TCP/IP-basierte Dienste zur Speicherung von Multimedia-Daten

Diplomarbeit
Frank Gadegast
Matrikelnummer: 92528

Fachbereich Informatik
Technische Universität Berlin
Fachgebiet Offene Kommunikationssysteme (OKS)

Betreuer: Prof. Dr. Radu Popescu-Zeletin
Assistierender Betreuer: Dipl. Ing. Tom Pfeifer

13. Dezember 1995

Der Autor respektiert diese und alle im folgenden Text erwähnten, eingetragenen Warenzeichen und Copyrights.

Der Text und die Diagramme dieser Arbeit wurden mit dem Textverarbeitungsprogramm Winword 2.0 der Firma Microsoft erstellt.

Der Autor erklärt hiermit ausdrücklich, daß er diese Arbeit selbständig und eigenhändig und unter ausschließlicher Verwendung der genannten Hilfsmittel angefertigt hat.

Datum **Unterschrift**

Über den Autor

Der Autor, Frank Gadegast, wurde am 1. August 1966 in Frankfurt a.M. geboren. Mit 8 Jahren zog er nach Berlin und bekam mit 15 den ersten Kontakt mit Computern. Seit 1988 studierte er Informatik an der TU-Berlin, Schwerpunkte waren Betriebssysteme und Netzwerke, Kommunikation, Multimediaformate und Kompressionstechniken.

Seit 1989 arbeitete er selbständig für verschiedene Firmen, produzierte Software und CD-ROMs im Kompressionsbereich und war als Autor und Berater tätig. Zusätzliche Anstellungen bei Firmen wie der GMD Fokus in Berlin finanzierten das Studium und verschafften Praxisnähe und Teamgeist.

Diese vorliegende Arbeit schließt seine Studien an der TU-Berlin ab.

E-mail: phade@cs.tu-berlin.de
Internet: http://www.cs.tu-berlin.de/~phade

Danksagung

Der Autor möchte sich für die vielfältige Unterstützung aus Lehre, Forschung, Beruf, Familie und dem privaten Bereich bedanken. Ohne diese wäre nicht nur diese Arbeit undenkbar gewesen, sondern der persönliche Werdegang völlig anders verlaufen. Unterstützt durch seinen Betreuer, Tom Pfeifer, konnte die Beschäftigung mit dem Multimediabereich konstant fortgeführt werden. Der GMD Fokus dankt der Autor für das ihm entgegengebrachte Vertrauen.

Weiterhin dankt der Autor für die Geduld und Einsicht seiner Freundin und Lebensgefährtin, Alice, die unzählige Abende der Konkurrenz und Belästigung der digitalen, audiovisuellen Neuerungen im Computerbereich ertragen mußte.

Inhaltsverzeichnis

Kapitel 1 - Einleitung 1
1.1 Zielsetzung 1
1.2 Abgrenzung 2
1.3 Internet 2
1.4 Personal Communication Support System 3
1.4.1 Hardware- und Softwareplattform 4
1.5 Einsatzgebiete 5
1.5 Aufbau 7

Kapitel 2 - Grundlagen 8
2.1 TCP/IP-basierte Protokolle 8
2.1.1 OSI-Referenzmodell 8
2.1.2 Internet Protokoll 11
2.1.3 Transmission Control Protocol 11
2.1.4 File Transfer Protocol 12
2.1.5 HyperText Transmission Protocol 12
Uniform Resource Locator 13
MIME-Typen 13
2.1.6 Remote File Copy 14
2.2 Global Store 14
2.2.1 Referenced Data Transfer Protocol 15
2.3 Kodierungs- und Kompressionstechniken 15
2.3.1 Dokumentformate 16
ASCII 16
HTML 16
PostScript 17
2.3.2 Bildformate 17
BMP 19
GIF 19
JPEG 20
PBM 21
TIFF 22
XBM und XPM 22
2.3.3 Video 23
MPEG-1 24
MPEG-2 25
H.261 30
SMP/G.711 30
2.3.4 Audio 33
MPEG-1 33
MPEG-2 36
G.711 μlaw und a-law 36

G.721 ADPCM 37
ETSI GSM 06.10 39
WAV 40

Kapitel 3 - Konzeption 41
3.1 Definition des Message-Stores 41
3.2 Probleme bei der Global-Store Benutzung 42
3.3 Realzeitzugriff 43
3.4 Multimedia-Messages 44
3.5 Modifikationen in den PCSS-Profiles 45
3.5.1 Userprofiles 45
3.5.2 General Media Content 45
3.6 Selektion der Formate 47
3.6.1 Selektion nach Leistungsfähigkeit 48
3.6.2 Selektion nach Integrationsfähigkeit 49
3.6.3 Verwendete Formate 50
3.7 Selektion der Protokolle 50
3.7.1 Verwendete Protokolle 51
3.8 Modularisierung und Generik 52
3.9 Integration 54

Kapitel 4 - Implementierung 56
4.1 Message-Store 56
4.2 Modulstruktur 57
4.3 Libraries 57
4.3.1 LIBMMPCSS 57
4.3.2 LIBMMACCESS 58
4.3.3 LIBMMAUDIO 59
4.3.4 LIBMMFILE 60
4.4 Player/recorder 60
4.4.1 MMTEXT 62
4.4.2 MMPICS 63
4.4.3 MMAUDIO 64
4.4.4 MMVIDEO 65
4.4.5 MMSAVE 66
4.5 Test der Mechanismen, Protokolle und Dienste 67
4.5.1 Performanz 68
4.5.2 Wiederverwendbarkeit der Komponenten und Applikationen 70

Kapitel 5 - Ausblick 71
5.1 Erweiterungsmöglichkeiten 71
5.2 Verwendung in anderem Umfeld 72
5.3 PowerWeb Technology 72
5.4 Zusammenfassung 74

Index **77**

Abkürzungverzeichnis **81**

Literaturhinweise **83**

Anhang **87**

A. Installationshinweise 89

A.1 Player/Recorder 89

A.2 Message-Store 89

B. Softwarebeschreibung 91

B.1 MMTOOLS 91

B.2 LIBRARIES 92

Abbildungsverzeichnis

Abb. 1: PCSS-Komponenten und Realisierungsumgebung 4
Abb. 2: Incoming Message Manager..... 5
Abb. 3: Die sieben Schichten des OSI-Referenzmodells 9
Abb. 4: Client-Server-Beziehung bei FTP 12
Abb. 5: BERKOM MM-Mail als Beispiel für eine Global Store Anwendung 15
Abb. 6: 4:1:1 YUV-Farbraum..... 18
Abb. 7: Beispiel einer Grafik im XBM-Format..... 23
Abb. 8: Beispiel einer Grafik im XPM-Format 23
Abb. 9: Abhängigkeiten der frame-Typen eines MPEG-1 Videostroms 24
Abb. 10: Zeitliche Scalability 27
Abb. 11: Qualitäts-Scalability 28
Abb. 12: Pan-Scan-Scalability..... 28
Abb. 13: Generelles Blockschaltbild eines MPEG-1 Audiokodierers 35
Abb. 14: PCSS Modularisierung..... 52
Abb. 15: Integration des MM-PCSS 54
Abb. 16: MM-libraries..... 55
Abb. 17: Interface des Library-Testprogrammes 59
Abb. 18: Standard Datei- und Hilfemenü 62
Abb. 19: MMTEXT-Interface 63
Abb. 20: MMPICS-Interface 63
Abb. 21: MMAUDIO-Interface 1 64
Abb. 22: MMAUDIO-Interface 2 64
Abb. 23: MMVIDEO-Interface 65
Abb. 24: MMSAVE-Interface 66
Abb. 25: Ablauf der doppelten Referenzierung 73

Tabellenverzeichnis

Tab. 1: MPEG-2 Profiles 26
Tab. 2: MPEG-2 Levels 27
Tab. 3: Sampling Größen und Bitraten 27
Tab. 4: Datenrate der zeitabhängigen Formate 50
Tab. 5: Ping-Zeiten 68
Tab. 6: TCP/IP-Durchsatz 68
Tab. 7: Durchsatz bei FTP und HTTP 69
Tab. 8: Anzahl der möglichen Übertragungen 69

Diese Seite wurde freigelassen.

Kapitel 1 - Einleitung

Elektronische Nachrichten- und Informationssysteme haben in den vergangenen Jahren immer mehr an Bedeutung gewonnen und sind zu wichtigen Hilfsmitteln an modernen Arbeitsplätzen geworden. Im Gegensatz zu den traditionellen Nachrichtensystemen wie Telefon und Briefpost können computerbasierte Nachrichten nach ihrem Erhalt weiter verarbeitet werden. Im Zusammenhang mit der immer komplexer werdenden globalen Vernetzung lokaler Informationssysteme können Nachrichten nicht nur in Sekundenschnelle übertragen, sondern auch gespeichert, weitergeleitet und auf andere Kommunikationsmedien umgesetzt werden.

Im Zuge der Weiterentwicklung von modernen, computergestützten Kommunikationssystemen gewinnen Multimediadaten, insbesondere Text, Bild, Video und Audio, sowie komplexe Dokumentformate, zunehmend an Wichtigkeit. Die Übertragung, Speicherung und Integration dieser hochvolumigen Daten ist jedoch nicht ohne Schwierigkeiten. Die weitgehenden Einsatzmöglichkeiten von Computern in der Kommunikation lassen bereits jetzt Mängel an den Speicher- und Übertragungsystemen bzw. Übertragungsprotokollen vermuten. Hochvolumige Multimediadaten, die bei der Realisierung von z. B. Konferenzsystemen, Bildschirmtelephonie und Video-/Audio-Datenbanken entstehen, lassen sich jedoch bei geschickter Verwendung und Einbindungen von z.B. Komprimierungshardware doch auf bestehenden Netzen übertragen, speichern und wieder abrufen.

1.1 Zielsetzung

Die vorliegende Arbeit erklärt und analysiert die zugrunde liegenden Verfahren zur Übertragung und Speicherung solcher multimedialer Daten und erklärt die im Kontext der Echtzeitübertragung im Internet anwendbaren Multimediaformate und deren kodierungsinherenten Eigenheiten. Weiterhin werden die Protokolle und Methoden erklärt und analysiert, die notwendig sind, um diese Daten in global verfügbaren Archiven abzulegen, und bei Bedarf wieder abzuspielen und darzustellen.

Im Rahmen dieser Diplomarbeit soll untersucht werden, inwieweit in neuen, komprimierten Formaten gespeicherte Multimediadaten mit Hilfe von bekannten Netzwerkprotokollen im Internet übertragen und gespeichert werden können.

Eine exemplarische Einbindung der besprochenen und theoretisch analysierten Methoden und Protokolle in das vom Bereich "Offene Kommunikationssysteme" (OKS) der Technischen Universität (TU) Berlin und bei der GMD Fokus, Berlin, entwickelte und persönliche Mobilität in der Telekommunikation unterstützende System *Personal Communication Support System* (PCSS), zeigt, inwieweit die entwickelten Methoden praxisnahe Verwendung finden können.

Die im Rahmen dieser Diplomarbeit entwickelten Softwarelösungen kann man als die Multimediaerweiterung des PCSS verstehen. Sie sollen insbesondere dafür sorgen, Nachrichten zu erzeugen, wieder darzustellen, zu speichern, zu verwalten und zu löschen. Weiterhin wird ein weltweit verfügbarer Dienst zur Speicherung dieser Nachrichten definiert und zur Verfügung gestellt, der im weiteren als Message-Store bezeichnet werden wird.

1.2 Abgrenzung

Im Folgenden soll das Umfeld dieser Diplomarbeit genauer beschrieben und eingegrenzt werden.

Da bis jetzt kaum einheitliche, standardisierte Text-, Bild-, Video- und Audiokodierungen vorliegen, für jede Multimediaanwendung eigene Formate Verwendung finden und die meisten Hard- und Softwarehersteller eigene Formate definiert haben und auch benutzen, muß eine Vorauswahl bei den zu betrachtenden Formaten getroffen werden.

In dieser Arbeit wird man sich auf Formate konzentrieren, die

- in universitätsnahen Institutionen (BERKOM, TUBKOM, GMD Fokus) oder an der TU Berlin selbst benutzt werden,
- bis jetzt schon durch internationale Gremien standardisiert sind (ISO/IEC, CCITT, ITU) und
- kommerzielle Formate, die sich einer weitgehenden Verbreitung erfreuen und als De-Facto-Standards bezeichnet werden können.

Dabei sollen zwar für jede Multimediaart mehrere Formate betrachtet werden, jedoch nur die im Rahmen der Implementierung sinnreichen Formate in die praktische Arbeit integriert werden.

In Betracht kommen bisher die Formate:

- für Audio: G.711 μ-law, a-law, PCM, ADPCM, MPEG und WAV
- für Video: SMP/G.711, Sun Cell, H.261 bzw. MPEG
- für Bilder: GIF, TIFF, JPEG, XBM/XPM, PBM und BMP
- für Text: ASCII, PostScript sowie verschiedenste Dokumentformate

Die bei der Übertragung zu betrachtenden Dienste sollen die im Internet auf dem TCP/IP-Protokoll bereits standardisierten Dienste sein, wie z.B.

- FTP (*File Transfer Protocol*)
- HTTP (*Hypertext Transfer Protocol*)
- RCP (*Remote Copy Protocol*)
- RDT (*Referenced Data Transfer Protocol*)

1.3 Internet

"Als Internet wird die Verbindung all jener Computer bezeichnet, die über das Protokoll TCP/IP miteinander kommunizieren." [24]

Das Internet ist ein weltweiter, generischer Kommunikationsverbund unterschiedlichster Computer, Netze und Technologien. Heutzutage sind bereits mehr als 5 Millionen Computer und 35 Millionen Nutzer an das Internet angeschlossen. Laut Hochrechnungen sollen am Ende dieses Jahrhunderts mehr als 500 Millionen Menschen über das Internet kommunizieren. Es ist damit das weltweit größte Kommunikationsmedium, abgesehen vom analogen Telefonnetz, und wird zukünftig bekannte Kommunikationsmedien integrieren bzw. ersetzen.

Einzelne Rechner werden in lokalen Netzen (*Local Area Network* (LAN)) verwaltet. Die Kopplung der verwendeten Netze (*Internetworking*), geschieht hier mit speziellen Netzknoten, *Router* genannt. Auch die im Rahmen des PCSS benutzten Rechner sind so vernetzt.

Da im Internet austauschbare Netzverbindungen, Übertragungsmedien und alle Arten von bekannter Computerhardware verwendet werden, wurde, um eine einheitliche, funktionierende Kommunikation zu garantieren und um allen Internetdiensten eine einheitliche Schnittstelle zur Verfügung zu stellen, das Kommunikationsprotokoll TCP/IP spezifiziert. TCP/IP ist zwar nicht international standardisiert, läßt sich jedoch funktionell mit den entsprechenden Protokollen des OSI-Referenzmodells vergleichen und somit einordnen. Letzteres und das Protokoll TCP/IP selber wird weiter unten ausführlich beschrieben.

1.4 Personal Communication Support System

In Verbindung mit den neuen, durch Computer gestützten, Kommunikations- und Informationssystemen gewinnt die Benutzermobilität immer mehr an Bedeutung.

Bei herkömmlichen Medien, wie Telefon, Fax oder Briefpost wird bei der Kontaktaufnahme nie direkt der gewünschte Kommunikationspartner angesprochen oder referenziert, sondern eine geräte- oder raumbezogene Adresse, an der der gewünschte Partner "vermutet" wird, oder an der er oft zugegen ist. Obwohl z.B. bei der Briefpost in der Adressierung explizit der Name des Kommunikationspartners (Adressat, Empfänger) angegeben wird, bedeutet dies nicht, das der Adressat auch zum Zeitpunkt der Zustellung dort persönlich anwesend ist. Ausgehend von diesen Problemen wird durch das Fachgebiet OKS der TU Berlin in Zusammenarbeit mit der GMD Fokus, Berlin, zur Zeit das *Personal Communication Support System* (PCSS) entwickelt (siehe [9]).

PCSS, das für den Einsatz im Bereich kleinerer, organisationslokaler Netzwerke ("*In House*"-Netzwerke) entworfen wurde, soll sowohl die über das Netzwerk angebotenen Dienste erweitern und kombinieren, um eine flexible und einfache Benutzermobilität zu unterstützen, als auch eine einheitliche Konfiguration der angebotenen Dienste ermöglichen.

Die Abbildung 1 zeigt die logischen Komponenten innerhalb des PCSS. Auf die weiß hinterlegten Service wird weiter unten eingegangen, die Multimedia-Service und der Message-Store werden innerhalb dieser Arbeit entwickelt (dunkelgrau hinterlegt), der schraffiert dargestellte Bereich des *Management Information Service* (MIS) realisiert die Verwaltung der Profiles innerhalb des PCSS. Für eine ausführliche Beschreibung des MIS wird auf [9] verwiesen.

Innerhalb des PCSS wird ein Benutzer dann eindeutig über eine persönliche Kennung referenziert. Das PCSS unterstützt den Aufbau von Kommunikationsverbindungen der computergestützten Kommunikationsmedien durch das räumliche Auffinden des Kommunikationspartners, durch die Analyse der an dem Kommunikationsort vorhandenen Kommunikationsgeräte (wie Telefon, Multimedia-Workstation oder Text-Terminals) und durch den Aufbau einer an alle Beteiligte angepaßte, bestmögliche Verbindung.

Weiterhin sorgt das PCSS dafür, daß bei Nichtzustandekommen von Kommunikations-verbindungen, z.B. Anrufweiterleitung und -speicherung sowie Anrufbeantworterfunktionalitäten aktiviert werden.

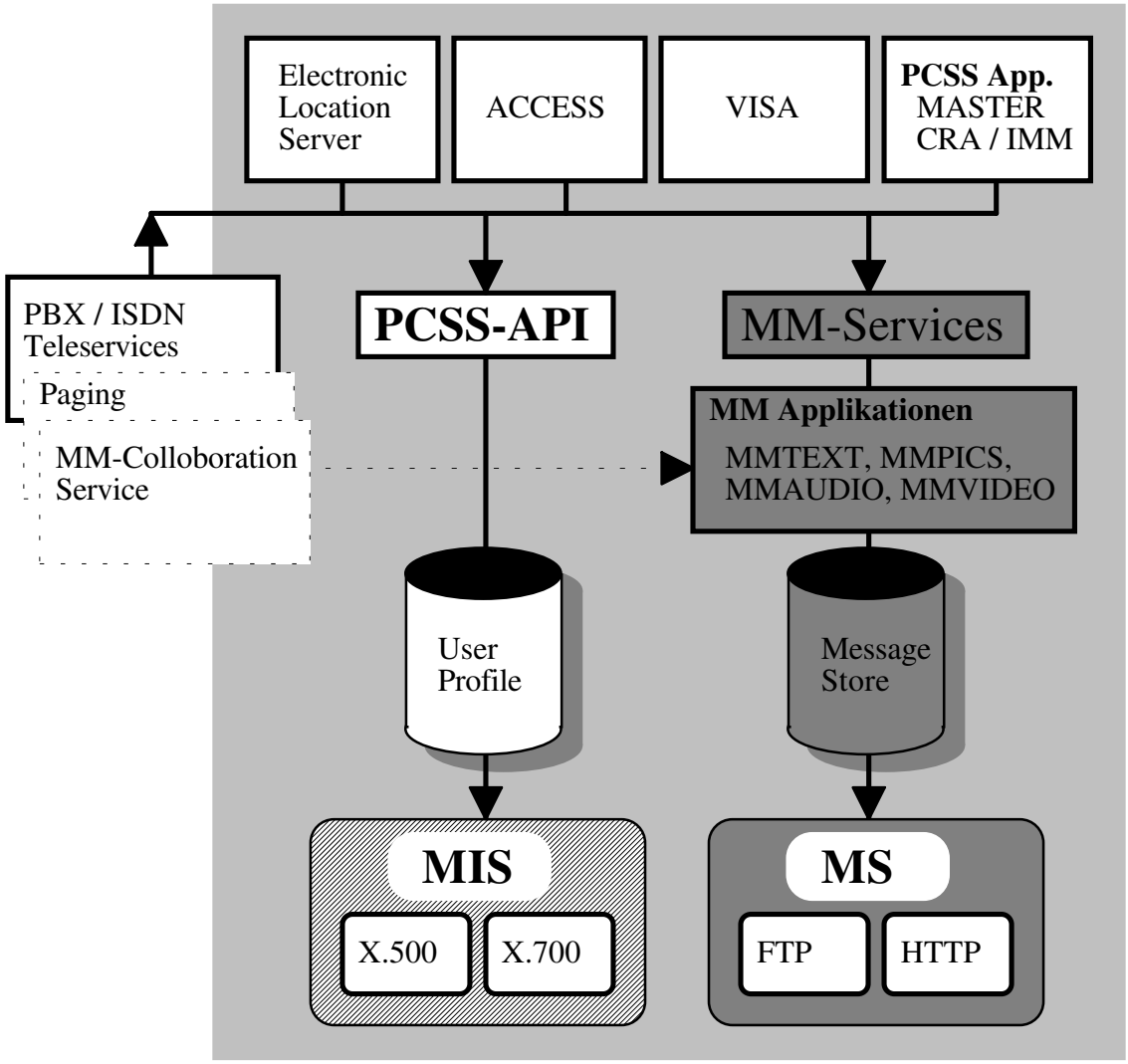


Abb. 1: *PCSS-Komponenten und Realisierungsumgebung*

Dafür müssen die multimedialen Text-, Bild-, Video- und Audionachrichten von geeigneten Applikationen aufgenommen und in den Message-Store übertragen bzw. gespeichert werden. Die Nachrichten werden dann auf der Empfängerseite als eingegangene Nachricht vermerkt.

1.4.1 Hardware- und Softwareplattform

Die für das PCSS verwendete Hardwareplattform sind die Unix-Workstations der Firma Sun, insbesondere die Baureihe SparcStation 10 und 20, mittlerweile auch die SparcStation 5. Diese Workstations enthalten alle ein Audio*device* mit der Möglichkeit, Audiodaten in einer der Audio-*Compact Disc* (CD) vergleichbaren Auflösung zu digitalisieren und wieder abzuspielen. Für die zur Videodigitalisierung notwendige Hardware werden sowohl die SunVideokarte der Firma Sun als auch die Parallax-Karte ins Auge gefaßt. Softwareseitig wird das Betriebssystem Solaris der Firma SUN und die graphische Oberfläche X11 bzw. OpenWindows der Firma SUN eingesetzt.

Bei der Entwicklung der graphischen Oberflächen hat man sich zwecks Einheitlichkeit des Designs der Applikationen auf den Einsatz der folgenden Systeme geeinigt:

- **Tcl** - interpreterbasierte Scriptsprache
- **Tk** - Interfacescriptsprache basierend auf Tcl

- **Tix** - zusätzliche Interfaceklassen in einem einheitlichen Design
- **BLT** - *drag&drop* Funktionalität als Erweiterung für Tk

1.5 Einsatzgebiete

Innerhalb des PCSS gibt es mehrere Module, welche die im Rahmen des praktischen Teils dieser Diplomarbeit entwickelten Softwarelösungen verwenden werden. Diese sind:

- IMM-Manager

  Der *Incoming Message Manager* hat die gleichen Anforderungen an die multimediale Erweiterung des PCSS wie der CRA-Manager, er verwaltet anstatt CRAs eingehende Nachrichten und ermöglicht deren Anzeige und Löschung aus dem Message-Store.

Abb. 2: *Incoming Message Manager*

- CRA-Manager

  CRA steht für *Customized Recorded Announcements*. Der entsprechende Manager verwaltet die PCSS-spezifischen Ansagetexte eines Benutzers innerhalb des PCSS. Er muß mit Hilfe der hier implementierten Applikationen multimediale Nachrichten erzeugen, verändern, anzeigen und löschen, sowie diese Nachrichten in den Message-Store übertragen können.

- ACCESS

  Das *Advanced Communication Control Environment and Scheduling System* (ACCESS) wird den Zugriff auf sämtliche Inhalte des persönlichen User-Profiles ermöglichen. Mit der Management-Applikation ACCESS kann der Teilnehmer am PCSS seine im PCSS enthaltenen, benutzerbezogenen, persönlichen Datenstrukturen wie z.B. den Systemzugang manipulieren. Weiterhin kann ein PCSS-Teilnehmer durch ACCESS seine individuelle Kommunikationsumgebung gestalten, z.B. Ansagetexte aufnehmen und aufgezeichnete Nachrichten darstellen. Letzteres wird durch die hier entwickelten Applikationen realisiert.

- MASTER

  MASTER, oder auch *Profile Administration Tool* (PAT) genannt, ist die Applikation innerhalb des PCSS, die es ermöglichen soll, sämtliche Profiles (Zonen, *Virtual Access Points* (VAP), *Service Access Points* (SAP) und die User Profiles) zu administrieren, d.h. erstmalig zu erstellen und dann zu modifizieren oder zu löschen. Weiterhin soll MASTER auch die Konfiguration von Zugriffsrechten und die Verwaltung von Passwörtern ermöglichen. Im Zusammenhang mit dieser Arbeit muß von MASTER auch die Initialisierung der Benutzerbereiche innerhalb des Message-Store möglich sein. Nachrichten innerhalb des Message-Stores müssen von MASTER notfalls gelöscht oder verschoben werden können. Das Konzept von sogenannten System-Nachrichten innerhalb des Message-Stores ist bereits vorgesehen.

- VISA

  Das *Visitor Information Service Application* (VISA) System wird im Rahmen des PCSS als eine der Applikationen entworfen, die den Nutzen der Integration von PCSS Mechanismen (sprich persönlicher Mobilität) in andere Systeme zeigen soll. Durch die Realisierung des multimedialen Besucherinformationssystem VISA können sowohl multimediale als auch administrative Aspekte des PCSS anschaulich demonstriert werden.

  Das Info-Terminal VISA nimmt im weitesten Sinne die Aufgaben eines Sekretärs oder einer "Empfangsdame" und Pförtners wahr, und ermöglicht dadurch auch eine Form der Eingangskontrolle. Der Besucher kann die in Gruppen oder Projekten zugeordneten Mitarbeiter mit Hilfe des PCSS lokalisieren und mit ihnen per Telefon oder Video-Phone kommunizieren, bzw. audiovisuelle Nachrichten hinterlassen.

  VISA wird momentan auf Basis der Scriptsprache Metacard auf einer mit einer SunVideo-Karte ausgerüsteten SparcStation 20 entwickelt und innerhalb der Räume der GMD Fokus getestet. Eine Integration der im Rahmen dieser Diplomarbeit entstandenen Multimedia-Applikationen in das VISA-System wird angestrebt. Daher müssen der Aufbau und gegebenenfalls die Funktionen der Multimedia-Applikationen leicht an die graphische Struktur von VISA anzupassen sein. Dieses ist eine weiteres Entwurfskriterium des praktischen Teils dieser Arbeit.

- Phone-Mail-Server

  Der PM-Server ist die Komponente des PCSS, die die multimediale Anbindung an eine digitale ISDN-Vermittlungsstelle realisiert. Er muß dabei CRAs aus dem Message-Store über ISDN-Telefone abspielen und eingehende Nachrichten im Message-Store ablegen können [39].

Weiterhin sind viele Einsatzmöglichkeiten für die hier entwickelten Applikationen und Mechanismen innerhalb der Internetwelt denkbar, wie z.B. die Verknüpfung der Multimedia-Applikationen mit dem *WorldWideWeb* (WWW) zum zeitabhängigen Rezipieren von multimedialen Nachrichten, sowie der Einsatz im Rahmen von *Multimedia Mail* und *Multimedia Colloboration Service*, die von der GMD Fokus entwickelt werden. Dies sollte bei der Realisierung bzw. Implementierung konzeptionell berücksichtigt werden und sich in einem modularisiertem Aufbau der Dienste, Mechanismen und Methoden wiederspiegeln.

Konzepte eines Message-Stores werden bei zunehmender Komplexität der globalen Kommunikation weiter an Wichtigkeit gewinnen. Nachrichten müssen zunehmend weltweit in abruf- und modifizierbarer Form gespeichert werden.

1.5 Aufbau

Im Rahmen dieser Diplomarbeit werden die folgenden Grundthemen behandelt:

- Kapitel 1
  Einführung in die Anforderungen der zu entwickelnden Methoden, Definition von Begriffen und Beschreibung des Implementierungsumfeldes.

- Kapitel 2
  Erklärung der grundlegenden Protokolle, Kompressionsmethoden und -formate.

- Kapitel 3
  Auswahl und Konzeption von speziellen Protokollen und Multimediaformaten, sowie eines Message-Stores.

- Kapitel 4
  Implementierung eines *Message-Stores* (MS) im Rahmen des PCSS sowie der benötigten Zugriffsfunktionen und geeigneter Aufnahme- und Präsentationssoftware, sowie die Analyse und der Test der implementierten Dienste.

- Kapitel 5
  Einordnung des Gesamtkonzeptes, abschließende Bemerkungen, Asublick und Zusammenfassung.

Grundwissen über Digitalisierung von analogen Signalen und der grundsätzlichen Vernetzung von Computern sowie fachspezifische Englischkenntnisse werden bei der Lektüre dieser Arbeit vorausgesetzt. Themen der Sicherheit von Computersystemen im Allgemeinen werden hier nicht behandelt.

Kapitel 2 - Grundlagen

In diesem Kapitel werden die grundlegenden Dienste und Protokolle der Übertragung von digitalen Daten sowie die Kodierungen und Kompressionstechniken von multimedialen Daten vorgestellt und erläutert.

2.1 TCP/IP-basierte Protokolle

Um eine Einordnung der bei der Übertragung benutzten, und auf den im Internet verbreiteten Übertragungsprotokoll TCP/IP basierenden, Kommunikationsprotokolle wie HTTP, FTP und RCP zu ermöglichen, wird hier zuerst das OSI-Schichtenmodell erklärt. TCP/IP läßt sich mit dort standardisierten Protokollen vergleichen. Hierbei werden auch gleichzeitig die grundlegenden Probleme der digitalen Kommunikation bzw. der digitalen Übertragung besprochen.

2.1.1 OSI-Referenzmodell

Eine komfortable Kommunikation ist nur möglich, wenn die auf den physikalischen Leitungen existierenden Probleme wie Übertragungsfehler, Überlastung der Leitungen, Synchronitätsverlust oder Unerreichbarkeit der an der Kommunikation beteiligten Systeme überwunden werden. Es gibt zwei grundsätzlich verschiedene Ansätze, um diesen abzuhelfen:

- Protokollerweiterung

  In die bereits verwendeten und bekannten Protokolle werden Mechanismen eingebaut, um die zusätzlichen Probleme zu lösen. Das würde bedeuten, immer wieder die gleichen grundlegenden Probleme mit den gleichen Mitteln (wie Protokollmeldungen, Timeouts etc.) in verschiedenen Übertragungsprotokollen (wie z.B. LAPB, Datex-P) zu lösen.

- Strukturierter Ansatz

  Die bei einer Kommunikation auftretenden Probleme werden in strukturierter, modularisierter Weise angegangen, indem man verschiedene Protokolle für verschiedene Zwecke entwickelt, die hierarchisch angeordnet sind. Jede Protokollschicht erweitert die Funktionalität der darunterliegenden Schicht. Man erhält also eine Hierarchie von austauschbaren Protokollen.

Die Internationale Standardisierungsorganisation ISO hat daher ein Modell für eine solche hierarchische Kommunikationsarchitektur entwickelt, das sogenannte Referenzmodell für Offene Kommunikationssysteme, *Open System Interconnection*, kurz OSI-Modell. Das OSI-Modell wurde 1984 in der Norm ISO 7498 festgeschrieben (siehe [26]).

Jede Schicht des OSI-Modells hat die Aufgabe, einen bestimmten Aspekt der Kommunikation zu behandeln. Diese Bereitstellung einer Kommunikationsleistung nennt man Dienst (*Service*). Oberhalb einer Schicht ist nur ihr Dienst bekannt und nicht, wie er von dieser Schicht unter Zuhilfenahme der darunterliegenden Schicht erbracht wird.

