JPEG der Kompressionsfaktor aber kleiner als 10 eingestellt werden. Die Kompression unter MPEG ist viel aufwendiger als die Dekompression, was zur Asymmetrie in der Bearbeitungszeit führt. Diese Asymmetrie bewirkt, daß nicht, wie bei M-JPEG, die gleiche Zeitspanne für Kodierung und Dekodierung benötigt wird. Außerdem bietet M-JPEG durch einfacher ausgelegte Algorithmen eine von Grund auf schnellere Methode, die auch Softwarelösungen (besonders bei der Kompression) nicht ausschließt. Außerdem sind die Hardwarekosten für MPEG zur Zeit noch viel höher als für M-JPEG. Tabelle 9 gibt den Überblick über den Vergleich beider Verfahren.

[10] Intra Coded Picture

[11] Predictive Coded Picture

[12] Bidirectionally Predictive Coded Picture

[13] DC Coded Picture

	M-JPEG	MPEG
Komprimierungsfaktor	1:100	1:240
Bearbeitungszeit	Symmetrie	Asymmetrie
Algorithmen	Einfach	aufwendig
Hardwarekosten	Ab 400 DM	Kodierung > 10000 DM Dekodierung ab 800 DM

Tabelle 9: Vergleich verschiedener Eigenschaften von M-JPEG und MPEG

4.3 Video-Overlay

Einige Graphikadapter ermöglichen, ein Videobild auf den Computerbildschirm einzumischen und es an die Größe eines vorgegebenen Fensters anzupassen. Dies geschieht durch Überlagerung der Echtzeitvideosignale mit dem Ausgangssignal des Videoadapters. Bei einigen Graphikadaptern wird es dazu digitalisiert, dann in einem Video-RAM zwischengespeichert und von dort mit der Frequenz der Computergraphikkarte ausgelesen und auf das gewünschte Format gebracht. Diese Technik wird als Video-Overlay bezeichnet. Seine Wirkungsweise ist graphisch in Abbildung 16 dargestellt.

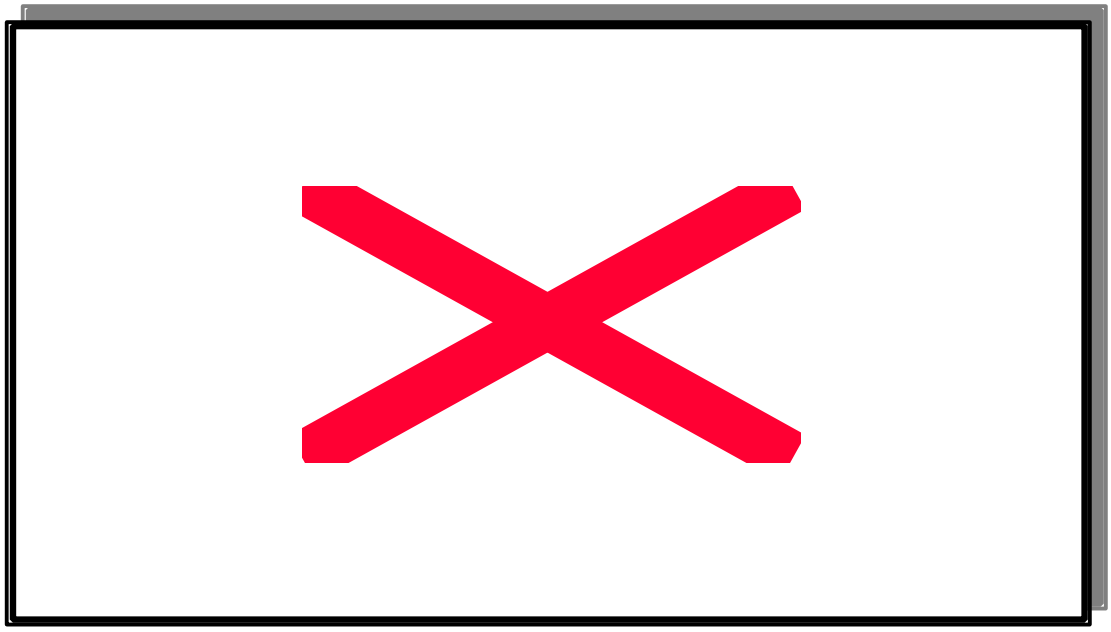


Abbildung 16: Wirkungsweise des Video-Overlay

Graphikadapter, die Video-Overlay unterstützen, haben außer dem üblichen Bildspeicher noch einen sog. Overlay- oder Alpha-Speicher. Dieser ist bei 32-Bit-Farbtiefe völlig äquivalent zum Farbspeicher: Es stehen 8 Bit pro Pixel für die Attributzuordnung zur Verfügung. Doch auch bei einer Farbtiefe von 16 Bit steht immer noch 1 Bit pro Pixel für Overlay-Zwecke zur Verfügung. Bei der Abfrage des Bildspeichers entscheidet der Alpha-Wert darüber, ob das betreffende Pixel das Live-Video oder die Computergraphik wiedergeben soll. Es gibt zwei Arten der Overlaydarstellung: die örtliche und die inhaltliche Darstellung.

Bei der *örtlichen Overlaydarstellung* wird eine Fläche festgelegt (z.B. ein Rechteck oder eine Fläche, die auch wesentlich komplexer sein darf), in der das Live-Video erscheinen soll. Die Vorgabe der Fläche kann über Koordinaten oder über ein Zeigegerät (z.B. die Maus) erfolgen.

Bei der *inhaltlichen Overlaydarstellung* entscheidet dagegen der Inhalt des Farbspeichers. So kann z.B. ein bestimmter Farbwert der Graphik festgelegt werden, für den die betreffenden Pixel nur noch das Live-Video zeigen. Daneben ist natürlich auch die Definition von Farbwertbereichen möglich. Die Manipulation des Alpha-Speichers geschieht i.a. per Software, deshalb ist eine kurze Berechnungszeit unvermeidbar. Je nach Komplexität der Graphik entsteht ein mehr oder weniger bizarres Gebilde, in dem nun das Live-Bild erscheint.

In diesem Abschnitt wurde die Wirkungsweise des Video-Overlay nur als Überblick gegeben, weil dies eigentlich bereits vom Graphikadapter (falls er Video-Overlay unterstützt) und dessen Treiber übernommen wird. Bei der Programmierung muß lediglich für die Einstellung und Einhaltung von Anschlußbedingungen gesorgt werden, um das Video-Overlay zu aktivieren.
Für weitere Informationen zur Overlaytechnik sei auf einschlägige Literatur [MULT95] hingewiesen.

5 Implementierung

Im folgenden Kapitel werden die Grundlagen des entwickelten Videokonferenz-Systems vorgestellt. Zur Illustration werden kurze Auszüge aus dem Quellcode gegeben.

5.1 Entwicklungsumgebung und Übersicht

Eine Softwarelösung für die Kompression bzw. Dekompression der Videodaten mit einer Bildqualität wie PAL ist aus folgenden Gründen zur Zeit noch nicht realisierbar:

- Ein durchschnittlicher Rechner (Pentium 120 MHz) von heute ist nicht in der Lage, 25 Bilder (720x576 Bildpunkte) in eine Sekunde mit Motion-JPEG-Verfahren softwaremäßig zu kodieren bzw. zu dekodieren. Im Test konnte ein Rechner mit Pentium-Prozessor 120 MHz, 32 MB RAM nur zwei Bilder in PAL-Qualität in einer Sekunde mit dem JPEG-Verfahren softwaremäßig dekodieren.
- Der Prozessor wird allein durch die Kompression-/Dekomressionsaufgabe voll in Anspruch genommen und führt zu einer Vollbelastung des ganzen Systems. Im genannten Test war der Prozessor mit der Dekompression zu nahezu 100% ausgelastet. Weitere parallel ablaufende Prozesse wie z.B. Senden und Empfangen von Daten oder Systemprogramme müssen aber ebenfalls von Prozessor abgearbeitet werden.

Die Darstellung des empfangenen Bildes ist auf der Seite des Empfängers ein wichtiges Kriterium. Sie darf nicht zu viel Zeit in Anspruch nehmen, da sonst eine Symmetrie der Bearbeitungszeit bei Sender und Empfänger nicht mehr eingehalten werden kann. Eintreffende Frames können nicht rechtzeitig bearbeitet und müssen verworfen werden. Es ist außerdem wünschenswert, daß der Prozessor dabei so wenig wie möglich belastet wird. Eine reine Softwarelösung für die Darstellung empfangener Frames kann diesen Anforderungen zur Zeit nicht nachkommen.

Aufgrund der oben besprochenen Probleme sollten die Kodierung bzw. Dekodierung und Darstellung der Videodaten durch die Hardware erfolgen. Derartige Hardware, welche die Kompression bzw. Dekompression des Videostromes und Videowiedergabe per Overlaytechnik anbietet, wird kommerziell bereits kostengünstig angeboten. Deshalb wurden Videokarten verschiedener Hersteller bezüglich Kompatibilität und Funktionalität untersucht. Eine Liste der getesteten Videokarten wird am Ende dieses Abschnittes gegeben.

Für die Entwicklung des Systems standen drei Rechner zur Verfügung, die miteinander über Ethernet (10 Mbit/s) und ATM (155 Mbit/s über ein Switch IBM-8260) verbunden waren. Die Rechner waren jeweils mit folgenden Komponenten bestückt:

- Pentium Prozessor 120 Hz, 32 MB RAM
- einer Graphikkarte Miro Crystal VR4000
- einer Soundkarte SB 16 oder einer Avance Logic ALS100 (vollduplex)
- einer Videokarte FPS60 oder einer AVMaster von FAST
- einer Videokarte DC30 von MIRO
- einer Kamera von SONY
- einer Ethernet-Karte von ACCTON 10 Mbit
- einer ATM-Karte von MADGE 155 Mbit

Die Implementierung wurde in C++ geschrieben und besteht aus folgenden Modulen:

- dem Hauptmodul Main
- dem Benutzerprogramm
- dem Audiomodul
- dem Videomodul
- dem Netzwerkmodul

Das Programm läuft auf einer Windows-32 Bit-Plattform und wurde auf Windows95 und WindowsNT mit Ethernet und ATM LAN Emulation getestet.

5.2 Das Programmierkonzept

Durch die Aufteilung der Implementierung in getrennte unabhängige Module ist jederzeit eine komponentenweise Änderung oder Verbesserung der Software des Systems möglich. Für die Audio-, Video-, sowie Netzwerkmodule wurden jeweils eigene C++ Klassen implementiert. Diese einzelnen Module bzw. Klassen können unabhängig voneinander für weitere Anwendungen benutzt werden.

Die den Aufgabenbereichen zugeordneten Module bzw. Modulgruppen agieren völlig unabhängig voneinander. Die Kommunikation zwischen den Modulen und die Koordination der Modulaktivität wird von einem Hauptmodul (Main) überwacht (Abbildung 17). Die Kommunikation unter Windows erfolgt über Nachrichten. Der Vorteil der Abwicklung aller Aktivitäten und Nachrichten innerhalb der Anwendung über das Modul Main besteht darin, daß eine zentrale Kontrolle der Anwendung in einem Modul ermöglicht wird. Nachteilig wirkt sich diese Konzeption auf die Verarbeitungsgeschwindigkeit aus: Da fast jede interne Nachricht über das Modul Main laufen muß, entstehen Verzögerungen, die die Ausführung verlangsamen.

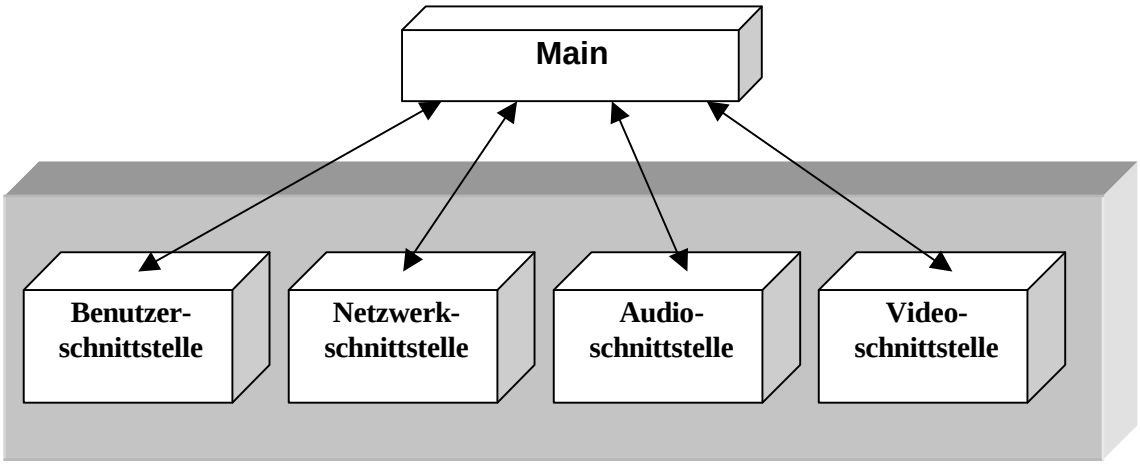


Abbildung 17: Übersicht des Programmaufbaus

Folgende Punkte wurden bei der Implementierung berücksichtigt und realisiert:

- Benutzerfreundlichkeit.
- Unterstützung verschiedener Audio- und Videoformate.

- Ausnutzung der Multitaskingfähigkeit von WindowsNT und Windows95.
- Hardwareunabhängige Programmierung d.h. das lauffähige Programm ist von der installierten Systemhardware unabhängig.