Es sind 7 Schichten von der CCITT benannt und im Rahmen der X.200-Serie veröffentlicht worden. X.200 enthält die Beschreibung des OSI-Modells, X21*n* definiert dabei die Dienste der Schicht *n*, X.22*n* spezifiziert das zugehörige Protokoll. Die Schichten bauen wie folgt aufeinander auf [26]:

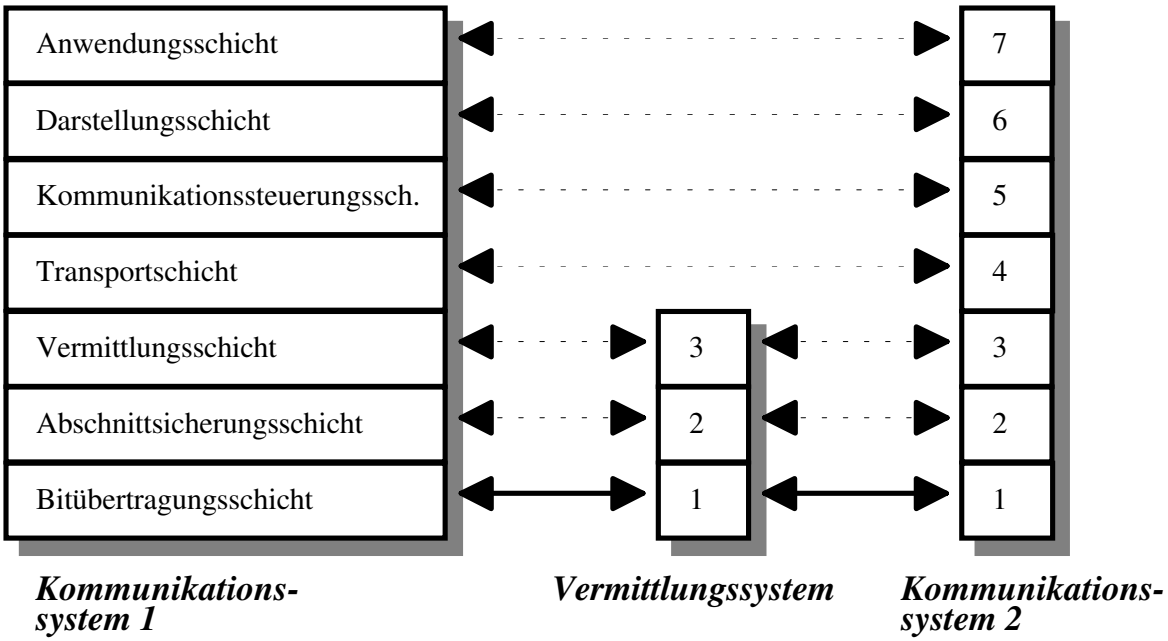


Abb. 3: *Die sieben Schichten des OSI-Referenzmodells*

- **Bitübertragungsschicht**

  Diese Schicht beschreibt die Übertragung von Signalen auf einem konkreten Kommunikationsmedium wie z.B. dem analogen Telefon oder digitalen ISDN-Leitungen. Dabei werden die Parameter der Bit- und Bytesynchronisation, Bit- und Bytetakt, Informationspegel, Repräsentation von Signalen, Übertragungsraten und Übertragungsrichtung festgelegt.

  Die Bitübertragungschicht stellt hierbei als Dienst eine Übertragung von einzelnen Bits über ein zu einem Kommunikationspartner direkt vorhandenes Kommunikationsmedium zur Verfügung. Es wird nur eine Kommunikation auf einem einzelnen Übertragungsabschnitt betrachtet. Werden z.B. Modems (Modulator/Demodulator) verwendet, können hier die international standardisierten Protokolle V.34 oder V.21 als Schicht-1-Protokoll benutzt werden.

- **Abschnittssicherungsschicht**

  Der Dienst der Abschnittssicherungsschicht stellt eine vor Übertragungsfehlern weitgehend gesicherte und vor Überlastung geschützte Übertragung von größeren Informationseinheiten (*frames* [1]) zur Verfügung und bedient sich dabei der Dienste der Bitübertragungsschicht. Wie bei der Bitübertragungsschicht wird hier auf einem einzelnen Übertragungsabschnitt operiert.

  Zusätzlich zur Übertragungsgüte der Schicht 1 realisiert die Schicht 2 den Bytetakt, Gruppierung und Sortierung von Informationseinheiten, Reduktion von Übertragungsfehlern

[1] *Frames* sind grundsätzlich nur eine Anzahl von digitalen Werten, die zu einer Informationseinheit zusammengefaßt werden. In der Videotechnik werden *frames* jedoch anders definiert.

bzw. die Wiederholung der Daten bei Datenverlust und Schutz vor einer Datenüberschwemmung beim Empfänger durch Anwendung von Flußkontrollmechanismen.

- **Vermittlungsschicht**

  Da es generell unmöglich ist, zwischen allen potentiellen Kommunikationspartnern direkte Leitungen zu verlegen, werden die Endkommunikationsstellen zumeist über ein Vermittlungssystem der Schicht 3 angeschlossen.

  Dabei verwaltet das verwendete Protokoll der Vermittlungsschicht einen Adressraum, um über mehrfach genutzte Leitungen einzelne Kommunikationspartner anzusprechen. Das Schicht 3 Protokoll muß dabei die Probleme der Wegewahl durch das Netz lösen, virtuelle Verbindungen über eine physikalische Leitung schaffen und Maßnahmen zur Netzkopplung ergreifen.

  Die Protokolle der Vermittlungsschicht sollen hier nicht genauer erläutert werden. Das im Internet grundlegende Protokoll der Schicht 3 namens IP wird weiter unten auf Seite 11 erklärt.

- **Transportschicht**

  Die Hauptaufgabe der Transportschicht besteht in der Kostenoptimierung und der Verbesserung der Dienstqualität. Die Transportschicht kann dabei die gewünschte Qualität der Verbindung durch geeignete Protokollmechanismen tatsächlich erbringen. Um die verschiedenen Abstufungen der Dienste der Transportschicht zu erbringen, wurden von der ISO dabei vier Klassen von Transportprotokollen angeboten, sie unterscheiden sich nach folgenden Funktionalitätsgruppen:

  1. Adressierung von Anwendungen innerhalb der Endsysteme
  2. *Segmenting/Reassembling*
  3. Fehlererkennung und -behebung
  4. Kostenoptimierung
  5. Sonstiges wie *Resequencing*, *Splitting* und *Recombining*

  Die verschiedenen Transportprotokolle sollen hier nicht genauer erläutert werden, das im Internet und im Rahmen dieser Arbeit verwendete Protokoll namens TCP wird weiter unten auf Seite 11 erklärt.

- **Kommunikationsschicht, Darstellungsschicht**

  Diese beiden Schichten ermöglichen grundsätzlich eine gesteuerte Kommunikation von Anwendungen, in dem z.B. die Rollen bei der Übertragung und Teile verschiedener zu übertragener Dateien und Dokumente genau identifiziert werden. Dazu werden auch Syntaxen zur Aushandlung von Übertragungsmodi und Transferprotokollen definiert (z.B. die *Abstract Syntax Notation 1* (ASN.1) [12], mit der z.B. die Strukturen der PCSS-internen Daten definiert werden). Die später vorgestellten Dienste benutzt keine Protokolle dieser beidern Schichten.

- **Anwendungsschicht**

  In der Anwendungsschicht können Dienste dann abstrakt gekapselt und modelliert werden, d.h. die Kommunikation mit einem Dienst wird festgelegt, Implementierung und darunterliegende

Protokolle bleiben jedoch frei wählbar. Die Anwendungsschicht ist im Rahmen dieser Arbeit nur von Interesse, da mit ihrer Hilfe Dienste allgemein festgelegt wurden, deren konkrete internetspezifische Ausprägung später genauer betrachtet wird. Zu nennen sind hier z.B. *File Transfer, Access and Management* (FTAM), eine Client/Server-Lösung zur Realisierung eines virtuellen Dateienspeichers. Sein Gegenstück im Internet ist dabei das *File Transfer Protocol* (FTP).

2.1.2 Internet Protokoll

Das gegenwärtig wichtigste Protokoll zur Netzkopplung ist das *Internet Protocol* (IP). Die Hauptaufgabe des IP ist die netzübergreifende Adressierung, es verbessert jedoch kaum die Dienste der darunterliegenden Netze bzw. Schichten. IP arbeitet verbindungslos durch den Austausch von Datagrammen. Datagramme bestehen dabei aus Datenpaketen, versehen mit einem IP-*header*, der alle Informationen betreffs Länge, Überprüfungssummen, Sender- und Empfängeradressen und Protokollbezeichner enthält.

Sender- und Empfängeradresse werden in 32-bit-Feldern angegeben und bestehen aus einer Angabe des Netzes und des darin befindlichen Anschlusses (Rechner, *host*). Ein Datagramm besteht dabei aus einem IP-*header* und dem IP-Datenfeld. Der IP-*header* enthält unter anderem die Adresse des End-Empfängers. Falls sich dieser in einem anderen Netz befindet als der Sender, ist keine direkte Versendung möglich, stattdessen wird ein Router (im Internet wird dieser auch *Gateway* genannt) im eigenen Netz angesprochen, der die Weiterleitung der Informationen anhand sogenannter *Routing*-Tabellen (sinngemäß als Wegweiser zu anderen Netzen zu verstehen) übernimmt.

2.1.3 Transmission Control Protocol

Das *Transmission Control Protocol* (TCP) ist innerhalb der Internet-Protokolle das verbindungsorientierte Transportprotokoll oberhalb von IP. Seine Funktionalität entspricht der eines ISO-Transportprotokolls der Klasse 4 der Transportschicht und wird angewendet, wenn der Dienst der darunterliegenden Schicht als nicht akzeptabel angesehen wird oder explizit verbindungsorientiert gearbeitet werden muß. Grundsätzlich bietet TCP folgende Funktionalität:

- Auf- und Abbau von End-zu-End-Verbindungen zwischen adressierten Anwendungen
- End-zu-End-Fehlererkennung und -behebung durch Prüfsummen
- End-zu-End-Flußkontrolle
- *Sequencing* (die darunter liegenden Netze müssen keine Sequenzerhaltung bieten, IP arbeiten mit Datagrammen)
- Übertragung von Eildaten

TCP bietet daher eine zuverlässige und reihenfolgeerhaltende Übertragung von Nutzinformationen. TCP geht von einer volumenabhängigen Tarifierung aus, eine Kostenoptimierung ist daher nicht realisiert. Geboten wird eine Übertragung eines Bytestroms, größere zur Übertragung generierte Informationspakete sind für die Anwendungen nicht erkennbar.

Verbindungen zwischen zwei Kommunikationsprogrammen durch TCP/IP erfolgen dabei via *sockets* (Sockel). Diese *sockets* werden zumeist von einem TCP/IP unterstützenden Betriebssystem beim Aufbau der Verbindung realisiert und können von den Programmen mit den gleichen

Methoden wie zum Zugriff auf Dateien benutzt werden. *Sockets* können geöffnet und geschlossen werden, von ihnen können Daten gelesen oder in sie geschrieben werden.

2.1.4 File Transfer Protocol

Das *File Transfer Protocol* (FTP) ist im *Request For Comment* (RFC) 765, sowie diversen Zusätzen (RFC 412, 480, 640, 697) definiert und festgelegt, wird auf TCP/IP realisiert und entspricht dabei einer Applikation der Schicht 7 des OSI-Modells.

In einer FTP-Verbindung werden zwei reelle über *sockets* realisierte Verbindungen benutzt, eine *control connection* und eine *data connection*. Dateien werden dabei über die *data connection* in beide Richtungen übertragen, die Kommunikation zwischen *client* und *server* geschieht wechselseitig über die *control connection*, der die Kommunikation anstoßende Partner ist dabei der *client*.

Die FTP-Kommandos (wie z.B. user, pass, get, put, chmod, delete etc.) werden zusammen mit nötigen Parametern wie Dateinamen oder Zugriffsrechten an den *server* übertragen, dieser antwortet mit einem Aufbau einer zusätzlichen *data connection* oder mit Fehlermeldungen über die *control connection*.

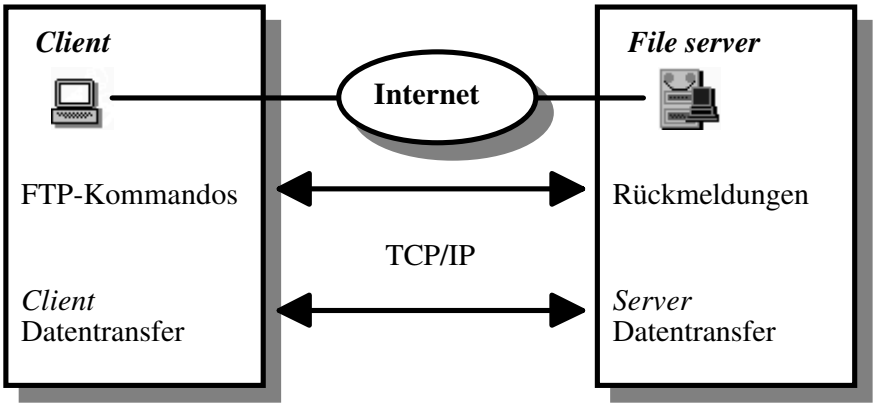


Abb. 4: *Client-Server-Beziehung bei FTP*

Der Dateitransfer in beide Richtungen, Änderung von Zugriffsrechten, Löschen von Dateien, Traversieren und Auflisten von Verzeichnisbäumen, sowie eine Benutzerautorisierung werden vom FTP-Protokoll unterstützt. Serverseitig werden mehrere simultane *client*-Verbindungen unterstützt.

2.1.5 HyperText Transmission Protocol

Das *HyperText Transmission Protocol* (HTTP) ist als Internet-Draft definiert und festgelegt, wird auf TCP/IP realisiert und in die Schicht 7 des OSI-Modells eingeordnet [10].

Es ist das zugrundeliegende Protokoll für den im Internet bestehenden Netzwerkservice *WorldWideWeb* (WWW, W3). WWW verbindet die Übertragung von Dokumenten und deren einheitliche Darstellung (spezifiziert in der Seitenbeschreibungssprache *HyperText Markup Language* (HTML)), mit einer Anbindung an externe Programme (z.B. Datenbanken). "World Wide Web implementiert damit auf dem elektronischen Netzwerk ein logisches Netzwerk von

miteinander verbundenen Dokumenten auf verschiedenen Internet-hosts" [24] im Hypertext-Format [2].

Zur Darstellung der in HTML spezifizierten Dokumente wird ein sogenannter Hypertext-*browser* oder Web-*browser* eingesetzt, die Dokumente selber werden im Datensystem eines Servers-*hosts* abgelegt. *Browser* und Server kommunizieren zum Datenaustausch mit dem Protokoll HTTP.

Uniform Resource Locator

Zur weltweit eindeutigen Referenzierung von Dokumenten werden sogenannte *Uniform Resource Locator* (URL) verwendet. Diese kann man als eine um das Zugriffsprotokoll, den Hostrechner, und den Pfad erweiterten Dateinamen verstehen, definiert wird er wie folgt:

<protocol>://<host>/<path>/<file>
z.B. **http://www.cs.tu-berlin.de/user/phade/mpeg.html**

Als Protokoll können dabei alle im Internet benutzten Kommunikationsprotokolle angegeben werden, z.B. telnet, ftp oder http. Als *host* wird der Rechnername, der das Dokument verwaltet, einschließlich Netzwerknamen (*domain*) in Punkt- [3] oder voller Schreibweise angegeben. Die Pfad- und Dateiangabe ergänzen die Referenzierung.

Zur Dereferenzierung, d.h. dem Zugriff auf ein Dokument, baut der Web-*browser* via TCP/IP eine *socket*-Verbindung zum Server auf und schickt das Kommando 'get URL', wobei URL eine Referenz auf eine Dokument darstellt. Daraufhin schickt der Server mehrere Textzeilen als *header* [4] , der Dateiname, Länge und Typ des geforderten Dokuments oder eine Fehlermeldung enthält, sowie das geforderte Dokument selbst über den gleichen *socket* zurück, welches dann vom *browser* für den Benutzer dargestellt wird. Ein zweites Kommando (put oder post) ermöglicht dem *browser* spezielle Daten an den Server zu schicken, z.B. die Daten eines vom Benutzer ausgefüllten Formulars.

MIME-Typen

MIME-Typen bieten eine internetspezifische Möglichkeit, um das Format von übertragenen oder gespeicherten Dateien zu spezifizieren. MIME-Typen werden z.B. innerhalb des *headers* einer HTTP-Übertragung angegeben. MIME bedeutet *Multipurpose Internet Mail Extensions* (MIME) und wurde im RFC 822 zur einheitliche Gestaltung der *header* im Bereich Internet-e-mail definiert und gilt als de-facto-Standard.

MIME-Typen bestehen aus einem durch einen Schrägstrich getrennten Haupt- und einen Subtypen.

---

[2] Hyptertext-Verbindungen sind in ein Dokument integrierte Querverweise auf andere Dokumente, über die der Benutzer direkt auf diese anderen Dokumente zugreifen kann. Diese Dokumente können wiederum Hyptertext-Verbindungen enthalten.

[3] Unter Punktschreibweise versteht man die Angabe der IP-Adresse, welche aus vier durch Punkte getrennte 8-bit-Zahlen in Dezimalschreibweise besteht, z.B. 192.135.35.10. Damit nicht beim Kommunikationaufbau komplizierte Zahlenfolgen verwendet werden müssen, steht im Internet der *Naming Service* zur Verfügung, der die Zahlenfolgen auf symbolische Namen abbildet. In dieser vollen Schreibweise wird ein Rechner durch die Angabe des Rechernamens und des Netzwerknamens eindeutig identifiziert, z.B. www.tu-berlin.de.

[4] Als *header* (Kopf) werden im folgenden Daten verstanden, welche die nachfolgenden Daten einer Datei genauer spezifizieren oder erklären.

<main-typ>/<sub-typ>
z.B. **audio/x-mpeg**

Der Haupttyp beschreibt die Rezeptionsart (audio, video, text, application, document etc.), der Subtyp das Format, in dem die Daten vorliegen (z.B. wav, mov, txt, exe, ps). Einem in Standardisierung befindlichen Format wird dabei im Subtyp ein 'x-' vorangestellt. Serverseitig wird die Verbindung zwischen Datei und MIME-Typ über die sogenannte Dateinamenerweiterung realisiert.

HTTP ist eines der einfachen Protokolle der Internet-Kommunikation und eignet sich deshalb besonders, um neue entstandene Programme auf einfache Weise mit weltweit gespeicherten Dokumenten bzw. multimedialen Dateien zu verbinden.

2.1.6 Remote File Copy

Remote File Copy (RCP) ist seit dem Unix-Betriebssystem BSD 4.3 der Universität Berkeley eine verbreitete Methode um Dateien zwischen Dateisystemen verschiedener im Internet vernetzter Rechnersysteme zu kopieren. RCP ist auf den meisten Unix-Varianten als Programm implementiert, welches die zur Datenübertragung notwendige *socket*-Verbindung durch das Starten einer *remote-shell* [5] auf dem entfernten Rechnersystem ermöglicht. Die Dateien werden dann über die Ein- und Ausgabekanäle der verbundenen Applikationen (RCP und *remote-shell*) übertragen.

RCP setzt die gleichen Authorisierungsmethoden wie eine *remote-shell* voraus. Da eine Speicherung eines Benutzernamens für jedes beteiligte LAN innerhalb des PCSS nicht praktikabel ist, muß der Benutzer sowohl auf dem eigenen, als auch auf dem entfernten Rechner den gleichen Benutzernamen und das gleiche persönliche Passwort benutzen. Dies ist zumeist nur in lokalen Netzwerken der Fall. Eine Übermittlung eines Passwortes für den entfernten Rechner ist nicht möglich.

2.2 Global Store

Das *Distributed Office Applications Modell* (DOAM) [14] beschreibt ein Konzept der globalen Referenzierung von multimedialen, globalen Objekten namens *References Object Access* (ROA).

Innerhalb dieses Modells werden mehrere Datentypen definiert, unter anderem die *Distinguished Object Reference* (DOR). Diese ermöglicht eine weltweit eindeutige Referenzierung eines Objektes, vergleichbar der Funktionalität einer URL. Weiterhin wird dort ein Zugriffsprotokoll zum ROA Modell namens *Referenced Data Transfer* (RDT) definiert.

Bei der GMD Fokus wurde ein DOA-Modell implementiert, um innerhalb eines von BERKOM und CIO (Race-Projekt 2060) entwickelten *Multimedia Mail Systems,* Nachrichten mit einem Referenzmechanismus auszustatten. Multimediale Komponenten dieser Nachrichten werden dann

[5] Unter einer *remote-shell* versteht man einen auf einem entfernten Rechner gestarteten Kommandozeileninterpreter. Aus diesem heraus können nun alle möglichen Befehle auf dem entfernten Rechner ausgeführt werden. Die *socket*-Verbindung der *remote-shells* wird zumeist über betriebssysteminterne Server-Prozesse ermöglicht, ähnlich FTP und HTTP.

innerhalb eines *Global Stores* (GS) gespeichert und mittels RDT übertragen. Referenzen auf diese Objekte werden mit Hilfe des ebenfalls bei der GMD Fokus entwickelten Services *External Reference Production* (ERP) erzeugt, die dann resultierenden Verweise als DORs in den Nachrichten versandt und gespeichert [38].

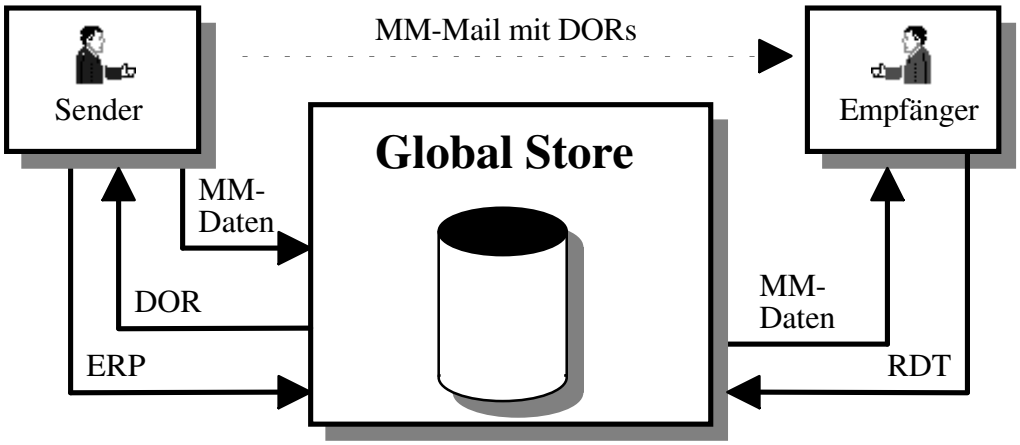


Abb. 5: *BERKOM MM-Mail als Beispiel für eine Global Store Anwendung*

Somit brauchen multimedialen Komponenten nicht generell innerhalb der Nachricht übertragen werden. Die Komponenten werden erst aus dem weltweit zugänglichen GS übertragen, wenn der Empfänger der Nachricht dies explizit wünscht.

2.2.1 Referenced Data Transfer Protocol

Referenced Data Transfer Protocol (RDT) ist das Zugriffsprotokoll zum DOA-Modell, in unserem Beispiel dem *Global Store*. Sowohl der RDT-Nutzer als auch der den RDT erbringende Server können die Funktionen des RDT Protokolls ausführen, der Zugriff ist symmetrisch.

Die erste Operation innerhalb des RDT-Protokolls ist die Operation "übertragen" (*transfer*). Sie ermöglicht das Anstoßen einer Übertragung einen Objektes aus dem *Global Store*, das Objekt wird dabei durch die DOR referenziert. Die zweite Operation ermöglicht die Änderung des *Quality of Service* (QoS) der Übertragung eines Objektes und die direkte Modifikation einer DOR (*extend*).

2.3 Kodierungs- und Kompressionstechniken

In diesem Abschnitt sollen die Probleme des Volumens von multimedialen Daten, ihrer Kompressionstechniken, des rechnerischen Aufwands der Komprimierung und der Kodierungen und Protokolle selbst erläutert werden.

Diese Analyse soll insbesondere im nächsten Kapitel zu einer Bewertung der ausgewählten Formate und Protokolle führen, die zeigt, wie die Kodierungen ausgenutzt werden können, um die Belastung von Netzen zu minimieren, bzw. paketweisen Zugriff zu ermöglichen und die durch die Formate ermöglichten Rezeptionsweisen zu unterstützen.

2.3.1 Dokumentformate

ASCII

Der mit 7 bit zu beschreibende *American Standard Code for Information Interchange* (ASCII) ist der am meisten verbreitete Kommunikationscode nicht nur innerhalb der Vereinigten Staaten, sondern weltweit.

Die zur Kommunikation benötigten Schriftzeichen wurden von der ANSI in der Vorschrift X3.4, seine internationalen Erweiterungen von der CCITT in der Vorschrift T.50 und ISO 646 festgelegt. Letztere benutzen dann ein achtes Bit, um länderspezifische Sonderzeichen integrieren zu können. Unformatierter Text wird zumeist mittels ASCII-Zeichen gespeichert und übertragen, alle in den Kommunikationsprotokollen benötigten Befehle werden ASCII-kodiert übertragen, selbst *Sourcecode* für Programme wird in ASCII geschrieben.

HTML

Die *HyperText Markup Language* (HTML) ist der Internet-Standard, der auf Basis des ASCII-Zeichensatzes den "Internet Hypertext" definiert. Als Hypertext bezeichnet man durch Querverweise verbundene Dokumente. HTML enthält generische Kodierung zur Beschreibung von Dokument- und Verweiselementen, ähnlich PostScript (siehe Seite 17). HTML ist von der generischen und international standardisierten Seitenbeschreibung SGML abgeleitet worden. Die *Structured Generalized Markup Language* (SGML) wurde für das Anwendungszenario 'Verlagswesen' entwickelt und 1986 als ISO-Standard 8879 (siehe auch [13]) verabschiedet. Sie ist eine (Meta)Sprache zur Auszeichnung von Dokumentstrukturen.

Wie SGML, ermöglicht auch HTML Dokumentelemente mit einem "deskriptiven Markup" zu versehen, HTML ist also menschenlesbar. Da alle Hypertextelemente in ASCII spezifiziert werden, können HTML-Dokumente mit herkömmlich textbasierten Editoren bearbeitet werden. Hypertextelemente werden durch *tags* vom restlichen Text abgetrennt (normalerweise durch die kleiner/größer Zeichen '<' und '>'. Folgende Gruppen von Elementen lassen sich unterscheiden:

- **Anker** sind Verweise auf andere Stellen innerhalb des Dokuments oder Verweise auf andere Dokumente. Letztere werden zumeist in URL-Schreibweise innerhalb des *tags* angegeben. Eine Applikation, die HTML-Dokumente darstellt, ermöglicht die Selektion der Anker durch den Benutzer, um ein weiteres Dokument anzuzeigen. Anker können auch auf Applikationen oder multimediale Dokumente wie z.B. Video- und Audioströme, Bilder und Texte verweisen.

- **Titel** spezifizieren den beschreibenden Namen eines Internet Hypertext Dokumentes.

- **Paragraphen**: Absätze werden von den HTML-Applikationen umgebrochen. Paragraphen ermöglichen die Abtrennung einzelner Absätze eines Dokumentes.

- **Überschriften** können in verschiedenen Größen angelegt werden und bedingen einen neuen Absatz.

- **Listen:** Mehrere verschiedene Listenarten werden definiert.

- **Grafiken:** Der Import von verschiedenen Grafiken wird unterstützt.

- **Sonderzeichen:** Zeichen, wie z.B. länderspezifische Umlaute können so schriftsatzunabhängig angegeben werden.

- **Unformatierter Text:** Der Zeilenumbruch durch die HTML-Applikation kann so unterbunden werden.

HTML wird zur Zeit um die Einbindung von weiteren Applikationen, um neue Dokumenttypen und neue Dokumentelemente erweitert (HTML 3.0, HTML+). Die Trennung von Information innerhalb von Dokumenten und Informationsverarbeitung innerhalb von Applikationen wird daher weiter verschwimmen.

PostScript

PostScript wurde von der Firma Adobe ca. 1985 konzipiert und ist in erster Linie eine ASCII-basierte, komplexe Seitenbeschreibungssprache, die hauptsächlich zur Ansteuerung von Laserdruckern benutzt wird [27].

Um unabhängig von der modellabhängigen Auflösung einer Seite eines Dokumentes zu sein, werden die von einer Computerapplikation zur Darstellung zu bringenden Elemente als vektorisierte Elemente innerhalb eines PostScript-Dokumentes kodiert. D.h. ein Kreis wird durch seinen Ursprung und seinen Durchmesser, ein Wort durch die Angabe des Schriftsatzes, der Größe und Position gespeichert. Dadurch kann die schlußendliche Repräsentation des Dokumentes erneut berechnet und beliebig skaliert werden.

In PostScript sind Elemente wie Text, Bilder (Rasterformate), Zeichnungen (vektorisierte Kreise, Rechtecke, Bogen etc.), Farben und komplette (leere) Seiten spezifizierbar. Alle diese Elemente können auf den Seiten positioniert, skaliert, gruppiert, in Tabellen oder verschiedenen Schichten überlappend oder durchsichtig angeordnet, sowie rotiert und projiziert werden.

PostScript-Dokumente können durch PostScript interpretierende Applikationen oder durch Hardware (z.B. in Laserdruckern) angezeigt werden. Die Erstellung der schlußendlichen Dateien ist sehr komplex und wird zumeist nur von Applikationen ermöglicht, die ihre Elemente intern auch vektorisiert speichern. Auch die Berechnung der dann auflösungsabhängigen Repräsentation erfordert ein bestimmtes Mindestmaß an Rechenleistung und Speicher. Mittlerweile wird, um weitgehende Hardwareunabhängigkeit bei Monitoren zu erreichen, softwareseitig das sogenannte Display-PostScript eingesetzt.

2.3.2 Bildformate

Grundsätzlich werden bei digital gespeicherten Bildern zwei Kodierungsarten verwendet: Pixel- und Vektorformate.

Pixelformate teilen ein Bild in Zeilen und Spalten auf und speichern den Farbwert jedes Bildpunktes. Vektorformate speichern die Parameter der im Bild enthaltenen graphischen Elemente, z.B. die Position und den Umfang eines Kreises, sowie die Breite der Linien und deren Farbe. Im Folgenden sollen ausschließlich Pixelformate betrachtet werden, da sie im Bereich der graphischen Darstellung am weitesten verbreitet sind. Sie sind plattformunabhängig definiert und

werden deshalb von verschiedenen Programmen auf jeder Plattform unterstützt. Weiterhin sind die ihnen zugrundeliegenden Algorithmen bekannt (im Gegensatz von zumeist von einzelnen Herstellern verwendeten Vektorformaten) und frei verfügbar.

Bei der Erklärung dieser Bildformate werden grundlegende Komprimierungsvorgänge erläutert, die insbesondere auch bei der Videokomprimierung Verwendung finden.

Die meisten Algorithmen zur Farbquantisierung arbeiten im RGB-Farbmodell. Es wird zumeist eingesetzt, um 24-bit Farbwerte (dies entspricht 16777216 verschiedenen Farben) zu beschreiben, sowie auf eine Palette mit 256 Farben abzubilden. Den RGB-Farbraum, der von einem kartesischen Koordinatensystem aufgespannt wird, kann man sich als Würfel vorstellen, der die Menge der auf dem Monitor darstellbaren Farben umspannt. Eine beliebige Farbe wird mit einem dreidimensionalen Vektor beschrieben, dessen Länge Auskunft über die Helligkeit (Luminanz) der Farbe gibt.

Der erste Schritt in der Farbkompression bzw. Farbreduktion ist oft eine uniforme Prequantisierung der Farbdaten von 24 bit auf 15 bit. Mit diesem Schritt wird die Datenflut eingedämmt, aber der Farbinformationsgehalt des Bildes, bezüglich unseres Farbsensorsystems Auge-Gehirn nicht merklich eingeschränkt, da nach Expertenmeinung der Mensch nur 50.000 bis 300.000 Farben unterscheiden kann. Nach der Prequantisierung wird die Häufigkeit, mit der die Farben im Bild vertreten sind, in Form eines dreidimensionalen Histogramms ermittelt. Bei Verwendung des Popularitätsalgorithmus wird nun aus den häufigsten Farben in der Verteilung die Farbpalette gebildet. Das Ergebnis ist aber in vielen Fällen nicht befriedigend, da mit dieser Methode, die ausgewählte Farbe in keinem Verhältnis zu der Anzahl der Bildpunkte steht, die mit dieser Farbe gesetzt werden. Eine Verbesserung der Histogramm-Auswertung wird mit der Verwendung des Median-Cut Algorithmus (auch abnehmende Quantisierung genannt erreicht.

Hierzu werden zuerst die Grenzen des RGB-Würfels so zusammengezogen, daß alle Bildpunkte und Farben des Bildes darin enthalten bleiben. Damit entsteht ein Quader oder ein Würfel als Spezialfall. Dieser Quader wird nun längs der längsten Kante geteilt, so daß in den entstehenden Subquadern etwa die gleiche Menge von Pixeln enthalten ist (Medianschnitte). Die Kontrahierung und weitere Teilung der enstandenen Quader wird nun solange fortgesetzt, bis die vorgegebene Menge von Subquaderen entstanden ist, oder die Subquader nicht mehr teilbar sind. Im Idealfall enthält ein Subquader nur noch einen Farbwert, der dann direkt einen Palettenwert darstellt. Da im Regelfall aber noch mehrere Farbwerte in einem Subquader enthalten sind, wird für jeden dieser Subquader ein Mittelwert errechnet.

Da sich gezeigt hat, daß das menschliche Auge Farbwerte in einer geringeren Auflösung als Helligkeitswerte wahrnimmt, wird der gängige RBG-Farbraum (Rot-Grün-Blau) bei den digitalen Bildstandards zumeist in drei *planes* umgewandelt, eine Luminanz-*plane* (Grauwerte, Y) und zwei Chrominanz-*planes* (Farbwerte U/V oder auch Cr/Cb).

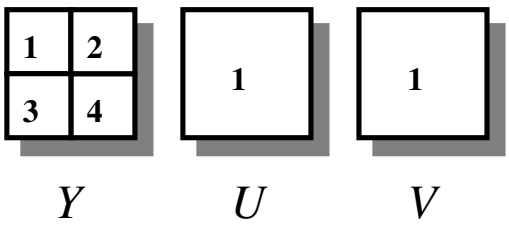


Abb. 6: *4:1:1 YUV-Farbraum*

Die Farbwert*planes* können dann wie z.B. bei JPEG (siehe Seite 20) und MPEG (siehe Seite 24) von ihrer Kantenlänge halbiert werden (das ist dann ein Viertel der Fläche !). Dies reduziert die Datenmenge enorm, ist jedoch ein Datenverlust. Diese Art der Kodierung nennt man den 4:1:1 YUV-Farbraum. Sie wird schematisch in der Abbildung 6 gezeigt.

BMP

BMP ist das unter dem Betriebsystem Windows 3.0 entstandene und durch die weiteren Versionen dieser Betriebssystemfamilie häufig verwendete Bildformat.

Das Format ist unabhängig vom verwendeten Ausgabegerät definiert (*device-independent*), es kann sowohl auf verschiedenen Bildschirmen oder Druckern ausgegeben werden und wird mittlerweile auch von Programmen anderer Betriebsysteme unterstützt. Das RGB-Farbmodell wird zugrunde gelegt. Durch geeigneten Aufbau von Farbtabellen ist jedoch auch die Kodierung von monochromen und von Graustufenbildern möglich.

Das Format definiert einen *header*- und einen Datenbereich. Ersterer wird BITMAPINFO genannt und enthält Angaben wie Größe, Farbtiefe, Farbtabelle und Kompressionsart. Der Datenbereich enthält nachfolgendend die Farbwerte jedes Punktes einer Linie, Linien werden bündig auf einen duch 32 teilbaren Wert verlängert und mit Null-Werten aufgefüllt.

Als Werte für die Farbtiefe sind 1, 4, 8 und 24 zugelassen. Der *Run-Length-Encoding* Algorithmus wird für Bilder mit einer Farbtiefe von 4 oder 8 bit/Bildpunkt angeboten, wobei jeweils zwei Bytes als Informationseinheit aufgefaßt werden. Enthält der erste Bytewert eine Null und ist der zweite Wert größer als drei, enthält der zweite die Anzahl der folgenden Bytes, die die Farbe des nächsten Bildpunktes als Verweis in die Farbtabelle enthalten (keine Kompression). Ansonsten bedeutet der erste Bytewert die Anzahl der nächsten Bildpunkte die mit der Farbe des zweite Bytewertes als Verweis in die Farbtabelle gesetzt werden sollen. Ist das Bild mit 4 bit/Bildpunkt kodiert werden auch nur vier Bit für diese Informationen benutzt, zwei Werte werden also in einem Bytewert kodiert.

BMP definiert weiterhin im *header* die Möglichkeit eine Farbtabelle anzugeben, deren Farben unbedingt zur Anzeige des Bildes benutzt werden müssen.

GIF

Das *Graphics Interchange Format* (GIF) wurde vom Online-Dienst CompuServe 1987 entwickelt, um den Austausch von Bildern über eben diesen Online-Dienst, unabhängig von der vorhandenen Hardware zu ermöglichen. GIF zeichnet sich durch seine verlustfreie aber doch recht effiziente Kompression und durch die mögliche Schachtelung von mehreren Bildern innerhalb einer Datei aus. Die Spezifikation liegt mittlerweile in zwei unkompatiblen Versionen (GIF87a und GIF89a) vor.

Grundsätzlich kann ein GIF-Bild aus den folgenden Abschnitten bestehen, wobei die letzten vier beliebig und wiederholt angeordnet werden können:

- *header***:** Enthält die GIF-Erkennungssequenz und die Versionsnummer des verwendeten Algorithmus. Sein Ende kennzeichnet den Anfang des Datenbereichs.

- *application*: Ermöglicht die Kodierung von Versions- und Namensinformation des Programmes, welches das GIF-Bild erzeugt oder bearbeitet hat.
- *trailer*: Markiert das Ende des GIF-Datenstroms.

- *control*: Dieser Abschnitt steuert die Darstellung jeweils eines folgenden *image* Blocks. Die Angabe von zeitlichen Verzögerungen, sowie der Farbe, die den Hintergrund durchsichtig erscheinen läßt, können angegeben werden.
- *image*: Ein *image* Block besteht jeweils aus einem *image header*, einer optionalen Farbtabelle und den Daten der Bildpunkte.
- *comment*: Textuelle Kommentare zu einem *image* Block.
- *plain text*: Dieser Block ermöglich die ASCII-basierte Kodierung von Texten innerhalb eines Bildes. Sowohl der Schriftsatz, die Farbe, die Größe, Position und die Streckung und Ausrichtung als auch der eigentliche Text können spezifiziert werden.

Ein GIF-Bild (*image block*) wird grundsätzlich als bit-Strom kodiert. Ein einzelnes Bild besteht wiederum aus dem *Logical Screen Descriptor* zur Kodierung von Größe, Position und Art der Farbtabelle des Bildes, optionalen globalen oder lokalen Farbtabellen (globale gelten für alle kommenden Bilder) und den Bildpunktfarben, die als Verweise in die Farbtabelle kodiert werden. Die Bildpunktfarben werden durch den LZW Algorithmus mit variabler Längenkodierung komprimiert. Durch diesen Algorithmus ist es möglich, sich wiederholende Bitmuster variabler Länge im Datenbereich zu lokalisieren. Während der Analyse wird ein *codetable* aufgebaut, der die auftretenden Bitmuster durch kurze Bitfolgen ersetzt. Die häufigsten Bitmuster werden dabei durch die kürzesten Bitfolgen repräsentiert (zur weiteren Erläuterung des LZW-Algorithmus siehe [23]).

Insbesondere durch den *plain text* und den *control* Block sowie durch die Schachtelung mehrerer Bilder eignet sich das GIF-Format besonders, um Bildabläufe und Animationen in kleinerem Rahmen zu kodieren. GIF ist nur für Bildformate mit einer 8-bit Farbtabelle definiert.

JPEG

Der internationale Standard der *Joint Picture Experts Group* (JPEG) (ISO/IEC SC29/WG10 ISO 10918) [15] ist ein Standard zur Kodierung von photographischen Standbildern. Der Basis-algorithmus beschreibt eine Transformationskodierung basierend auf der Diskreten-Cosinus-Transformation (DCT) im YUV-Farbraum.

Aus den zur Verfügung stehenden Transformationen hat sich die DCT (*Discrete Cosine Transform*, abgeleitet aus der Diskreten-Fourier-Tranformation) als besonders effizient erwiesen. Eine auf einen 8*8 Pixelblock angewendete DCT ergibt wiederum einen 8*8 Pixelblock. Die Koeffizienten der DCT lassen sich als Spektrum des 8*8 Eingabeblock interpretieren.

"Während die Energie des Bildsignals zufällig verteilt sein kann, konzentriert sich die Energie des korrespondierenden DCT-Blocks vorzugsweise auf Koeffizienten mit niedrigen Frequenzen" [31]. Werden die Koeffizienten im Zick-Zack durchnummeriert, ergeben sich als zu speichernde Werte ein DC-Koeffizient, dann wenige niedrige AC-Koeffizienten und viele AC-Koeffizienten nahe Null. Die DCT ist ein verlustfreies Verfahren, da die Kodierung komplett umkehrbar ist. Nun eignen sich diese Werte jedoch optimal um sie mit dem Huffman-Verfahren (eine Art Top-Down Shannon-Fano-Kodierung; häufige Bytewerte werden durch kurze Bitfolgen ersetzt) und anschließend nach dem Lauflängen-Verfahren zu kodieren (sollten sieben Nullen im Strom nacheinander folgen, braucht man nur die 7 und die 0 zu kodieren).

Ein JPEG-Bild setzt sich dann aus verschiedenen Bildkomponenten wie dem *picture-header*, Quantisierung- und Huffman-Tabellen, *frame-header*, *scan-header* (zur Aufteilung des Bildes in verschiedene Bereiche) und den Bilddaten zusammen. Die Kompressionrate des zu speichernden Bildes ist frei einstellbar. Innerhalb des JPEG-Verfahrens sind vier verschiedene Kompressionmodi möglich:

1. Basis-Verfahren (basiert wie hier beschrieben auf der Anwendung der DCT)
2. Erweitertes DCT-Verfahren (mit höherer, wahlweiser progressiver bit-Genauigkeit)
3. Verlustfreies Verfahren (nicht auf DCT-basierend)
4. Hierachie-Verfahren (Kombination von mehreren Bildern)

Die YUV-Farbwerte werden in einer, dem MPEG-Verfahren vergleichbaren Weise kodiert (Blockbildung, DCT, Entropy- und Huffman-Kodierung; siehe Seite 24). Das Verfahren ermöglicht ein effiziente Komprimierung von Bildern mit "natürlichem" Inhalt, es ist nicht besonders zur Speicherung von computergenerierten Bildern oder Grafiken geeignet, da die Farben von Objekten mit scharfen Objektkanten nur ungenügend genau kodiert werden.

Werden JPEG-Bilder aneinandergereiht, spricht man von Motion-JPEG (M-JPEG). Dieses Verfahren ist nicht international standardisiert, wird aber häufig bei der Videodigitalisierung benutzt, um die entstehenden Datenmengen im vorherein zu reduzieren. Da M-JPEG keine temporären Redundanzen ausnutzt, reicht die Komprimierung nicht aus, um sie zur Übertragung in lokalen Netzen zu nutzen.

PBM

PBMplus ist ein Softwarepaket, welches die Konvertierung von Bildern verschiedener Bildformate sowie die scriptbasierte Änderung digitalisierter Bilder ermöglicht. PBMplus wurde 1991 vom Autor Jef Poskanzer im Internet vorgestellt und hat dank seiner vielfältigen Einsetzbarkeit weite Verbreitung gefunden und ist zumindest auf Unix-basierten Workstations zum de-facto-Standard geworden.

Intern werden vier verschiedene Bildformate (PBM für farbige Bilder, PGM für Grauwertbilder, PPM für Echtfarbbilder und PNM zur formatunabhängigen Manipulation) angeboten. Diese können jeweils textuell (zur Einbindung in C-Programme, vergleiche XBM/XPM auf Seite 22) als auch binär kodiert werden. Da im Softwarepaket sowohl Konvertierungsprogramme für die internen Grafikformate, als auch für jedes sonstige Format mindestens ein Konvertierungsprogramm zu einem der internen Formate vorliegt, läßt sich somit jedes Format in jedes beliebige andere Format konvertieren und während der Konvertierung manipulieren.

Zum Manipulieren der internen Bildformate werden Filterprogramme für die folgenden Funktionen angeboten:

- Farbreduzierung, Quantisierung und Analyse der Farbwerte
- Veränderung von Kontrast, Helligkeit und Farbsättigung
- Beschneidung, Mischung von mehreren Bildern, Spiegelung des Bildes und Veränderung der Größe
- Erzeugung von Texturen und fraktalen Hintergründen
- Relief-, Schlier-, Ecken- und Mosaic-Filter

TIFF

Das *Tag Image File Format* (TIFF) wurde 1987 von der Aldus Corporation und Microsoft entwickelt, um die Portabilität und Hardwareunabhängigkeit eines Bildes bei der Kodierung zu berücksichtigen.

TIFF beschleunigt und vereinfacht die Verwendung und Ansteuerung von z.B. Scannern (Bildeinzugs- oder Bilddigitalisierungsgeräten) oder Faxgeräten, da lästige Umkodierung zwischen Geräten, Treibern und Bildapplikationen entfallen. TIFF ist weder eine Seitenbeschreibungs- noch eine Druckersprache, obwohl es oft auch dafür genutzt wird. TIFF ist nicht international standardisiert und liegt auch in mehreren, verschiedenen Varianten und Komplexitätsgraden vor.

Ein TIFF-Bild besteht aus den folgenden Abschnitten, die durch Markierungen (*tags*) voneinander getrennt werden:

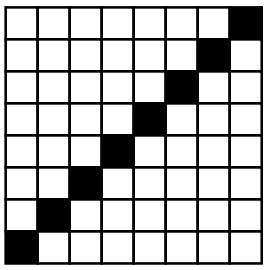
- *header, directory*: Enthält Angaben zur Byteorder, Versionsnummer, Verweise auf *directories* (welche jeweils Unterabschnitte bzw. einzelne Bilder enthalten).
- *structure:* Enthält Angaben über die verwendete Kodierungstechnik und die Anzahl der folgenden *tag-fields*.
- *fields:* Definieren die im folgenden Bild benutzten einzelnen Kodierungblöcke, wie Zeilen, Objekte, Zellen oder Blöcke und deren Eigenschaften wie die der verwendeten Kompression, Ausrichtung und Auflösung. Weiterhin können Elemente definiert werden, die das verwendete Seitenformat beschreiben.
- *data fields:* Die einzelnen graphischen Elemente des aktuellen Bildes werden dann in vorher nicht zu bestimmender Reihenfolge als *fields* deklariert.

Der entscheidene Unterschied zu den anderen Bildformaten besteht in dem generischen Ansatz. Generell kann man innerhalb des TIFF-Formates graphischen Inhalte auf verschiedene Arten kodieren, z.B. wird oft ein *preview*-Bild in einem zeilenorientiertem und unkomprimiertem Format zusätzlich zum folgenden, wesentlich komplexeren Bild kodiert, um ein schnelles Suchen innerhalb von Bildarchiven zu ermöglichen. Ein Ausage zur erreichten Kompression kann nicht getroffen werden, diese hängt zu sehr von der bildinternen Repräsentation ab.

XBM und XPM

X11-*Bitmap* (XBM) und X11-*Pixmap* (XPM) sind in der Unixwelt häufig verwendete Grafik-Formate, um Programmsymbole (*icons*) oder Hintergründe zu speichern. Sie ermöglichen die Definition von monochromen (XBM) oder farbigen (XPM) Bildern innerhalb von Programmcode der Programmiersprache C.

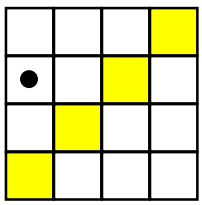
Im monochromen XBM-Format werden die Bildpunkte in einer Liste von Byte-Werten (*byte-array*) der Programmiersprache C kodiert, wobei jeweils 8 Bildpunkte zu einem Byte-Wert zusammengefaßt werden. Die Dimension des Bildes erfolgt durch zwei zusätzliche C-Definitionen. Die Abbildung 7 zeigt ein Beipiel.



```
#define demo_xbm_width 8
#define demo_xbm_height 8
static unsignedchar demo_xbm_bits [] = {
  0x80,
  0x40,
  0x20,
  0x10,
  0x08,
  0x02,
  0x01
};
```

Abb. 7: *Beispiel einer Grafik im XBM-Format*

Im XPM-Format werden die Bilddaten zusammen mit dem *header* in einer Liste von Zeichenketten (*string-array*) der Programmiersprache C kodiert. In der ersten Zeile wird die Dimension des Bildes und der sogenannte *hot-spot* [6] definiert. Die folgenden Zeilen beschreiben die verwendeten Farben, wobei jeweils eine textuell bzw. durch einen RGB-Farbwert angegebene Farbe durch ein Zeichen aus dem ASCII-Zeichenvorrat substituiert wird. Die Angabe einer Farbe, die den Hintergund durchscheinen läßt, wird durch das Symbolfolge "s None" ermöglicht. Die nachfolgende Zeilen listen die Zeilen des Bildes mit ihren substituierten Farbwerten auf. Die Abbildung 8 zeigt ein Beispiel.



```
static char *demo_xom[] = {
"4 4 1 2",
"     s None  c None",
"X    c yellow",
"   X",
"  X ",
" X  ",
"X    "};
```

Abb. 8: *Beispiel einer Grafik im XPM-Format*

Weder XPM nocht XPM kodierte Bilder werden komprimiert noch effizient gespeichert, die Repräsentation durch 8-bit ASCII-Werte erzeugt den gleichen Datenumfang, wie z.B. eine unkomprimiert gespeichert BMP-Datei. Auch hier können nur bis 256 verschiedene Farb- oder Grauwertstufen kodiert werden.

2.3.3 Video

Die digitale Darstellung eines Studio-TV-Signals erfordert gemäß der CCIR Recommendation 601 [5] eine Nettodatenrate von 166 Mbit/s. Keine der bereitstehenden digitalen Übertragungsmedien im Computerbereich kann eine solche Datenrate bieten. Mit den hier behandelten Standards können jedoch auch Video- und Audio-Signale auf bereits bestehende Medien übertragen und gespeichert werden, da sie die Menge der Daten durch verschiedenste Techniken reduzieren bzw. komprimieren.

---

[6] Ein *hot-spot* wird bei der Verwendung der Bildes als Cursor-Symbol benutzt, um den exakten Bildpunkt zu definieren, der die Selektion auslößt. Im XBM-Format wird der *hot-spot* durch zwei weitere C-Definitionen angegeben.

MPEG-1

MPEG steht für "Motion Picture Expert Group" und ist die Gruppe der ISO, die sich mit der Standardisierung im Videobereich beschäftigt. 1993 wurde ein internationaler Standard (ISO 11172) mit dem übersetzten Titel "Kodierung von Bewegtbildern und assoziierten Audio für digitale Speichermedien mit bis zu 1,5 Mbit/s" verabschiedet [16].

Dieser internationale Standard ISO 11172 (ISO/IEC JTC1/SC2/WC11) ist in drei Einzelstandards unterteilt:

- Video: dieser Teil beschreibt die Videokodierungs- und Kompressionstechniken
- Audio: dieser Teil beschreibt die Audiokomprimierung mit Hilfe des sogenannten psychoakustischen Modells
- System: dieser Teil behandelt die Synchronisation und das *Multiplexing* von Video- und Audiodatenströmen

Video- und Audioteil werden hier explizit vorgestellt.

Ein MPEG-Video-Strom wird durch Aneinanderreihung von *intra-* (I), *predicted-* (P), und *bidirectional-* (B) *frames* [7] beschrieben. I-*frames* sind in einem Format abgelegt, dessen Techniken grundsätzlich mit denen des JPEG-Formats (siehe Seite 20) übereinstimmen, in P-*frames* werden nur die Differenzen zu einem I-*frame* kodiert, ein B-*frame* kodiert die mittleren Differenzen eines I- und eines P-*frames* (siehe Abbildung 9).

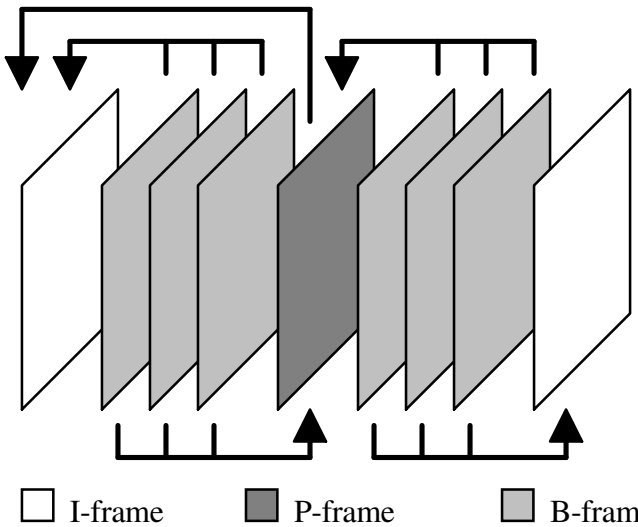


Abb. 9: *Abhängigkeiten der frame-Typen eines MPEG-1 Videostroms*

Jeder *frame* wird wiederum in drei *planes*, eine Luminanz-*plane* (Grauwerte, Y) und zwei Chrominanz-*planes* (Farbwerte, Cr und Cb), zerlegt. Alle *planes* werden in sogenannte Makroblöcke unterteilt. Alle *planes* werden in 8*8 Pixel-Blöcke aufgeteilt. Jeder dieser Blöcke wird mittels einer Diskreten-Cosinus-Transformation (DCT) kodiert, dabei wird der erste Wert jedes erechneten Blocks als DC-Koeffizient (DC) bezeichnet; die restlichen Differenzwerte als AC-Koeffizienten (AC). DCs werden untereinander auch als Differenzen gespeichert.

---

[7] Als *frames* werden bei den Videoformaten Einzelbilder verstanden. Um ein flüssiges Bild zu erhalten, ist die Anzahl der Einzelbilder pro Sekunde (*frame rate*) entscheidend. In der europäischen Fernsehwelt sind z.B. 25 Einzelbilder pro Sekunde, in der amerikanischen sogar 30 Bilder pro Sekunde üblich.

Zusätzlich können *frames* auch als Verweise der Makroblöcke auf gleiche, in vorherigen *frames* gespeicherte Blöcke kodiert werden (hier bezieht man sich jedoch auf 16*16 Pixel-Blöcke). Dieses Verfahren nennt man *motion compensation*. Im Gegensatz zu JPEG werden in MPEG feste Quantisierungs- und Huffmann-Tabellen verwendet, diese sind zwar nicht für jede Video-Übertragung optimal, können dann jedoch auch in Hardware kodiert werden. Dies ist unerläßlich für die Fernsehtechnik. Die Bildgröße ist variabel (in 16-Pixel-Schritten) und kann maximal horizontal 720 *pels* und vertikal 576 *pels* betragen, dies ist die Auflösung eines normalen S-VHS Signals.

Durch das Kodierungsverfahren sind Einzelbildzugriff, schnelles Suchen (Positionierung auf I-*frames*) und die Editierbarkeit des Datenstroms gegeben.

MPEG wurde durch die Wahl der Kodierung auf asymmetrische Anwendungen (Fernseh-Technik, Multimedia-Mail, etc.) zugeschnitten, mit größerem Hardware-Aufwand können jedoch auch symmetrische Verfahren implementiert werden. Insbesondere benötigt die Dekodierung im Computerbereich kaum Hardwareunterstützung.

MPEG ist in Bezug auf die Komprimierung und die zur Verfügung gestellten Kombinations- und Editierungsmöglichkeiten keine optimale Videokodierung, da viele Parameter wie z.B. Bildgröße, die Anzahl der möglichen Spuren und die verwendeten Huffman-Tabellen begrenzt bzw. fest vorgegeben werden. Allerdings ist MPEG der einzige internationale Konsens in diesem Bereich und findet, auch dank seiner Verwandtschaft mit den Telekommunikations-Standards (H.261, S-VHS), weite Verbreitung.

Bei der geplanten Integration des Formates MPEG in das PCSS-System wird sich die Kombination von Audio- und Video-Daten sowohl bei der Aufnahme als auch bei der Wiedergabe als sehr problematisch erweisen. Wenn es mit der nächsten Software-Version der SunVideo-Karte nicht möglich ist, Audiosignale synchron zum Video aufzunehmen, müssen die Audiodaten separat synchron aufgenommen werden. Ist eine kombinierte Aufnahme mit der SunVideo-Software nicht möglich, so müssen die Daten anschließend getrennt in zwei Dateien in den Message-Store übertragen oder vorher ineinander geschachtelt (*multiplexed*) werden.

MPEG-2

Im November 1994 wurde der internationale Standard MPEG-2 (ISO 13818) mit dem übersetzten Titel "Generische Kodierung von Bewegtbildern und synchronisiertem Audio" vorgelegt. [17]

MPEG-2 ist keine Weiterentwicklung des bereits bestehenden Standards MPEG-1 im technischen Sinne. Eine Entwicklung des Standards MPEG-2 heißt nicht, daß MPEG-1 nun veraltet wäre. MPEG-2 ist nur eine komplexere, umfassendere Kodierung, geschaffen für ein vollständig anderes Anwendungsprofil. Die verwendeten Mechanismen von MPEG-2 sind jedoch sehr ähnlich zu MPEG-1. MPEG-2 wird die Erweiterung von MPEG-1 werden, die Rücksicht auf die hardwaretechnischen Neuerungen und Erfordernisse nimmt.

Besonders ist hierbei die neu hinzugefügte Funktionalität der "Scalability" hervorzuheben, die ähnlich dem Standard *Open Document Architecture* (ODA), ein der Hardware angepaßtes Konsumieren von Videos ermöglichen wird. Insbesondere wird dadurch bei einem für HDTV

(zukünftiges hochauflösendes Fernsehen) kodierten 16:9-Bild die Abwärtskompatibilität zum herkömmlichen 4:3 Fernsehbild garantiert.

MPEG-2 verfolgt genau wie MPEG-1 das grundlegende Prinzip des ungleichen Enkodier- und Dekodierverhältnisses (Studio-/Home-Hardware), um die Kodierung zu optimieren. Folgende grundlegende Anforderungen werden durch das Format erfüllt:

- Innerhalb der Bereiche *Satellite Broadcast*, *Electronic Cinema* und *Digital Home Television* wird eine angepaßte Bandbreite < 3 Mbit/s geboten.

- Der Videoteil der Kodierung muß eine erhebliche Flexibilität aufweisen. Unter anderem müssen verschiedene Bildformate, eine wahlfreie Bildqualität, flexible Bitraten, wahlfreier Zugriff auf verschiedene Video-Kanäle (sogenanntes *channel hopping*), eine nachträgliche und einfache Editierung des kodierten Bitstreams und verschiedenste *trick modes* (für z.B. effektreiche Überblendungen) garantiert werden. Eine Wiederholung des Kodier-/Dekodiervorgangs darf nicht zu weiteren Qualitätsverlusten führen.

- Der Audioteil der Kodierung muß mehrere multilinguale Kanäle und niedrigere Sampling-Frequenzen unterstützen.

Auch der MPEG-2-Standard besteht wiederum aus drei Hauptteilen, nämlich dem Video-, dem Audio- und dem Systemteil. Im Standard selber wird zwar ein weiterer, vierter *Conformance*-Teil (der die Charakteristiken von MPEG-2-Bitstreams und Testergebnisse enthalten wird) erwähnt, er wurde jedoch bis jetzt noch nicht publiziert.

Drei grundlegende neue Kodierungsmethoden wurden in MPEG-2 beschrieben:

Profiles und Levels

Es werden sogenannte *Profiles* und *Levels* definiert. Diese schränken die zur Verfügung stehenden Parameter der Kodierung ein, um diese Einschränkungen dann in den Kompressionsalgorithmen ausnutzen zu können. Gleichzeitig standardisieren die *Profiles* und *Levels* auch die Kodierungsparameter.

Die Kombination von *Profiles* (*complexity of compression*) und *Levels* (*sample rate, frame dimension, coded bitrates*) läßt dabei aber immer genügend Spielraum für die unterschiedlichsten Anwendungen. Die in Tabelle 1 und 2 aufgelisteten *Profiles* und *Levels* sind bis jetzt im Standard festgeschrieben. Tabelle 3 zeigt eine Auswahl der möglichen *Sampling*raten.

Profile	**B pictures?**	**Syntax Comments**
Simple	No	Intended for software applications, perhaps CATV.
Main	Yes	Most decoder chips, CATV, satellite. 95% of users.
Main+	Yes	Main with Spatial scalability.
Next	Yes	Main+ with 4:2:2 marcoblocks.

Tab. 1: *MPEG-2 Profiles*

Level	Max. frame sample size	Pixels/ sec	Max. bitrate	Significance
Low	352 x 288	3.05 M	4 Mb/s	CIF, consumer tape equiv.
Main	720 x 480	10.40 M	15 Mb/s	CCIR 601, studio TV
High 1440	1440 x 1152	47.00 M	60 Mb/s	4x 601, consumer HDTV
High	1920 x 1080	62.70 M	80 Mb/s	production SMPTE 240M std

Tab. 2: *MPEG-2 Levels*

Dimensions	Coded rate	Comments
352x480x24 Hz (progressive)	2 Mbit/sec	Half horizontal 601. Looks almost NTSC broadcast quality, and is a good (better) substitute for VHS. Intended for film src.
544x480x30 Hz (interlaced)	4 Mbit/sec	PAL broadcast quality (nearly full capture of 5.4 MHz luminance carrier). Also 4:3 image dimensions windowed within 720 sample/line 16:9 aspect ratio via pan&scan.
704x480x30 Hz	6 Mbit/sec	Full CCIR 601 sampling dimensions.

Tab. 3: *Sampling Größen und Bitraten*

Scalability

Die *Scalability* von Video und Audio wurde verwirklicht. Sie liefert die Möglichkeit von unterschiedlicher Endnutzung bei ein- und derselben feststehenden Kodierung. So kann der Endnutzer entscheiden, welche Teile einer Übertragung er konsumieren möchte. "[Scalability is] the ability of a decoder to ignore some portions of a total bitstream and produce useful audio and video output from the portion which is decoded." [7] Dadurch wird MPEG-2 weitgehend speichermedienunabhängig. Innerhalb der Einzelbild- *Scalability* lassen sich z.B. die beiden folgenden Modi realisieren (andere Kombinationen sind natürlich möglich):

Erstens gibt es die zeitliche Scalability (siehe Abbildung 10), d.h. sollten die Ressourcen des konsumierenden Systems ausreichen, kann die Bildrate erhöht werden, indem in den normalen Ablauf des "Base Layers" zusätzliche B-*frames* des "Enhancement Layers" dekodiert werden.

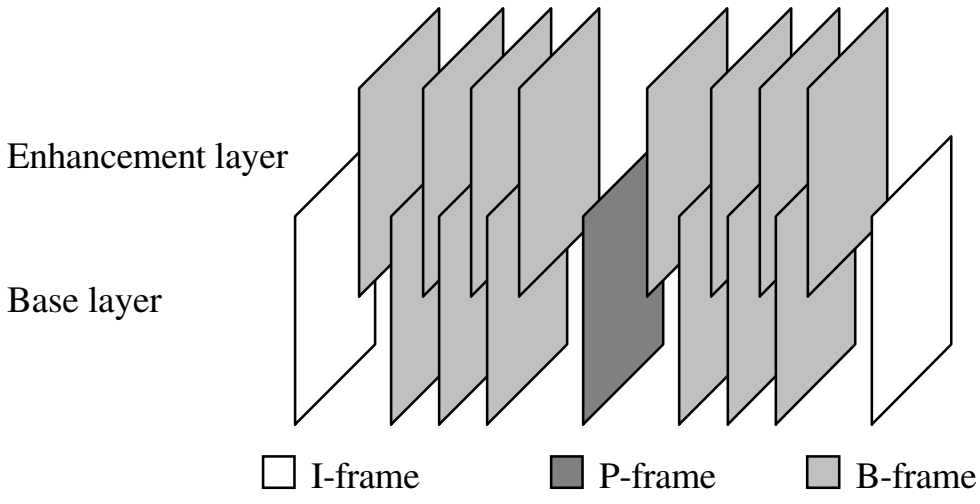


Abb. 10: *Zeitliche Scalibility*

Zweitens wird die Qualitäts-*Scalability* definiert (siehe Abbildung 11), d.h. anstelle der ungenaueren B-*frames* des "Base Layers" werden P-*frames* des "Enhancement Layers" dekodiert und angezeigt. Dies erfordert natürlich mehr Rechenzeit.

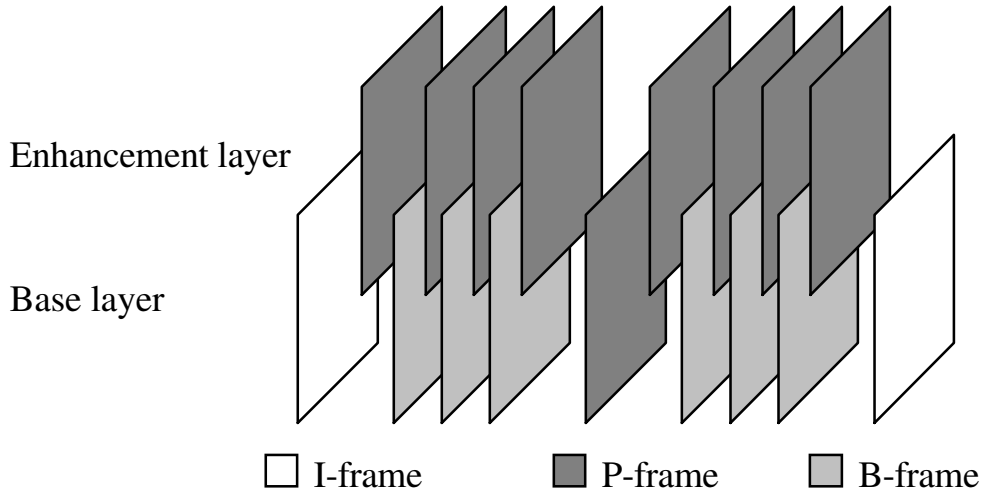


Abb. 11: *Qualitäts-Scalibility*

Zusätzlich sind noch die Kodierung von 3-dimensionalen Bildern (*stereoscopic view*) und die sichere oder geheime Übertragung von Teilströmen mit der *Scalability*-Funktion möglich.

Weiterhin wird noch die Makroblock-*Scalability* definiert. Dabei handelt es sich um einen zusätzlichen Eintrag in jedem Makroblock, der auf den nächsten "wichtigen" (mit vielen Änderungen gegenüber dem letzten Bild) Makroblock verweist, sollte also die Dekodierzeit nicht ausreichen, können "unwichtige" Makroblöcke übersprungen werden.

Schließlich gibt es noch die *Pan-Scan-Scalability* (siehe Abbildung 12). Diese ermöglicht die Definition von Ausschnitten im aktuellen Bild. Diese Ausschnitte können zwar von Bild zu Bild unterschiedlich eingeteilt werden, die Hauptanwendung ist jedoch die Definition eines 4:3 Fernsehbildes innerhalb eines 16:9 HDTV-Bildes.