5.3 Das Audiomodul

5.3.1 Übersicht

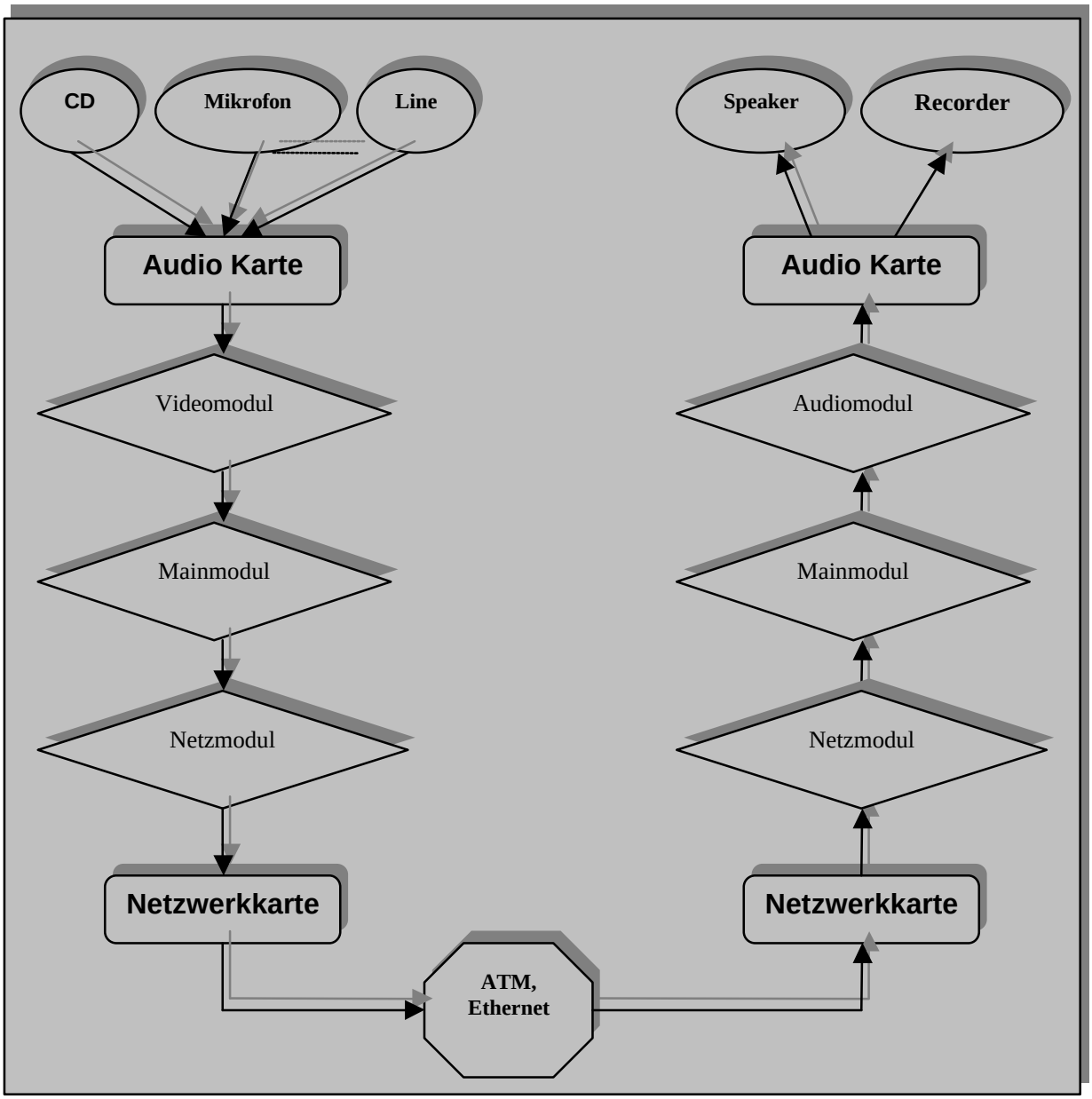


Abbildung 18: Audioaufnahme und -wiedergabe

Audiosignale können vom Mikrofon (bzw. CD-Spieler, Line-In, etc.) durch den Audiotreiber aufgenommen, digitalisiert und über das Netzwerkmodul unabhängig vom Videostrom übertragen werden. Empfangene Audiodaten werden vom Netzwerkmodul an das Audiomodul übergeben und durch entsprechende Maßnahmen für eine Synchronisation mit dem Videostrom wiedergeben (siehe Abbildung 18).

5.3.2 Audioaufnahme

Für die Aufnahme der Videodaten ist die C++-Klasse ***CWaveIn*** implementiert. Die Audiodaten können dadurch unabhängig vom Video aufgenommen werden - Voraussetzung dafür ist eine installierte Soundkarte. Da aber jeder zusammen mit der Videohardware gelieferte Capturetreiber die Audio-Aufnahme steuern kann, wird dies dem Capture-Treiber überlassen. Dadurch wird eine Synchronisation zwischen Audio- und Videoströmen über die gleichzeitige Aufnahme der beiden Datenströme ermöglicht. Die Wiedergabe-Funktionen werden im Audiomodul mit Zwischenspeicher-Funktionen versehen.

Die Aufnahme der Audiodaten wird also vom Video-Capturetreiber übernommen. Der Capturetreiber stellt einen Puffer für die Aufnahme der Audiodaten bereit und reserviert diesen. Es muß aber die Größe dieses Puffers bestimmt werden. Dessen Größe und das Audioformat entscheiden, wieviele Abtastungen in einer Sekunde erfolgen können (10 Abtastungen bedeuten, daß alle 100 ms der Audiopuffer mit Daten gefüllt wird). Es sollen z.B. Audiodaten mit dem folgenden Format 11025 Hz, Stereo und 16 Bit pro Sample aufgenommen werden. Das bedeutet eine Datenrate von (11025*2*16 = 352800 Bit) 44100 Byte pro Sekunde. Eine Puffergröße von 4100 Byte zwingt dem Capturetreiber dazu, 10 mal in einer Sekunde den Puffer mit Audiodaten füllen zu müssen. Es dauert genau 100 ms, bis der Puffer mit Audiodaten gefüllt wird. Ohne Berücksichtigung der Rechen- und Übertragungszeit entsteht schon eine Ende-zu-Ende-Verzögerung von 100 ms. Eine Zwischenspeicherung beim Empfänger erhöht diese Verzögerung noch um ein Vielfaches der Warteschlangenlänge. Die Qualität der Kommunikation ist dadurch nicht mehr akzeptabel.

Um die Ende-zu-Ende Verzögerung zu minimieren, muß daher bereits den Aufnahmepuffer so klein wie möglich gehalten werden. Das hat wiederum folgende Nachteile:

- Es müssen viele kleine Audiopakete übertragen werden, große Pakete lassen sich aber effektiver übertragen (siehe Performance-Untersuchung, Abschnitt 5.7.2)
- Nicht alle Soundkarten (abhängig von der Qualität) können sehr kleine Audiopakete (10 – 20 ms) exakt aufnehmen bzw. wiedergeben.

Die optimalen Parameter beim Setzen der Audio-Abtastfrequenz und bei der Ermittlung der Warteschlangengröße für den Empfänger werden Abschnitt 5.7.3 bestimmt und wiedergegeben, vgl. Tabelle 10.

5.3.3 Audioformat

Audiodaten werden im WAVE-Format aufgenommen und wiedergegeben. Für das WAVE-Format können alternativ verschiedene Qualitäten eingestellt werden. **Tabelle 8** im Abschnitt **4.1** über das **WAVE-Format für digitale Audiodaten** hat einen Überblick über die möglichen Werte gegeben.

Da das Übertragungssystem verbindungslos arbeitet, hat der Empfänger keine Kenntnis über das Format sowie die Abtastfrequenz der aufgenommenen Audiodaten. Für eine korrekte Wiedergabe benötigt der Empfänger aber alle Informationen über die empfangenen Daten. Die Wiedergabe der Audiodaten in einem falschen Format kann fatale Folgen haben: Im schlimmsten Fall wird die Arbeit des Audiotreibers abgebrochen, und das gesamte System muß neu gestartet werden.

Die Abtastfrequenz spielt für den Empfänger eine große Rolle bei der Kompensierung des Delay Jitter. Mit ihrer Hilfe kann die Größe der Warteschlange (d.h. die Anzahl der zwischenzuspeichernden Audiopakete, siehe

2.5) entsprechend der gewünschten gesamten Ende-zu-Ende Verzögerung eingestellt werden. Mehr darüber wird in Abschnitt 5.3.6 besprochen.

5.3.4 Audioübertragung

Es wurde bei der Initialisierung des Capturetreibers eine Callback-Funktion[14] installiert, die jedesmal vom Capturetreiber aufgerufen wird, wenn der aufzunehmende Audiopuffer vollständig mit Daten gefüllt ist.
Die Audiodaten werden zusammen mit der entsprechenden Formatangabe sowie der Angabe über die Abtastfrequenz im Audio-Paket gekapselt und anschließend vom Netzwerkmodul übertragen.

5.3.5 Audioempfang

Vor der Wiedergabe werden die empfangenen Audiodaten zuerst zwischengespeichert, um Delay Jitter zu kompensieren. Wie bereit im Kapitel **2** über die Eigenschaften und Anforderungen des Systems besprochen wurde, sollen zur Bestimmung der Größe der Warteschlange verschiedene Werte wie die Fehlerrate und die gesamte Ende-zu-Ende-Verzögerung berücksichtigt werden.

Um eine flüssige Kommunikation zwischen Konferenzteilnehmern zu erreichen, ist es erstrebenswert, die gesamte Ende-zu-Ende-Verzögerung zu minimieren. Dies ist allerdings nur in bestimmten Grenzen möglich, da sie schon von der Aufnahmefrequenz abhängig ist und außerdem eine kleine Gesamtverzögerung gleichzeitig eine kleine Warteschlangenlänge und schlechtere Kompensierung des Delay Jitter bedeutet.

Tabelle 10 in Abschnitt 5.7.3 (Performance-Untersuchung) gibt einen Überblick über die bei der Untersuchung festgestellten optimalen Parameter.
Weil das System verbindungslos arbeitet und eine Verlustrate nicht vorher einschätzbar ist, muß deshalb der Empfänger selbst die Größe der Warteschlange ermitteln, in dem er eine bestimmte maximale Ende-zu-Ende Verzögerung erlaubt. Anhand dieser Größe kann die Anzahl der

[14] Rückruf-Funktion

zwischenzuspeichernden Datenpakete errechnet werden. Die Berechnung der Warteschlangenlänge wird im nachfolgenden Abschnitt diskutiert.

5.3.6 Die Kompensierung der Übertragungsschwankungen

Kommen die Audiopakete mit großen Abstandsschwankungen oder Verlust am Ziel an, gerät die Taktrückgewinnung außer Tritt und es kommt zu Tonstörungen. Daher müssen die Audiodaten im Empfangsspeicher gehalten werden.

Die gesamte Ende-zu-Ende-Verzögerung **Tg** vom Beginn der Audioaufnahme bis zu ihrer Wiedergabe setzt sich aus folgenden Teilen zusammen:

$$Tg = Ta + Tü + Tp$$

Gleichung 3: gesamte Ende-zu-Ende Verzögerung

Wobei
- ***Tg*** *die gesamte Ende-zu-Ende Verzögerung [ms]*
- ***Tü*** *die Übertragungszeit [ms]*
- ***Ta*** *die Aufnahmedauer der Daten [ms]*
- ***Tp*** *die Wartezeit in der Warteschlange [ms]*

sind. Die Maßeinheiten sind im eckigen Klammern „[…]“ ausgeführt.

Dabei sind die anderen Verzögerungen wie die Dauer der Kopie der Datenpuffer vernachlässigbar klein (< 1 ms). Außerdem wird hier die Übertragungszeit ***Tü*** zunächst nicht berücksichtigt, weil sie stark vom Transportnetzwerk abhängig ist. Sie wird als fest vorgegeben betrachtet und ist nicht Gegenstand der Untersuchungen im Rahmen dieser Arbeit. Deshalb reduziert sich die Gleichung 3 zur Bestimmung der Warteschlange wie folgt:

$$Tg = Ta + Tp$$

Gleichung 4: gesamte Ende-zu-Ende Verzögerung

Ta, die Aufnahmezeit für ein Audiopaket, ist von der Abtastfrequenz ***AF*** abhängig und ist gleich ***Ta = 1000/AF***

Ist ***A*** (numerische Zahl) die Größe der Warteschlange, so ist

$$Tp = Ta * A$$

⇨ $$Tg = Ta + Ta*A$$

$$A = Tg / Ta - 1$$

Gleichung 5: Bestimung der Warteschlangeslänge

Um eine maximale Ende-zu-Ende Verzögerung von ca. 150 ms zu halten, wobei die Aufnahme der Audiodaten mit einer Abtastfrequenz von 25 Paketen pro Sekunde erfolgt (d.h. 40 ms pro Paket), kann beim Empfänger eine Zwischenspeicherung von ca. zwei oder drei Paketen (150 / 40 –1 = 2,8) stattfinden (siehe Abschnitt 5.7.3, Tabelle 10).

Der Wiedergabezeitpunkt wird daher entsprechend um die maximal einzuhaltende Verzögerungszeit verzögert. Dies geschieht durch einen Thread[15], der nach dem Eintreffen des ersten Pakets und der Initialisierung des Wiedergabetreibers die Wiedergabe um diese Zeit hinausschiebt. Während der Verzögerungszeit werden eintreffende Pakete in der Warteschlange zwischengespeichert. Beim Test auf der Basis von ATM-LAN-Emulation wurde sogar ohne Zwischenspeicherung bei einer Audio-Abtastfrequenz von 8-15 Audioframes pro Sekunde u.U. störungsfreie Übertragung erzielt.