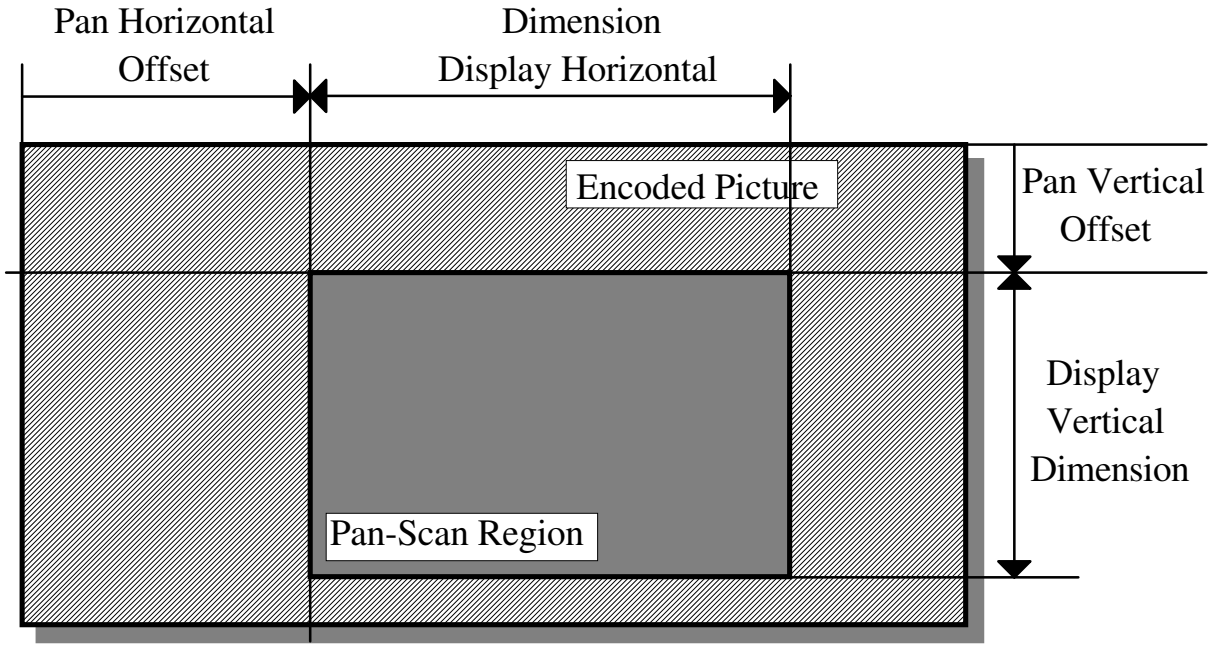


Abb. 12: *Pan-Scan-Scalibility*

Sicherheit

MPEG-2 ist der erste der vielen Multimedia-Video-Definitionen- und -Standards, in dem auch der Bereich der Sicherheit (Vertraulichkeit und Integrität) mit konzipiert wurde.

MPEG-1 hatte schon im Audio-Strom einen rudimentären CRC-check eingeführt. MPEG-2 ist robust gegenüber Bitfehlern und Zellenverlusten. Ein speziell für MPEG-2 optimierter, schneller Cypher-Algorithmus ist beschrieben, der ein *Scrambling* des kompletten Datenstroms ermöglicht.

Kodierungstechniken

MPEG-2 soll laut Anwendungsprofil auf alle Eventualitäten in der digitalen Video-, Übertragungs- und Fernsehtechnik vorbereitet sein. Ein wichtiges *feature* ist daher die Rückwärtskompatibilität zu existierenden digitalen Kodierungen wie MPEG-1 oder H.261. Diese Rückwärtskompatibilität bedeutet keine Bitübereinstimmung der resultierenden Datenströme, sondern sorgt über die Benutzung und Variierung der *Profiles* und *Levels* nur dafür, daß Hard- und Software die z.B. zum Einsatz von MPEG-1 bestimmt war, auch mit MPEG-2 (mit reduzierten Möglichkeiten dann natürlich, denn MPEG-2 ist von der benötigten Rechenleistung zur Dekodierung viel aufwendiger) zu betreiben ist. Allerdings sind auch sogenannte Sub-Koder innerhalb eines MPEG-2-Koders realisiert, die exakte, rückwärtskompatible Datenströme erzeugen.

Z.B. ergibt die Kombination von *Video Main Profile* (wie MPEG-1) und *Video Main Level* (CCIR 601, Studio TV) eine MPEG-1-ähnliche Kodierung, die abgekürzt als MP@ML bezeichnet wird.

Eine erweiterte Komprimierungsmethode mit dem Namen *Block-based Motion Compression Prediction* (MCP) wird in MPEG-2 benutzt. Sie ermöglicht *Frame Motion Prediction* (das Auffinden von ganzen, gleichen Frames innerhalb des Datenstroms), *Field Motion Prediction* (das Auffinden von gleichen, wie auch immer geformten Teilen in verschiedenen Bildern und auch innerhalb des aktuellen Bildes) und das zeitliche Vorhersehen von Kodierungsfehlern. Zusätzlich werden sogenannte *Backward Motion Vectors* eingeführt, in MPEG-1 waren Bewegungsvektoren immer nur auf vorherige Bilder gerichtet. Bewegungsvektoren in MPEG-2 können auch genau bis auf einen halben Bildpunkt innerhalb des Bildes bestimmt werden. Dies ermöglicht das Auffinden von langsamen Verschiebungen.

Diese Methoden ermöglichen zwar eine wesentliche genauere Analyse der Videodaten, erhöhen den benötigten Rechenaufwand jedoch enorm.

Zusätzliche Funktionen des MPEG-2-Videoteils sind (neben den oben und vorher genannten):

- variable Makroblockgrößen
- *downloadable length coding tables*
- Huffman oder JPEG-Methoden zur Variablen-Kodierung
- *quant matrix download extension*
- *interlace prediction and compression extension*
- alternative Scanmethoden für "Interlaced Video"
- Trick Modi durch Ausnutzung der *Scalability*

Um die unterschiedlichsten Anwendungsprofile zu unterstützen wird auch ein variabler Farbraum eingeführt. 4:1:1 wird wegen MPEG-1 weiterhin unterstützt, für das *Broadcasting* (Übertragung von Fernsehbildern zwischen verschiedenen Sendern) und dem professionellen Einsatz von digitaler Studiotechnik ist jedoch ein Farbverlust nur in geringem Maße oder gar nicht zu akzeptieren. Deshalb werden auch die Farbräume 4:2:2 und 4:4:4 unterstützt.

H.261

"Der H.261-Standard kann zur Kompression von Bildsequenzen für Videokonferenzen oder für das Bildfernsehen verwendet werden." [31].

Als Übertragungsmedium für Bildtelefon und Videokonferenzen dient ISDN mit p*64 kbit/s [8] , der Datenstrom ist nicht byteorientiert und wird durch eine *forward-error-correction* (BCH) geschützt, die es ermöglicht, 2-bit-Fehler zu beheben und 3-bit-Fehler zu erkennen. Die Wahrscheinlichkeit eines 1-bit-Fehlers bei ISDN ist entsprechend geringer.

Als Einzelbild werden das sogenannte *Common Interchange Format* (CIF), oder das *Quarter-CIF* (QCIF) verwendet. Diese bestehen aus 352 Punkten in 288 Linien. Bei einer gewünschten Übertragung von 30 Bildern/s der NTSC-Norm über einen ISDN-B-Kanal, muß im QCIF-Format immerhin noch eine Kompression von 142,5:1 erreicht werden.

Die einzelnen *frames* werden in ihre Farbseiten des YUV-Farbmodells (wie bei MPEG) zerlegt und dann als Blöcke übertragen, für jeden wird einzeln entschieden, ob er übertragen wird und wenn ja, ob er einer Transformkodierung, sprich Intrakodierung wie z.B. bei MPEG und/oder *Motion Compensation*, unterzogen wird. Die restlichen Blöcke werden einer DCT-Transformation und anschließend noch einer speziellen *entropy*-Kodierung unterzogen. Dazu werden spezielle Tabellen benutzt, die es ermöglichen, daß der sogenannte *picture start code* (PSC) nicht innerhalb eines *frames* erscheinen kann.

Zusammengefaßt hat H.261 folgende Charakteristika:

- Standard zur Übertragung von digitalen Video-Sequenzen der CCITT
- p*64 kbit/s
- bereits als Hardware vorhanden
- synchrones Kodierungs-/Dekodierungsverhältnis
- einfache Umsetzung RGB-YUV
- Intra-Frame-Kodierung (einschließlich einer Vorhersage zu erwartender Übertragungsfehler)
- auch DCT und Huffmann

H.261 wird bei Übertragungen über ISDN (oder lokale Netzwerke) zumeist von dem Audioformat G.711 begleitet.

SMP/G.711

Software Motion Pictures (SMP) wurde von der Digital Equipment Corporation (DEC) entwickelt und ist ein Kompressionsalgorithmus sowie eine komplette Softwareimplementierung für digitales Video [32].

[8] *p* bezeichnet dabei die Anzahl der verwendeten ISDN-B-Kanäle (normalerweise zwischen 1 und 32).

Maßgeblich für die Entwicklung von SMP war eine Abschätzung zwischen den entstehenden Kosten und der allgemeinen Verfügbarkeit von Ausgabeeinheiten. Das Ziel war, eine Methode zu entwickeln, die es ermöglicht, auf nahezu jedem Desktop Computer ohne zusätzliche Dekompressionshardware digitales Video abspielen zu können. Die existierenden Standards für digitales Video verlangen eine hohe Rechenleistung, die von den meisten Endgeräten nur mit zusätzlicher Dekompressionshardware und Bildspeicher erbracht werden kann; die benötigten Datenraten sind so hoch, daß nur noch Hochgeschwindigkeitsnetzwerke die Datenflut bewältigen können.

Der Kompressionsalgorithmus wurde speziell für eine schnelle Softwaredekompression konstruiert und kann in alle verbreiteten Window-Systeme integriert werden. SMP generiert deterministische Datenraten, was bei einem Einsatz von CD-ROM Laufwerken und den verwendeten Netzwerken sehr vorteilhaft ist. Im Gegensatz zu den existierenden Standards für digitales Video werden bei SMP die einzelnen Bilder nur mit einer Farbtiefe von 8 bit/Bildpunkt und mit 15 Bilder/s kodiert. Eine typische Applikation von SMP auf einer *low-end workstation* kann digitales Video mit einer Auflösung von 320 * 240 Bildpunkten und einer Datenrate von 1.1 Mbit abspielen. Die Datenmenge für eine Sekunde kodiertes Video beträgt zwischen 135 und 145 kBytes, für die Audiokomponente sind 8 bis 15 kBytes vorgesehen.

SMP-Kompressionsalgorithmus

Der SMP-Algorithmus ist eine pixel-basierte, mit Verlust behaftete Methode, die für eine minimale CPU-Belastung ausgelegt ist. Die Eigenschaften sind eine für Kommunikationsanwendungen akzeptable Bildqualität, eine mittlere Kompressionsrate und eine feste Datenrate. Es werden keine rechnerisch aufwendigen Transformationen oder Entropiekodierungen vorgenommen.

Der SMP-Kompressor arbeitet in fünf Phasen:

- Skalierung des Eingabeformats
- *Block Truncation* Coding (BTC) oder *Color Cell Compression* (CCC)
- Selektion von *flat* und *structured blocks*
- Farbquantisierung
- Kodierung des SMP-Clips

Die Skalierung des Eingabeformats ist genaugenommen kein Teil des Algorithmus, kann aber bei verschiedenen Formaten notwendig werden, um auf ein Vielfaches der Blockgröße zu kommen. Die berechneten Werte für die Luminanz- und Farbwerte werden gespeichert, um in den folgenden Phasen Verwendung zu finden. Um die Kompressionsrate zu erhöhen, wird bei den Blöcken ein festes *Subsampling* vorgenommen. Die horizontale Auflösung eines Blocks wird halbiert, indem die Farbwerte von zwei nebeneinanderliegenden Bildpunkten durch Mittelwertbildung zusammengefasst werden. Damit reduziert sich die Anzahl der benötigten Bits für die Maske von 16 bit auf 8 bit, so daß pro Block ein Byte eingespart wird. Diese Reduktion hat zur Folge, daß vertikale Kanten verschwommen dargestellt werden, was aber für Videobilder vertretbar erscheint.

Block Truncation Coding (BTC) ist eine Kompressionstechnik für Grauwertbilder. Das Ausgangsbild wird zuerst in Blöcke von 4*4 Bildpunkten zerlegt, danach wird für jeden Block ein Mittelwert der Bildpunktwerte gebildet. Dieser dient anschliessend als Schwellwert, um die Bildpunkte in zwei Gruppen einzuordnen. Für jede der beiden Gruppen von Bildpunkten wird nun erneut ein Mittelwert berechnet, der untere und der obere Durchschnittswert wird jeweils durch

einen 8-bit Grauwert repräsentiert, so daß der durchschnittliche Luminanz- und Kontrastwert der Bildpunkte erhalten bleibt. Durch die Bitmaske wird die Struktur des Blocks bewahrt. Mit dieser Kodierungsmethode wird ein Block aus 16 Pixeln mit einer 16-bit Maske und zwei 8-bit Werten zusammen mit 32 Bit kodiert, dies entspricht einer Rate von 2 Bit pro Pixel.

Wie beim BTC arbeitet der *Color Cell Compression* (CCC)-Algorithmus auf der Basis von 4*4 Blöcken. Für die Segmentierung der Bildpunkte in die zwei Gruppen wird auch in diesem Verfahren die Luminanzwerte der einzelnen Bildpunkte zugrunde gelegt. Die Berechnungen der Farbwerte für die beiden Bildpunktgruppen erfolgt nun für die Rot-, Grün- und Blaukomponenten seperat. Für diese Berechnung und auch für die folgenden wird das Ausgangsbild im 24-bit RGB Farbmodel benötigt. Da bei diesem Algorithmus zum einem das RGB-Farbmodell, zum anderen der Luminanzanteil eines Bildpunktes Verwendung findet, ist ein Umrechnen in ein anderes Farbmodell (z.B. in den 4:1:1 YUV-Farbraum, siehe Seite 18) notwendig.

Nach der Farbquantisierung mittels des *Median-Cut Algorithmus* werden für die Kodierung eines Blocks noch drei Bytes benötigt (eine 8-bit Maske und zwei 8-bit Indizes in die generierte Farbpalette), dies entspricht 1,5 bit/Bildpunkt. Um auf die angestrebte Kompressionsrate von 1 bit/Bildpunkt zu kommen, wird noch eine weiteres adaptives Subsamplingverfahren angewandt. Blöcke mit einem geringen Abstand der Luminanzwerte für die Vordergrund- und Hintergrundfarbe (dies kann auch als Kontrast interpretiert werden), werden nur noch durch einen Farbwert und ohne Maske repräsentiert. Mit der Einführung einer zweiten Ausprägung für einen Block muß eine weitere Kodierungsinformation hinzugefügt werden. Jedem Block wird ein Bit einer Bitmap zugeordnet, welches anzeigt, ob ein normaler CCC-Block (*structured*) oder ein *subsampled* Block (*flat*) vorliegt. Die Einteilung in *structured* und *flat* Blöcke ist abhängig von der gewählten Datenrate. Ausgehend von diesem Wert wird berechnet, wieviele normale Blöcke verwendet werden können, weitere Berechnungen sind nicht notwendig, da für diese Einteilung die Luminanzwerte benutzt werden. Bei gleicher Anzahl der verschiedenen Blocktypen ergibt sich eine Kompressionsrate von 1 bit/Bildpunkt, die Anzahl der Bits für die Farbpalette und die Informationen über den Blocktyp sind nicht berücksichtigt.

Abschließend werden die Vordergrund- und Hintergrundfarben der einzeln Blöcke so gut wie möglich auf Einträge der erzeugten Farbpalette der Quantisierungsphase abgebildet und der Ausgabestrom geschrieben.

Wird nur mit Grauwertbildern gearbeitet, entfallen die CCC, die Farbquantisierung, sowie das Suchen der geeigneten Farben in der Palette. Bei leistungsfähigen Rechnern ist eine Kompression von Grauwertbildern in Echtzeit möglich. Eine Softwarekodierung von Farbvideosequenzen in Echtzeit ist erheblich aufwendiger. Für den kompletten Kodiervorgang, von der Skalierung bis zum Schreiben der SMP-Daten, werden 220 Maschineninstruktionen auf einem MIPS R3000 Rechner für jeden Bildpunkt benötigt, im Gegensatz dazu nur 8 Instruktionen bei einem Grauwertbildpunkt.

Dekompression

Eine Zielsetzung bei der Entwicklung von SMP war die Möglichkeit eine Softwaredekompression in Echtzeit auf einer breiten Basis von Rechnerarchitekturen zu erreichen [9]. Bei der Dekompression wird nur der inverse Schritt der letzten Kodierungsphase ausgeführt. Die Wiederherstellung der 4 * 4 Pixelblöcke mit zwei Farben und einer Maske erfordert nur eine geringe Belastung der CPU, da keine Berechnungen ausgeführt werden und Größe und Ausrichtung bekannt sind. Der weitaus

[9] Selbst auf einem PC mit 386-Prozessor und einer einfachen VGA-Grafikkarte sind mehr aus 15 Bilder/s möglich.

höhere Anteil der Arbeit liegt in dem Kopieren der dekodierten Daten in den Bildschirmspeicher und der Anpassung an die Kapazitäten der Ausgabeeinheit (90 % der CPU-Zeit bei einem 24-bit Display). Bei der Verwendung von 8-bit Ausgabeeinheiten ergeben sich zwei Probleme.

1. Es können bei keinem Window-System alle 256 Farben für eine Applikation reserviert werden.
2. Das Setzen der Farbtabelle und das Laden eines Bildes wird nicht gleichzeitig ausgeführt.

Diese Probleme lassen sich nicht im Ganzen für alle Window-Systeme gleichzeitig lösen, sondern nur speziell für einzelne Systeme. Dies steht aber im Widerspruch zur Unabhängigkeit vom Ausgabegerät.

Ein SMP-Clip setzt sich zusammen aus einem Datei*header* und den Einzelbildern. Der Dateiheader enthält Informationen über die Anzahl, Dimension und Größe der Einzelbilder, sowie die Anzahl der Bilder pro Sekunde. Die einzelnen Bilder haben eine feste Länge und somit auch einen festgelegten Abstand zum folgenden Bild, damit ist es nicht nötig, zusätzliche Informationen, wie z.B. Marken für den Anfang eines Bildes, zu kodieren. Ein einzelner *frame* ist wie folgt aufgebaut. Das erste Byte gibt die Anzahl der Farben in der darauf folgenden Farbtabelle an. Ist der Wert 0, so handelt es sich um ein Graustufenbild. In diesem Fall wird die Grauwertpalette nicht kodiert, die Werte für Vorder- und Hintergrund der Blöcke sind dann direkt Luminanzwerte.

Zur Übertragung von Video und Ton wird das Format SMP zumeist mit einem Audiodatenstrom im Format G.711 synchonisiert. Dieses wird weiter unten erläutert.

2.3.4 Audio

Die Qualität einer digitalisierten Audiosequenz hängt von der Auflösung der einzelnen *samples* [10] und der Anzahl der Abtastungen in einem Zeitinterval ab. Je besser die Quantisierungsauflösung gewählt wird, umso geringer wird der Rauschanteil der Sequenz gegenüber dem Nutzsignal und umso mehr Daten müssen pro *sample* verarbeitet werden. Durch mehr Abtastungen pro Zeitinterval nähert sich das digitale Signal besser dem analogen Ausgangssignal an. Mit heutiger Technologie ist es problemlos möglich, die Unterscheidungsverzögerung des menschlichen Gehörs zu übertreffen. Für eine grundlegende Beschreibung von Audiosignalen wird auf [18] verwiesen.

MPEG-1

Der Audio-Teil des MPEG-1 Standards beschreibt Mechanismen und Algorithmen, mit denen die digitale Speicherung von Audiosignalen auf kostengünstigen Speichermedien auf der einen Seite und die digitale Übertragung von Audiosignalen auf Kanälen mit begrenzter Kapazität auf der anderen Seite ermöglicht wird.

Bei diesem Zielstreben steht jedoch die Erhaltung der Qualität in einem bestimmten Bereich im Vordergrund. Im Audiobereich wird eine der *Compact Disc* (CD) nahekommende Qualität erreicht. MPEG wird zumeist ausschließlich im Zusammenhang mit der Videokomprimierung gesehen. Dabei wird oft übersehen, daß die eigentliche Multimedia-Revolution soeben im Audiobereich stattgefunden hat. Die Übertragung von Musik und Sprache über z.B. LANs und die

[10] Als s*ample* wird ein einzelner bei der Digitaliserung anfallender Wert bezeichnet. Ein *sample* ist die digitale Entsprechung eines analogen Signals zu einem bestimmten Zeitpunkt.

Speicherung von mehr als 10 Stunden Audio auf einer CD-ROM, und das alles in CD-Qualität, kann nicht hoch genug eingeschätzt werden. Die Übertragung übers Internet soll hier später besprochen und realisiert werden.

Bei einem zu erzielenden Datendurchsatz von 1,5 Mbit/s werden maximal 386 kbit/s für den Audiobereich verwandt.

Die Darstellung eines stereophonen Audiosignals im Studioformat erfordert eine Abtastfrequenz von 48 kHz und eine gleichförmige Quantisierung von 16 Bit pro Abtastwert. Daraus ergibt sich eine Datenrate von 768 kbit/s für ein Monosignal, als Produkt der Multiplikation der 48 kHz mit den 16 bit pro Abtastwert. Daraus resultierend ergibt sich für ein Stereosignal eine Datenrate von 2x768 kbit/s, also ca. 1,5 Mbit/s. Als Vergleich dazu wird auf einer CD mit einer Abtastfrequenz von 44,1 kHz bei der gleichen Quantisierung von 16 bit pro Abtastwert gearbeitet, wodurch sich eine Datenrate von ca. 706 kbit/s (Mono) ergibt.

Im MPEG-Audio-Standard werden drei Abtastfrequenzen verwendet, 32, 44.1 und 48 kHz. Aber im Gegensatz zu den oben beschriebenen Fällen ergeben sich hier im Endeffekt Datenraten zwischen 32 kbit/s und 192 kbit/s für ein Monosignal. Für ein Stereosignal liegen sie zwischen 128 kbit/s und 384 kbit/s. Mit einer Datenrate unter 112 kbit/s (mono 56 kbit/s) können leider noch keine zufriedenstellenden Ergebnisse erzielt werden. Diese untere Grenze wird im Bereich der Video-CD (VCD) verwandt.

Das Ziel des Standards ist es, mit einer von 1,5 Mbit/s im Studioformat auf 384 kbit/s reduzierten Datenrate eine der Compact Disc ebenbürtige Qualität zu erreichen, wobei auch bei niedrigeren Datenraten wie 192 kbit/s bis hinunter zu 112 kbit/s noch akzeptable Qualitäten erzielt werden sollen. Das menschliche Gehör ist im allgemeinen bei Störungen im Audiobereich empfindsamer als im visuellen Bereich, d.h. kurzzeitiges Rauschen und Knacken ist störender als z.B. Flimmern.

Innerhalb der Kodierung sind vier Modi zu unterscheiden. *Single Channel Coding* für Monosignale, *Dual Channel Coding* zur Kodierung von z.B. bilingualen Monosignalen (wie z.B. Zweikanalton im Bereich des TV), *Stereo Coding* zur Kodierung eines Stereosignals, bei dem die beiden Kanäle separat kodiert werden. Zusätzlich ist das *Joint Stereo Coding* zu nennen, das ebenso wie das *Stereo Coding* zur Kodierung eines Stereosignals benutzt wird. Bei diesem Verfahren wird die Datenredundanz und -irrelevanz zwischen den beiden Kanälen ausgenutzt und somit eine höhere Kompressionsrate erreicht.

Das digitale Eingangssignal wird in 32 gleichförmige Spektralkomponenten (Frequenzgruppen, Teilbänder) zerlegt, dieses Grundprinzip entspricht dem Vorgang im menschlichen Gehör (Psychoakustik). Der Vorgang wird als Zeit-Frequenzbereichs-Umsetzung bezeichnet. Die Spektralkomponenten werden dann in Abstimmung auf die Wahrnehmungseigenschaften des menschlichen Gehörs kodiert. Die schlußendliche Kodierung wird von einer der drei definierten Schichten (*Layer*) durchgeführt.

Sowohl Quantisierung als auch Kodierung werden unter Einbeziehung einer Maskierungsschwelle realisiert. Diese Maskierungsschwelle wird vom psychoakustischen Modell für jede Komponente individuell mit Hilfe der Diskreten-Fourier-Transformation (DFT) berechnet und gibt die maximal erlaubte Quantisierungsfehlerleistung an, mit der noch kodiert werden darf, ohne daß eine Wahrnehmung dieses Fehlers durch das menschliche Gehör befürchtet werden muß.

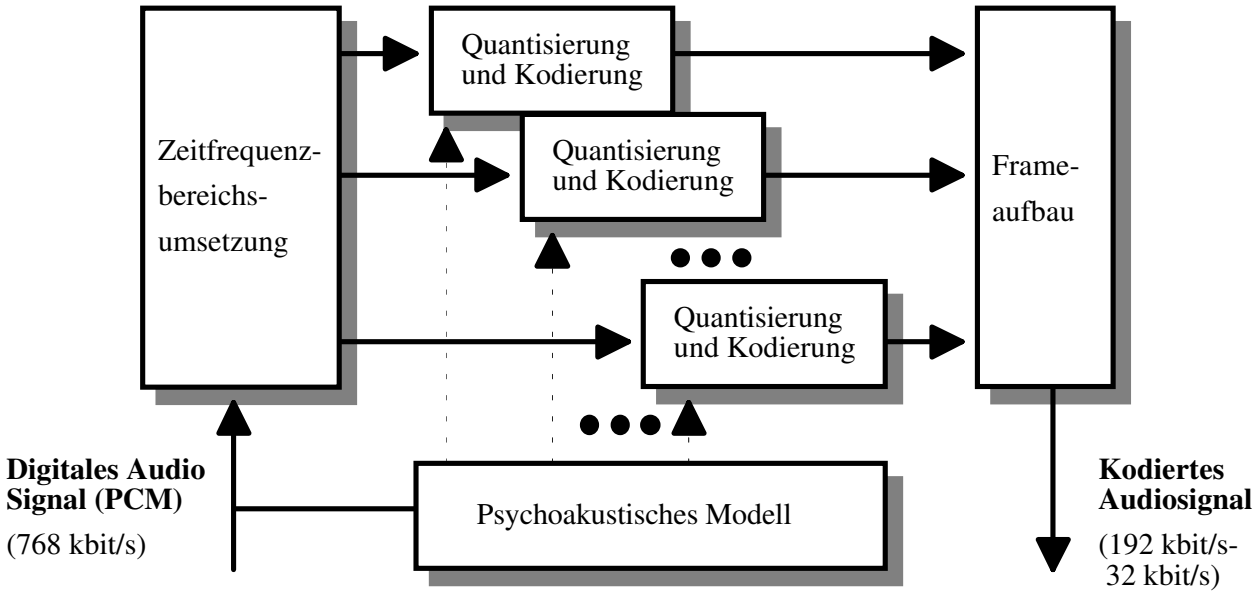


Abb. 13: *Generelles Blockschaltbild eines MPEG-1 Audiokodierers*

Die oben erwähnten drei *Layer* des MPEG-Audio-Standard arbeiten alle nach dem beschriebenen Grundprinzip. Die Zerlegung des Eingangssignals und der Kodierung unterscheidet sich jedoch sowohl in der benötigten Rechenleistung als auch in der erreichten Kompressionsrate. *Layer* II ist im Verhältnis zum *Layer* I komplexer aber im Bezug auf die Kodierung auch effizienter.

Der *Layer* III besitzt die größte Komplexität und zugleich die höchste Effizienz. Er kann mittlerweile auf leistungsfähigen Computern (Pentium-PC, PowerMac oder Workstations) auch in Echtzeit in Software dekodiert werden. Im Unterschied zu *Layer* I und II verwendet dieser *Layer* analog zur Umsetzung im Videoteil eine *Modified Discrete Cosine Transformation* (MDCT), die es erlaubt, die Anzahl der notwendigen Quantisierungswerte optimal zu errechnen. Laute, niederfrequente Signale überdecken dabei leise, hochfrequente Signale, die letzteren müssen dann also nicht mit der vollen Bit-Tiefe, sprich Quantisierung, kodiert werden. Die Art der Generierung der Quantisierungswerte mit Hilfe der MDCT wird "Psychoakustisches Modell" genannt und wurde durch langwierige Hörtests von Musikern, Tontechnikern und anderen Spezialisten optimiert.

Wie im Videoteil wird zusätzlich eine Entropie- und Huffmann-Kodierung vorgenommen, die die Werte nach der Umsetzung durch die MDCT optimal komprimieren. Die CD-Qualität wird hierbei mit der selben Datenrate wie in Layer II erreicht (384 kbit/s). Seine Stärke zeigt die Kodierung durch den *Layer* III erst bei niedrigeren Datenraten, bei denen immer noch eine beeindruckende Qualität erzielt werden kann.

Insgesamt erhält man bei der MPEG-Audiokomprimierung als sehr nützlichen Nebeneffekt die Rauschfreiheit, d.h. es sind keine zusätzlichen Verfahren zur Geräuschminderung wie z.B. das Dolby-System nötig.

Die *Layer* I und II sind bereits vollständig als integrierte Schaltungen realisiert. Die *Layer* I Kodierung wurde bereits seit der Einführung der DCC-Recorder (*Digital-Compact-Cassette*) verwendet. Die *Layer* II Kodierung wird vom europäischen DAB (*Digital Audio Broadcasting*) System verwendet. Zukünftiges Ziel dürfte es sein, mit dem *Layer* III die CD-Qualität schon bei einer Datenrate von 2*64 kbit/s zu erreichen, hierzu wird eine ideale Realisierung des *Joint Stereo Coding* notwendig sein. Dies ist in naher Zukunft zu erwarten.

MPEG-2

Der Audioteil von MPEG-2 ist wiederum genau wie bei MPEG-1 auf die Kodierung von Audiosignalen hoher Qualität (CD, Studio) zugeschnitten, weist aber eine viel höhere Flexibilität auf. Zusätzlich zu den festgelegten bereits bei MPEG-1 benutzten Sampling-Frequenzen von 33, 44 oder 48 kHz sind auch solche von 8, 11, 16, 22 oder 24 kHz möglich, dies insbesondere zur Kodierung von Signalen aus digitalen Quellen wie ISDN, G.722 oder *samples* aus dem Studiobereich. Ein variables Bit/Sample-Verhältnis wird erlaubt.

Um die Rückwärtskompatibilität zu MPEG-1 und insbesondere die Unterstützung von HDTV und Dolby-Surround zu ermöglichen, kann man in MPEG-2 bis zu 5 Kanäle kodieren (plus einen Kanal mit tiefstfrequenten Anteilen bis zu 100 Hz für Spezialeffekte). Diese Kanäle können in drei verschiedene aufgeteilt werden (links, mitte, rechts = HDTV), die anderen beiden werden als Raumkanäle genutzt (Dolby-Surround). Speziell über die drei Kanäle lassen sich vielfältige Kombinationen zur multilingualen Übertragung wählen.

Durch die Vielfältigkeit der Kodierung und Kanäle werden auch neue Möglichkeiten zur Kompression der Audiosignale erschlossen (zusätzlich zu den von MPEG-1 bekannten Mechanismen). Zu nennen sind

- *Intensity Stereo Coding* (ISC)
- *Phantom Coding of Center* (PCC)
- *Dynamic Transmission Channel Switching*
- *Dynamic Cross Talk*
- *Adaptive Multi-Channel Prediction*

Eine erweiterte Funktionalität im Bereich *Motion Compensation* und *Motion Estimation* wurde definiert. Die Integrität aller *samples* (bzw. *Offset*-Tabellen und Subbandinformationen) kann nun durch zugefügte Prüfsummen geschützt werden.

G.711 µlaw und a-law

Die µ-law Transformation ist eine grundsätzliche Audiokompressionstechnik, die von dem *Comité Consultatif Internationale de Télégraphique et Téléphonique* (CCITT) in der Recommendation G.711 im Jahre 1969 beschrieben und somit international standardisiert wurde.

Ausgegangen wird von einer logarithmischen Umsetzung und einer Wandlerauflösung von 13 bit, die nach der Wandlung auf 8 bit reduziert wird. Die Umwandlung geschieht über die in der Recommendation fest vorgeschriebenen, logarithmischen 13-zu-8 bit- und 8-zu-13 bit-Tabellen. Durch diese Art der Umwandlung wird die Kodierung an das logarithmische Hörvermögen des menschlichen Ohres angepaßt. Eine geringere Kodierungsauflösung entspricht damit qualitativ der höheren Quantisierungsauflösung.

Die Transformation läßt sich mathematisch wie folgt beschreiben:

$$y = 255 - \ln(1+x) \quad \text{for } x = 0$$
$$y = 127 - \ln(1+x) \quad \text{for } x < 0 \text{ [37]}$$

wobei *x* ein normalisierter Signalwert zwischen 0 und 1 ist. Es ergeben sich also Werte zwischen 0 und 255. Die µ-law Transformation wird gewöhnlich in Nordamerika und Japan für die Übertragung über das *Integrated Services Digital Network* (ISDN) benutzt, da dort Übertragungsstrecken existieren, die die dauernde Übertragung von Nullen nicht zulassen, da sonst der Bittakt verloren geht. In Europa wird für das ISDN-Netz der Telekom die sogenannte a-law Kodierung der G.711 benutzt, diese benutzt nur andere 13 zu 8 bit-Umsetzungstabellen und Werte zwischen -128 und 127.

"Die Auflösung von 8 bit wurde [...] aufgrund der Tatsache gewählt, daß bei einer internationalen Verbindung bis zu 14 oder 15 PCM-Umsetzungen in Kette geschaltet sein konnten. Bei einer ISDN-Verbindung gibt es jedoch nur eine Kodierung und eine Dekodierung, so daß die Beschränkung auf 8 bit [...] bei 8 kHz [...]" [34] nicht mehr sinnvoll erscheint. Dennoch halten die Hard- und Softwarehersteller aus den folgenden Gründen daran fest:

- 8-bit-PCM-Codecs sind als hochintegrierte Bausteine erhältlich
- ADPCM soll zukünftig ins ISDN integriert werden
- bestehende Hard- und Softwarelösungen (digitale Vermittlungstellen, die Audio-Hard- und Software z.B. der SparcStation) lassen sich nicht mehr auf andere Datenbreiten und Samplingraten umkonfigurieren.

Für den Einsatz hochqualitativer Audiospeicherung, -verarbeitung und -übertragung eignet sich das Format G.711 nicht. Einerseits wird nicht daran gedacht, das bestehende Format zu ändern, andererseits ist eine Kompressionsrate von 8/13 nicht ausreichend für hochvolumige Daten wie z.B. die einer CD.

Da es hier allerdings zumeist um die Übertragung von Sprache oder Datenflüsse, die durch die Kapazität der vorhandenen Leitungen begrenzt sind, gehen wird, wird dieses Format später genauer betrachtet werden müssen.

G.711 wird heute zur Kodierung von Audiosignalen über Leitungen im ISDN und zur Übertragung der Audiodaten bei H.261 und SMP verwendet. µ-law ist die Soft- und Hardware basierte Audiolösung der SUN Workstations, die als Plattform für das PCSS gewählt wurde.

G.721 ADPCM

Die Empfehlung der CCITT G.721 beschreibt ein Verfahren zur Kompression von typischen 64 kbit/s-Audiosignalen (kodiert in µ-law, a-law oder linear) auf 32 kbit/s. Es wird allgemein *Adaptive Differential Pulse Code Modulation* (ADPCM) genannt. Das grundsätzliche Verfahren beruht auf der Ähnlichkeit (nicht Gleichheit !) benachbarter Signale, es wird versucht aus dem bisherigen Signalverlauf auf den zukünftigen zu schließen.

- ADPCM-*Encoder*

  Grundsätzlich benutzt der *Encoder* die Funktionen des *Decoders*, um den Signalverlauf vorauszusagen. Zunächst werden die eventuell logarithmischen Eingangswerte in lineare umgewandelt (*uniform* PCM). Die so requantisierten Werte werden dann mit einem entsprechenden *Offset* (abhängig von der Implementierung des Quantisierers) gewichtet. Dieser *Offset* wird benutzt, um den uniformen PCM-Wert innerhalb des Quantisierungsbandes zu positionieren. Somit kann der ADPCM-*Encoder* die Charakteristik des Signals adaptieren,

indem er die Anzahl der Quantisierungs- und / oder Vorhersagestufen (*predictor steps*) variiert. Die Ausgangssignale werden dann nur als Differenzen übertragen. Deshalb werden innerhalb des ADPCM-kodierten Ausgangssignales manchmal diese Anzahlen mitkodiert, um unabhängige Implementierung der benutzten Quantisierer und Prediktoren kompatibel zu halten.

Die entsprechende Implementierung des Quantisierers hängt zumeist von den zu erwartenden Eingangssignalen ab, und wird hier anhand des Algorithmus der *Interactive Multimedia Association* (IMA) kurz vorgestellt. Andere Algorithmen werden in der Recommendation G.721 oder G.723 bzw. durch den *Compact Disk Interactive Audio Compression Algorithm* (CD-I) beschrieben.

- IMA-Algorithmus

  Dieser Algorithmus erreicht eine Kompression von 4:1. Seine Schnelligkeit resultiert aus der Einfachheit seines *Predictors*.

  Der Vorhersagewert ist einfach das vorherige Audio*sample*, deshalb ist dieser Schritt nicht adaptiv zu nennen. Die Adaption geschieht im Quantisierungsblock. Dabei weisen die 3 bits der Anzahl der Quantisierungstufen des gerade quantisierten Signals als Index in eine Tabelle, der dort verzeichnete Wert wird davon abgezogen und fungiert als Index in eine zweite Tabelle und wird für die nächste Iteration gespeichert. Der Wert aus der zweiten Tabelle ist dann die neue Anzahl der Quantisierungsstufen des nächsten s*amples*. Dieses Verfahren ist komplett umkehrbar und somit werden keine Zusatzinformationen innerhalb des resultierenden Bitstroms benötigt.

  Zusätzlich bietet das ADPCM-Verfahren Möglichkeiten zur *Error Recovery*, bei Bitfehlern innerhalb des komprimierten Datenstroms werden einzelne *samples* nur innerhalb der jeweiligen Quantisierungsstufe verfälscht. Fehlende Bits resultieren nur in einer Normalisierung der 88 *code words* der zweiten Tabelle, nach 88 *samples* rekonstruiert sich das Signal von selbst (siehe auch [37]).

- ADPCM-*Decoder*

  Der *Decoder* wendet zunächst den *Inverse Adaptive Quantizer* auf das Ausgangssignal an und rekonstruiert dann das Differenzsignal. Das *Sample* wird dann mit Hilfe des Differenzsignals und dem *Adaptiv Predictor* rekonstruiert und von dem uniformen Signal in ein normales PCM-Signal überführt.

  ADPCM wird heute in der digitalen Telefonie (ISDN) benutzt, da es gut zur Übertragung von Sprache geeignet ist (die Kompressionsmethoden sind auf den bei der Telefonie vorherrschenden Frequenzbereich abgestimmt). Weiterhin wird ADPCM direkt von der Audiohardware der im PCSS benutzten Sun Computer verwandt. Das Verfahren und vorhandene, diese Verfahren verwendende Programme sind auch auf andere Plattformen zu Portieren und auf heutigen Rechnern auch ohne Hardwareeinsatz kodier- und dekodierbar.

ETSI GSM 06.10

GSM ist eine Empfehlung der ETSI-Standardisierungsorganisation zur Komprimierung von Sprache und wird heute z.B. im D2-Telefonnetz benutzt, es sind also schon hochintegrierte Schaltungen erhältlich. GSM läßt sich nur mit hohem Aufwand in Software realisieren. GSM "benötigt zur Sprachübertragung eine Bandbreite von lediglich 13 kbit/s. Der Algorithmus kodiert Blöcke von je 160 Sprach*samples*, die im 13 bit uniform PCM-Format vorliegen, in Blöcken von 260 bits, was bei einer Samplingrate von 8 kHz eine Bitrate von 13 kbit/s ergibt." [34]

Das Eingangssignal wird als 13 bit uniform PCM digitalisiert (Wertebereich von -4096 bis 4095, *signed*). Durch eine Kreuzkorrelation wird die mathematische Ähnlichkeit zweier *samples* errechnet.

- *Encoder*

  Das Verfahren der GSM-Kodierung ist recht mathematisch und muß hier stark vereinfacht dargestellt werden.

  Der GSM-*Encoder* arbeitet auf Blöcken zu je 160 s*amples* des digitalisierten Eingangssignals. Diese werden skaliert, durch einen Hochpaß- [11] und einen Preemphasisfilter [12] bearbeitet. Wie bei ADPCM wird nun innerhalb des sogenannten LPC Blocks eine Voraussage der nächsten *samples* getroffen und die Differenz beider Werte gespeichert. Die Sequenz der Differenzwerte des Blockes (schon korreliert) wird dann als Filterparameter für einen *Linear Prediction Filter* benutzt, quantisiert und kodiert, dann als *Logarithmic Area Ratios* (LAR) bezeichnet und zum Dekodierer gesandt. Aus der Sequenz der LAR-Parameter wird dann ein sogenanntes *Short Term Residual Signal* (STR) interpoliert, daß den Fehler der Vorhersage für die 160 *samples* darstellt. Dann werden die STR-Signale mit den *samples* kreuzkorreliert und ergeben die sogenannten *Long Term Prediction* Signale (LTP). Aus diesen läßt sich dann der sogenannte *gain*-Parameter berechnen, der angibt, wie die LTP-Signale skaliert werden müssen, um die STR-Signale möglichst gut abzubilden. Die LTP-Signale werden anhand von dem vorher bei der Berechnung angefallen *lag*-Parametern verschoben und mit dem *gain*-Parameter skaliert. Die entstehenden Signale heißen dann *Long Term Estimate* (LTE) und sind eine langfristige Vorhersage des Signalverlaufs. Es wird von STR abgezogen, dann wird noch ein Tiefpaßfilter darauf angewandt, schlußendlich findet noch ein *Downsampling* mit Faktor 3 (nur jeder dritte Wert wird benutzt) statt und anschließend werden sie mit einem APCM-Verfahren kodiert. Im bit-Strom werden zuerst *frame*weise alle berechneten Parameter übertragen, dann folgen die APCM-Werte.

- *Decoder*

  Der GSM-*Decoder* funktioniert analog, da er schon beim Kodieren mitgelaufen ist, um die Qualität der Prediktion zu überprüfen. Die Filterbänke sind invertiert, aus dem Preemphasisfilter wird ein Deemphasisfilter, das *Downsampling* kehrt sich in ein *Upsampling* (wieder Faktor 3, die fehlenden *samples* werden mit Nullen aufgefüllt).

Im Bereich des *Intelligent PCSS* wird eine direkte Integration der Mobilfunkanbindung mit Hilfe der GSM-Komprimierungstechnik geplant.

---

[11] Resultiert in einer Beschneidung des Frequenzganges.
[12] Der Frequenzgang wird modifiziert, um Verluste bei der Übertragung zu minimieren.

WAV

Von auf Microsoft-Betriebssystemen (wie Windows 3.1, NT oder Windows 95) laufenden Programmen werden Audiodateien zumeist als WAV-Datei abgelegt. Innerhalb dieses Datei können mehrere verschiedene Datenströme, sogenannte RIFFs (*Resource Interchange File Format*) kodiert werden. Jeder dieser Ströme kann mit einem eigenen Kompressionsalgorithmus kodiert und mit den anderen Strömen synchronisiert werden. Audio-Daten werden innerhalb einer RIFF-Datei durch Angabe einer Identifikation, gefolgt von einer Längenangabe und dann die als ein ununterbrochener c*hunk* (Strang) und byteorientiert vorliegenden PCM-*samples* gespeichert. Die Zusammensetzung des RIFF-Formats für Audiodaten stellt sich somit wie folgt dar:

"RIFF" <Länge>
"WAVE"
"fmt " <Länge><Format-Daten>
"data " <Länge><Nutzdaten>

Die Reihenfolge der höherwertigen und der niederwertigen Bytes ist dabei Intel-spezifisch. Die PCM-*samples* können je nach Datei, als 8- oder 16-bit Werte, komprimiert oder unkomprimiert, mono oder stereo und in jeder denkbaren Frequenz vorliegen.

Theoretisch sind innerhalb einer WAV-Datei nicht nur verschiedene Kodierungsparameter, sondern durch die Angabe der **fmt**-Daten, auch beliebig viele verschiedene Kompressionsalgorithmen für die gespeicherten Daten möglich.

Kapitel 3 - Konzeption

In diesem Kapitel werden diejenigen Multimediaformate ausgewählt, deren Übertragung und Speicherung innerhalb eines LAN's bzw. weltweit im Internet unproblematisch ist. Dabei soll eher ein möglicher Realzeitzugriff und nicht Anforderungen der Formate an die Übertragungsleistung im Vordergrund stehen (wobei der Datenumfang natürlich einer Realzeitübertragung oft im Wege steht). Es werden dabei die Probleme des sequentiellen und auch simultanen, mehrfachen Zugriffs besprochen und generelle Mechanismen gefunden, die die Übertragung von verschiedenen Arten von Multimediadaten (Video, Audio) garantieren, um dadurch den späteren Implementierungsaufwand zu minimieren. Dies ist gleichzusetzen mit einer Auswahl der TCP/IP-basierten Kommunikationsprotokolle. Außerdem werden die Multimediaformate hinsichtlich ihrer Integration ins PCSS bewertet und ausgewählt sowie die Zugriffsprotokolle des Message-Stores festgelegt.

3.1 Definition des Message-Stores

Als Message-Store (Nachrichtenspeicher) wird im Rahmen dieser Arbeit ein System verstanden, in das Multimedia-Nachrichten übertragen, strukturiert und benutzerbezogen gespeichert und wieder gelöscht werden können. Ein Message-Store wird benötigt, um multimediale Nachrichten innerhalb des PCSS-Systems zentral zu speichern. Ein verteilte Speicherung auf mehreren oder beliebigen Rechnern kann weder eine garantierte Erreichbarkeit (die Verfügbarkeit beliebiger, einzelner Rechnersysteme ist sowohl von der Netzanbindung als auch softwareseitig nicht immer gegeben) noch eine garantierte Übertragungsleistung bieten. Durch einen zentralen Message-Store können sowohl der Verwaltungs- als auch der Implementierungsaufwand minimiert werden.

Da lange System-Reaktionszeiten innerhalb des PCSS oder an einem VISA-Terminal zu einer geringeren Benutzer-Akzeptanz des gesamten Systems und des Informationsterminals führen würde, sollte der Message-Store eine möglichst kurze Zugriffszeit besitzen. Da es sich außerdem meist um Daten mit einigen Megabyte Umfang handelt, sollte eine audiovisuelle Präsentation der Daten bereits während des Lesens der Daten aus dem Message-Store möglich sein. Müßten die Daten erst komplett gelesen werden bevor eine Präsentation beginnen kann, würde sich die Antwortzeit des VISA-Terminals auf Benutzer-Aktionen stark verlängern. Um Audio- und Video-Daten synchron abspielen zu können, müssen zwei Dateien gleichzeitig aus dem Message-Store gelesen werden können, wenn sie getrennt abgelegt sind. Das Erzeugen der schlußendlichen Dateien hingegen muß nicht in Echtzeit erfolgen, da der ACCESS-Benutzer seine Ansage nach der Erstellung eventuell noch überprüfen oder modifizieren möchte. Ansagen werden deshalb zuerst auf dem Rechner des ACCESS-Benutzers lokal gespeichert und erst in ihrer finalen Form im Message-Store abgelegt. Mit Nachrichten eines VISA-Benutzers an einen PCSS-Benutzer wird ebenso verfahren. Nicht mehr benötigte Ansagen und gelesene Nachrichten müssen aus dem Message-Store entfernt werden können, um den begrenzten Platz vernünftig nutzen zu können.

Alle Zugriffsmethoden sollen auf dem TCP/IP-Kommunikationsprotokoll basieren und weltweiten Zugriff und eine eindeutige weltweite Referenzierung ermöglichen (die Global Store-Implementierung der GMD kann aus den weiter unten erläuterten Gründen nicht innerhalb dieser Arbeit verwendet werden). Insbesondere müssen die Methoden zur Rückübertragung (Darstellung)

der abgelegten Nachrichten datenstrombasiert sein, um eine sofortige Darstellung zu ermöglichen. Eine komplette Übertragung einer gespeicherten Nachricht zur nachträglichen Darstellung ist in Hinsicht auf den Umfang von multimedialen Nachrichten unakzeptabel. Die vorhandene und jeweils unterschiedliche Ausrüstung der die Nachricht empfangenden Hardwareplattformen würde die maximale Länge von Nachrichten zu sehr beschränken.

Im Gegensatz dazu muß die Übertragung in den Message-Store nicht strom- sondern kann auch dateibasiert sein. Da zumeist Nachrichten mit lokaler Hardware aufgenommen werden und die Nachrichten oft mehreren, zeitlich unabhängigen, Änderungen bis zur Fertigstellung unterworfen sind und das Ergebnis mehrere Male kontrolliert wird, ist dieses Vorgehen auch anzuraten. Nur eine fertiggestellte Nachricht sollte in den Message-Store übernommen werden.

Der hier verwendete Begriff des Message-Stores entspricht nicht dem im Bereich der *Multimedia-Mail Teleservice* (MMM) und X.400 verwendeten gleichlautenden Begriffes (siehe dazu [6]).

3.2 Probleme bei der Global-Store Benutzung

Die momentane Implementierung des Global Store der GMD Fokus wurde unter dem Betriebssystem SunOS 4.1 auf dem Fokus-Netz vorgenommen. Implementierungen im Umfang der vorliegenden Diplomarbeit erfolgen jedoch auf Rechnern unter Solaris 2.4 im OKS-Netz der GMD Fokus. Es ergeben sich also erhebliche Portierungsprobleme, da SunOS-Sourcen unter Solaris 2 übersetzt werden müßten. Dies betrifft die Zugriffsprotokolle des Global Store RDT, ERP, XRM sowie im besonderen RTT, auf welches auf Seite 42 noch eingegangen wird. Des weiteren werden im Fokus-Netz proprietäre Programme wie *gmdmake* verwendet, deren Vorgehensweise beim Übersetzen des Programmcodes kaum nachzuvollziehen ist.

Es ergeben sich außerdem die folgenden Widersprüche zu denen bereits geforderten Funktionalitäten eines Message-Stores:

1. Die aktuelle ISODE-Implementierung verkraftet keine großen Objekte, da das gesamte Objekt im Hauptspeicher gehalten und in selbigem zusätzlich noch (mehrfach) dupliziert wird. Dies kann nur verhindert werden, wenn ein komplett neues Zugriffsprotokoll für den Global Store entwickelt wird.

2. RDT ist wegen Punkt 1 sowie der Tatsache, daß die *Distributed Object References* (DORs) nur in Dateien aufgelöst werden können, nicht echtzeitfähig. Als Alternative bietet sich das von Ghamsari in seiner Diplomarbeit implementierte *RealTime-Transfer* (RTT) Protokoll [13] an [11]. Allerdings ist die Anwendung von RTT für die Zwecke des PCSS mit sehr großem Portierungsaufwand verbunden, da es bisher nur unter SunOS 4.1 sowie in der Fokus-Umgebung funktioniert. Letzteres hängt mit der starken Integration von RTT in andere GMD-Fokus-Projekte wie MMC zusammen, aus denen das Protokoll herausgelöst werden müßte. RTT müßte außerdem auf die Möglichkeit hin untersucht werden, ob mehrere DORs parallel gelesen werden können, sowie gegebenenfalls modifiziert werden, falls dies nicht gegeben ist. Da RTT nur das Format SMP/G.711 unterstützt, müßte es für die generische, paketweise Nutzung mit binären Datenströmen modifiziert werden.

[13] RTT kann DORs über *sockets* in Echtzeit und im Speicher auflösen, so daß Objekte paketweise aus dem Global Store gelesen werden können.

3. ERP ist ebenfalls wegen Punkt 1 nur bedingt, d.h. bis zu gewissen Objektgrößen, einsetzbar. Die Verwendung von NFS als Application Service anstatt ERP/RDT könnte auch mit größeren Objekten funktionieren, allerdings nur, wenn der ERP-Client NFS-Zugriff auf den Global Store hat und ISODE nicht versucht, das gesamte Objekt im Speicher zu halten. Ein NFS-Zugriff auf den Message-Store ist aber nur lokal zu verantworten.

4. Das Löschen von Objekten aus dem Global Store funktioniert nur über sogenannte Verfallsdaten. Für die hier angestrebte Verwendung von DORs ist dies nicht akzeptabel, da in vielen Fällen die Nutzungsdauer des Objektes nicht im voraus zu bestimmen ist.

Zusammenfassend wären für eine Nutzung dieser Implementierung des Global Stores als Message-Store innerhalb des PCSS folgende Änderungen erforderlich:

- RTT-Client sowie -Server müßten nach Solaris 2 portiert und von anderen MMC-Projekten losgelöst werden.

- Der eigene, modifizierte und auf RTT basierende Server müßte beim Global Store angemeldet werden, um DORs mit einem eigenen *Application Service* an den Client übertragen zu können.

- RTT müßte so modifiziert werden, daß beliebige DORs übertragen werden können und nicht nur SMP/G.711 Daten in einem für AVC angepaßtem Format.

- Das ERP-Problem mit großen Objekten könnte durch die Verwendung von NFS gelöst werden, wobei dann alle ACCESS-Benutzer sowie VISA NFS-Zugriff auf den Global Store haben müßten.

- Das Problem der nicht vorhandenen Löschoperation auf DORs müßte umgangen werden. Dies würde eine Erweiterung der Global Store Definition erfordern.

Aus den genannten Gründen wird im Rahmen der vorliegenden Diplomarbeit auf eine Integration verzichtet.

3.3 Realzeitzugriff

Der Realzeitzugriff auf im Message-Store gespeicherte Multimedia-Nachrichten (siehe Seite 44) ist ausschließlich für die strombasierten, zeitabhängigen Formate von Belang. Text und Bild haben weder bei der Rezeption noch bei der Aufnahme bzw. Digitalisierung eine zeitliche Dimension. Audio und Video sowie Text und Bild unterscheiden sich also hinsichtlich der Synchronität:

- **isochrone Kommunikation** - der zeitliche Abstand zwischen zwei Nachrichtenteilen ist auf Sende- und Empfangsseite identisch. D.h. alle Nachrichteneinheiten werden mit der gleichen Verzögerung übertragen (mit einer gewissen Varianz der Übertragungsverzögerung).

- **asynchrone Kommunikation** - es bestehen keine Beschränkungen für Verzögerungen und Verzögerungsschwankungen

- **synchrone Kommunikation** - die Übertragungsverzögerung ist für alle Nachrichteneinheiten beschränkt. Dies gilt für Kommunikationsbeziehungen wie z.B. dem Zwiegespräch,

Gruppenbesprechungen oder Konferenzen und ist im Bereich der hier besprochenen multimedialen Nachrichten nicht von Belang.

Audio- und Bewegtbildkommunikation verlangen eine isochrone Übertragung. In der Praxis wird dies durch eine synchrone Übertragung mit genügend kleiner Verzögerung angenähert.

Bei der Aufnahme von Audionachrichten müssen alle zeitlich vorkommenden Signale in Betracht gezogen und bei der Rezeption auch alle Audiosignale aus dem Message-Store zum Rezipienten übertragen und dort in Echtzeit abgespielt werden. Eine zeitliche Verzögerung bis zum Start der Rezeption ist hier akzeptabel, jedoch keine Unterbrechung oder zeitliche Verzerrung während der Wiedergabe.

Videonachrichten wiederum können sowohl bei der Aufnahme als auch bei der Wiedergabe zeitlich und räumlich skaliert werden. D.h. sollte es technisch nicht möglich sein, die volle Anzahl der Einzelbilder pro Sekunde (*frame rate*), den vollen Farbinhalt oder die volle Größe der Einzelbilder zu erhalten, beeinträchtigt dies nicht (bis zu gewissen Grenzen natürlich) den Inhalt der Nachricht. Solange bei der Wiedergabe noch die in der Videonachricht enthaltenen Einzelbilder zum relativ gleichen Zeitpunkt wie bei der Aufnahme abgespielt werden, genügt eine solche Videonachricht den Anforderung eines Message-Stores. Ein zeitliche Streckung der Videonachricht aufgrund fehlender Ressourcen ist hierbei unakzeptabel, da dies die Synchronität mit eventuell assoziiertem Audio behindert. Auch bei Videonachrichten ist eine zeitliche Verzögerung bis zum Start der Wiedergabe in gewissem Rahmen möglich.

3.4 Multimedia-Messages

Als Multimedia-Messages werden im folgenden alle digitalen Nachrichten innerhalb des PCSS bezeichnet, die

- im Sinne der Anrufweiterleitung oder durch die Funktionalität des Anrufbeantworters mit Unterstützung des PCSS behandelt werden,
- von in das PCSS integrierten Applikationen aufgezeichnet und wieder abgespielt werden,
- in dem an das PCSS angegliederten Message-Store abgelegt werden oder
- die entweder Text-, Bild-, Audio- oder Videonachrichten darstellen.

Eine Multimedia-Message kann dabei eine Kombination der vier Medien-Typen Text, Bild, Audio und Video darstellen. Falls möglich, sollten bei der Aufnahme mehrere Medien-Typen die gleiche Nachricht wiederspiegeln, und somit mehrfach in der Multimedia-Message gespeichert werden, um bei der Rezeption der Nachricht unabhängig von vorhandener Ausrüstung zu werden. Sollte z.B. keine Videoausrüstung vorhanden sein, könnte dann wahlweise der Text angezeigt werden. Obwohl eine Multimedia-Message also eine Kombination aus mehrere Datenströmen sein kann, beschreiben diese jedoch immer sinngemäß die gleiche Nachricht. Ein Ausnahme bildet die Kombination Video und Audio. Sollten Video- und Audiodatenströme innerhalb einer Multimedia-Message vorhanden sein, wird sie als Videonachricht mit synchronem, assoziiertem Audio interpretiert. Dies soll eine Speicherung eines Films (*movies* [14]) mit in getrennten Dateien vorliegenden Video- und Audioinformationen ermöglichen.

[14] Als *movie* wird im weiteren immer eine Videonachricht mit Video- und Audioteil verstanden.

3.5 Modifikationen in den PCSS-Profiles

Um Multimedia-Nachrichten innerhalb des PCSS zu verwalten, mußten Änderungen und Erweiterungen an den im folgenden erklärten User-Profiles vorgenommen werden.

3.5.1 Userprofiles

Grundlegende Datenstruktur des PCSS ist das *User Profile*. Es nimmt alle Informationen auf, die zur Steuerung und Kontrolle der integrierten Telekommunikationsdienste notwendig sind. Für jeden Benutzer existiert dabei eine Instanz des *User Profiles*, in die über geeignete, ebenfalls durch das PCSS bereitgestellte Dienste die persönlichen Konfigurationsdaten eingetragen werden (wie z.B. die bedingungsabhängige Behandlung ankommender Anrufe) (siehe auch [9]).

Weiterhin sollen im *User Profile* auch alle eingegangenen Anrufe für den Benutzer gespeichert werden. Diese *Incoming Messages* werden im *User Profile* in der sogenannten *Message Box* gespeichert. Die Anrufbeantworterfunktionalität des PCSS bedingt auch die Erzeugung und Verwaltung von multimedialen Ansagen. Diese, *Customized Recorded Announcements* (CRAs) genannten, Ansagen werden innerhalb des *User Profiles* im sogenannten *CRA Profile* abgelegt.

Die oben definierten Multimedia-Messages wurden im Rahmen dieser Diplomarbeit innerhalb des *User Profiles* durch die Objektklasse *GMcontent* definiert. Diese Klasse wird im folgenden erklärt.

3.5.2 General Media Content

Der *General Media Content* (GMcontent, genereller medialer Inhalt) definiert die Objektklasse **pcssGMcontent**, die innerhalb des PCSS benutzt wird, um jede Art von persönlichen Multimediadokumenten innerhalb der PCSS Profiles zu definieren. Diese Klasse wurde innerhalb des *Deliverable* 6 des BERKOM II Projektes PCSS vom Juni 1995 an Hand der dieser Diplomarbeit zugrunde liegenden Erkenntnisse festgeschrieben (siehe auch [9], Seite 59). Objekte dieser Klasse werden insbesondere von den CRAs und IMMs referenziert. Die Klasse ist wie folgt in ASN.1-Notation definiert:

```
pcssGMcontent              OBJECT-CLASS ::= {
        SUBCLASS OF        top
        MUST CONTAIN       { pcssGMcontentIDKey }
        MAY CONTAIN        { description |
                           pcssTextComponent |
                           pcssAudioComponent |
                           pcssVideoComponent |
                           pcssImageComponent }
        ID                 { pcssDSObjectClass.xzy} }

pcssGMcontentIDKey ATTRIBUTE ::= {
        WITH SYNTAX                    CaseIgnoreString
        QUALITY MATCHING RULE          caseIgnoreMatch
        ID                 {pcssDSAttributetype.xyz} }
```

```
pcssTextComponent ATTRIBUTE ::= {
        WITH SYNTAX                     GMComponent
        ID                  {pcssDSAttributetype.xzy} }

pcssAudioComponent ATTRIBUTE ::= {
        WITH SYNTAX                     GMComponent
        ID                  {pcssDSAttributetype.xzy} }

pcssVideoComponent ATTRIBUTE ::= {
        WITH SYNTAX                     GMComponent
        ID                  {pcssDSAttributetype.xzy} }

pcssImageComponent ATTRIBUTE ::= {
        WITH SYNTAX                     GMComponent
        ID                  {pcssDSAttributetype.xzy} }

GMComponent := SEQUENCE {
        pcssReference [0] PcssReference,
        pcssComponentType [1] PcssMimeType,
        pcssByteLength [2] PcssByteLength,
        pcssDuration [3] PcssDuration OPTIONAL }

PcssMimeType ::= IA5String

PcssByteLength ::= INTEGER

PcssDuration ::= INTEGER

PcssReference ::= CHOICE {
        [0] GS          ANY
        [1] URL         PcssURL }

PcssURL ::= IA5String
```

Wie bereits dargelegt, wird eine Multimedianachricht innerhalb des PCSS-Profiles durch mehrere Multimediadatenströme beschrieben. Diese werden dadurch repräsentiert, daß innerhalb des **GMcontent** wahlweise jeder Medientyp (Text, Bild, Video und Audio) keinmal oder einmal vertreten sein kann. Jeder Medientyp einhält zur Beschreibung eine Referenz, einen Mime-Typen, eine Bytelänge und eine *Duration* (Zeitdauer). Diese werden im folgenden genauer erklärt:

Als Referenz wird hier der Verweis auf eine Datei mit einem sogenannten *Uniform Resource Locator* (URL, siehe Seite 13) verstanden. Die URL ermöglicht die weltweite und globale sowie auch eine lokale, jeweils eindeutige, Referenzierung einer Datei und der gewünschten Zugriffsmethode auf diese Datei. Laut der letzten Definition des User-Profiles in [9] kann weiterhin ein Datenstrom innerhalb des Global Stores als DOR angegeben werden.

Typ, Größe und zeitliche Länge einer Nachricht wurden ins Profile verlegt, um Nachrichten anhand ihrer Abspielbarkeit zu selektieren, ohne auf den Message-Store zugreifen und die gespeicherten Dateien teilweise lesen und analysieren zu müssen. Die entsprechenden Daten und Meßwerte fallen bei der Aufnahme der Nachrichten an und können somit einfach gespeichert werden.

Um innerhalb der Internet-Terminologie zu bleiben, hat man sich für die Verwendung der MIME-Typen zur Beschreibung des Datenformats der Nachrichten entschieden. Außerdem liefert auch ein HTTP-Server vor der Übermittlung der eigentlich angeforderten Datei einen *header*, in dem u.a. auch der MIME-Typ der folgenden Datei angegeben wird. Diese Angabe kann bei der Darstellung von Nachrichten aus dem Message-Store berücksichtigt und analysiert werden.

Die Größe einer Nachricht wird gespeichert, um vor der Darstellung der Nachricht Aussagen machen zu können, die Angabe einer zeitlichen Länge macht natürlich nur bei den Medientypen Audio und Video Sinn, sie wird bei Text und Bild ignoriert.

Ein gleichzeitiges Vorkommen einer Video- und einer Audiokomponente kann als ein gespeichertes *movie* interpretiert werden (abhängig vom Datenformat). D.h. ein *movie* wurde nur als zwei getrennte Datenströme gespeichert. Dies sollte zwar nicht die Regel sein. *Movies* sollten als *interleaved* Video/Audiodatenströme abgelegt werden, um nicht bei der Übertragung zweier Dateien eine Synchronisierung zu erschweren.

3.6 Selektion der Formate

Um aus den für eine Implementierung in Frage kommenden und oben vorgestellten Formate auszuwählen, wurden folgende Kriterien zur Bewertung herangezogen:

- **Realzeitdekodierung** - alle im Message-Store befindlichen multimedialen Datenströme müssen den oben erklärten Anforderungen an die Realzeitdarstellung genügen. Eine Realzeitkodierung bei der Aufnahme ist nur bei den Videoformaten notwendig. Eine eventuell notwendige Komprimierung der anfallenden Daten kann insbesondere bei CRAs und IMMs nach der Aufnahme erfolgen.

  Alle Text- und Bildformate lassen sich in Echtzeit darstellen. Unkomprimierte Audioformate bereiten hier auch keinerlei Schwierigkeiten, soweit die einzelnen *samples* in Echtzeit aus dem Message-Store übertragen werden können (bei Übertragungen aus einem nicht-lokalen Message-Store innerhalb eines LANs, trifft dies für die beschriebenen Audioformate nicht zu). Komprimierte Audioformate lassen sich zwar leicht übertragen, haben dann jedoch hohe Anforderungen an die Systemleistung bei der Dekodierung. Alle besprochenen komprimierten Audioformate können jedoch auf den verwendeten SparcStations in Echtzeit dekodiert werden. Von den Videoformaten kann sowohl SMP als auch MPEG-1 in Echtzeit dekodiert werden, für MPEG-2 sind bis heute weder echtzeitfähige Soft- noch Hardware für die im PCSS verwendeten Hardwareplattformen frei verfügbar.

- **Softwareimplementierung** - die Dekodierung der Datenströme muß in Software erfolgen können, d.h. ein Hardwareunterstützung bei der Darstellung darf nicht nötig sein (Videoformate können hier jedoch eine Ausnahme bilden). Das bedeutet weiter, daß die den Datenformaten inneliegenden Algorithmen bekannt seien müssen, bereits in anderen Programmen Verwendung gefunden haben und daß Beispielprogrammcode vorliegt.

  Für alle besprochenen Formate, bis auf SMP, gibt es frei verfügbaren und integrierbaren Programmcode. SMP wird in Lizenz von der Firma DEC vertrieben. Der Punkt der Integrierbarkeit spielt weiter unten in Abhängigkeit von der beim PCSS verwendeten Software eine Rolle.

- **Plattformunabhängigkeit** - die ausgewählten Formate sollten zwecks einer zukunftssicheren Auswahl international standardisiert sein, de-facto Internetstandards sein oder auf allen möglichen Hardwareplattformen weite Verbreitung gefunden haben.