5.3.7 Ausgleich von Paketverlusten

Bei Paketverlust entsteht eine Lücke im Audiostrom. Das einfache Ignorieren dieses Pakets hat zwar keine negative Wirkung auf die Synchronisation, führt

[15] ein parallel laufender Kindprozeß

aber zu einem nicht-kontinuierlichen Verlauf der Wiedergabe. Dies geschieht auch dann, wenn ein Datenpaket zu spät ankommt. Ein Ausgleich kann durch Wiederholung des zuvor empfangenen Audiopakets oder durch Hinzufügen eines leeren Audiopakets erfolgen. Letzteres hat eine akustisch bessere Wirkung auf den Zuhörer, da z.B. gesprochener Text durch Wiederholung von Fragmenten wesentlich unverständlicher als durch fehlende Textfragmente wirkt.

5.4 Das Videomodul

Das Videomodul enthält Funktionen für die Aufnahme und Wiedergabe von Videoströmen. Es wurden zwei C++-Klassen implementiert:

- eine für die Aufnahme und
- die andere für die Wiedergabe des Videostromes.

Der Sender benutzt ein Objekt der C++-Klasse ***CVideoCapture*** für die Steuerung bei der Aufnahme der komprimierten Frames (z.B die Einstellungen der Aufnahmefrequenz oder des Video- bzw. Audioformates). Auf der Seite des Empfängers wird dagegen die C++-Klasse ***CVideoRender*** für die Zwischenspeicherung und die Darstellung der Videodaten benutzt.

5.4.1 Videobild-Aufnahme und Übertragung der komprimierten Frames

Das Einfangen und das Bereitstellen der digitalisierten Videobilder werden hardwaremäßig von der Videokarte übernommen. Der Sender muß aber folgende Schritte durchführen, um den Capturetreiber anzusprechen, den Kompressor zu initialisieren und den Preview (Voransicht) der Videobilder anzuzeigen:

- Bereitstellung des Capture-Fensters
- Verbindung zum Capture-Treiber
- Setzen des Videoformates
- Vorzeigen der Videobilder auf dem Bildschirm
- Aufnahme durch den Einsatz einer Callback-Funktion[16]

[16] Rückruf-Funktion

5.4.1.1 Bereitstellung des Capture-Fensters

Zuerst muß ein Fenster generiert werden, auf dessen Fläche das Vorzeigen der Videobilder erfolgen kann.

Es geschieht durch den Aufruf der Memberfunktion ***OpenOverlayWindow().*** Die zu übergebenden Parameter bestimmen die Position, den Namen und den Vater dieses Capture-Fensters.

5.4.1.2 Verbindung zum Capture-Treiber

Nach der Erzeugung des Capture-Fensters wird die Verbindung zwischen diesem Fenster mit einem Capture-Treiber hergestellt Die Verbindung geschieht durch den Aufruf von ***ConnectOverlayWindow***(). Als Funktionsparameter muß der Index des Capture-Treibers (es können mehrere Capture-Treiber in einem System vorhanden sein) übergeben werden. Der Wert Null veranlaßt das Programm, den Default-Treiber zu initialisieren.
Der folgende kurze Auszug aus dem Programm illustriert das Vorgehen:

{ Auszug aus dem Programm-Code }

```
CVideoCapture * pCapture;
PCapture = new CvideoCapture;

//Erstellung des Capture-Fenster
if ((result=pCapture->OpenOverlayWindow(
                      Form1->Handle,
                             PanelLeft,
                      PanelTop,
                      PALWidth,
                      PALHeight,
                      NULL,//"Capture Window",
                      WID_PRWWIND))!=VIDEOERR_NOERROR){
     goto Error_VideoCapture;
   }
//Verbindung zum Treiber
```

```
if ((result=pCapture->ConnectOverlayWindow(Adapter,TRUE))
!=VIDEOERR_NOERROR){
               goto Error_VideoCapture;
               }
.. .. ..
}
```

5.4.1.3 Setzen des Videoformates

Nach der Herstellung der Verbindung zum Capturetreiber kann das Videoformat abgefragt und neu gesetzt werden. Dabei wird eine Dialogbox angezeigt. Entsprechende Werte wie der Kompressionsfaktor, die Datenrate oder die Bildgröße können eingestellt werden. Das Videoformat wird vom Videomodul gespeichert und später mit den Videodaten übertragen.

Die Einstellung und die Abfrage des Videoformates erfolgen durch den Aufruf der Memberfunktionen :

***CVideoCapture::GetCatureFrameHeader*(); //Abfrage des Videoformates**

und

CVideoCapture::SetParam(); //Setzen des Videoformates

{ Auszug aus dem Programm-Code }

```
.. .. .. ..
//setzen der Parameter
if ((result=pCapture->SetParam(VideoFPS,
                               AudioFPS,
                               AudioKanal,
                                  BitsPerSample,
                               AudioFrequenz))
                            !=VIDEOERR_NOERROR){
          goto Error_VideoCapture;
          }
```

```
.. .. ..
}
```

Wie bereit im Abschnitt 5.3.2 besprochen, ist der Video-Capture-Treiber auch für die Aufnahme der Audiodaten verantwortlich. Beim Setzen des Videoformates müssen außerdem die Abtastfrequenzen der Video- und Audiodaten eingestellt werden.

5.4.1.4 Anzeigen der Videosignale auf dem Bildschirm

Nach einer erfolgreichen Verbindung zwischen dem Capture-Treiber und dem Capture-Fenster können Videosignale direkt im Capture-Fenster durch den Aufruf der Memberfunktion ***CVideoCapture::SetupOverlay()*** angezeigt werden. Diese Anzeige erfolgt hardwaremäßig durch Video-Overlay.

5.4.1.5 Einsatz der Callback-Funktionen

Vor der Aufnahme der Video- und Audiodaten müssen bestimmte Callback-Funktionen installiert werden, deren Einsatz der Anwendung ermöglicht, Nachrichten vom Capture-Treiber über den Zustand und den Status der Aufnahme zu empfangen.

Zwei wichtige Callback-Funktionen werden durch den Aufruf der Memberfunktionen ***SetAudioCallback***() und ***SetVideoCallback***() installiert. Sie werden jedesmal dann sofort vom Capture-Treiber aufgerufen, wenn der Aufnahmepuffer vollständig mit Audio- bzw. Videodaten gefüllt wurde. Das Videomodul kann somit den Inhalt des Aufnahmepuffers und die Angaben über das Aufnahmeformat (Bildformat, Kompressionsfaktor und Aufnahmefrequenz) an das Netzwerkmodul zur Übertragung übergeben.

{ Auszug aus dem Programm-Code }

```
.. .. ..
//Install von Video-Callback
if ((result=pCapture->SetVideoCallback(
                (LPARAM)(LPVOID)CaptureVideoCallback)
```

```
     )!=VIDEOERR_NOERROR){
          goto Error_VideoCapture;
          }
//Install von Audio-Callback
if ((result=pCapture->SetAudioCallback(
             LPARAM)(LPVOID)CaptureAudioCallback)
    )!=VIDEOERR_NOERROR){
          goto Error_VideoCapture;
          }
.. .. ..
```

Die Aufnahme erfolgt durch einen Thread und ist unabhängig von anderen Aktivitäten (z.B. von der Übertragung oder der Wiedergabe).

5.4.2 Dekompression und Wiedergabe der empfangenen Frames

Die Dekompression und die Darstellung der empfangenen Frames werden hardwaremäßig von der Videokarte übernommen. Der Empfänger benutzt ein Objekt der C++-Klasse ***CVideoRender*** für die Steuerung dieser Aktivitäten, die in folgenden Schritten durchgeführt werden:

- die Initialisierung des Dekompressors
- die Kompensierung des Delay Jitters
- die Wiedergabe durch Video-Overlay

5.4.2.1 Initialisierung des Dekompressors

Videodaten werden zusammen mit den Angaben über das Format (Bildformat, Kompressionsfaktor und Aufnahmefrequenz) übertragen. Das erste empfangene Paket wird für die Initialisierung des Dekompressors benutzt. Dabei wird im gesamten System nach einem installierten Dekompressor gesucht, der das Videoformat dekodieren kann. Erst nach einer erfolgreichen Initialisierung (d.h. es wurde ein Dekompressor gefunden, der das Videoformat dekodieren und die dekomprimierten Daten direkt in die Hardware zur Darstellung ausgeben kann) werden die empfangenen Frames dem Dekompressor zur Dekodierung und Darstellung übergeben. Wärend der Initialisierung (ca. 500 ms) werden

ankommende Frames ignoriert. Nach der erfolgreichen Initialisierung werden Bildframes zwischengespeichert (falls die Warteschlange aktiviert wurde - die Bestimmung der Warteschlangenlänge wurde in Abschnitt 5.3.6 besprochen) oder direkt wiedergegeben.

{Auszug aus dem Programm-Code}

```
.. .. ..
CVideoRender * pRender;
PRender=new CVideoRender;
.. .. ..
if (pRender->Init(
                lpBmpInf,
               //Videoformat
                lpEmpfVideoFrame[0],
               //empfangspuffer
                EndToEndDelay,     //gewünschte Verzögerung.
                        ((struct SeqHeader *)wparam)-
>DgId)==0){
     //Buffermanager aufrufen- Warteschlange aktivieren
.. .. ..
}
else {
// Fehler ! kein Dekompressor gefunden
.. .. ..
}
```

5.4.2.2 Kompensierung des Delay Jitters

Die Memberfunktion ***CVideoRender***::**BufferManager**() leitet notwendige Maßnahmen für die Kompensierung des Delay Jitters ein. Diese Kompensierung funktioniert nach dem gleichen Prinzip wie die beim Audiostrom (siehe Abschnitte 5.3.5 und 5.3.6).

5.4.2.3 Darstellung der Videoframes

Die Darstellung der Videoframes erfolgt über das örtliche Video-Overlay. Der Empfänger benutzt dabei ein individuelles Fenster, auf dessen Fläche die Videobilder angezeigt werden. Eine entsprechende Skalierung der Videobildgröße mit der Fenstergröße wird automatisch vorgenommen. Dieses Fenster wird durch den Aufruf der Memberfunktion ***CVideoRender::*OpenPlaybackWindow**() geöffnet.

Einzeln empfangene Videoframes werden durch den Aufruf der Menberfunktion ***CVideoRender::Draw()*** dekodiert und dargestellt. Diese Funktion wird von der Funktion ***CVideoRender::BufferManager***() aufgerufen.

5.4.2.4 Ausgleich von Paketverlusten

Der Ausgleich von Paketverlusten des Videostromes für dieses System ist aus folgenden Gründen nicht zwingend notwendig: Im Gegensatz zum Audiostrom (siehe Abschnitt 5.3.7) ist beim Videostrom die Darstellung eines leeren Pakets sehr störend – der entsprechende Bereich des Videobildes wird schwarz dargestellt. An dieser Stelle ist eine Eigenschaft des Video-Overlays von Vorteil: Das zuvor angezeigte Videobild bleibt so lange auf dem Bildschirm, bis es durch das Anzeigen eines neuen Bildes ersetzt wird. Durch diesen Effekt ist beim Verlust eines Paketes die Wiedergabe eines anderen Paketes oder das Wiederholen des Vorgängers nicht nötig. Es muß jedoch die Stelle des verlorenen Paketes in der Warteschlange markiert werden, um die Synchronisation mit dem Audiostrom zu gewährleisten.

5.4.3 Synchronisation mit dem Audiomodul

Durch gleiche Maßnahmen für die Kompensierung der Übertragungsschwankungen bei der Video-Übertragung (Abschnitt 5.4.2.2) und bei der Audio-Übertragung (Abschnitt 5.3.6) wird die Gesamtverzögerung der Audio- und Videoströme entsprechend der Eingabewerte eingehalten. Eine kleine Zeitdifferenz zwischen beiden Strömen entsteht dadurch, daß große Videopakete

eine größere Übertragungszeit als die wesentlich kleineren Audiopakete benötigen, um zum Ziel zu gelangen. Im Ergebnis erhalten wir eine Lippen-Synchronisation beider Datenströme (siehe Tabelle 11 in Abschnitt 5.7.4 Performance-Untersuchung).

5.5 Das Netzwerkmodul

Im Netzwerkmodul befindet sich das Multicast-Interface. Das Netzwerkmodul benutzt ein Übertragungsprotokoll, welches auf UDP basiert und Funktionen, die unter anderem die eigentlichen Übertragungen, das Eintreten in eine und das Austreten aus einer Multicast-Gruppe, die Fragmentierung und Defragmentierung der Datenpakete abwickeln. Das Interface wurde als eine C++ - Klasse implementiert. Folgend ist eine Übersicht über diese Klasse gegeben. Die Funktionalität wird weiter unten in diesem Abschnitt erläutet.

```
class CUdpSocket {

private:
     int sock;                          // Socket-ID
     struct sockaddr_in my_addr; // eigene IP-Adresse
     LPSTR SARSendBuf;                  //Sendepuffer =
iMaxUdpDg
     HWND hParent;                 //Vater Fenster
     UINT msg;                     //Nachricht für Vater
     struct sockaddr_in from_addr; //IP-Adresse des Senders
     int fromlen;                       // Länge dieser
Adresse
     HANDLE tHd;                        //Handle des
Empfangsthreads
     unsigned long tId;            //Thread-ID
     struct RecvStructT RecvStruct;//User Data, an Thread
```

```
.. .. .. weitere Membervariable, siehe Anhang

public:
/* Init()- Erzeugen des Sockets (Port, Netzwerkadapter)
int Init(int port, int adapter);
void Close();

// Multicast Funktionen
/* JoinGroup()- Eintreten einer Multicastsgruppe */
int JoinGroup(struct ip_mreg *);
/* LeaveGroup()- Austreten
int LeaveGroup(struct ip_mreg *);

/*RecvData() – Empfangsfunktion
int RecvData(LPSTR buf,int* plen);
/*SendData() – Sendensfunktion
int SendData(LPSTR buf,int* plen, struct sockaddr_in *
pAddr);

// Senden und Empfangen mit Fragmentierung & Reassemblierung
/* InitSARSend() – Initialisierung maximale Paketgröße und
Nachrichtsaustausch mit Parent-Fenster*/
int InitSARSend(unsigned short ui,HWND,UINT);

/*SARSeg()- Senden mit Fragmentierung
BOOL SARSeg(LPSTR buf,DWORD dw,struct sockaddr_in * pAddr);


/* InitSARRecv() – Initialisierung des Empfangspeichers und
Nachrichtaustausch mit Parent-Fenster. Es wird ein Thread
gestartet, der im Hintergrund auf Daten wartet und Daten
empfängt */
int InitSARRecv(LPSTR buf,DWORD dw,HWND hwnd,UINT msg);
```

```
… weitere Memberfunktionen, siehe Anhang
};
```

5.5.1 Das Übertragungsprotokoll

Für die Funktionalität des Systems ist die Tatsache der verbindungslosen Kommunikation von besonderer Bedeutung. Es besteht keinerlei Vereinbarung zwischen Sender und Empfänger bezüglich der übertragenen Daten. Der Empfänger kennt zu Übertragungsbeginn nicht das Format, den Kompressionsfaktor, die Paketlänge sowie die Frequenz der vom Sender gesendeten Video- und Audiodaten, bis zu dem Zeitpunkt, zu dem er die Daten erfolgreich empfangen hat. Der Sender kann außerdem jederzeit eigenwillig Änderungen vornehmen z.B. die maximale Paketlänge, das Videoformat oder Audioformat, den Videokompressionsfaktor oder die Abtastfrequenz neu setzen.

Auf der Seite des Empfängers funktioniert der Empfang ähnlich wie bei einem Fernseher: Der Benutzer schaltet den Fernseher an und kann die vorgegebenen empfangenen Programme anschauen. Er weiß aber vorher nicht, ob das Programm in normalem oder breitem Format, der Ton in Mono oder Stereo ausgestrahlt werden. Umgekehrt kann der Fernsehsender die Anwesenheit dieses Benutzers nicht wahrnehmen. Bei einer Bild- oder Tonstörung kann der Benutzer nicht sofort erkennen, ob dies am Empfang liegt oder ob beim Sender gerade eine Störung vorliegt.

Es ist deshalb notwendig, alle nötigen Informationen über Format, Kompressionsfaktor, Frequenz und Paketlänge in den übertragenen Daten mitzuschicken. Wir bezeichnen im folgenden die Daten, die in einem UDP-Datagramm als Nutzdaten geschickt werden sollen, als Sequenz. Ein Sequenz besteht aus einem Sequenzkopf und Sequenzdaten (siehe Abbildung 19).

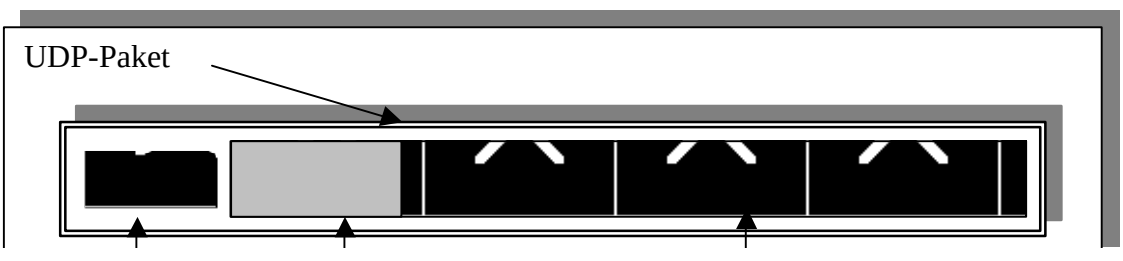


Abbildung 19: UDP-Paket und Video-Frames

Wie bereits oben besprochen wurde, muß daher ein Protokoll für die Übertragung der Informationen implementiert werden. Dafür wurde ein Header (Sequenzkopf) eingeführt, der für die jeweilige Sequenz alle nötigen Informationen enthält. Nachfolgend wird der Aufbau dieses Headers besprochen, vgl. Abbildung 20.

Der Parameter DgId bezeichnet dabei die Identifikationsnummer eines UDP-Datenpakets. Es ist eine vorzeichenlose 2-Byte-Zahl, die am Anfang der Übertragung per Zufall generiert wurde und immer um 1 erhöht wird, wenn ein UDP-Paket erfolgreich abgeschickt wird.

DgId	**SeqSum**	**SeqId**
PaketLen	**DataLen**	**SeqCRC**
time		

Abbildung 20: der Aufbau eines Sequenzkopfes.

Weil ein komprimiertes Videoframe viel größer als die gewählte maximale Größe eines UDP-Datenpaket sein kann, muß das Frame in mehrere Fragmente zerlegt werden. Die Informationen darüber werden im Sequenzkopf gehalten, um eine korrekte Reassemblierung zu ermöglichen. Der Parameter SeqSum ist eine 1 Byte große vorzeichenlose Zahl, die die Anzahl der Fragmente eines Frames

bezeichnet. Der Parameter SegId ist eine 1 Byte große vorzeichenlose Zahl, die ein einzelnes Fragment eines zerlegten Frames identifiziert.

Für die mögliche Fehlererkennung in einem übertragenen Segment wird eine einfache Prüfsumme gebildet, die über dem Parameter SeqCRC (4 Byte) gegeben wird. Der Parameter Time enthält einen Zeitstempel der lokalen Sendestation im Format Stunde:Minute:Sekunde.

Der Sequenzkopf enthält außerdem Informationen über die Länge der Sequenzdaten (Parameter DataLen, 4 Byte) und die benutzte UDP-Paketlänge (Parameter PaketLen, 4 Byte). Weil diese UDP-Paketlänge jederzeit vom Sender neu gewählt werden und von Sender zu Sender unterschiedlich groß sein kann, muß auch dieser Wert übertragen werden, um eine korrekte Reassemmblierung zu ermöglichen.

5.5.2 Fragmentierung/Reassemblierung eines Bildframes

Abbildung 21 illustriert die Fragmentierung eines Bildframes (z.B. ein Bildframe der Größe von 200 KByte, PAL-Qualität, Kompressionsfaktor 4). Das Frame wird in mehreren Fragmenten mit der Länge der maximal erlaubten UDP-Paketlänge zerlegt. Ein UDP-Paket ist normalerweise 64 KByte groß, d.h. in dem Beispiel wird das Frame in 4 Fragmente zerlegt. Anschließend wird für das jeweilige Fragment ein Sequenzkopf generiert. Der Wert des Parameters SeqSum entspricht der Anzahl der Fragmente der Zerlegung (in Beispiel ist SeqSum=4) und ist gleich 1, falls das Frame nicht fragmentiert wurde. Der Wert des Parameters SeqId durchläuft die Werte von 1 bis zum Wert des Parameters SeqSum.

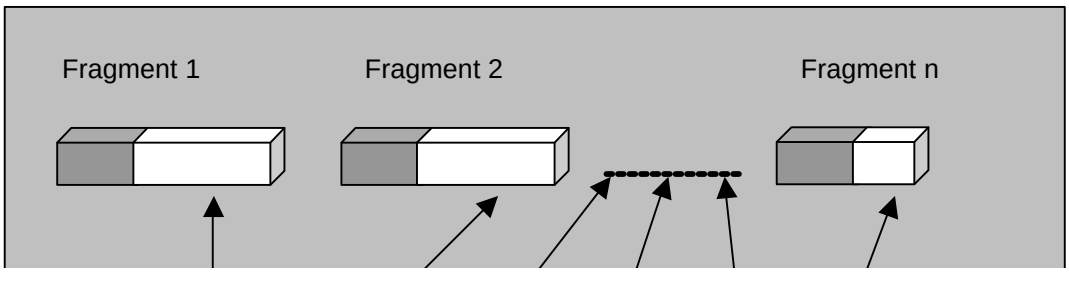


Abbildung 21: Fragmentierung eines Frames

Die Reassemblierung der Segmente muß möglichst schnell geschehen, um eine Synchronisation zwischen Sender und Empfänger zu gewährleisten. Es wurde auf zusätzliche Empfangspuffer verzichtet. Die Fragmente werden direkt in den für die komprimierten Bilddaten reservierten Speicher kopiert. Die Einordnung eines empfangenen Fragmentes in den reservierten Bildspeicher erfolgt über die Sequenznummer SeqId im Sequenzkopf. Der Empfänger führt dabei ein Protokoll über die bereits empfangenen Fragmente. Die Datagrammkennzeichnung wird jeweils aktualisiert, so daß ein nicht mehr aktuelles, zu spät eintreffendes Paket sofort erkannt wird. Ein reassembliertes Frame gilt als komplett, wenn die Anzahl der empfangenen Fragmente mit gleicher Datagrammkennzeichnung gleich der Summe der Fragmente im Parameter SeqSum ist.

Ein doppelt empfangenes Fragment wird erkannt und ignoriert. Ein zu spät eintreffendes Fragment mit einer nicht-aktuellen Datagrammkennzeichnung im Parameter DgId wird verworfen. Die Reihenfolge der eintreffenden Fragmente spielt keine Rolle bei der Reassemblierung.

Abbildung 22 illustriert eine nicht erfolgreiche Defragmentierung. Ein Frame mit einer Identifikationsnummer, z.B. 20, besteht aus vier Fragmenten. Die ersten zwei Fragmente wurden zunächst korrekt empfangen und deren Inhalte in den

Bildpuffer kopiert. Das Eintreffen des ersten Fragmentes des Frames mit der Identifikationsnummer 21 veranlaßt dann das Verwerfen des noch nicht fertig defragmentierten Bildpuffer-Inhalts und damit des Frames. Der Inhalt dieses ersten Fragmentes des neuen Frames (Nr. 21) wird in den Bildpuffer überschrieben. Das später eintreffende dritte Fragment sowie das letzte Fragment des Frames Nummer 20 werden ignoriert.

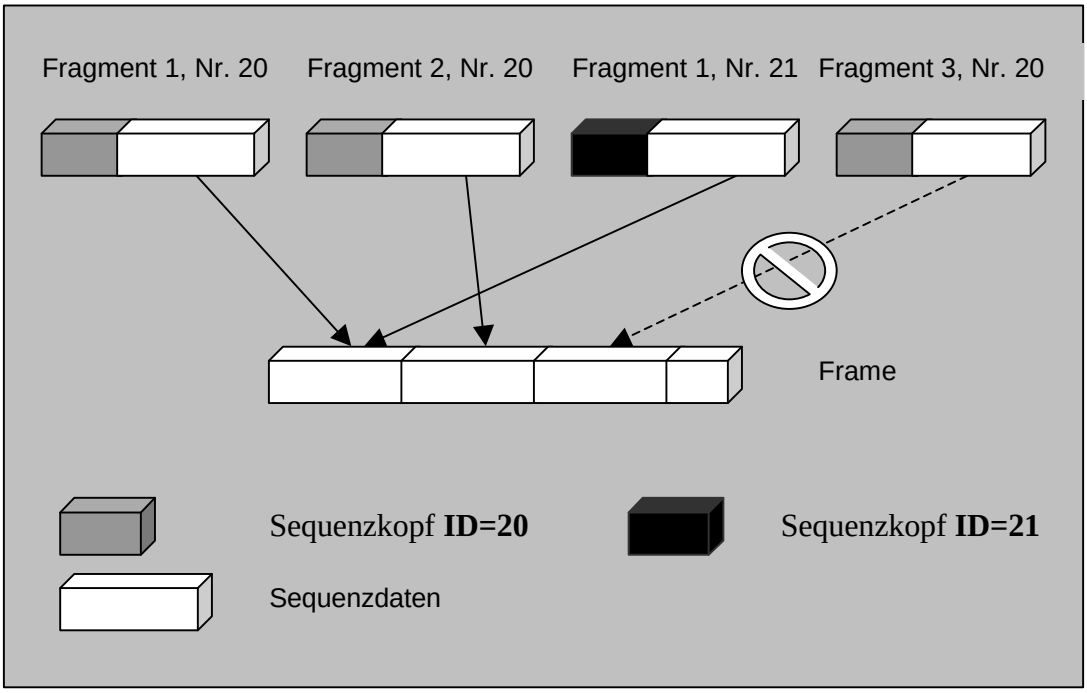


Abbildung 22: Darstellung einer nicht erfolgreichen Defragmentierung eines Frames , z.B. Nr. 20

Komplett empfangene Frames werden sofort an das Video- bzw. Audiomodul weitergeleitet. Dort werden die Daten entsprechend dem Benutzerwunsch entweder gleich wiedergegeben oder in einer Warteschlange zwischengespeichert. Der Grund, warum diese Zwischenspeicherung nicht bereits im Netzwerkmodul stattfindet, liegt an folgendem Problem: Da das System verbindungslos arbeitet, ist eine korrekte Taktwiedergabe (Anzahl der Bild- bzw. Audiopakete pro Sekunde) erst möglich, nachdem die ersten Video- bzw. Audiopakete empfangen und vom Video- bzw. Audiomodul ausgewertet wurden. Im Fall einer Zwischenspeicherung wird danach ein Zeitgeber gestartet, welcher dem System ermöglicht, periodisch in exakten Zeitabständen Audio-

bzw. Videoframes aus der Warteschlange herauszuholen und wiederzugeben (siehe Abschnitt 5.3 Audiomodul). Um die Funktionalität zwischen Netzwerk- und Multimediamodul voneinander zu trennen, darf die Auswertung der Multimediapakete nicht im Netzwerkmodul stattfinden.