  Alle besprochenen Text-, Bild- sowie Audioformate sind auf allen heutigen Hardwareplattformen und Betriebssystemen kodier- und dekodierbar. Bei den Videoformaten ist H.261 wegen seiner benötigten Hardwareunterstützung leider wenig verbreitet, das insbesondere für den Einsatz der CD-ROM spezifizierte asynchrone Format MPEG erfreut sich im Internet (dank genügend freier Software und mittlerweile recht preiswerter Hardware) größter Beliebtheit. MPEG-2 wird durch die Einführung der ersten Set-Top-Boxen für den Endkonsumenten im Fernsehbereich mehr Verbreitung finden. Soft- und Hardwareimplementierungen werden dann folgen. SMP wird auf nicht Unix-basierten Systemen so gut wie nicht eingesetzt.

  WAV ist leider nur in seiner am meisten verbreitesten Form, nämlich unkomprimiert, plattformunabhängig. Das Vorhandensein generell beliebig vieler Komprimierungsalgorithmen erfordert den jeweiligen Dekomprimierungsprogrammcode in jeder das WAV-Format nutzenden Applikation. Da Microsoft immer mehr Subformate integriert, ist hier mit Schwierigkeiten zu rechnen. Allerdings werden innerhalb des PCSS die Audiodaten mit den PCSS-internen Applikationen aufgenommen und abgespielt und somit die gleichen Subformate unterstützt. Der Import schon im WAV-Format vorhandener Daten gestaltet sich jedoch auf der Solaris-Plattform weiterhin als schwierig.

- **Erkennbarkeit** - es sollte möglich sein, gespeicherte Datenströme an Hand eines *headers* (d.h. an formattypischen Informationen am Anfang der Datei) bzw. der Dateinahmenerweiterung eindeutig zu erkennen. Sollte es innerhalb des Formates verschiedene Kodierungsmodi geben, sollten diese ebenfalls leicht unterscheidbar sein.

  ASCII-Text läßt sich, da es ein rein zeichenbasiertes Format ist (d.h. jedes Byte innerhalb des Datenstroms entspricht einer inhaltlichen Information, nicht z.B. einer Formatierung oder *Header*informationen), leider nicht automatisch erkennen, auch unterschiedliche Zeichensätze lassen sich nicht unterscheiden. Eine eventuell notwendige Unterscheidung verschiedener Kodierungen des ASCII-Formats läßt sich notfalls über die Angabe des MIME-Typs realisieren. PostScript enthält, obwohl ausschließlich mit Hilfe von ASCII-Zeichen kodiert, *header*zeilen. Alle besprochenen Bild- und Videoformate enthalten Kennungen am Anfang der Dateien, die eine eindeutige Identifizierung ermöglichen. Bei den Audioformaten enthalten MPEG, G.711 (wenn richtig kodiert, zu oft existieren jedoch auch Dateien ohne *header*, dies ist jedoch kein Manko des Formats an sich), WAV und GSM Identifizierungsinformationen. Sogenannte *raw* Audiodaten werden zumeist ohne Header gespeichert und sollen hier nicht weiter betrachtet werden.

Die zwei folgenden Kriterien müssen hier genauer ausgeführt werden.

3.6.1 Selektion nach Leistungsfähigkeit

HTML und PostScript können nicht weiter betrachtet werden, da die Erzeugung dieser Formate zu komplex und für den Benutzer noch nicht intuitiv genug ist. Außerdem wird es z.B. nicht nötig sein, komplexe Dokumente am VISA-Terminal zu erstellen, nur um Nachrichten an PCSS-Benutzer zu schicken.

Bei den Bildformaten sind hinsichtlich der Leistungsfähigkeit weder bei der Übertragung noch bei der Kodierung und Dekodierung Abstriche zu machen.

Die Videoformate können nur begrenzt in lokalen Netzwerken aus dem Message-Store übertragen werden (z.B. benötigt MPEG-1 eine Datenrate von ca. 1,2 Mbit/s, dies entspricht ca. einem Fünftel der echten Übertragungsleistung eines auf Ethernet basierenden Netzwerkes; MPEG-2 kann variabel bis 10 Mbit/s kodiert werden). Allerdings können schon bei der Aufnahme die Kodierungsparameter z.B. bei MPEG so gestaltet werden, das die Kompressionsrate stark steigt und die Qualität des dekodierten Videos für Nachrichten innerhalb des PCSS noch als ausreichend zu betrachten ist. Eine Übertragung von Videos auf weltweiter Ebene im Internet ist nicht möglich.

Hochvolumige, unkomprimierte Audioformate können nicht innerhalb des Internet übertragen werden, da die benötigte Übertragungsleistung (Bandbreite) zumeist nicht zu Verfügung steht. Auch innerhalb eines LANs sollten unkomprimierte Audiodateien wie z.B. WAV nicht übertragen werden, sie würden andere Übertragungen (z.B. NFS) zu sehr belasten. Die Übertragung von komprimierten Formaten wie MPEG-1 und ADPCM ist generell unproblematisch innerhalb eines LANs, ist die Anbindung des LANs an das Internet höher dimensioniert (ab 2 ISDN B-Kanälen z.B.) ist selbst die Übertragung dieser im Internet möglich. MPEG-1 Layer 3 kann, da noch kein echtzeitfähiger, freier Progammcode vorliegt, nicht weiter betrachtet werden. Layer 2 bietet jedoch schon eine genügend gute Qualität bei einer hohen Komprimierungsrate.

3.6.2 Selektion nach Integrationsfähigkeit

Da im Rahmen des praktischen Teils dieser Diplomarbeit nicht alle verwendeten Formate von Grund auf neu implementiert werden können, müßten die Formate im Plattformumfeld des PCSS bereits als freier Programmcode vorliegen und einfach in die bestehenden Entwicklungsysteme integrierbar und auch für andere Programme wiederverwendbar sein.

Dies bedingt zumeist das Vorhandensein von Beispielprogramcode in der Programmiersprache C. C++ kann nur bei Verwendung von zusätzlichen Entwicklungssystemen (z.B. O-Tcl oder Object-Tcl) in Tcl eingebettet werden. Tcl selbst enthält ein einfach zu benutzendes C-Interface. Andere Programmiersprachen, wie Pascal, Modula, Fortran, ADA oder Objective-C können daher nicht benutzt werden.

Für HTML und PostScript sind die notwendige Software und Hilfsprogramme nur schwer zu finden. Eine Integration bei den im Rahmen dieser Diplomarbeit entwickelten, darstellenden Programmen ist möglich, jedoch beim Fehlen der Programme, die diese Formate erzeugen, nur schwer weiter zu verfolgen. Neu-Implementierungen würden den Rahmen der Arbeit sprengen.

Bei den Bildformaten sind GIF-, XBM/XPM-, PBM- and TIFF-Erweiterung für Tk und Tix vorhanden, diese Formate können also einfach integriert werden. Eine Erweiterung für JPEG in Tix soll laut Angaben der Entwickler folgen.

Bei den Videoformaten muß man auf eine Integration von SMP/G.711 leider verzichten, da die zugrunde liegende Software lizenzpflichtig ist. Da die Digitalisierung von Videosignalen eine Hardwareerweiterung bedingt und die Verwendung der Parallax-Karte aus Kostengründen (bis zu DM 15.000.- pro Karte) entfällt, bleiben nur Formate, die von der SunVideo-Karte unterstützt werden. Dies sind zwar mehrere, aber allen Auswahlkriterien scheint nur MPEG-1 zu genügen.

Einschränkungen bei der Integration der Audioformate bestehen nur bei der Anbindung von MPEG-Audio. Die einzige verwendbare, da in Echtzeit dekodierende, Lösung wurde von Tobias Bading an der TU-Berlin in der Programmiersprache C++ entwickelt (siehe [1]). Um dieses sehr gut verwendbare und hochqualitative Format benutzen zu können, sollte eine Anbindung trotzdem versucht werden. Freier Programmcode für ADPCM wird z.B. von der CCITT zur Verfügung gestellt, G.711 wird bereits hardwareseitig durch die SparcStations unterstützt. WAV ist in seiner unkomprimierten Form ein vom Aufbau her einfaches Format.

3.6.3 Verwendete Formate

Zusammenfassend kann man sich für die Verwendung der folgenden Formate im Rahmen des PCSS entscheiden:

- **Text** - ASCII
- **Bild** - GIF, XBM/ XPM, PBM, TIFF
- **Video** - MPEG-1
- **Audio** - MPEG-1 (Layer 2), G.711, ADPCM, WAV

Die zur Darstellung benötigte Datenrate der zeitabhängigen Formate wird in der Tabelle 4 gezeigt.

Format	Datenrate
MPEG-1 Video laut Standard	1.2 Mbit/sec
MPEG-1 Video Mindestanforderung 320*240, *6 frames/sec*	200 kbit/sec
MPEG-1 Audio laut Standard (Stereo)	384 kbit/sec
MPEG-1 Audio Mindestanforderung 33 kHz, Mono	96 kbit/sec
G.711	64 kbit/sec
G.721	32-40 Mbit/sec
WAV unkomprimiert oder RAW-PCM	64-1400 Mbit/sec

Tab. 4: *Datenrate der zeitabhängigen Formate*

3.7 Selektion der Protokolle

Beim Zugriff auf den Message-Store werden folgende Funktionalitäten benötigt:

- eindeutige, globale Referenzierung einzelner Nachrichten bzw. ihrer Teile

  Hierbei bietet sich der Mechanismus der URLs an. Nicht nur Zugriffsprotokoll, sondern auch Rechner-, Pfad und Dateiname können komplett angegeben werden.

- paketweise, strombasierte Dereferenzierung der Dateien

  Nur HTTP ermöglicht die strombasierte Rückübertragung der Dateien, der Datenfluß kann also jederzeit umgelenkt und abgebrochen werden. Eine Darstellung während der Übertragung der Dateien ist also möglich. FTP arbeitet durch die Teilung von Steuerungs- und Datenfluß dateibasiert, Daten werden also paketweise ins Datensystem übertragen und können nicht

während der Übertragung im Speicher verwaltet und verändert werden. Die Benutzung von RCP muß aus dem gleichen Grund verworfen werden.

- gesicherte, weltweite Übertragung in den Message-Store

  HTTP kann zwar mit dem POST-Kommando Daten an einen HTTP-Server übertragen, diese Daten dienen jedoch aussschließlich der Serversteuerung bzw. der Anforderung von weiteren Diensten. Dateien können mit HTTP nicht übertragen werden. FTP bietet jedoch einen gesicherten, dateibasierten Transfer, der auch bei teilweiser Nichtverfügbarkeit des *Clients* bzw. des *Servers* nicht versagt. RCP wiederum kann nur Dateien weltweit übertragen, wenn für beide beteiligte Systeme die Nutzerkennung und das Passwort bekannt sind. Eine Trennung des UNIX-Benutzerzugangs vom Nutzerzugang zum Message-Store auf dem Message-Store-System ist nicht möglich, da die Dateien im Message-Store über das NFS gespeichert werden und diese Speicherung die Verbindung mit einem dort bekannten UNIX-Benutzer erfordert.

- Modifikation gespeicherter Dateien

  Einzig das FTP Protokoll ermöglicht eine Modifikation der entfernten, im Message-Store gespeicherten Dateien. Die Modifikation des Namens, des Datums und der Zugriffsrechte sowie die Löschung der Nachricht sind möglich.

- gleichzeitiger, mehrfacher Zugriff auf den Server

  Alle Protokolle bieten *server*seitig die Möglichkeit, mehrfach und gleichzeitig auf verschiedene Dateien zuzugreifen. Ungünstig hat sich jedoch das *Server*verhalten mit dem Protokoll FTP erwiesen. Es scheint Prioritäten bei gleichzeitiger Übertragung aus dem Message-Store unterschiedlich zu verteilen; es kommt leider oft zu unakzeptablen Unterbrechungen im Sekundenbereich, die andererseits während der Speicherung in den Message-Store unkritisch sind.

- gesicherte Identifizierung von berechtigten Benutzern

  Alle beschriebenen Protokolle bieten die Möglichkeit der Benutzerautorisierung. Zusätzlich bieten sowohl FTP- als auch HTTP-Server die Möglichkeit den Zugriff zu protokollieren.

3.7.1 Verwendete Protokolle

Folgende Protokolle und Mechanismen sollten aus den genannten Gründen beim Übertragen von Nachrichten in und vom Message-Store verwendet werden:

- **HTTP** - zur Dereferenzierung
- **FTP** - zur Speicherung bzw. Modifikation
- **URL's** - zur eindeutigen Referenzierung der Dateien

3.8 Modularisierung und Generik

Da, wie im Kapitel 1 beschrieben, die hier entwickelten Applikationen von mehreren Modulen innerhalb des PCSS benutzt werden, müssen praktisch bedingte Aufrufstrukturen und Implementierungsabhängigkeiten von der reinen Funktionalität abgekapselt werden können.

Eine Benutzung der Applikationen innerhalb anderer Projekte soll durch die Einbindung in das PCSS nicht verhindert werden. Einzelne Submodule sollen mehrfach und gleichzeitig von anderen Applikationen benutzt und adaptiert werden können.

Innerhalb des PCSS wurden mehrere funktionale Einheiten (sogenannte *building blocks*) definiert. Die für die multimediale Erweiterung des PCSS relevanten Blöcke und die dazugehörigen Benutztrelationen, sind in der Abbildung 14 dargestellt.

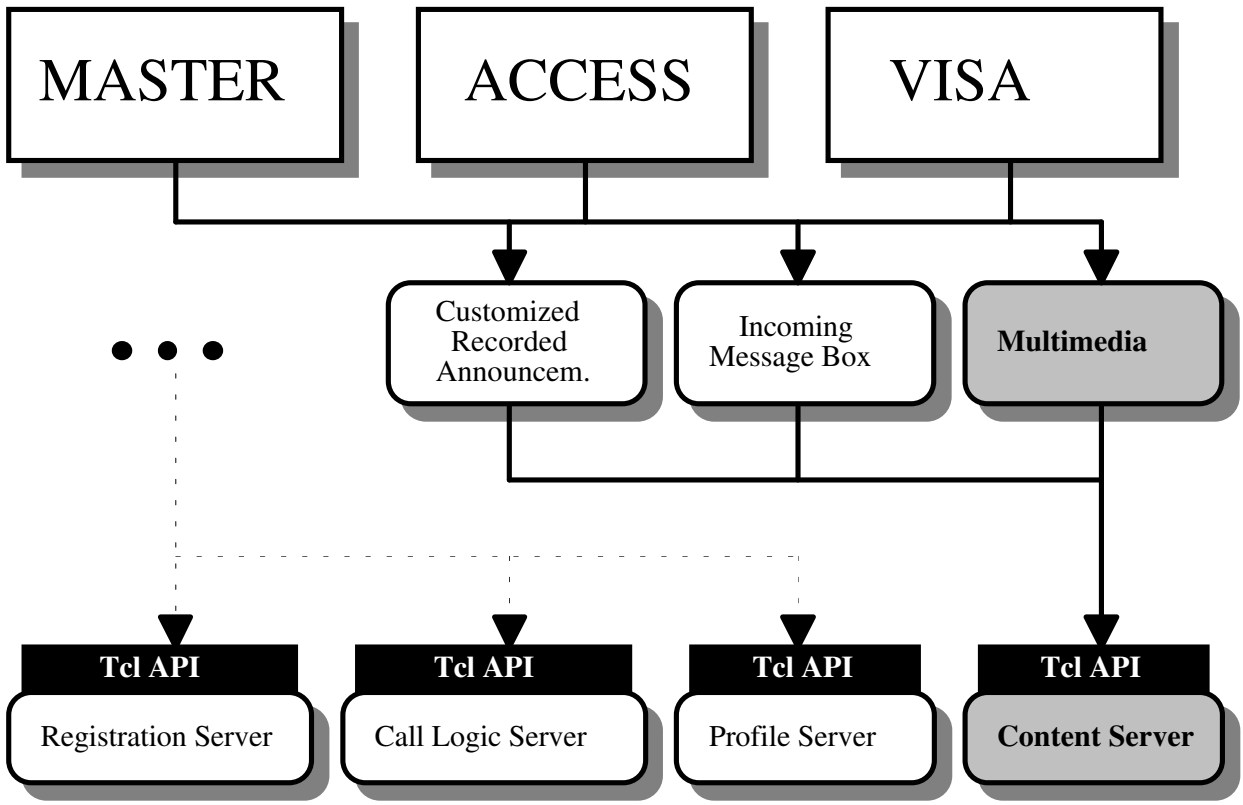


Abb. 14: *PCSS Modularisierung*

Innerhalb der multimedialen Erweiterung des PCSS (MM-PCSS) werden Funktionen in folgenden Kategorien implementiert:

Unterstützung der PCSS-Komponenten

Die hier implementierte, multimediale Erweiterung des PCSS muß dessen Komponenten mit den folgenden Funktionalitäten unterstützen:

- Aufrufstrukturen für darstellende und aufnehmende Applikationen [15]
- globale und benutzerabhängige *player/recorder* Integration zur Laufzeit

[15] Im weiteren *player* und *recorder* genannt.

- globale und benutzerabhängige Konfiguration der unterstützten Formate zur Laufzeit
- Kapselung der unterstützten Formate durch Verwendung von MIME-Typen
- *player/recorder*-unabhängige Bestimmung der Aufnahmezeitdauer
- Tcl-Funktionen zur Dateibehandlung wie copy, remove, rename und tmpnam

Zugriff auf den Message-Store

Die Implementierung muß hinsichtlich des Zugriffs auf den Message-Store,

- *socket*-Verbindung zum PCSS-spezifischen Message-Store aufbauen, diese parametrisieren und über das Protokoll HTTP Dateien übertragen,
- durch URLs spezifizierte Dateien mit Hilfe des Protokolls HTTP dereferenzieren und im lokalen Dateisystem speichern,
- URLs erzeugen, umkodieren und in lokale Dateinamen verwandeln und
- Dateien in den Message-Store via FTP übertragen oder gegebenenfalls löschen können.

Audiounterstützung

Die Audioerweiterung des PCSS umfaßt:

- Tcl-Funktionen zur Ansteuerung der Audioschnittstellen der verwendeten Hardwareplattform,
- Funktionen zur automatischen Erkennung von Audioformaten und
- Kompressionsfunktionen.

Zur einheitlichen Integration der *player/recorder* wurde die vom MIME-Typ abhängige und im Internet weit verbreitete Art der Konfiguration übernommen. MIME-Typen werden systemweit definiert, die Referenzierung eines MIME-Typs eines Formates wird Anhand der *filename-extension* (Dateinamenserweiterung) festgelegt.

<main-typ>/<sub-typ>	**Extension, 2. Extension, 3. ...**

z.B.

audio/wav	**wav**
audio/ulaw	**au, ulaw**

In der Internetwelt wird diese Konfigurationsdatei innerhalb des HTTP-Servers zumeist einfach *mimetypes* genannt, innerhalb des PCSS wird sie als **.mmpcssmimetypes** gespeichert.

Die Zuordung der somit ermittelten MIME-Typen eines Formats zu den sie verwendenden Applikationen geschieht in einer weiteren Datei (normalerweise *mailcap* genannt). Da im Rahmen des PCSS jedoch sowohl aufnehmende Applikationen als auch darstellende verwendet werden, werden jeweils zwei Dateien für diese Zuordung benutzt und in Anlehnung **.mmpcssrecordcap** und **.mmpcssplaycap** genannt. Das Wort *cap* steht hier jeweils für *capabilities,* also Möglichkeiten. Durch diese Art der Konfiguration lassen sich systemweite Vorgabewerte realisieren, die der PCSS-Benutzer aber jederzeit durch seine persönliche Konfiguration überschreiben kann. Der Aufbau dieser Konfigurationsdateien sieht wie folgt aus:

<main-typ>/<sub-typ> <application-path> <parameters> %s

z.B.

audio/wav mmaudio -simple %s
text/text mmtext -recorder %s

Das Kürzel '%s' wird dabei vor dem Aufruf der Applikation durch den entsprechenden Dateinamen ersetzt. Dadurch lassen sich, bezogen auf den jeweiligen MIME-Typen, in einfacher Weise auch andere Applikationen zur Aufnahme und Wiedergabe einbinden, oder das PCSS um neue Formate ergänzen.

3.9 Integration

Die einzelnen Komponenten des MM-PCSS werden, wie aus der Abbildung 15 ersichtlich, von anderen PCSS-Modulen benutzt, die grau dargestellten Elemente beziehen sich auf die im Rahmen dieser Arbeit entwickelten Module.

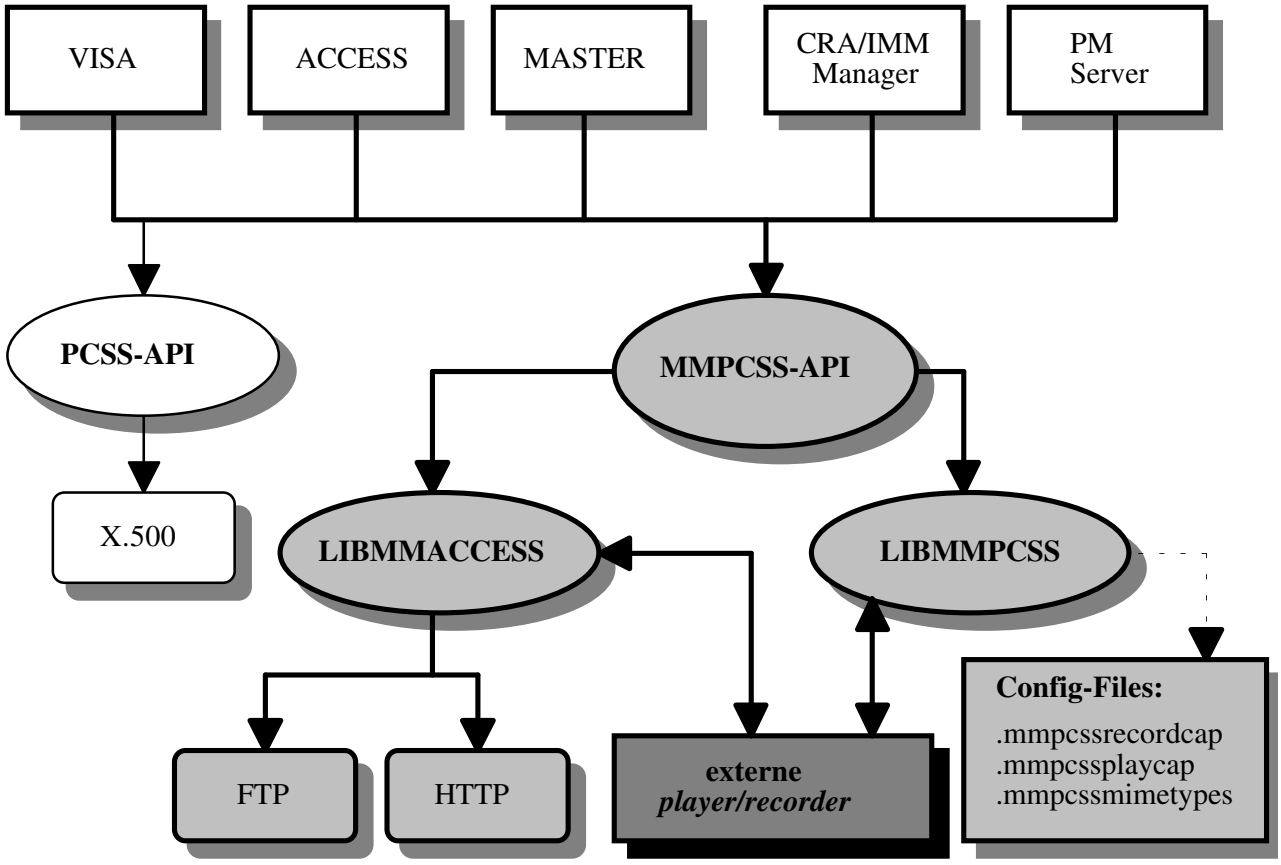


Abb. 15: *Integration des MM-PCSS*

Innerhalb der *player/recorder* Applikationen lassen sich mehrere Funktionsgruppen separieren, die auch anderen Modulen zugänglich gemacht wurden (via Tcl- und C-API).

Die weiß hinterlegten Libraries der Abbildung 16 müssen dabei allen Applikationen innerhalb des PCSS zugänglich gemacht werden, die auf den Message-Store zugreifen oder die die *player/recorder* Applikationen verwenden wollen.

Die hellgrau hinterlegten Libraries werden ebenfalls anderen Applikationen zugänglich gemacht, die direkten Zugriff auf das Audio*device* oder zusätzliche Funktionen zur Manipulation von Dateien benötigen.

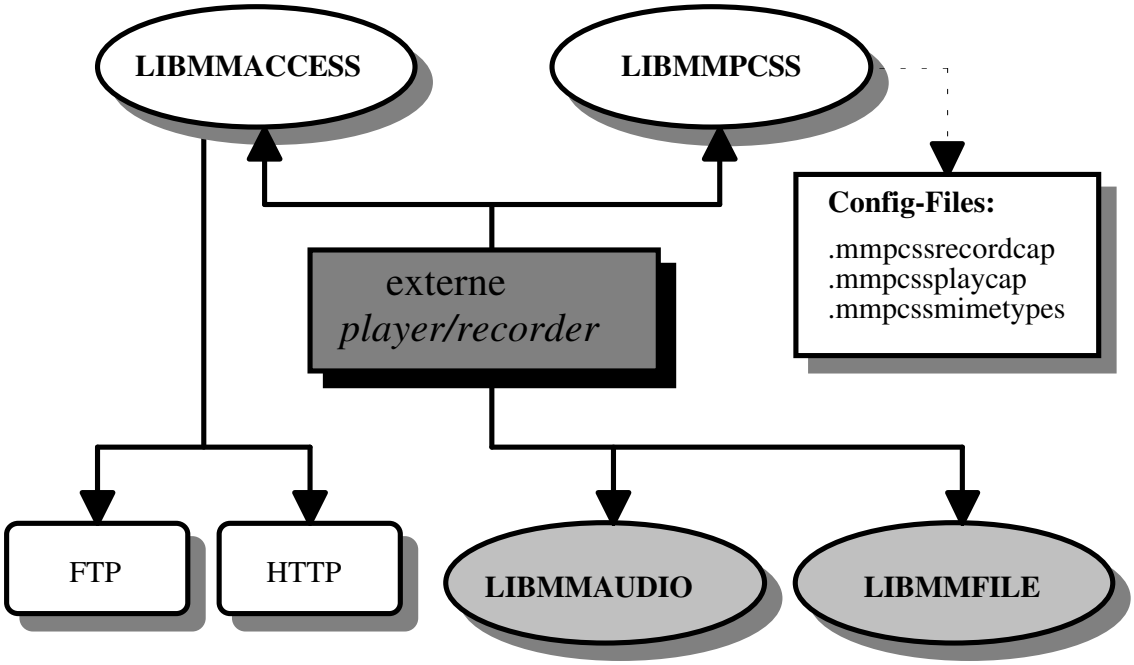


Abb. 16: *MM-libraries*

Der direkte Protokollzugriff via FTP oder HTTP sowie die direkte Manipulation der Konfigurationsdateien soll durch die Libraries LIBMMACCESS bzw. LIBMMPCSS vor den sie verwendenden Applikationen versteckt werden.

Kapitel 4 - Implementierung

Die Implementierung der konzeptionell im vorherigen Kapitel vorgestellten multimedialen Erweiterung des PCSS mit einer Integration eines Message-Stores mit Hilfe von Protokollen auf Basis von TCP/IP wurde im Sommer 1995 am Computernetz des Fachbereich OKS in den Räumen der GMD Fokus begonnen und Ende 1995 abgeschlossen.

In diesem Kapitel soll diese Implementierung, ihre Struktur und ihre Module, sowie die Lösungen praktischer Probleme und die Realisierung der Integration mit den anderen Modulen des PCSS erläutert werden. Hierbei werden auch die exemplarisch zu entwickelnden, multimedialen Aufnahme- und Abspielprogramme des MM-PCSS vorgestellt.

Am Ende des Kapitels sollen die entwickelten Mechanismen und Applikationen getestet und bezüglich ihrer Wiederverwendbarkeit analysiert werden.

4.1 Message-Store

Der Message-Store wurde auf einem der Workstations, Modell SparcStation 20 mit dem Betriebssystem Solaris 2.4 konfiguriert, wobei die vom PCSS via URLs referenzierten Multimediadateien in einem Unterverzeichnis des FTP-Servers abgelegt werden.

Als FTP-Server wurde die Implementierung der Firma Sun verwendet, die bereits im Betriebsystem Solaris 2.4 enthalten ist. Der Server mußte jedoch für anonymen Zugriff konfiguriert werden. Innerhalb des PCSS-Unterbaums im FTP-Bereich wurden dann für jeden PCSS Benutzer Unterverzeichnisse angelegt. Dadurch lassen sich hinterlassene Nachrichten eindeutig demjenigen Benutzer zuordnen, der die Nachricht erhalten hat. Da die PCSS-Module des Phone-Mail-Servers und des VISA-Systems Dateien innerhalb von Unterverzeichnissen anderer Benutzer hinterlassen und somit eine Schreibberechtigung auf diese Verzeichnisse benötigen, wurden, um nicht zusätzlich umständliche Sicherheitsmechanismen entwickeln zu müssen, alle Verzeichnisse für jeden Anwender schreibbar gemacht.

Da dies jedoch aus sicherheitstechnischen Gründen eine sehr unbefriedigende Lösung darstellt, wurde bereits eine Benutzerkennung und ein Benutzerpasswort in das persönliche User-Profile aufgenommen, welches zu einem späteren Zeitpunkt der Implementierung des PCSS-Systems zur Identifizierung des Benutzers gegenüber dem Message-Store benutzt werden wird. Der Phone-Mail-Server und VISA müssen dann als privilegierte Applikationen auf den Message-Store zugreifen können (die zu den Applikationen gehörigen Unix-Benutzer sollten dann als FTP-Administratoren eingetragen werden).

Weiterhin wurde der FTP-Server so konfiguriert, daß gleichzeitig mehrere Dateien in den Message-Store überspielt werden können.

Als HTTP-Server wurde die frei verfügbare Implementierung der Universität Cern, Schweiz, (httpd-1.3.tar.gz, zu finden auf URL=http://www.cern.de) auf demselben Computer installiert und konfiguriert. Um eine einfache Abbildung lokaler Dateinamen auf URLs und umgekehrt, sowie eine

Abbildung der zum Wurzelverzeichnis des FTP-Servers relativen Dateinamen auf globale URLs und umgekehrt zu vereinfachen, wurde als Wurzelverzeichnis des HTTP-Servers das des FTP-Servers benutzt.

4.2 Modulstruktur

In der Abbildung 16 aus Seite 55 wurde bereits die Modulstruktur des MM-PCSS gezeigt. Grundsätzlich sind alle Libraries in der Programmiersprache C realisiert, die für andere Applikationen innerhalb des PCSS sowie für alle *player/recorder* notwendigen Funktionen werden via Tcl/Tk-Aufrufe zur Verfügung gestellt (*function wrapping*).

Somit müssen alle Applikationen, die Funktionen einer dieser Libraries benutzen wollen, ein simples C-Hauptprogramm benutzen, um die Libraries zu initialisieren. Diese Vorgehensweise ist Tcl/Tk-intern notwendig und bedingt dann die Erzeugung eines applikationsabhängigen Tcl/Tk-Interpreters (siehe dazu auch die Softwarebeschreibung im Anhang auf Seite 91).

4.3 Libraries

Im folgenden sollen die Funktionen der implementierten Libraries [16] genauer erklärt werden, ohne jedoch auf die Aufrufstruktur einzugehen (diese findet man im Anhang B.2 auf Seite 92). Die PCSS-relevanten (also nicht MM-PCSS internen) Funktionen können sowohl von C-, C++ oder Tcl/Tk aus aufgerufen und benutzt werden.

4.3.1 LIBMMPCSS

Die *player/recorder* für die verschiedenen Medien-Typen wurden als eigenständige Applikationen konzipiert, um sie, erstens in mehreren anderen Systemen verwenden zu können, zweitens mehrere *player/recorder* gleichzeitig benutzen zu können und drittens bei Änderung von Teilkomponenten des PCSS-Systems sowie bei der Integration neuer Datenformate flexibel und generisch reagieren zu können.

Um den Prozess einer Aufnahme und der Darstellung von multimedialen Datenströmen von der Verwendung der eigentlichen externen Applikationen zu trennen, wurde auf den Verweismechanismus der WWW-*browser* zurückgegriffen. Datenströme werden mittels MIME-Typ spezifiziert, die jeweiligen Applikationen des Formats den MIME-Typen zugeordnet.

Die Ausführung der richtigen Applikation für den richtigen Datenstrom ist eine der Hauptaufgaben der Library MMPCSS. Sie überprüft sowohl globale, systemweite als auch benutzerspezifische Einstellungen und startet die externe Applikation. Bei der Darstellung kehrt die entsprechende Funktion innerhalb der Library MMPCSS zum aufrufenden Programm zurück. Wird eine Aufnahme eines Datenstroms gewünscht, startet die entsprechende Funktion den eingestellten *recorder* und blockiert bis dieser *recorder* terminiert. Da im vorherein nicht bekannt sein kann,

[16] Unter dem Begriff Libraries wird im folgenden die Zusammenfassung von verschiedenen Funktionen und Prozeduren im programiertechnischen Sinn verstanden. Im Deutschen werden diese als Funktionsbibliotheken bezeichnet.

unter welchem Dateinamen der Benutzer eine aufgenommene Datei abspeichert und zumindesten Video- und Audio*recorder* die Länge des aufgenommenen Datenstroms zurückliefern sollen, muß eine Kommunikation zwischen der Library MMPCSS und den externen *recorder*n stattfinden. Die Kommunikation zwischen Library und externen *recorder* wurde mit den sogenannten *Cutbuffern* [17] des XServers realisiert, welche für alle Applikationen schreibbar sind. Hat ein *recorder* eine Datei erfolgeich gespeichert, muß er, bevor er terminiert, *Cutbuffer* 1 mit dem Dateinamen und *Cutbuffer* 2 mit der *duration* füllen. Diese *Cutbuffer* werden vor dem Aufruf der externen *recorder* gelöscht. Sollte daher eine nicht innerhalb des PCSS entstandene Applikation als *recorder* eingesetzt werden, muß diese mit einem voreingestellten Dateinamen aufgerufen und auf die Verwendung der *duration* verzichtet werden. Vor dem eigentlichen Aufruf kann durch zwei Funktionen geprüft werden, ob der entsprechende MIME-Typ im System bekannt, bzw. ob eine Applikation zur Darstellung oder Aufnahme dieses Types eingestellt ist.