Aus diesem Grund wurde im Netzwerkmodul nur ein Empfangspeicher für ein einziges Frame implementiert, sodaß ein zu spät eintreffendes Fragment das Verwerfen des ganzen Frames veranlassen muß. Sobald eine Sequenz mit einer neueren Datagrammkennzeichnung ankommt, wird das noch in der Defragmentierung befindliche Frame verworfen. Für die hier geführten Untersuchungen unter den eingerichteten Bedingungen konnte jedoch bisher kein Verwerfen aufgrund falscher Reihenfolge eines Frames verifiziert wurden.

5.5.3 Multicast

Bevor das Senden oder Empfangen stattfinden kann, muß durch den Aufruf der Funktion ***InitSocket****(port, adapter)* das Kommunikationsocket erfolgreich erzeugt werden. Die Variable *adapter* kennzeichnet dabei einen Netzwerkadapter (falls die Station mehrere besitzt), über den gesendet bzw. empfangen werden soll. Der Sender kann einen Default-Port (Null) für das Senden benutzen. Der Empfänger bestimmt dagegen durch den Port, ob er Videodaten (durch den reservierten Video-Port) bzw. Audiodaten (über den Audio-Port) empfangen will. Das Kommunikationsocket wird für das verbindungslose UDP-Protokoll (SOCK-DGRAM) und die Internet-Adreßfamilie AF_INET geöffnet.

Nach der erfolgreichen Initialisierung des Sockets kann die Anwendung in beliebige Multicast-Gruppen eintreten bzw. daraus austreten. Die implementierte Klasse ***CUdpSocket*** hat dazu jeweils eine Memberfunktion definiert:

*int JoinGroup(struct ip_mreq *)* Beitreten in eine Multicast-Gruppe

*int LeaveGroup(struct ip_mreq *)* Austreten aus einer Multicast-Gruppe

Als Parameter muß ein Zeiger auf die Struktur ip_mreq übergeben werden, wobei ip_mreq wie folgt festgelegt wurde:

struct ip_mreq {

struct in_addr in_group; die Internetadresse der Multicast-Gruppe

struct in_addr in_interface; die Internetadresse des Netzwerkadapters

}

Die mit einem Socket verknüpfte Mitgliedschaft in einer Multicast-Gruppe wird auch aufgegeben, wenn der Socket geschlossen oder der zugehörige Prozeß beendet wird. Es können auf einem Hostrechner mehrere Sockets Mitglied ein und derselben Gruppe sein. Die Übergabe der Multicast-Datagramme an einzelne Sockets basiert jedoch auf dem Port des Empfängers. Für unsere Anwendung werden zwei verschiedene Ports für Audio- und Videoempfang benutzt. Der Audioport hat den Wert 2001, der Videoport 2000. Um Multicast-Datagramme zu empfangen, die an einen bestimmten Port gerichtet sind, muß sich der Socket an diesen Port binden.

Um Datenpakete an eine Multicast-Gruppe zu schicken, benutzt der Sender die Multicast-Gruppen-Adresse als Zieladresse. Das Senden kann durch folgende Memberfunktionen durchgeführt werden:

SARSeg(SendPuffer, PufferLänge, ZielAdresseZeiger);

oder *SendData(SendPuffer, PufferLängeZeiger, ZielAdresseZeiger);*

Als Parameter müssen die Adresse und die Länge des Sendepuffers sowie die Zieladresse in Form einer Internet-Adreß-Struktur übergeben werden. Die Funktion *SARSeg*() fragmentiert automatisch das Frame, falls dessen Größe die maximale Netzwerk-Paketgröße überschreitet.

Sobald ein Socket durch den Memberfunktionsaufruf *JoinGroup()* in einer Multicast-Gruppe beigetreten ist, kann die Anwendung Multicast-Datagramme durch einen einzelnen Aufruf der Memberfunktion *RecvData() oder InitSARRecv()* empfangen. Die Memberfunktion *RecvData*() empfängt Daten und kehrt nach dem Empfang sofort zurück. Die Memberfunktion *InitSARRecv*() empfängt keine Daten, sondern startet einen Thread, der im Hintergrund auf Daten wartet und

Daten empfängt. Dieser Thread und die Memberfunktion RecvData() rufen ihrerseits die Standard API-Funktion recvfrom() für den Empfang auf.

5.5.4 Multitasking

Das Multitasking unter Windows95 und WindowsNT ermöglicht die parallele Ausführung von mehreren Teilaufgaben. Diese Fähigkeit ist für den Empfänger besonders wichtig, wenn mehrere Prozesse gleichzeitig ablaufen sollen, wie z.B. das Empfangen und Wiedergeben von Daten.

Die folgende Abbildung illustriert die Funktionsweise dieses Empfangs-Threads *RecvThread*() (siehe Abbildung 23).

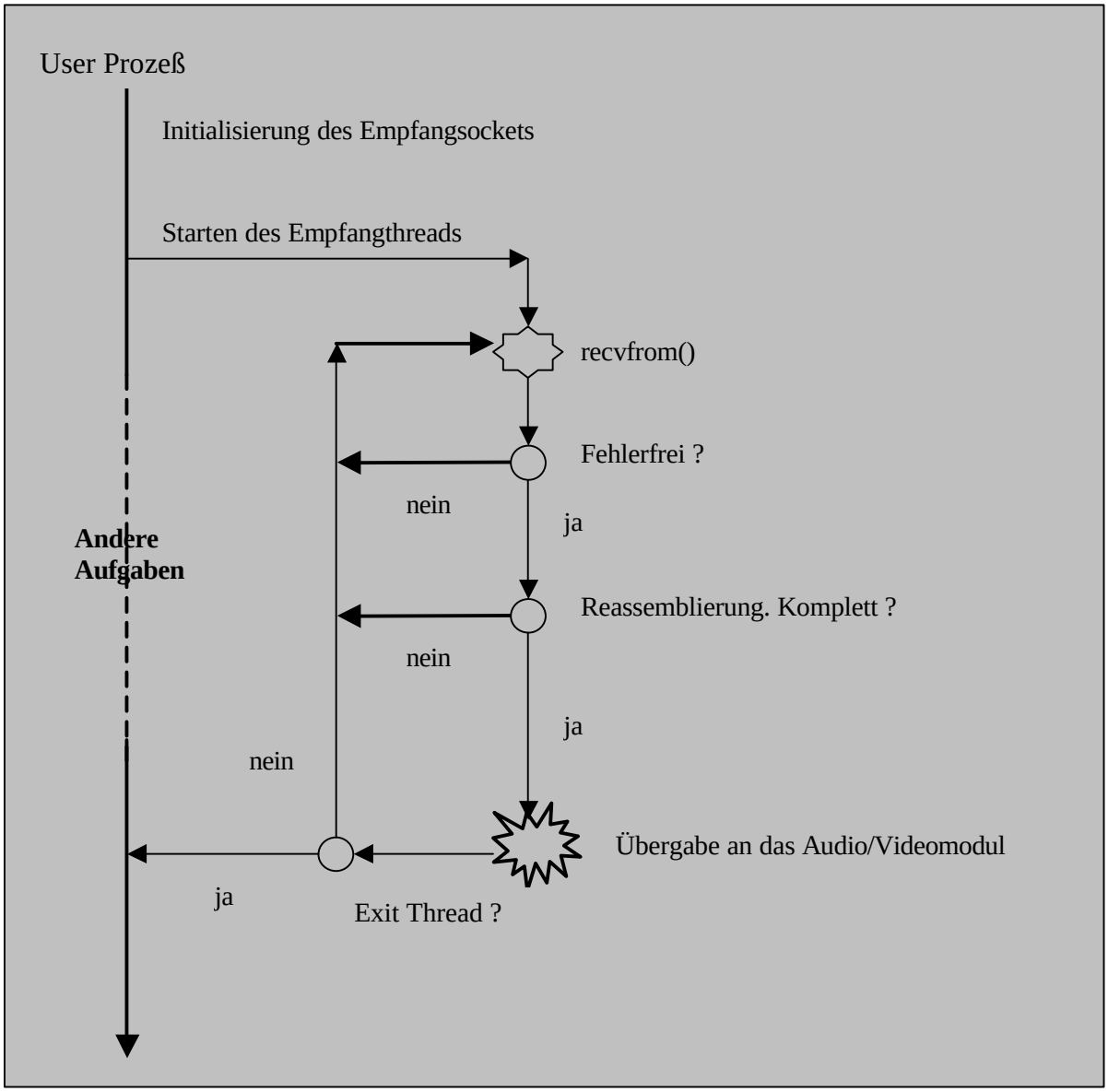


Abbildung 23: Arbeitsweise von RecvThread()

Ein recvfrom()-Aufruf ist ein blockierter Aufruf (engl. Blocking Call), d.h., die Kontrolle wird erst an die Anwendung zurückgegeben, nachdem recvfrom() geführt wurde, die Datagramme also empfangen wurden. Ein erfolgreich empfangenes Datagramm wird durch eine Prüfsumme auf die Richtigkeit geprüft. Fragmente eines Audio- oder Videoframes werden anschließend reassembliert. Die dafür nötigen Informationen werden aus dem Sequenzheader gelesen. Ein komplett empfangenes Video- bzw. Audioframe wird für das Video- bzw. Audiomodul bereitgestellt. Während der Wartezeit auf ein Datagramm blockiert recvfrom() den aufrufenden Mutterprozeß. Im Fall der Zwischenspeicherung kann recvfrom() eine nicht-kontinuierliche Wiedergabe verursachen. Hier kommt dem System die Multitaskingfähigkeit von Windows95 und Windows NT zu Gute.

Beim Empfänger wird z.B. durch den Aufruf der Memberfunktion *InitSARRecv*() ein Thread gestartet, der im Hintergrund Daten empfängt. Ein anderer Thread führt dann parallel in periodischen Zeitabständen die Wiedergabe der Video- bzw. Audiodaten aus. Die Aufnahme der Video- bzw. Audiosignale wird auf der Seite des Senders ebenfalls durch einen Thread übernommen. Parallel dazu kann die Anwendung andere Aufgaben erledigen. Es müssen allerdings Verfahren implementiert werden, die gleichzeitige Zugriffe der parallel laufenden Prozesse auf gemeinsame Speicher verhindern.

Windows bietet verschiedene Mechanismen zur Vermeidung gleichzeitiger Zugriffe auf gemeinsame Objekte. Ein davon ist der geschützte Eintritt in einen kritischen Bereich. Bevor ein Thread auf einen gemeinsamen Speicher zugreifen kann, muß er warten, bis er in den kritischen Bereich eintreten darf.

```
{ein kurzer Auszug aus dem Programmkode}

//Initialisieren
MMRESULT CWaveOut::Init(WAVEFORMATEX * pWef, int AbtFreq,
                        int MaxDelay, WORD FirstId,
                        LPSTR lpRecvData, WORD byteRead)
```

```
{
... ... ...
//Festlegen eines kritischen Bereiches
InitializeCriticalSection(&StartData.CritSect);
... ... .. .. .. ..
}
//-----------playsound thread-------------------------
void CALLBACK AudioTimerCallback(UINT wTimerID, UINT msg,
                                 DWORD dwUser, DWORD dw1,
                                 DWORD dw2)
{
//variable
.. .. ..
AudioStartDataT *pData;  //zeigt auf eine Datenstruktur

pData=(AudioStartDataT *)dwUser;

//eintreten des kritischen Bereiches
 EnterCriticalSection(&pData->CritSect);

 //prüfen, ob überhaupt noch audiopaket in queue
 if (pData->OutputPending >= 1)         {

//play sound aus erstem Puffer
.. ... .. .. ..
 //RingBuffer nachreihen
 .. .. .. .. ..
}
else { //Warte, bis Queue wieder gefüllt sind
... ... ...
}
//verlassen des kritischen Bereiches
LeaveCriticalSection(&pData->CritSect);
}
```

```
//-------------Warteschlange-Thread-----------------
MMRESULT BufferManager( WORD DgId, LPSTR lpRecvData,
                                    WORD ByteRead, HWND hP)
{
//Falls keine Warteschlange nötig
... ... ...
//sonst Pufferung

//Eintreten des kritischen Bereiches
EnterCriticalSection(&StartData.CritSect);

//Warteschlange mit empfangenen Daten füllen
... ... ...

//Verlassen des kritischen Bereiches
LeaveCriticalSection(&StartData.CritSect);
}
return MMSYSERR_NOERROR;
}
```

5.6 Das Benutzermodul

Das Benutzermodul beinhaltet Dialoge und Menüs, mit denen das ganze System durch den Benutzer per Eingabe und Auswahl gesteuert wird. Die Nachrichten werden vom Benutzermodul über das Mainmodul an die entsprechenden Module weitergereicht.

Der Benutzer kann unter anderem folgende Optionen ändern:

- Maximale Größe des verwendenden Datagramms
- Maximale Anzahl der Subnetze, über die ein Multicast-Datagramm vermittelt werden soll
- Format und Abtastfrequenz der Audiodaten
- Format, Kompressionsfaktor und Aufnahmefrequenz der Videodaten

- Auswahl von installierten Video- bzw. Audiotreibern
- Betrachtung der Empfangsprotokolle (Liste, Graphik)
- AutoReset
- Eintreten in eine und Austreten aus einer Multicast-Gruppe
- Größe der Warteschlange (durch eine gewünschte gesamte Ende-zu-Ende-Verzögerung beim Empfänger)
- Betrachten der aktuellen Übertragungsrate sowie Fehlerrate

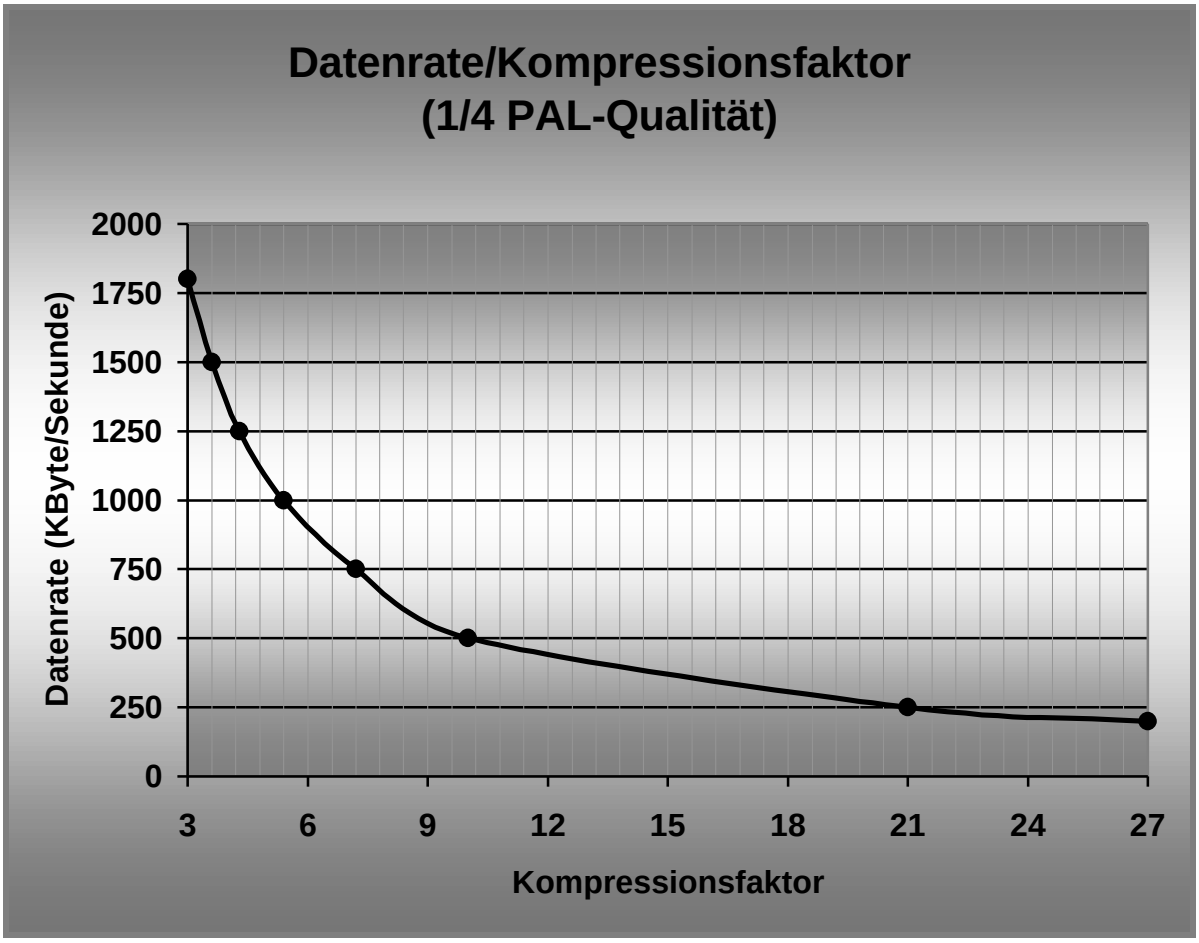
5.7 Performance-Untersuchung

Dieser Abschnitt befaßt sich mit den Ergebnissen der durchgeführten Tests für das in dieser Arbeit entwickelte Konferenz-System. Es wird auf die Eigenschaften und die Beschränkungen des implementierten Systems eingegangen.

5.7.1 Kompressionsfaktor/Datenrate

Durch eine entsprechende Einstellung des Kompressionsfaktors kann man die gewünschte Datenrate bzw. die durchschnittliche Größe eines Videobildes festlegen.

Abbildung 24: Verhältnis Kompressionsfaktor zu Datenrate mit ¼ PAL-Bildqualität

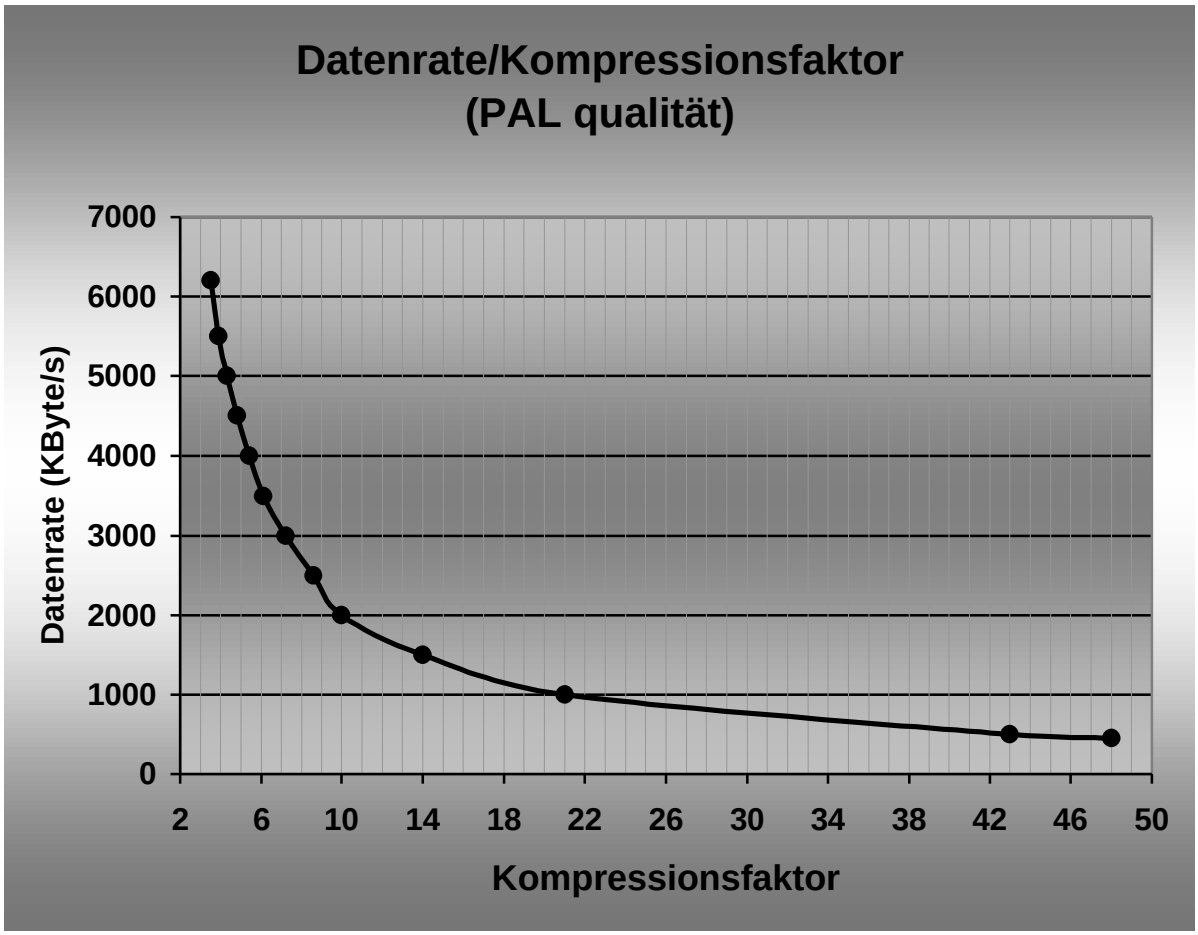


(M-JPEG Verfahren)

Abbildung 24 und Abbildung 25 geben jeweils einen Überblick über das Verhältnis zwischen Datenraten und Kompressionsfaktoren beim M-JPEG Verfahren. Die Messungen erfolgten bei einer Bildwiederholfrequenz von 25 FPS. In Abbildung 24 ist die Bildqualität ¼ PAL und in Abbildung 25 ist sie volles PAL. Die Meßgenauigkeit liegt bei 97,5 %.

Wie bereits bei der Behandlung des Kompressionsverfahrens M-JPEG im Abschnitt 4.2.1 besprochen wurde, ist durch einen hohen Kompressionsfaktor zwar eine kleine Datenrate erreichbar, allerdings ist dabei auch hoher Qualitätsverlust zu verzeichnen. Eine von vielen Seiten akzeptabler Kompromiß liegt für den Kompressionsfaktor bei dem Wert 10.

Abbildung 25: Verhältnis Kompressionsfaktor zu Datenrate bei PAL-Bildqualität (M-JPEG Verfahren)



5.7.2 Verhältnis von Paketgröße zu Übertragungszeit

Das Verhältnis von Paketgröße zu Übertragungszeit wurde bei Übertragungen per ATM-LAN Emulation gemessen (siehe Abbildung 26). Die Fehlergrenzen liegen für Frames bis 30 KByte bei ca. +/- 1 ms, für Frames von 30 bis 60 KByte bei +/-3 ms. Die Schwankungen, die hier zu sehen sind, entstehen durch die Meßungenauigkeit.
Das Verhältnis steigt kontinuierlich mit der Paketgröße. Aus dem nahezu linearen, nicht proportionalen Ausstieg ist ersichtlich, daß sich große Pakete effektiver als kleine Pakete übertragen lassen. Um beispielsweise 60 KByte zu übertragen braucht man für 1 Paket ca. 21-22 ms, überträgt man 6 Pakete zu 10 KByte, ergibt sich eine größere Übertragungszeit (ca. 30 ms). Die Fragmentierung der Videoframes erhöht zusätzlich die Verlustrate, da beim Verlust eines Fragments das ganze Frame verworfen werden muß.

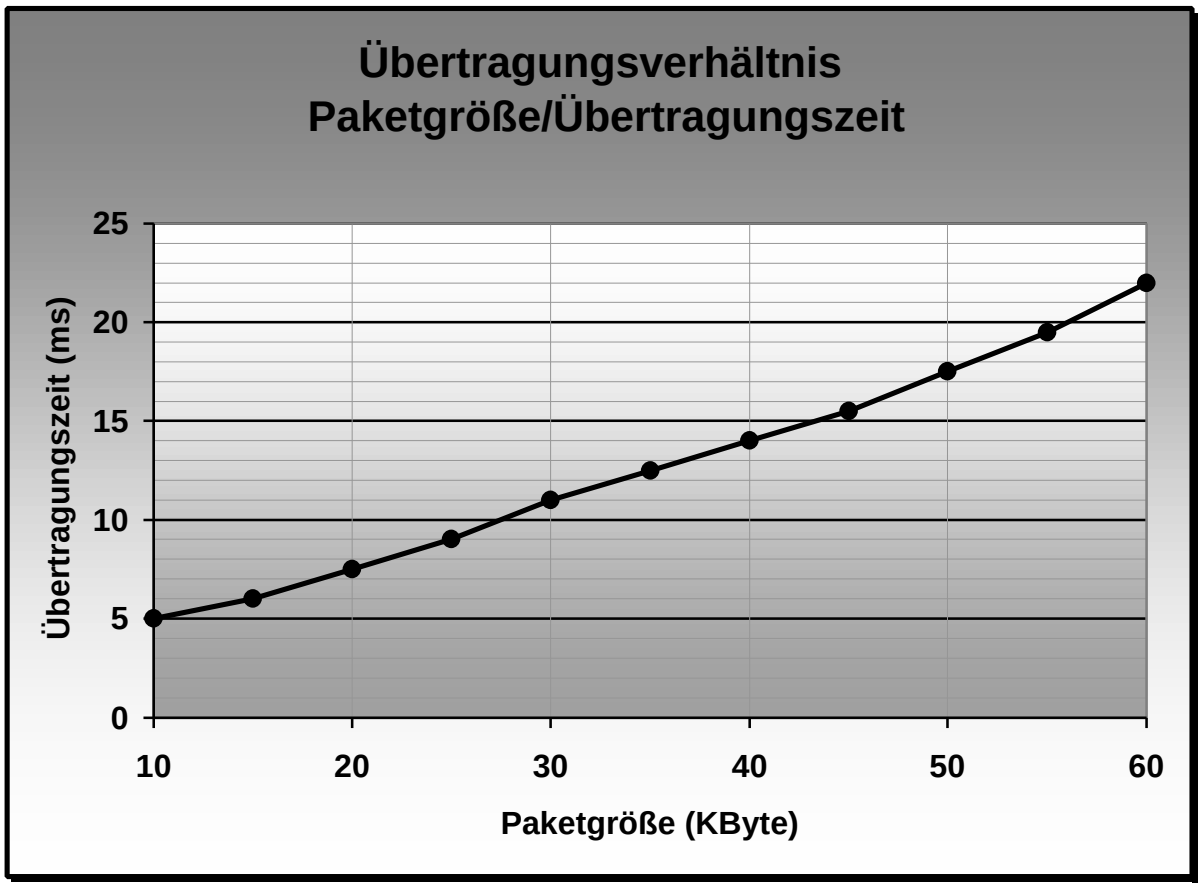


Abbildung 26: Verhältnis Paketgröße zu Übertragungszeit

Für die Anwendung muß daher immer versucht werden, die maximal mögliche Größe eines Datenpakets bei der Übertragung zu benutzen. Ein UDP-Datagramm hat eine maximale Größe von 64 KByte – ein Paket dieser Größe ist ausreichend für die Übertragung eines Frames in ¼ PAL-Qualität bei einem akzeptablen Kompressionsfaktor (z.B. Kompressionsfaktor 3,6 bei einer Datenrate von 1500 KByte/s oder ca. 60 KByte pro Videobild, siehe Abbildung 24).