Weiterhin ermöglicht die Library MMPCSS, die Konfigurationsdateien des Benutzers zu modifizieren und mit den globalen Konfigurationsdateien abzugleichen. Eine manuelle Editierung der benutzerspezifischen Konfigurationsdateien innerhalb des PCSS ist also nicht nötig. In den globalen Konfigurationsdateien vorgenommene Erweiterungen werden automatisch in die persönliche Konfiguration der Benutzer übernommen.

Eine zusätzliche Funktion erzeugt außerdem temporäre Dateinamen zur Speicherung von Dateien im Message-Store. Somit werden bestehende Nachrichten nie durch neu eingehende überschrieben.

4.3.2 LIBMMACCESS

Diese Library stellt haupsächlich die Kommunikation mit den für den Message-Store spezifizierten Protokollen FTP und HTTP zur Verfügung, wobei letztere soweit als möglich verborgen werden sollen. Die Kommunikation mit dem Message-Store wird auf einfache Funktionen reduziert, wie:

- das Öffnen einer im Message-Store gespeicherten Datei,
- das Lesen einer Anzahl von Daten einer so geöffneten Datei,
- das Lesen einer Anzahl von Zeilen einer so geöffneten Datei, falls es sich um ASCII-Text handelt,
- das Übertragen einer Datei aus dem Message-Store in das lokale Filesystem via HTTP,
- die Überprüfung, ob eine Datei komplett gelesen wurde,
- das Schließen einer Datei,
- das Übertragen einer Datei aus dem lokalen Dateisystem in den Message-Store eines Benutzers via FTP,
- das Ändern des Datums und der Zugriffsrechte einer Datei im Message-Store und
- das Löschen einer Datei im Message-Store.

Dabei ermöglichen mehrere Funktionen der Library MMACCESS die Umkodierung von Dateibezeichnung in URLs und umgekehrt, z.B. liefert der Aufruf der Funktion zum Speichern im Message-Store nicht nur den Erfolg der Operation, sondern auch eine gültige URL als Verweis auf die gespeicherte Datei, die somit sofort in der Message-Box gespeichert werden kann.

Zum Test der Funktionsweise der in den Libraries LIBMMPCSS und LIBMMACCESS enthaltenen Funktionen ist eine weitere Applikationen mit Tcl/Tk-Interface entstanden (siehe

[17] Als *Cutbuffer* bezeichnet man innerhalb des XServers der graphischen Oberfläche X11 bis zu acht freie, allgemein zugängliche Speicherbereiche, die insbesondere zum Austausch von Daten via *cut&paste* benutzt werden.

Abbildung 17 auf Seite 59). Sie sollte haupsächlich verwendet werden, um während der Entwicklung anderer Applikationen die Anbindung dieser an den Message-Store zu testen und zu überprüfen.

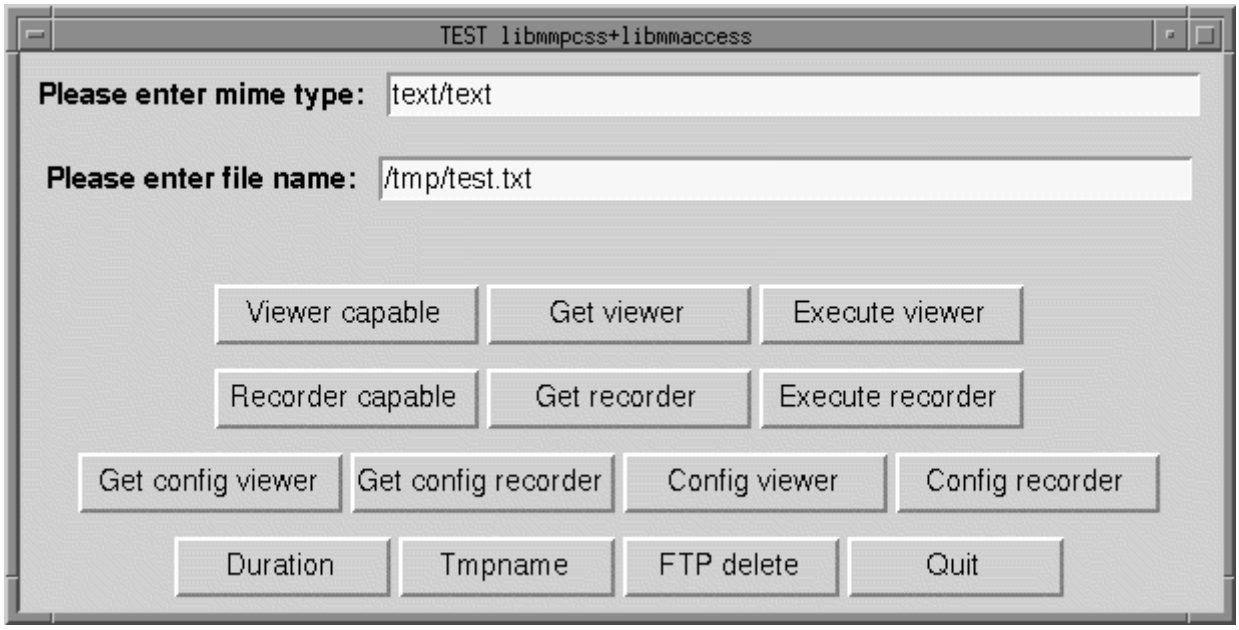


Abb. 17: *Interface des Library-Testprogrammes*

4.3.3 LIBMMAUDIO

In dieser Library sind alle Funktionen implementiert, die die Anbindung der Audio*devices* von Solaris bzw. Linux ermöglichen. Die Anbindung an das PC-basierte Unixsystem "Linux" wurde implementiert, um die oberen Schichten plattformunabhängig zu gestalten, eine Plattformunabhängigkeit der Audioformate zu garantieren und Fehler in der Implementierung durch sogenanntes *cross-compiling* [18] schneller zu lokalisieren. Alle entwickelten Funktionen lassen sich auch von Tcl/Tk aus benutzen.

Implementiert wurden Funktionen, um

- das Audio*device* und das Audio*controldevice* [19] zu öffnen,
- das Audio*device* anzuhalten, dessen Buffer zu löschen und es zu schließen,
- den aktuellen Status des Audio*devices* abzufragen,
- die Lautstärkenskalierung einzustellen,
- die Aufnahme- und Wiedergabelautstärke einzustellen,
- die Eingangs- und Ausgangsleitungen innerhalb des Mischers zu schalten,
- die Audiosignale aufzunehmen und in einem Puffer oder einer Datei abzulegen,
- das Abspielen von Audiodateien bzw. Audiopaketen mit Hilfe des HTTP-Protokolls zu ermöglichen,

[18] Als *cross-compiling* wird normalerweise der Vorgang bezeichnet, ein ausführbares Programm auf einer Hardwareplattform zu erzeugen, welches jedoch für eine andere Hardware-Plattform bestimmt ist. Auf der Zielplattform benötigt man somit keine Entwicklungsumgebung. Vom *cross-compiling* wird insbesondere bei der Entwicklung von neuen Betriebssystemvarianten Gebrauch gemacht, um dort erstmalig ein Entwicklungssystem zu generieren. Hier wurde *cross-compiling* eingesetzt, um Entwicklungsfehler schneller zu lokalisieren.

[19] Welches zur Einstellung des Audiomischers, z.B. *LineIn/LineOut* oder der Lautstärke und zur Einstellung von Hardware-Komprimierungsparametern angesprochen werden muß.

- das Abspielen anzuhalten sowie "an den Anfang zurückzuspulen",
- RAW-PCM Daten in das MPEG-Format zu komprimieren.

Unterstützt werden die Formate

- G.711 linear, μ-law oder a-law
- G.721 ADPCM
- RAW-PCM jeder Art
- WAV [20]
- MPEG 16-bit, Mono/Stereo, 33, 44.1 und 48 kHz

Die MPEG-Unterstützung erstreckt sich hierbei auf die drei im Standard festgeschrieben *sampling*-Frequenzen. Die MPEG-Echtzeitdekodierung wurde über eine signalgesteuerte Anbindung einer externen Applikation ermöglicht (siehe dazu den Abschnitt: Player/Recorder - MMAUDIO auf Seite 64). Während des Abspielens, des Aufnehmens und der Komprimierung von Audiodaten gibt die Library MMAUDIO in regelmässigen, die Echtzeit des jeweiligen Vorgangs nicht gefährdenden, Abständen die Bearbeitung an andere eventuell wartende Applikationen, bzw. die aufrufende Tcl/Tk-Applikation ab. Somit wird sichergestellt, daß das Interface seinen Inhalt neu darstellen und der Benutzer weiterhin mit der Applikation interagieren kann.

4.3.4 LIBMMFILE

Die Library MMFILE mußte implementiert werden, um den vielfältigen Zugriffs- und Verwaltungsmöglichkeiten von Dateien innerhalb des UNIX Dateisystems Rechnung zu tragen. Viele Funktionen waren nicht von Tcl/Tk aus verfügbar und andere mußten neu entwickelt werden, wie:

- das Kopieren einer lokalen Datei,
- das Löschen einer lokalen Datei,
- das Umbenennen einer lokalen Datei,
- das Erzeugung von temporären Datei- und Verzeichnisnamen und
- die sinnvolle Beschneidung von langen Dateinamen zur Darstellung im Interface bei begrenztem Platz.

4.4 Player/recorder

Mehrere Applikation wurden zur Darstellung von multimedialen Datenströmen (*player*) und zur Erzeugung bzw. Importierung derselben (*recorder*) implementiert. Applikationen für Text, Bild und Audio sind entstanden, die in Anfängen realisierte Videoapplikation wird basierend auf den Vorgaben und Resultaten und Empfehlungen dieser Arbeit momentan im Bereich einer Projektaufgabe einer Veranstaltung im Bereich OKS der TU Berlin erweitert und vervollständigt.

Folgende Vorgaben sind bei der Implementierung berücksichtigt worden:

---

[20] Das Audio*device* der SparcStation ist begrenzt. Insbesondere im 8-bit Bereich werden nicht alle Aufnahme- und Abspielfrequenzen unterstützt. Im 16-bit Bereich werden die meisten gängigen Frequenzen wie 8, 11, 22, 33, 44, sowie die Studioauflösung von 48 kHz sowohl Mono als auch Stereo unterstützt.

- **einheitliches Erscheinungsbild beim Benutzer:** Schon früh hatte sich die Projektgruppe zur Implementierung des PCSS-Systems auf die Verwendung der scriptbasierten Programmier- und graphischen Schnittstellensprachen Tcl, Tk und Tix festgelegt. Somit erhalten alle im Rahmen des PCSS entstehenden Programme eine einheitliche Repräsentierung beim Benutzer, z.B. werden Menüs immer gleich benutzt, Eingabefelder sehen immer ähnlich aus und Dateiselektordialoge stellen sich gleich dar.

- **einheitliche nach außen verfügbare Schnittstellen:** Tcl/Tk erlaubt nicht nur Interprozesskommunikation, sondern bietet auch, da es sich um eine Interpretersprache handelt, die Möglichkeit, Programmcode anderer Applikationen in die eigene zu integrieren, bzw. während der Laufzeit zu benutzen. Weiterhin bietet Tcl/Tk die Möglichkeit, Code der Programmiersprache C an die eigene Applikation zu binden, bzw. den für die eigene Applikation entwickelten C-code in einer Library zu sammeln und via Tcl/Tk-Aufrufen anderen Applikation nutzbar zu machen.

- **einheitlicher Funktionsumfang:** Soweit wie möglich sollen von den gleichen Stellen des sichtbaren Interfaces die gleichen Funktionalitäten abrufbar sein. Weiterhin soll die grundsätzliche Handhabung und der Arbeitsablauf der Applikationen ähnlich sein.

- **variable Darstellung der Applikationen:** Verschiedene Komplexitätsgrade der Benutzerschnittstellen der Applikationen müssen konfigurierbar sein. Applikationen, die sowohl als *player* und auch als *recorder* ihrer Medientypen verwendbar sind, müssen sich zu unterschiedlichen Zeiten der Benutzung angepaßt und somit unterschiedlich darstellen.

- **Anpassung an den Message-Store:** Die Rezeption einer aufgezeichneten Nachricht muß in Anpassung an den Message-Store über das Protokoll HTTP in Echtzeit erfolgen, d.h. die Daten, die paketweise, von der Datei aus dem Store gelesen werden, werden sofort zur Darstellung gebracht. Die Aufnahme der Nachrichten muß jedoch lokal geschehen, bevor die fertiggestellte Nachricht in den Message-Store überführt wird. Daraus folgt, daß alle Applikationen explizit eine unterschiedliche Darstellung von lokalen und im Message-Store abgelegten Nachrichten unterstützen müssen. Eine generische Lösung kann nur ansatzweise über die Implementierung der MMACCESS-Library realisiert werden. Das Problem der Speicherung einer fertig aufgenommenen Nachricht im Message-Store kann jedoch für alle Applikationen (und unabhängig von diesen) gelöst werden. Nach Beendigung der aufnehmenden Applikationen wird explizit durch die, die einzelnen Applikationen versteckenden, MMPCSS-Library über eine Speicherung und Weiterverarbeitung der Nachricht entschieden.

Alle folgenden Applikationen sind sowohl als *player* als auch als *recorder* konzipiert. Sollten während des Arbeitsablaufes *recorder*-spezifische Funktionalitäten oder spezielle Aufrufe unabdingbar sein, wird dieser Unterschied durch einen Aufrufparameter ("-recorder") spezifiziert. Alle Applikationen enthalten Funktionen zum Aufnehmen bzw. Laden eines multimedialen Datenstroms, zum Speichern desselben, zur Versionsanzeige, zum Terminieren der Applikation und zur Darstellung von Hilfetexten. Diese Menüs (siehe Abbildung 18 auf Seite 62) sind auch via Tastatur zu bedienen.

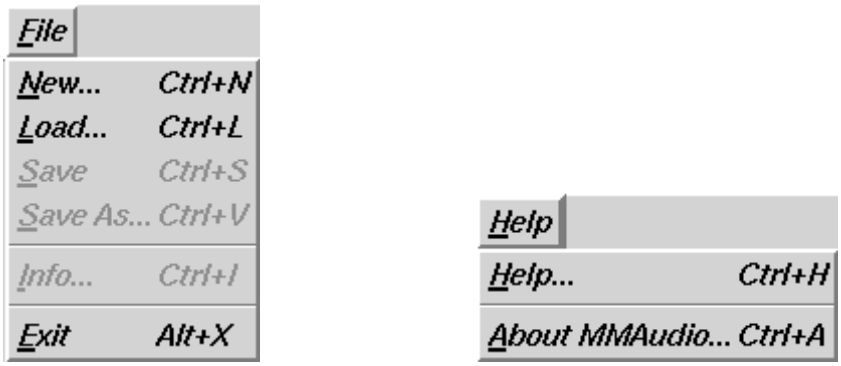


Abb. 18: *Standard Datei- und Hilfemenü*

Die Applikationen benutzen alle die Funktionen der MMACCESS Library, um Dateien paketweise vom Message-Store zu lesen.

Weiterhin enthalten sie Funktionen zum Parsieren der Dateinamenerweiterung der zu bearbeitenden Datei. Damit (und einem Parameter zur expliziten Nennung eines MIME-Typen) kann der MIME-Typ des Datenstroms bestimmt werden, bevor dieser zur Darstellung gebracht wird. Sollte dies nicht zur Identifizierung ausreichen werden Funktionen zur Analyse des Dateiformats benutzt (z.B. analysiert die Audioapplikation MMAUDIO die *header* der Dateien).

Es werden Balkendiagramme zur Darstellung des Umfangs der vom Message-Store entgegen-genommenen Daten benutzt. Sowohl benutzte Dateinamen als auch der letzte bzw. aktuelle Status der Applikationen wird zur Anzeige gebracht.

Alle Applikationen erscheinen mindestens in zwei in der Komplexität verschiedenen Interface-varianten. Die gewünschte Darstellungart wird durch einen Aufrufparameter festgelegt, der natürlich auch in die MM-PCSS Konfigurationsdateien eingetragen werden kann. Keine der Applikationen bietet die Möglichkeit, verschiedene Kodierungen in andere Formate zu konvertieren. Dieses Problem muß in zukünftigen Arbeiten gelöst werden.

Für die vier Medientypen Text, Bild, Audio und Video sind folgende Applikationen entstanden:

4.4.1 MMTEXT

Implementiert ist ein simpler ASCII-Editor, der sowohl zur Darstellung der Textdateien als auch zu deren Bearbeitung geeignet ist.

Der Editor basiert auf dem vom Tix zur Verfügung gestellten *widget* [21]. Funktionen zum Einfügen und Löschen von Zeichen, sowie zur Positionierung des Cursor sind bereits integriert.

Das Interface von MMTEXT (siehe Abbildung 19) steht als menügesteuerte Applikation oder als reine Editierapplikation zur Verfügung.

[21] Als *widget* bezeichnet man ein im Funktionsumfang gekapseltes Element, welches wegen des zur Verfügung stehenden, einfachen Interface, und der Parametrisierbarkeit desselben, zur Einbindung in andere Programme geeignet ist.

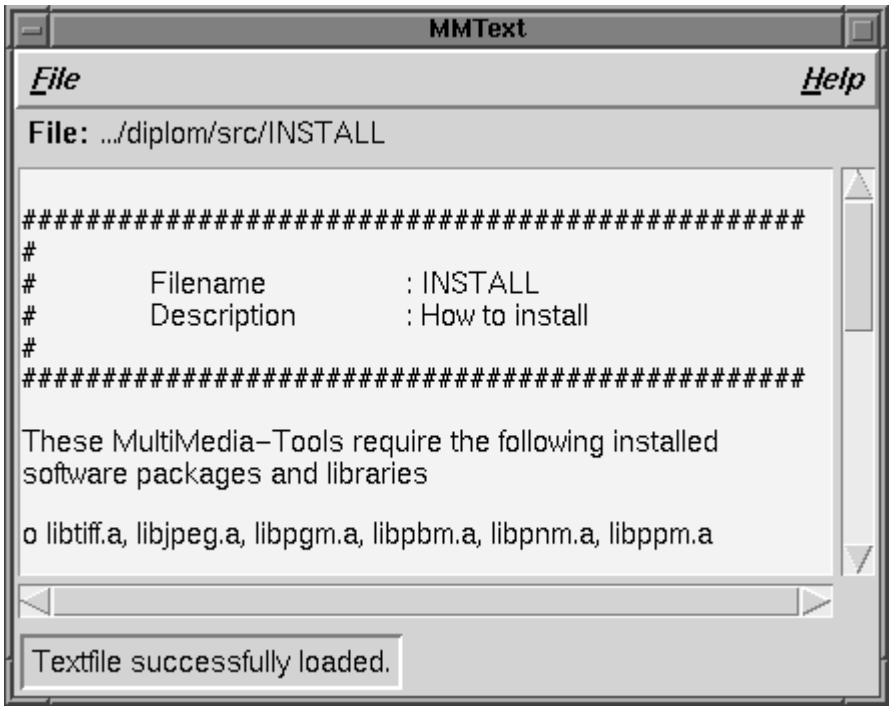


Abb. 19: *MMTEXT-Interface*

4.4.2 MMPICS

MMPICS stellt die ausgewählten Bildformate wie in Abbildung 20 dar. Unterstützt werden die Formate GIF, TIFF, XBM, XPM und alle PBM-Varianten. Das Interface paßt sich dabei der Größe des entsprechenden Bildes an. MMPICS ist nicht als *recorder* konzipiert, es bietet jedoch die Möglichkeit, Bilder zu laden bzw. zu importieren. Ein Bild*recorder* wäre z.B. ein Scannergerät oder ein Zeichenprogramm. Die Anbindung eines Scanners an das VISA-System kann, zur Übermittlung von mitgebrachten schriftlichen Mitteilung, durchaus als sinnvoll gelten. Die Aufnahme eines Standbildes durch die Videoapplikation kann auch eine sinnvolle Erweiterung darstellen, sollte jedoch nicht in die Bildapplikation MMPICS integriert werden.

Abb. 20: *MMPICS-Interface*

Das Interface von MMPICS (siehe Abbildung 20) steht als menügesteuerte oder als darstellende Applikation zur Verfügung.

4.4.3 MMAUDIO

Die vom Funktionumfang weitaus umfangreichste Applikation ist MMAUDIO, der Audio*recorder* und *-player*. Sowohl auf eine benutzergeführte Interfacegestaltung als auch auf die Unterstützung vielfältiger Audioformate wurde großer Wert gelegt. Selektierbar sind durch eindeutige Symbole repräsentierte Knöpfe, wie sie von z.B. von einem Kassetten- oder Videorekorder bekannt sind. *Play*, *Record*, *Pause*, *Stop* und *ToStart* (Sprung an den Anfang des Audiostroms) sind implementiert. Weiterhin können sowohl das Audioformat als auch alle relevanten Kodierungsparameter (entsprechend des gewählten Formates) eingestellt werden. Ergänzt wird das graphische Interface durch Schieberegler zur Regulierung der Lautstärke und der Aufnahmeempfindlichkeit. Weiterhin kann zwischen angeschlossenem Mikrofon und dem *Line-In*-Eingang umgeschaltet werden, um Aufnahmen von anderen, analogen Audioquellen zu ermöglichen.

Unterstützt wird die Aufnahme und das Abspielen von MPEG-1-Audio, µ-law, a-law, RAW-PCM, ADPCM und WAV. Die Aufnahme und Kodierung aller Formate geschieht dabei in Echtzeit, nur MPEG-1-Audio wird zuerst als eine Folge von linearen PCM-*samples* aufgenommen und nachträglich kodiert. Somit erfolgt die langwierige MPEG-Kodierung nur bei Aufnahmen, die explizit vom Benutzer gewünscht wurden.

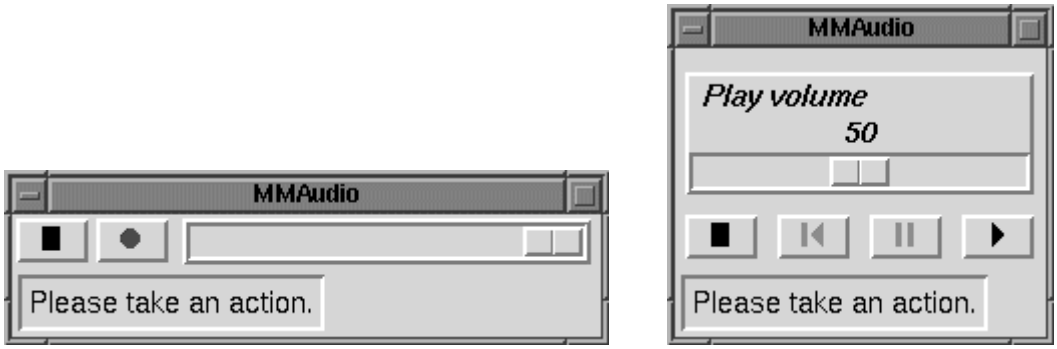


Abb. 21: *MMAUDIO-Interface (links: einfach, rechts: wenig komplex)*

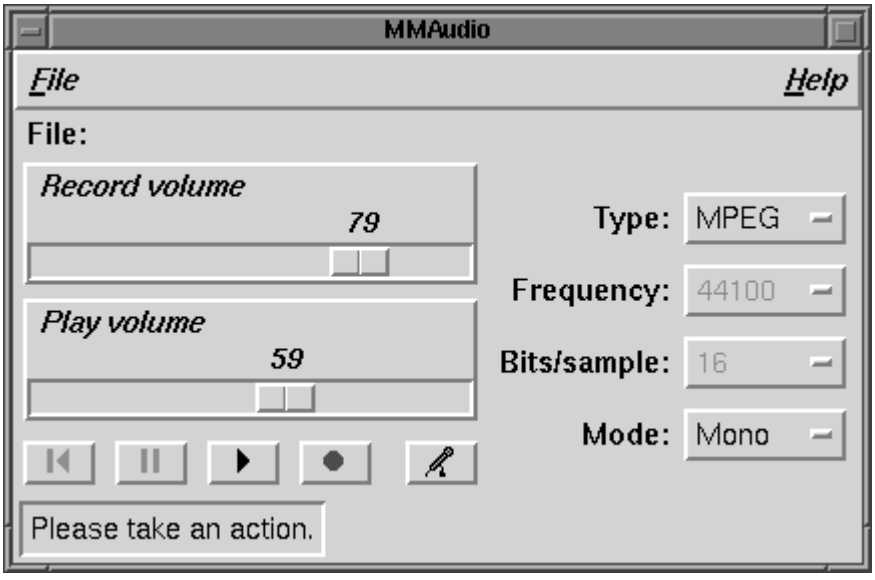


Abb. 22: *MMAUDIO-Interface (komplex)*

Das Interface von MMAUDIO steht in drei Grundvarianten zur Verfügung: einfach, mittel und komplex (letzteres siehe Abbildung 21). Das weniger komplexe Interface (siehe Abbildung 21, rechts) eignet sich für Aufnahme und Wiedergabe, bei denen nicht zwischen Formaten gewechselt werden muß, also der MIME-Typ schon feststeht. Das einfache Interface (siehe Abbildung 21, links) ist zur Einbindung in andere Applikationen gedacht, die zusätzlich Audiounterstützung benötigen. Weiterhin lassen sich alle Varianten explizit als *recorder* und mit einer für Touchscreens [22] geeigneten Größe der Knöpfe starten und konfigurieren. Zur weiteren Unterstützung des Benutzers wird die Interaktion mit Menüs und Schaltern nur erlaubt (graue Schaltflächen sind dabei inaktiv), wenn die entsprechende Funktion im Kontext der momentanen Ausführung sinnvoll bzw. erlaubt ist, z.B. kann der Benutzer während der Aufnahme weder die Aufnahmelautstärke noch das Format ändern.

Um das Format MPEG zu unterstützen, wurde die Kodierungssoftware der CCITT umgeschrieben, angepaßt und als Teillibrary (LIBENCODE) zur Verfügung gestellt. Als Dekodierer wurde die Entwicklung MAPLAY von Tobias Bading, die im Rahmen eines OKS-Projektes fertiggestellt wurde, als externe Applikation angebunden. Die Steuerung derselben erfolgt über sogenannte Unix-*Signals*, d.h. MAPLAY kann durch die Applikation MMAUDIO jederzeit gestartet, angehalten oder gestoppt werden. Die eigentlichen Daten werden dabei jedoch von der Applikation MMAUDIO (z.B. vom Message-Store) gelesen und mit Hilfe des *named-pipe*-Konzeptes [23] an MAPLAY zur Dekodierung weitergereicht.

4.4.4 MMVIDEO

Als erster Prototyp ist die in Abbildung 23 gezeigte Applikation entstanden, die zumindestens MPEG-1-Videoströme aus dem Message-Store lesen und dann darstellen kann.

Abb. 23: *MMVIDEO-Interface*

[22] Ein Touchscreen ist ein spezieller Computermonitor mit drucksensitiver Oberfläche und wird z.B. im VISA-System benutzt. Die menschliche Hand ersetzt also die Computermaus zur Selektion von Objekten.

[23] Eine *named pipe* innerhalb des UNIX-Dateisystems verhält sich wie ein Puffer gewisser Größe. Von ihr kann gelesen und in sie kann geschrieben werden, wie in jede andere Datei, ihre Größe bleibt jedoch konstant und eignet sich deshalb besonders um eine dateibasierte Kommunikation zwischen mehreren Applikationen herzustellen. Somit müssen Lese- und Schreibfunktionen der Applikationen nicht an eine eventuelle Interprozesskommunikation angepaßt werden.

Die vollständige Implementierung der erforderlichen Videoapplikation hättte den Rahmen dieser Arbeit gesprengt. Vielfältige Probleme, wie die gleichzeitige Dekodierung von MPEG-1-Video- und -Audioströmen per Software, sowie die synchrone Aufnahme von Video und Audio und das Multiplexing beider Medien, konnten in der Applikation nur in Ansätzen implementiert werden.

Die Aufnahme der Videos muß momentan noch mit der von der Firma Sun mitgelieferten Applikation SunVideo erfolgen.

4.4.5 MMSAVE

Weiterhin ist eine zusätzliche Applikation entstanden, die es ermöglicht, eine via URL spezifizierte, entfernte Datei in das lokale Dateisystem zu speichern.

Nach dem Start der Applikation öffnet diese eine Verbindung zum HTTP-Server, öffnet ein Dateiselektorfenster, wie in Abbildung 24 gezeigt, um dem Benutzer die Eingabe eines Dateinamens zum Speichern zu ermöglichen und transferiert dann die Datei.

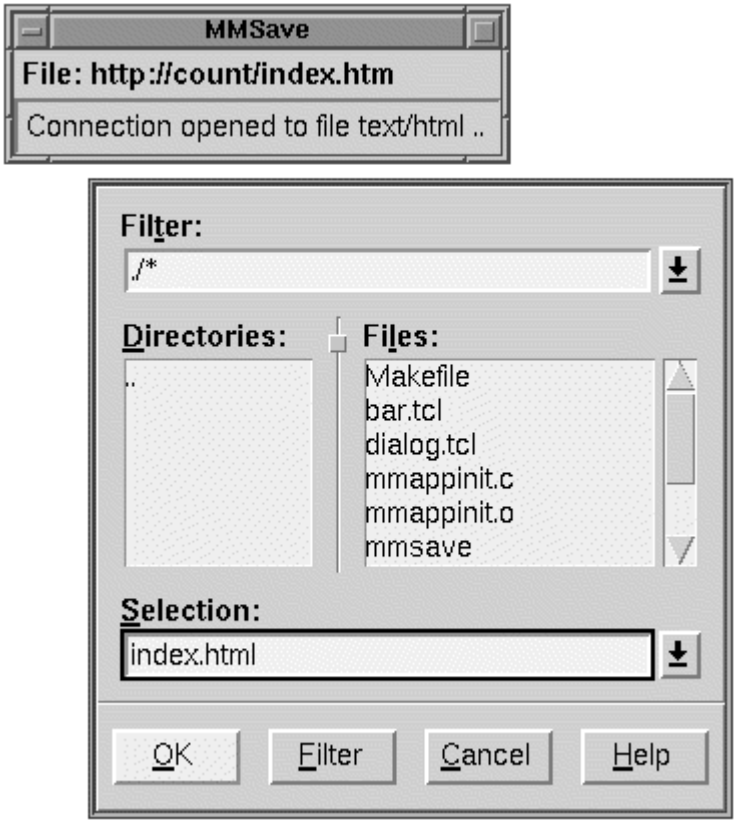


Abb. 24: *MMSAVE-Interface*

4.5 Test der Mechanismen, Protokolle und Dienste

Die implementierten Multimediaformate und Dienste sollen hier nun hinsichtlich ihrer:

- Hardware- und Plattformunabhängigkeit,
- Schnelligkeit,
- Brauchbarkeit bezüglich internationaler Konventionen,
- weitergehenden Integration in bestehende andere Multimediasysteme und -dienste und
- ihres niedrigen Anspruchs an vorauszusetzende Hard- und Software

getestet und gewichtet werden.

Insbesondere die Videoformate sind hardwareabhängig. Sie müssen nicht nur durch zusätzliche Hardware erzeugt werden, sondern auch durch Hardware komprimiert werden, um sie via TCP/IP übertragen zu können. Diese Hardwareabhängigkeit setzt sich natürlich bis in die sie verwendenden Applikationen fort. Die Formate müssen jeweils immer wieder reimplemtiert werden. Deshalb hätte die Implementierung einer entspechenden, umfassenden Applikation den Rahmen dieser Arbeit gesprengt.

Audioformate sind auch von der aufnehmenden und darstellenden Hardware abhängig, die grundsätzlichen Parameter, wie Frequenz, Bittiefe der *samples* und die Anzahl der verwendeten Kanäle, können jedoch zumeist an jeder Hardware einfach eingestellt werden. Da die Kodierung und Dekodierung der Audioformate in Software implementiert werden kann, ist bei den aufnehmenden und darstellenden Applikationen nur die unterste Hardwareanbindung zu realiseren gewesen. Allerdings muß bei den Audiodatenströmen die *byteorder* [24] beachtet werden.

Die Hardwareabhängigkeit der Video- und Audioformate bezieht sich auch auf die Darstellung bzw. Aufnahme. Das Audio- bzw. Video*device* kann jeweils nur von einer Applikation auf einem Rechner verwendet werden. Schon allein deshalb kommt es nur bei der gleichzeitigen Darstellung von Video- und Audiodateien zu Performanzproblemen. Die Aufnahme mehrerer Video- oder mehrerer Audioströme wird durch das Solaris-Betriebssystem nicht unterstützt.