Für die Übertragung des Videostromes in PAL-Qualität bei einem akzeptablen Kompressionsfaktor muß ein einzelnes Frame allerdings fragmentiert werden. Dafür sei ein Beispiel gegeben: Bei dem Kompressionsfaktor 6 ist die Datenrate 4000 KByte pro Sekunde bzw. die Größe eines Frames ca. 160 KByte. Jedes Frame muß daher in mindestens drei Fragmente zerlegt werden.

5.7.3 Audiostrom und Warteschlange

Sender AF[17]	Audiopaketlänge [ms]	Empfänger WL[18]	Verzögerung [ms][19]	Qualität *T*[20] [ms]
< 10	>100	0	> 100	***T*=0 ****
		1	>200	***T*=0 ****
		2	>300	***T*=0**
		≥3	>400	***T*=0**
20-25 *	40-50	0	40-50	***T*>0**
		1	80-100	***T*>0**
		2	>120	***T*=0**
		≥3	>160	***T*=0**
40 *	25	0	25	***T*>0**
		1	50	***T*>0**
		2	>75	***T*=0**
		≥3	>100	***T*=0**

Tabelle 10: optimale Warteschlangenlänge beim Empfänger in Abhängigkeit von der Audiopaket-Abtastfrequenz bei der Aufnahme

* nur bei Sound Blaster 16
** falls wenig Netzlast

Die Verzögerung berücksichtigt nicht die Übertragungsdauer der Pakete. Die Meßungenauigkeiten liegen bei +/- 1 ms.

Die Qualität der Audiokommunikation ist durch ***T*** bewertet. ***T*** ist die Zeitdifferenz zwischen zwei aufeinander folgenden Audiopaketen. ***T*** =0 bedeutet „flüssiger“ Audiostrom bei der Wiedergabe, ***T*** >0 bedeutet geringe Wiedergabequalität.
Es ist aus der Tabelle 10 ersichtlich, daß für große Audiopakete die Abspielfolge beim Empfänger (Paketlänge > 100 ms) „flüssiger“ erscheint als für kleine Pakete. Große Audiopakete bringen aber auch große Verzögerung mit sich, falls beim Empfänger noch Jitter-Kompensation durchgeführt wird. Sie beträgt ein Vielfaches der Audiopaketlänge. Kleine Audiopakete sind aber Delay--sensibel und bedürfen einer Zwischenspeicherung.

[17] Audiopaket-Abtastfrequenz (Pakete pro Sekunde)

[18] Warteschlangenlänge : Anzahl der zwischenzuspeichernden Audiopakete

[19] Verzögerung= Audiopaketlänge * (Warteschlangenlänge+1)

[20] durchschnittliche Unterbrechungszeit zwischen 2 Audiopaketen bei der Wiedergabe

Außerdem kann nicht jede Audiokarte Audiopakete jeder Länge in Echtzeit aufnehmen und wiedergeben. Eine geringe Übertragungsqualität ergibt sich dadurch, daß (z.B. bei Avance Logic ALS100) Pakete der Länge 25 ms nicht im 25 ms-Takt aufgenommen und wiedergegeben werden können. Das führt zur Asymmetrie und zum Aussetzen von Audiopaketen beim Empfänger, falls beide Seiten für die Übertragung kleiner Audiopakete Audiokarten verschiedener Qualität (z.B. SB 16 bzw. ALS100) benutzen.

Die SB16 Karte ist in der Lage, Audiopakete mit einer geringen Länge von 25 ms in Echtzeit aufzunehmen und wiederzugeben. Die Verzögerung ist daher gering (bei Jitter-Kompensation ca. >75 ms). Es ist deshalb empfehlenswert, Soundkarten mit derartiger Leistungsfähigkeit (mindestens gleichwertig zu SB16) einzusetzen.

Das System ist so implementiert, daß eine reine Audioübertragung ohne Videokarte möglich ist. Der Vergleich zu anderen Internet-Phone-Systemen wie VoxPhone, Speek Freely und Global Phone [INTERPHONE] zeigte, daß die Qualität der Audio-Kommunikation in dem hier vorgestellten System gleichwertig oder besser ist, da keine Komprimierung der Audiodaten durchgeführt wird. Es kann somit ohne Videohardware, nur mit Soundkarte, als Internet-Phone-System eingesetzt werden, falls eine Übertragungsrate von mindestens 64 Kbits (8000 Hz, Mono, 8 Bit pro Sample) erlaubt wird.

5.7.4 Synchronisation der Audio- und Videoströme

Die Gesamtverzögerung der Audio- und Videoströme wird durch gleiche Maßnahmen zur Kompensierung der Übertragungsschwankungen bei der Video-Übertragung (Abschnitt 5.4.2.2) wie bei der Audio-Übertragung (Abschnitt 5.3.6) entsprechend der Eingabewerte (Delay-Kompensation-Werte) eingehalten.

Tabelle 11 gibt einen Überblick über die Ergebnisse der durchgeführten Tests.

Audiopaket			**Videoframe ¼ PAL, 25 FPS**			**Qualität**		
AF[21]	**Länge [ms]**	**ÜD**[22] **[ms]**	**Länge [KByte]**	**Länge [ms]**	**ÜD [ms]**	**Kompres-sionsfaktor**	**Zeitliche Differenz [ms]**	**Synchro-nisation**
<10	>100	2	10	40	5	*21*	*57*	

[21] Abtastfrequenz: Audiopakete pro Sekunde

[22] Übertragungsdauer eines Paketes

			20		7	*10,5*	*55*	*Lippen-synchron*
			40		14	*5*	*48*	
			60		22	*3,5*	*40*	
25	40	2	10		5	*21*	*3*	
			20		7	*10,5*	*5*	
			40		14	*5*	*12*	
			60		22	*3,5*	*20*	
40	25	1	10		5	*21*	*19*	
			20		7	*10,5*	*21*	
			40		14	*5*	*28*	
			60		22	*3,5*	*36*	

Tabelle 11: Synchronisation der Audio- und Videoströme (1/4 PAL-Bildqualität, Audio 11025xStereox16Bit)

Die Fehlergrenzen der Messungen liegen für Videoframes bis 30 KByte bei ca. +/- 1 ms, für Videoframes von 30 bis 60 KByte bei ca. +/-3 ms. Die Zeitdifferenz bedeutet den maximalen zeitlichen Abstand der Ankunft gleichzeitig aufgenommener Audio- und Videopakete, z.B. bei einer Audiopaket-Abtastfrequenz von 10 Pakete pro Sekunde ist ein Audiopaket 100 ms lang. Nach 40 ms wird ein Videobild schon übertragen und beim Empfänger angezeigt. Erst nach 100 ms wird das Audiopaket übertragen. Es entsteht ein Abstand von 60 ms. Außerdem entsteht eine kleine Differenz (3 ms) wegen der unterschiedlichen Übertragungs- und Erzeugungsdauer. Das Ergebnis ist z.B. für Bildframes der Größe 10 KByte eine Differenz von ca. 60-3=57 ms vgl. Tabelle 11.

Für eine Audiopaket-Abtastfrequenz kleiner als 10 Pakete pro Sekunde ist die maximale Zeitdifferenz noch größer. Die kleinsten Zeitdifferenzen werden erzielt, falls Audio- und Videodaten mit gleicher Paket-Frequenz aufgenommen werden. Es entsteht dabei kein zeitlicher Abstand bei der Aufnahme, sondern nur bei der Übertragung.

Aus Tabelle 10 und Tabelle 11 ist ersichtlich, daß die optimale Parameterkombination bei „40-60-2“ (das bedeutet 40 Audiopakete pro Sekunde (Tabelle 10,Tabelle 11), 60 KByte Bildframe (Tabelle 11), Warteschlangenlänge 2 Pakete (Tabelle 10)) liegt. In diesem Fall müssen aber sehr viele kleine Audiopakete übertragen werden, dabei kommt es bei Netzlast häufig vor, daß Audio- bzw. Videoframes verworfen werden. Außerdem wird dabei das Gesamtsystem stark belastet, dafür jedes Paket ein Protokollkopf generiert werden muß und das System somit uneffektiv arbeitet. Für das System sind deshalb die Parameterkombination „25-60-2“ oder „8-60-0“ optimal.

5.7.5 Kompatibilität der eingesetzten Videokarten

Für die Untersuchungen wurden Videokarten verschiedener Hersteller herangezogen. Dabei wurden außer der Kompatibilität und der Funktionalität auch die Kosten der Videokarten berücksichtigt. Derzeit sind kostengünstige Videokarten für weniger als 1500 DM zu haben, können aber nur halbduplex arbeiten, d.h., sie können nur einen einzigen Datenstrom entweder komprimieren oder dekodieren. Videokarten, die mehrere Ströme gleichzeitig bearbeiten können, sind wesentlich teurer und kosten mehr als 10000 DM. Aus diesen Gründen wurde ausschließlich Hardware zur Untersuchung herangezogen, die nur einen Datenstrom verarbeiten kann. Da jedoch nicht alle Details des M-JPEG-Verfahrens von den verschiedenen Herstellern gleichermaßen behandelt wurden, sind die Produkte trotz Standardisierung nicht a priori kompatibel. Es ist dem jeweiligen Anwender der Produkte bzw. dem Software-Hersteller überlassen, die nicht dokumentierten Unterschiede zu finden. Weiterhin sind die Hardware-Baugruppen der einzelnen Hersteller nicht unbedingt kompatibel miteinander, da u.U. - von Anwender unveränderbare - Adreß-Einstellungen, Speicherbereiche und Interrupts von diesen Karten genutzt werden.

Wegen des oben genannten Halbduplex-Problems müssen in jedem Rechner zwei Videokarten installiert werden – die eine für die Aufnahme und Kodierung, die andere für die Dekodierung und Wiedergabe der Videodaten. Dabei treten Probleme bezüglich Kompatibilität und Funktionalität auf. Es ist deshalb nicht unmittelbar möglich, zwei Videokarten des gleichen Herstellers (z.B. zwei DC30 von MIRO oder zwei FPS60 von FAST) in einem Rechner gleichzeitig zu benutzen – da sie immer auf die gleichen Systemressourcen zurückgreifen. Das Unterbringen von zwei Videokarten verschiedener Hersteller in einem System konnte nur durch einen Trick gelingen, indem der Name des vorher installierten Videotreibers durch den Namen des danach installierten Videotreibers im Eintrag für den registrierten Wiedergabetreiber manuell ersetzt worden ist.

	DC30	**AVMaster**	**FPS60**
Bildqualität	¼ PAL	X[23]	X
	2 Halbbilder (vertikal), halbe Auflösung	X	X
	1 Halbbild, volle Auflösung	X	O[24]
	Volles PAL	X	O
Kompressionsfaktor	3 - 50	1,9-54	3,5 - 50

Tabelle 12: Kompatibilität der eingesetzten Videokarten DC30 von MIRO zu AVMaster bzw. FPS60 von FAST

[23] kompatibel

[24] inkompatibel

Im Testsystem wurde die Videokarte DC30 von MIRO für die Aufnahme beim Sender benutzt. Beim Empfänger wurden entweder die Videokarte FPS60 oder die Videokarte AVMaster von FAST für die Wiedergabe eingesetzt. Die Kompatibilität dieser Karten zur DC30 bezüglich Kodierung und Dekodierung wurde untersucht und in Tabelle 12 dargestellt. Die Kommunikationsmöglichkeiten werden in Abschnitt 5.7.7 besprochen.

5.7.6 Beschränkungen bezüglich der Betriebsysteme

Die Implementierung wurde für Windows-32 Bit–Betriebsysteme entwickelt. Allerdings konnte für WindowsNT 4.0 nur das Netzwerkmodul getestet werden, weil zum Zeitpunkt der Untersuchungen für die Videokarten keine entsprechenden Treiber für die DC30 von Miro unter WindowsNT zur Verfügung gestellt werden konnten. Die Nachfrage beim Hersteller ergab, daß bisher absichtlich keine Treiber für WindowsNT entwickelt wurden. Der Grund liegt darin, daß die untersuchten handelsüblichen Videokarten ursprünglich nicht für den Einsatz in Videokonferenz-Systemen, sondern nur als Videoschnittkarten zum Privatgebrauch vorgesehen wurden; das Betriebsystem WindowsNT dagegen war bisher nicht für den Home-Bereich vorgesehen.

Beim Test des Netzwerkmoduls unter Windows95 wurde ein Fehler in der als Standard mitgelieferten Socketbibliothek WinSock 1.0 entdeckt. Theoretisch muß ein UDP-Datagramm mit einer Größe bis 64 KByte gesendet und empfangen werden können. Mit WinSock 1.0 kann eine Station solche Datagramme zwar fehlerfrei senden, aber nicht empfangen. Bei der Untersuchung wurde festgestellt, daß sich Datagramme mit einer Größe von mehr als 12 KByte nicht empfangen lassen. Diesbezügliche Fehlermeldungen wurden nicht erzeugt. Die Größe von 12 KByte ist dabei unabhängig vom benutzten Netzwerkadapter, von der Bandbreite sowie von der Architektur des unterliegenden Netzwerks. Die Anwendungschicht bekommt keine Daten aus der Socketschnittstelle übermittelt. Die Ursache liegt darin, daß entweder der interne Empfangspuffer von WinSock 1.0 zu klein gehalten wurde oder die Defragmentierung in der IP-Schicht von WinSock 1.0 nicht richtig funktioniert.
Der von Microsoft neu entwickelte WinSock 2.0 ist frei von diesem Fehler und steht im Internet als Update zur Verfügung. Die damit durchgeführten Untersuchungen verliefen erfolgreich.

5.7.7 Kommunikationsmöglichkeiten

Die verschiedenen Optionen zur Übertragung sind in Tabelle 13 zusammengefaßt.

Feature		Kommunikation
Übertragung	Vollduplex	Nur zwei Stationen miteinander
	Halbduplex	Ein Sender, mehrere Empfänger

Bildqualität	¼ PAL	Möglich
	Zwei Halbbilder (vertikal), halbe Auflösung	Möglich
	Ein Halbbild, volle Auflösung	Möglich *
	Volles PAL	Möglich *
Kompressionsfaktor für Video		3,6 – 50
Audioqualität		verschiedene Qualitäten, unkomprimiert
Betriebsystem		Windows95

Tabelle 13: Verschiedene Optionen der Kommunikation

* bezüglich Kompatibilität der eingesetzten Videokarten (*siehe Abschnitt 5.7.3*)

5.7.8 Prozessorauslastung

Da die Kodierung bzw. Dekodierung und Wiedergabe der Videodaten hardwaremäßig erfolgt sind, ist der Prozessor von diesen aufwendigen Aufgaben verschont. Seine Hauptaufgabe ist das Kopieren der internen Puffer (z.B. vom Aufnahmepuffer zum Sendepuffer und umgekehrt vom Empfangspuffer zum Wiedergabepuffer). Durch den Einsatz mehrerer Threads ist die gleichzeitige Ausführung mehrerer Kopiervorgänge (z.B. bei Empfang und bei Wiedergabe) möglich. Das System ist durch diese Kopieraktivität kaum belastet. Die Ausführung von mehreren Zusatzaufgaben beim Videokonferencing wie z.B. der gemeinsamen Arbeit an Dokumenten, dem Austausch von Dokumentendaten zwischen Konferenzteilnehmern oder dem sog. Whiteboard können daher ohne eine große Beeinträchtigung der Konferenz durchgeführt werden.

5.7.9 Datenschutz

Das Konzept der IP-Multicast-Übertragung bietet keinen Datenschutz. Jede Station kann in eine Multicast-Gruppe eintreten - Voraussetzung dafür ist nur die Kenntnis der Adresse der Multicast-Gruppe, und Mitglied der Gruppe zu sein. Ein Mitglied kann alle Daten, die an diese Gruppe gesendet werden, problemlos empfangen. Die Auswertung der empfangenen Datenpakete ist unproblematisch, da alle Informationen über Nutzdaten (Format, Frequenz, Kompressionsfaktor, etc.) mit übertragen werden (im Sequenz-Header, siehe Abschnitt 5.5.1).

Das bedeutet, jeder Unbekannte, der im Besitz der Multicast-Gruppenadresse ist und der die nötigen Voraussetzungen hat (einen an ein Subnetz mit Multicast-Router angeschlossenen Rechner), kann die Videokonferenz mitsehen und mithören.

Eine Variante für den Datenschutz ist die Übertragung von kodierten Sequenz-Headern. Nach dem Empfang des ersten Datenpakets kennt der Empfänger die Adresse des Senders und kann bei ihm nach dem Schlüsselwort fragen. Im Besitz des Schlüssels kann der Empfänger den Sequenz-Header dekodieren und Frames korrekt wiedergeben. Der Sender muß daher einen Service für solche Aktivitäten bieten. Die Kommunikation muß gesichert z.B. auf Basis von TCP erfolgen.

Eine andere Variante ist das Vermeiden der Übertragung von Sequenz-Headern. Der Empfänger muß dann die Information über den Sequenz-Header beim Sender abfragen. Die hierfür nötigen Untersuchungen gehen aber über den Rahmen der vorliegenden Arbeit hinaus. Die angeführten Möglichkeiten wurden als Anregung zur Weiterführung des Themas gegeben.

5.7.10 Management-Ebene und Zusatzfunktionen

Zu einer Videokonferenz gehören außer Sehen und Hören noch mehrere zusätzliche Aktivitäten wie die gemeinsame Arbeit an Dokumenten, der Austausch von Daten und die Kontrolle über die an der Konferenz teilnehmenden Mitglieder (z.B. das Einladen eines bestimmten Teilnehmers, die Kontrolle über

das Eintreten sowie das Austreten eines Teilnehmers aus der Konferenzgruppe, das Vergeben von Passwörtern, etc.).

Es muß daher für ein Konferenz-System eine Management-Ebene implementiert werden, die all diesen Anforderungen nachkommen kann. Dies geht jedoch ebenfalls über den Rahmen dieser Arbeit hinaus.

6 Zusammenfassung

Mit dem vorgeschlagenen System können hochauflösende Videobilder (bis PAL Qualität) mit einer Bildwiederholfrequenz von 25 FPS in Echtzeit an mehrere Empfangstationen übertragen werden, wobei gewisse Beschränkungen in der Darstellung der Videobilder beim Empfänger nicht durch das entwickelte System und die Implementierung, sondern ausschließlich durch die im Rahmen der vorliegenden Arbeit noch nicht vollständig aufgehobene Inkompatibilität der eingesetzten Videokarten bestimmt wurden (zwischen FPS60 von FAST und DC30 von MIRO). Diese Beschränkungen können durch den Einsatz anderer miteinander kompatibler Videokarten (z.B. AVMaster) oder weitergehende Untersuchungen überwunden werden. Die gesamte Ende-zu-Ende Verzögerung im LAN bleibt weit unter dem Limit von 150 ms und genügt der Echtzeitanforderung. Die Zeitdifferenz zwischen dem Audio- und dem zugehörigen Videostrom liegt bei ¼ PAL-Qualität bei +/- 25 ms und bedeutet eine Lippensynchronität der Datenströme. Das System erlaubt eine vollduplexe Übertragung zwischen zwei Stationen, dabei können mehrere weitere Stationen mitsehen bzw. mithören.

Der Engpaß dieses Videokonferenzsystems ist nicht mehr die Bandbreite des physikalischen Netzes, sondern vielmehr die Betriebssystem- und Kommunikationssoftware sowie das Zusammenspiel der Hardwarekomponenten. Bei einer vollduplex Übertragung von Videodaten in PAL-Qualität ist die Datenrate von ca. 80 MBit/s nur die Hälfte der durch den ATM-Adapter zur Verfügung gestellten Bandbreite von 155 Mbit/s. Daraus wird deutlich, daß multimediale Anwendungen höhere Performance der Endsysteme als bisher verlangen, da die darunterliegenden Transportnetzwerke genügend Bandbreite garantieren können. Solche Aufgaben wie die Kodierung bzw. Dekodierung und die Darstellung der Daten müssen im Endsystem hardwaremäßig und mit genügend garantierter Kompatibilität erfolgen, weil der Einsatz von handelsüblichen Videokarten eine starke Herstellerabhängigkeit mitsichbringt.

Die Auswahl von UDP als Basis für das Transportprotokoll hat sich in mehrerer Hinsicht bewährt: Zum einen bietet das in der Anwendung implementierte Netzwerk-Modul einen transparenten Zugriff auf das Transportnetzwerk (ATM, unter besonderen Bedingungen auch Ethernet oder Tokenring), zum anderen ist die Multicast-Fähigkeit sehr einfach auf Basis von UDP zu implementieren. Das UDP/IP Konzept bietet auch eine einfache Anbindung zum Internet, sodaß bei einer kleinen Datenrate das Internet als Verbindungsmedium benutzt werden kann. Probleme können entstehen, wenn die Daten über herkömmliche Router und Getways, oder stark belastete, die Zeitsequenz noch nicht

garantierende ATM-Switches übertragen werden müssen. Dann kann die Echtzeitanforderung nicht mehr befriedigt werden.

Das System bietet damit Möglichkeiten, hochauflösende Videosequenzen in Echtzeit mit kostengünstigen Ausrüstungen zu übertragen. Die aufwendige Entwicklung spezieller Konferenz-Hardware kann somit durch eine softwaremäßige Verbindung vorhandener Hardware günstiger ersetzt werden. Auf Grund der hohen Bild-Qualität ist dessen Einsatz in der Praxis wie z.B. in der Medizin denkbar, Die Multicast-Fähigkeit gestattet mehreren Personen an verschiedenen Orten eine Operation live verfolgen und beraten zu können. Die eigentlich nachteilige Eigenschaft von UDP/IP, keine Paketwiederholung bei fehlerhafter Sendung zu gestatten, ist im Rahmen dieses Konzeptes zu einem entscheidenden Vorteil verwendet worden. Dennoch ist eine reine ATM-Schnittstelle denkbar, die eine Basis für zukünftige innovative Entwicklungen im Bereich von multimedialen Echtzeitsystemen darstellen kann.

Abkürzungen

AAL	ATM-Adaption-Layer
API	Application Programming Interface
ARP	Adress Resolution Protocol
ATM	Asynchronous Transfer Mode
B-ISDN	Broadband-Integrated Service Digital Network
CCITT	Comité Consultatif International Télégraphique et Téléphonique (heute ITU-T)
CLP	Cell Loss Priority
FPS	Frame Per Second
GAN	Global Area Network
GFC	Generic Flow Control
HEC	Header Error Control
HTDV	High Definition Television
IP	Internet Protocol
JPEG	Joint Photographic Expert Group
LAN	Local Area Network
MAN	Metropolitan Area Network
M-JPEG	Motion-Joint Photographic Expert Group
MPEG	Motion Picture Expert Group
OOP	Objektorientierte Programmierung
PAL	Phase Alternation Line
PCM	Pulse Code Modulation
PDH	Plesiochrone Digitale Hierarchie
PDU	Protocol Data Unit
PT	Payload Type
PVC	Permanent Virtual Connection
QoS	Quality of Service

RFC	Request For Comments
VBR	Variable Bit Rate
VC	Virtual Channel
VCC	Virtual Channel Connection
VCI	Virtual Channel Identifier
VP	Virtual Path

Literatur

[ATM 94]
Onvural, ATM Network: Performance Issues, Artec House ,1994

[ATM96]
Othmar Kyas. ATM-Netzwerke, Aufbau-Funktion-Performance. DATACOM, ISBN 3-89238-144-5, 1996.

[ATM97]
Mathias Hein, Nikolaus von der Lancken. ATM, Konzepte-Trends- Migration. THOMPSON Publishing, ISBN 3-8266-0214-5, 1997.

[INTERPHONE]
Speek Freely,
VoxPhone
GlobalPhone

[IPMUL97]
Chuck Semeria, Tom Maufer. Introduction to IP Multicast Routing. Networking Solutions Center, 3Com Corporation, 1997.

[MULT95]
Dieter Stotz, Computergestützte Audio-und Videotechnik. SPRINGER, ISBN 3-540-59144-3, 1995.

[MÜL94]
Michael Müller. Diplomarbeit, Entwurf, Bewertung und Implementierung von Protokollelementen für Audio- und Videodaten, 1994.

[NETZ97]
Alok K. Sinha. Netzwerkprogrammierung unter Windows NT 4.0. ADDISON-WESLEY, ISBN 3-8273-1103-9, 1997.

[NETZ92]
Andrew S. Tanenbaum. Computer-Netzwerke. Wolfram’s Fachverlag, 1992.

[RAE95]
Martin Raepple. Diplomarbeit, Realisierung eines Client/Server Systems zur Übertragung komprimierter Videoströme unter Echtzeitbedingungen über ATM unter Verwendung des SOM, 1995.

[RFC 1577]
Classical IP and ARP over ATM, IETF Network Working Group RFC 1577, Hewlett Packard Laboratories, January 1994.

[STEIN94]
R. Steinmetz. Multimedia Technologie, Einführung und Grundlagen. SPRINGER International, 1994.

[TCP94]
Kevin Washburn, Jim Evans. TCP/IP, Aufbau und Betrieb eines TCP/IP-Netzes. ADDISON-WESLEY, ISBN 3-89319-658-7, 1994.

[VIDEO96]
Richard Schaphorst. Videoconferencing and Videotelephony, Technologie and Standards. Artech House Publishes, 1996

Anhang

Header des Audiodatenformats WAV

Offset Hex	Länge Byte	Festwert Hex	Bedeutung	Bemerkung
00-03	4		RIFF	WAV-Kennung
04-07	4		Dateilänge	
08-0E	7		WAVEfmt	WAV-Kennung
0F	1	20	Leerzeichen	
10-13	4		Länge des nächsten Blocks	
14	1	01	Formattyp	LSB
15	1	00	Formattyp	MSB
16	1		Anzahl der Kanäle	LSB
17	1		Anzahl der Kanäle	MSB
18-1B	4		Einfache Sampleclock	
1C-1F	4		Gesamt-Sampleclock[25]	
20	1		Gesamtdatenbyte/Sample	LSB
21	1		Gesamtdatenbyte/Sample	MSB
22	1		Auflösung/Sample	LSB
23	1		Auflösung/Sample	MSB
24-27	4	‚data'	Kennung für Datenanfang	
28-2B	4		Gesamtanz. Datenbytes	
2C			Erstes Datenbyte	LSB bei 16-Bit-Auflösung

[25] Einfache Sampleclock*Auflösung/Sample*Anzahl Kanäle