Sowohl die verwendeten Bild- als auch die Textformate sind hardwareunabhängig (bei den Bildern muß gegebenenfalls die *byteorder* beachtet werden). Die Kodierung und Dekodierung ist komplett via Software möglich. Die gleichzeitige Darstellung und Aufnahme von Texten und Bildern andererseits ist durch das Betriebssystem bzw. die Hardware zu bewältigen.

Sowohl MPEG (Video und Audio), als auch G.711 und G.721 sind international standardisiert und werden im Umfeld der GMD Fokus zusammen mit den typischen Bildformaten unter dem Betriebsystems Solaris, wie XBM und PGM, häufig benutzt. Innerhalb des Internets werden die Formate GIF und WAV als gebräuchlicher Standard benutzt. Die Transportprotokolle FTP und HTTP werden im Internet als auch in anderen Netzen häufig eingesetzt und durch genügend (auch kommerzielle) Soft- und Hardware unterstützt, sodaß sie, obwohl eine weltweite Standardisierung z.B. durch die ISO fehlt, als de-facto-Standard zu bezeichnen sind.

---

[24] Bei verschiedenen Hardwareplattformen werden das höherwertige und das niederwertige *byte* eines 16-bit-Wertes oft in verschiedener Reihenfolge gespeichert. Die *samples* eines Audiodatenstroms in einem bestimmten Format müssen auf bestimmten Plattformen eventuell gedreht werden.

4.5.1 Performanz

Alle verwendeten Formate sind in Echtzeit von den verwendeten Computern innerhalb des PCSS erzeug- und wieder darstellbar.

Eine Ausnahme bildet MPEG-Audio, der PCM-Datenstrom muß nachträglich in MPEG-Audio umgewandelt werden. Dies benötigt auf einer verwendeten SparcStation-20 ca. die dreifache Abspieldauer; ist z.B. die Audionachricht 10 Sekunden lang, werden ca. 30 Sekunden zur Kodierung benötigt. Auf einer SparcStation 5 beträgt dieses Verhältnis 1:5. Problematisch ist die Darstellung von MPEG-Video mit synchronisiertem Audio. Dies stößt an die Grenzen der verfügbaren Rechenleistung. Hier ist allerdings noch keine entgültige Entscheidung gefallen, ob MPEG-Video z.B. auch mit G.711 synchronisiert werden kann.

Um die Übertragungsleistung der verwendeten Netze und Protokolle zu überprüfen, wurden zuerst die bestehenden Netzwerke hinsichtlich ihres Datendurchsatzes gemessen. Als Ping-Zeit wird diejenige Zeit angegeben, die verstreicht, bis der angesprochene Rechner auf das erste an ihn geschickte TCP/IP Datenpaket geantwortet hat (siehe Tabelle 5). Dieser Wert muß z.B bei der Anfangsverzögerung einer Übertragung aus dem Message-Store berücksichtigt werden. Da der Aufbau von *socket*-Verbindungen durch mehrere wechselseitige Abfragen und Nachrichten stattfindet, muß dieser Wert bei jedem Datenpaket, welches zum Aufbau der beidseitigen Verbindung benötigt wird, berücksichtigt werden.

Netz	Ping-Zeit
OKS-Netz, lokales LAN	1.4 ms
FOKUS-Netz über ATM-Hausanlage	3 ms
lokales Linux Netz mit 3 PC's	1.2 ms

Tab. 5: *Ping-Zeiten*

Der Aufbau einer Verbindung ist schwer zu messen, da er stark abhängig von der jeweiligen Implementierung ist. Eine Anfangsverzögerung zum HTTP-Server von ca. 0.3 bis 0.5 Sekunden ist normal und in unserem Rahmen vertretbar. Der Aufbau einer FTP-Verbindung dauert wenig länger.

Theoretisch ist auf den verwendeten Ethernet-Netzen eine Übertragung von 10 Mbit/sec möglich, da jedoch das TCP/IP Protokoll selbst sogenannten *Overhead* [25] produziert, Kollisionen von Datenpaketen nicht auszuschließen sind und verlorengegangene Daten erneut gesendet werden müssen, reduziert sich dieser Datendurchsatz, wie in Tabelle 6 gezeigt.

Netz	Durchsatz
OKS-Netz, lokales LAN	8.1 Mbit/sec
OKS- zu FOKUS-Netz über ATM-Hausanlage	6.2 Mbit/sec
lokales Linux Netz (Linux-Linux)	6.6 Mbit/sec
lokales Linux Netz (Linux-DOS/Windows)	2.6 Mbit/sec

Tab. 6: *TCP/IP-Durchsatz*

Weiterhin werden die Daten durch mehrere Schichten durchgereicht, innerhalb der beteiligten Applikationen mehrfach kopiert und wiederum auf anderen Datenträgern gespeichert. Auf z.B. im

[25] Als *Overhead* werden protokoll- oder formatspezifische Zusatzinformation bezeichnet, die keine Nutzdaten und zur reinen Realisierung des Dienstes eines Protokolls oder Formates notwendig sind.

Netz angeschlossene PC's ist das TCP/IP Protokoll zumeist nicht im Betriebsystem direkt, sondern über mehrere Zusatzprogramme (Treiber) realisiert. Dadurch werden die Nutzdaten oft zusätzlich kopiert und der normale Rechenablauf des Computers deshalb mehrfach unterbrochen. Zum Test wurden mehrere, große Dateien über das *Network File System* (NFS) übertragen, welches natürlich selbst auch *Overhead* erzeugt. Dies entspricht dem erreichten Durchsatz bei den verwendeten Protokollen FTP und HTTP (siehe Tabelle 7).

Netz	FTP	HTTP
OKS-Netz, lokales LAN	4.4 Mbit/sec	4.3 Mbit/sec
OKS- zu FOKUS-Netz über ATM-Hausanlage	4.1 Mbit/sec	3.9 Mbit/sec
lokales Linux Netz (Linux-Linux)	2.6 Mbit/sec	2.1 Mbit/sec
lokales Linux Netz (Linux-DOS/Windows)	1.4 Mbit/sec	1.3 Mbit/sec

Abb. 7: *Durchsatz bei FTP und HTTP*

Der Durchsatz liegt, da mehrere Applikationen am Datentransfer beteiligt sind, entsprechend niedriger. Die folgenden Werte sind "speicherzeit-bereinigt"; die Zeit, die benötigt wurde, um die empfangenen Daten zu speichern wurde bereits abgezogen.

Verglichen mit der Datenrate der verwendeten Formate kann die in Tabelle 8 angegebene Anzahl von multimedialen Dateien gleichzeitig und in Echtzeit zur Darstellung gebracht werden (hierbei werden natürlich nur zeitabhängige Formate berücksichtigt). Der Mindestdatendurchsatz von 4 Mbit/sec des HTTP-Protokolls im lokalen GMD-Fokus-Netz wurde hier zugrunde gelegt.

Format	Datenrate	Anzahl
MPEG-1 Video laut Standard	1.2 Mbit/sec	3
MPEG-1 Video Mindestanforderung 320*240, *6 frames/sec*	200 kbit/sec	20
MPEG-1 Audio laut Standard (Stereo)	384 kbit/sec	10.4
MPEG-1 Audio Mindestanforderung 33 kHz, Mono	96 kbit/sec	41.6
G.711	64 kbit/sec	62.5
G.721	32-40 Mbit/sec	125-100
WAV unkomprimiert oder RAW-PCM	64-1400 Mbit/sec	62.5-2.8

Tab. 8: *Anzahl der möglichen Übertragungen*

Daraus folgt abschließend, daß

- alle verwendeten Formate innerhalb eines auf Ethernet und TCP/IP basierenden, lokalen Netzwerkes in Echtzeit dargestellt werden können und
- alle verwendeten Formate (bis auf MPEG-1-Video) innerhalb dieser Netze problemlos, mehrfach übertragen werden können, ohne den sonstigen Datentransfer von z.B. Dateien zu sehr zu belasten, wenn man sich am geplanten Umfeld des PCSS orientiert. Für größere "In-House"-Systeme wie z.B. Universitäten mit hunderten, häufig benutzten Computern könnte dies unter Umständen nicht mehr ausreichend sein.

4.5.2 Wiederverwendbarkeit der Komponenten und Applikationen

Sowohl die beschriebenen Libraries als auch der Tcl-Programmcode können in anderen Applikationen wiederverwendet werden.

Die Libraries können direkt von der Programmiersprache C aus angesprochen werden, um z.B. die Ansteuerung der Multimedia-Applikationen über den MIME-Typ-Mechanismus in einem anderem Umfeld zu erleichtern oder eine Audioanbindung an die Betriebsysteme Solaris und Linux zu realisieren.

Die Anbindung der vorgestellten Mechanismen und globalen Applikationsaufrufe an die PCSS-Applikationen CRA-Manager und IMM-Manager ist bereits erfolgreich vollzogen worden. Beide Applikationen können über einfache Tcl-Funktionen (siehe Anhang B.2 auf Seite 92) die entsprechend benötigten multimedialen *recorder* und *player* starten und auf den Message-Store zugreifen. Die Anbindung an das VISA-Systems ist durch eine zusätzliche, auf Tcl/Tk und Tix basierenden, Applikation namens *NoteRecorder* realisiert worden. Diese ermöglicht die Verbindung von X.500 Einträgen mit den hier vorgestellten Libraries und Multimedia-Applikationen in Verbindung mit einer für TouchScreens optimierten Oberfläche.

Da die Applikationen in der Scriptsprache Tcl/Tk mit Unterstützung der C-Libraries entwickelt wurden, kann die gesamte Funktionalität jeweils einer der Multimedia-Applikationen in andere Programme integriert werden, in dem ein applikationsspezifischer Tcl/Tk-Interpreter mit den entstandenen C-Libraries kombiniert und die Tcl-Dateien der Multimedia-Applikationen in die neue Tcl/Tk-Applikation übernommen werden (siehe den Befehl *source* in Tcl). Durch Modifikation des letzten *pack*-Kommandos kann sowohl das komplette Interface in die Darstellung integriert, bzw. in einem neuen, untergeordneten Fenster dargestellt oder Teile des ursprünglichen Interfaces übernommen werden. Somit wird die Einbindung der hier programmierten Methoden in z.B. das VISA-System und ACCESS sehr erleichtert.

Da alle entstandenen Applikationen (bis auf die unterste Audioanbindung von MMAUDIO) als von der verwendeten Hardware getrennt zu betrachten sind, sie bereits auf die Unix-Betriebsysteme Solaris und Linux portiert wurden und die Programmpakete Tcl, Tk und Tix auf mehr als fünfzehn verschiedenen Unix-Derivaten übersetzt worden sind, ist damit zu rechnen, daß die entstandenen Multimedia-Applikationen ohne großen Aufwand auf andere Unix-Plattformen portierbar sind. Zu denken ist z.B. an die anderen, auf Sparc-Workstations existierenden, Betriebssystemvarianten wie SunOS und NeXTStep.

Kapitel 5 - Ausblick

Der abschließende Ausblick soll eine generelle Einordnung der entwickelten Mechanismen, Protokolle und Dienste bieten. Weiterhin soll die mögliche Weiterentwicklung der vorgestellten Mechanismen und Applikationen hinsichtlich der Integration weiterer multimedialer Formate diskutiert sowie eine Zusammenfassung der Resultate der vorliegenden Arbeit gegeben werden.

5.1 Erweiterungsmöglichkeiten

Alle *player* könnten um die Funktion erweitert werden, auf Wunsch eine, mittels des HTTP-Protokolls gelesene Datei, sowohl in Echtzeit darzustellen, als auch gleichzeitig als Datei in das lokale Dateisystem zu kopieren (*buffer* oder *cache* Mechanismus). Wenn der Benutzer die gleiche Datei erneut darstellen will, können die Applikationen die lokale Kopie benutzen und würden somit nicht den Message-Store belasten.

MMPICS sollte hinsichtlich des besseren Komprimierungsverhältnisses von JPEG um dieses Format erweitert werden.

Die Erweiterung von MMAUDIO kann sich in Hinsicht auf die weiteren, im PCSS-Umfeld benutzten und auch in Zukunft integrierten Hardwareplattformen (zu denken ist dabei an die neue SparcStation 4 und an die NeXT-Computer), hauptsächlich auf zusätzliche Audiokodierungen konzentrieren. Das Audioformat der MacIntosh Computer, sowie die von der digitalen Vermittlungstelle PABX der Firma Teles verwendete spezielle G.711 Version [26], könnten in MMAUDIO integriert werden. Um das Windows-Audioformat WAV besser einbinden zu können, könnten zusätzliche Windows-spezifische Komprimierer unterstützt werden.

MMVIDEO sollte auf Rechnerplattformen mit einem anderen Betriebssystem als Solaris 2.4 (z.B. SunOS, Linux) portiert werden. Die Anbindung an die SunVideo-Karte erfolgte duch die Verwendung der von Sun gelieferten XIL-Library. Da diese jedoch ausschließlich unter Solaris 2.4 verfügbar ist, müßte die Applikation von den Funktionen der XIL-Library getrennt und dieser Programmcode durch eine unabhängige MPEG-1-Video Implementierung ersetzt werden.

Die vollständige Portierung der Multimedia-Applikationen ist durch die Verwendung der Tcl/Tk- und Tix-Libraries erleichtert worden, letztere stehen sowohl unter Windowssystemen von Microsoft als auch auf dem MacOS der Macintosh-Computer zur Verfügung.

---

[26] Die ISDN-Vermittlungsstelle PABX der Firma Teles bieten die Möglichkeit, Ansagetexte zur Rufweiterleitung aufzunehmen. Da die PABX aber intern nicht die normale logarithmische Kompressionstabelle des Formates G.711 verwendet, sondern Tabellen, die mehrfach gedreht und gespiegelt sind, entspricht das intern gespeicherte Format nicht mehr G.711. Da auch die inverse Kompressionstabelle gedreht und gespiegelt ist, wird eine zur Rufweiterleitung aufgenommene und durch die PABX wieder abgespielte Nachricht korrekt rekonstruiert und die falsche interne Kodierung nicht entdeckt. Weiterhin wird bei der Kompression durch die PABX ein Quantisierungsintervall nicht berücksichtigt. Wenn also die interne Datei zur Rufweiterleitung nicht über ein angeschlossenes Telefon, sondern z.B. digital durch andere Quellen des PCSS erzeugt werden soll, muß über eine speziell zu berechnende Filterfunktion in das PABX-interne Audioformat umgewandelt werden.

5.2 Verwendung in anderem Umfeld

Die entstandenen Applikationen können zusammen mit dem Konzept des Message-Stores effizient in die bei der GMD existierenden Systeme MultiMediaMail (MMM) und Secure-MMM (ein bei GMD Fokus von Jürgen Meyer entwickelte, den Sicherheitsproblematiken Rechnung tragende Variante von MMM, siehe [25]) integriert und weiterverwendet werden. Die Trennung der Multimedia-Komponenten von der textuellen Nachricht mit der hier vorgestellten Verweistechnik via URL könnte in MMM integriert werden, um die Übertragung der Multimedia-Daten in der Nachricht zu verhindern. Der Empfänger einer Nachricht kann dann die Multimedia-Komponenten explizit aus dem Message-Store anfordern. Secure-MMM ist ebenfalls mit den Werkzeugen Tcl/Tk und Tix entstanden, eine optische Eingliederung sollte hier leicht möglich sein. Die entstandenen Multimedia-Applikationen könnten auch in das Multimedia-Colloboration-Service (MMC) Projekt der GMD Fokus integriert werden, um die Qualität der bestehenden Interface zu verbessern und neue hier implementierte Audioformate zu unterstützen. Letzeres würde jedoch einer Erweiterung der generellen Audiounterstützung durch MMC sowie dessen Übertragungsprotokolle erfordern.

Zusätzlich ist die Verwendung des Message-Store-Konzeptes im Bereich Video- oder Audio-*On-Demand* (auf Abruf) möglich. Multimediale Systeme wie Hotel-Videodatenbanken, digitale Radio- und Jukeboxsysteme im Internet oder Produkt- und Geschäftspräsentationen sind mit dessen Hilfe verteilt zu realisieren und weltweit abrufbar.

Ein weiteres wichtiges Einsatzgebiet für die entstandenen HTTP-fähigen Applikationen, ist die Verwendung in Verbindung mit WWW-*browser*-Applikationen, Dieses Konzept wird im folgenden vorgestellt und ist bereits an der TU-Berlin durch den Autor realisiert und installiert worden (URL: http://www.cs.tu-berlin.de/~phade/powerweb):

5.3 PowerWeb Technology

Durch keine der existierenden Web-*browser*-Applikationen wird eine gleichzeitige Darstellung von beliebigen MultiMedia-Formaten unterstützt. Die Übertragung von z.B. Audioströmen ist ausschließlich dateibasiert oder wird durch formatabhängige, nicht-generische Zusatzapplikationen realisiert. Zumeist blockiert die *browser*-Applikation, bis die entsprechende Datei gelesen ist, startet dann eine externe, zur Darstellung geeignete Applikation, die dann wiederum die gespeicherte Datei einlesen muß.

Wünschenswert wäre die Möglichkeit, zeitabhängige Formate generell während des Lesens vom Internet paketweise darzustellen, wie es die im Rahmen dieser Arbeit entstandenen Applikation in Verbindung mit dem Message-Store tun. Da diese Applikationen fähig sind, paketweise mit Hilfe des HTTP-Protokolls zu lesen, können sie zur Lösung dieses Problem genutzt werden.

Entscheidend ist dabei die Methode der doppelten Referenzierung, wie in Abbildung 25 auf Seite 73 dargestellt. Die Referenz auf eine via HTTP-Protokoll erreichbare Datei wird nicht mehr in der HTML-Seite installiert, sondern in eine Datei geschrieben, die dann selber als Referenz in die HTML-Seite eingefügt wird.

Eine zusätzliche Applikation wird nun mit dem MIME-Typen und der Dateinamenserweiterung dieser Referenzdatei durch den *browser* verbunden (vorgeschlagen ist **text/url** als MIME-Typ und **.url** als Extension). Ein Web-Server, der diese doppelte Referenzierung unterstützen will, muß

ebenfalls diesen MIME-Typen und die Extension deklarieren und bei der Dereferenzierung einer Referenzdatei übertragen.

Die zusätzliche Applikation wird dann bei der Selektion einer Referenzdatei durch den *browser* gestartet, der *browser* kann unverzüglich weiterarbeiten, da eine Referenzdatei sehr wenig Daten enthält. Die Applikation liest die Referenzdatei ein, überprüft die enthaltene URL, und kontaktiert den entsprechenden Server, um den MIME-Type der eigentlichen Datei zu erhalten. Nach der Parsierung der benutzerspezifischen Konfigurationsdateien kann die HTTP-fähige, externe Applikation gestartet werden. Ihr wird die URL übergeben, damit sie selbstätig die Daten der Multimedia-Datei lesen und gleichzeitig darstellen kann.

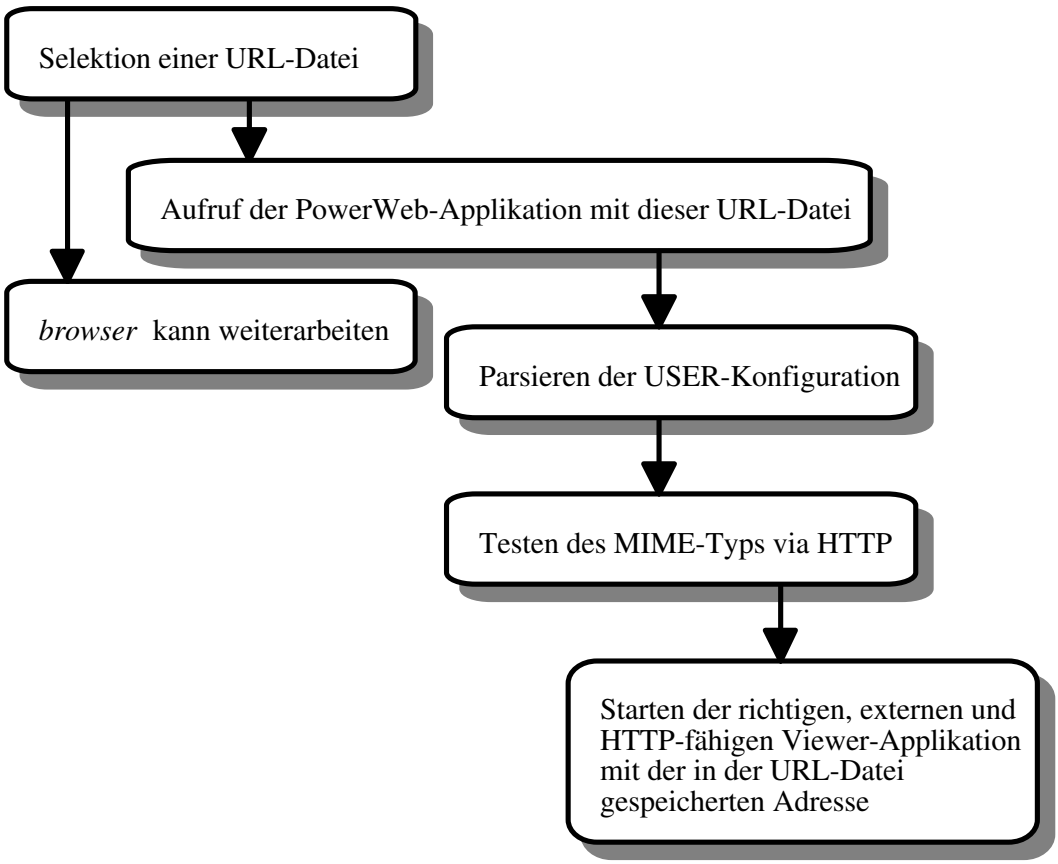


Abb. 25: *Ablauf der doppelten Referenzierung*

Somit können innerhalb des WorldWideWeb zeitabhängige Formate auch zeitabhängig konsumiert und diese Konsumierung zur Einsparung der begrenzten Übertragungskapazitäten jederzeit abgebrochen werden. Außerdem kann mit dieser Methode eine Datei innerhalb des WorldWideWeb auf eine andere Datei verweisen. Weiterhin können somit auch beliebig viele Dateien gleichzeitig gelesen werden. Diese Methode funktioniert unabhängig für alle *browser*-Applikationen, die es ermöglichen, externe Applikationen über die Angabe des MIME-Typs anzubinden und das sind alle dem Autor bekannten.

Zur weiteren Lektüre wird auf die Dokumentation des bestehenden PowerWeb-Systems verwiesen [4]. Das hier beschriebene, grundsätzliche Verfahren der doppelten Referenzierung sowie die in diesem Zusammenhang entstandene Software und Applikationen sind von der vorliegenden Arbeit getrennt zu verstehen und kein Bestandteil der Aufgabenstellung. Das PowerWeb-Verfahren wurde hier nur der Vollständigkeit halber und als eine der möglichen Umgebungen zum Einsatz der Multimedia-Applikationen erwähnt.

5.4 Zusammenfassung

In der vorliegenden Diplomarbeit wurden Konzepte und Applikationen zur Übertragung von multimedialen Daten über bestehende, auf dem internetspezifischen Übertragungsprotokoll TCP/IP basierende, Netze entwickelt und implementiert. Das vorgestellte Konzept eines Message-Store ließ sich mit der im PCSS verwendeten Hardware- und Netzwerkplattform verwirklichen, eine zentrale Ablage von multimedialen Nachrichten und deren zeitbasierten Rezeption wurde realisiert. Die multimedialen Formate, die innerhalb der zu Verfügung stehenden Bandbreite übertragen werden können, wurde ausgewählt und implementiert.

Das Protokoll HTTP hat sich als besonders geeignet zur zeitbasierten Übertragung erwiesen. Die hier beschriebene Implementierung des Global Stores konnte nicht zur Realisierung eines Message-Stores verwendet werden. Eine Erweiterung und Portierung des RTT-Protokolls würde jedoch die Probleme dieser Implementierung lösen.

Besonders erwähnenswert ist die Tatsache, daß qualitativ hochwertige Audionachrichten mit dem MPEG-1-Audioverfahren kodiert und problemlos innerhalb lokaler Netze übertragen werden können. Als problematisch erwiesen sich die nicht international standardisierten, bzw. generisch spezifizierten Formate, da deren Datenvolumen nicht genau genug analysiert werden konnte oder systemunabhängige Beschreibungen und Implementierungen nicht vorlagen. Weiterhin ist die Übertragung von Video in den bezeichneten Netzwerken als bis jetzt nicht zufriedenstellend zu lösen. Hier sollten weitergehende Analysen und Implementierungen nach einer Lösung suchen.

Die Widerverwendung der implementierten Applikationen ist durch die Verwendung der Scriptsprachen Tcl/Tk und Tix sowie der Gruppierung der Funktionen in Libraries garantiert. Die mögliche Spezifizierung der aufnehmenden oder abspielenden Applikationen während der Laufzeit unter Verwendung des MIME-Typ-Konzeptes erwieß sich sehr nützlich, um sowohl die multimedialen Applikationen als auch einen LAN-weiten Message-Store von mehreren Softwaresystemen aus gleichzeitig zu benutzen.

Diese Seite wurde freigelassen.

Index

A

ACCESS, 6; 41; 43; 70
ADPCM, 2; 37; 38; 39; 49; 50; 60; 64
ASCII, 2; 16; 17; 48; 50; 58; 62
Audio, 1; 2; 4; 23; 24; 25; 26; 27; 29; 33; 34; 35; 37; 38; 40; 41; 43; 44; 46; 47; 50; 60
Aufnahme, 25; 43; 44; 47; 49; 54; 56; 57; 58; 59; 61

B

Benutzer, 3; 13; 14; 16; 41; 43; 45; 48; 53; 56; 58; 60
Bild, 1; 2; 4; 17; 26; 28; 33; 43; 44; 46; 47; 48; 50; 60
Block Truncation Coding, 31
BMP, 2; 19; 23

C

CD-ROM, 31; 34; 48
Client, 11; 12; 43
Color Cell Compression, 31; 32
CRA, 5; 6; 45

D

Darstellung, 12; 13; 17; 34; 41; 42; 47; 50; 57; 58
Dateisystem, 53; 58
Datenrate, 23; 31; 32; 34; 35; 49; 50
Datenstrom, 30; 57
Datenstruktur, 45
Differenz, 39
Digitalisierung, 43
Dokumente, 10; 13; 16; 48
DOR, 14; 15

E

Echtzeit, 32; 35; 41; 44; 47; 50; 60; 61; 64; 68; 69
Echtzeitdekodierung, 60

F

FTP, 2; 8; 11; 12; 50; 51; 53; 55; 56; 57; 58
Funktion, 28; 57; 58

G

G.711, 2; 30; 33; 36; 37; 42; 43; 48; 49; 50; 60; 67; 68
generisch, 57
GIF, 2; 19; 20; 49; 50; 63; 67
Global Store, 14; 15; 41; 42; 43
GSM, 39; 48

H

HTTP, 2; 8; 12; 13; 14; 47; 50; 51; 53; 55; 56; 57; 58; 59; 61

I

IMM, 5
Integration, 1; 6; 25; 41; 42; 49; 50; 52; 53; 54; 56; 57
Interface, 49; 58; 59
Internet, 1; 2; 3; 8; 10; 11; 12; 13; 14; 16; 21; 34; 41; 47; 48; 49; 53
ISDN, 7; 9; 30; 36; 37; 38; 49

J

JPEG, 2; 19; 20; 21; 24; 25; 29; 49

K

Kodierung, 16; 24; 25; 26; 27; 28; 29; 30; 31; 32; 34; 35; 36; 37; 39; 49
Kommunikation, 1; 3; 7; 8; 9; 10; 12; 14; 16; 43; 58
Kompression, 30; 32; 36; 37; 38

L

Layer, 34; 35; 49; 50
Luminanz, 18; 24; 31; 32

M

MASTER, 6
Mechanismus, 50
Message-Store, 1; 3; 4; 5; 6; 7; 25; 41; 42; 43; 44; 46; 47; 49; 50; 51; 53; 55; 56; 58; 59; 61
Methode, 14; 31
Modell, 8; 14; 15; 34; 35; 56
Module, 5; 54; 56
Motion-JPEG, 21
MPEG, 2; 24; 25; 26; 27; 29; 30; 33; 34; 35; 36; 47; 48; 49; 50; 60
Multimedia, 3; 6; 7; 14; 25; 29; 33; 38; 41; 42; 43; 44; 45; 56
Multimediaformate, 41

N

Nachricht, 4; 15; 42; 44; 46; 47; 51; 56
Netz, 10; 11; 37; 42
NFS, 43

O

Objekte, 14; 15; 42; 45
ODA, 25

P

Parameter, 9; 17; 25; 26; 39
PBM, 2; 21; 49; 50; 63
PCSS, 1; 3; 4; 5; 6; 7; 10; 25; 37; 38; 39; 41; 42; 43; 44; 45; 46; 47; 48; 49; 50; 52; 53; 54; 55; 56; 57; 58; 61
Performanz, 68
Plattform, 18; 48
PostScript, 2; 16; 17; 48; 49
Profile, 6; 26; 29; 45; 46; 56
Protokoll, 2; 3; 9; 10; 11; 12; 13; 42; 51; 53; 59; 61
Puffer, 59

Q

Quantisierung, 21; 34; 35

R

RCP, 2; 8; 14; 51
RDT, 2; 14; 15; 42; 43
Realzeitzugriff, 41; 43
Referenz, 13; 46
RIFF, 40

S

SAP, 6
Schichten, 9; 11; 17; 34; 59
Server, 7; 11; 12; 13; 15; 43; 47; 51; 56
SMP, 2; 30; 31; 32; 33; 42; 43; 47; 48; 49
Standard, 13; 16; 20; 21; 24; 25; 26; 30; 34; 35; 50; 60; 62; 67; 69

T

TCP/IP, 2; 3; 8; 11; 12; 13; 41; 56
Terminal, 6; 48
Text, 1; 2; 3; 4; 16; 17; 43; 44; 46; 47; 48; 50; 58; 60
TIFF, 2; 22; 49; 50; 63
Transformation, 24; 30; 34; 35; 36; 37

U

URL, 13; 16; 46; 51; 56; 58

V

VAP, 6
Vernetzung, 1
Video, 1; 2; 4; 6; 23; 24; 25; 26; 27; 29; 30; 31; 33; 34; 41; 43; 44; 46; 47; 50; 58
VISA, 6; 41; 43; 48; 56
Volumens, 15

W

WAV, 2; 40; 48; 49; 50; 60
Wiedergabe, 25; 44; 54
WorldWideWeb, 7

X

XBM, 2; 21; 22; 23; 49; 50; 63; 67
XPM, 2; 21; 22; 23; 49; 50; 63

Y

YUV, 18; 19; 20; 21; 30; 32

Abkürzungverzeichnis

Folgende Abkürzungen werden im vorliegenden Text benutzt und erklärt:

ACCESS = *Advanced Communication Control Environment and Scheduling System*
ASCII = *American Standard Code for Information Interchange*
CD = *Compact Disc*
CRA = *Customized Recorded Announcements*
DCT = *Discrete Cosine Tranform*
DFT = *Discrete Fourier Transform*
DOR = *Distributed Object Reference*
FTP = *File Transfer Protocol*
HTML = *HyperText Markup Language*
HTTP = *Hypertext Transfer Protocol*
ISO = *International Standard Organisation*
GIF = *Graphics Intechange Format*
GS = *Global Store*
IMM = *Incoming Message Manager*
IP = *Internet Protocol*
ISC = *Intensity Stereo Coding*
LAN = *Local Area Network*
MIS = *Managment Information Service*
MMM = *Multimedia Mail*
MMC = *Multimedia Colloboration Service*
MM-PCSS = *MultiMedia PCSS*
MPEG = *Motion Picture Expert Group*
MS = *Message-Stores*
NFS = *Network File System*
ODA = *Open Document Architecture*
OSI = *Open System Interconnection*
PAT = *Profile Administration Tool*
PCC = *Phantom Coding of Center*
PCSS = *Personal Communication Support System*
PMS = *Phone Mail Server*
RCP = *Remote Copy Protocol*
RDT = *Referenced Data Transfer Protocol*
RFC = *Request For Comment*
RIFF = *Resource Interchange File Format*
RTT = *Realtime Transfer*
SAP = *Service Access Point*
SGML = *Structured Generalized Markup Language*
SMP = *Software Motion Picture*
TCP = *Transmission Control Protocol*
VAP = *Virtual Access Point*
VISA = *Visitors Information Service Application*
URL = *Uniform Resource Locator*

Literaturhinweise

[1] Bading, Tobias: Der MPEG Audio Decoder maplay, OKS Multimedia-Systeme-Projekt I, TU-Berlin, WS 93/94

[2] Born, Claudia; Freudenberg, Thomas Ch.: Grafikbearbeitung & Morphing, tewi-Verlag, 1994

[3] Gadegast, Frank; Meyer, Jürgen: Sicherheitsmechanismen für Multimedia Datenformate, Studienarbeit, TU-Berlin, 1994

[4] Gadegast, Frank: Powerweb Technology, Online-Dokumentation im WorldWideWeb, URL: http://www.cs.tu-berlin.de/~phade/powerweb, 1995

[5] CCIR 601: Encoding parameters of digital Television for Studios, Recommendation, 1982

[6] CCITT X.400: Message Handling, System and Service Overview, Recommendation, Genf, 1988

[7] Chiariglione, Leonardo: Multimedia Communication (MPEG-2), Script zum Vortrag, Brüssel, 1992

[8] CompuServe Incorporated: Graphic Interchange Format, Version 89a, Ohio, USA, 1989

[9] Eckhardt, Tim: Personal Communication Support System: Preliminary Specification of Profiles and Management Services, IN/TMN Integration Deliverable 6, Berkom-Projekt II, TU-Berlin, Juni 1995

[10] Fielding, R; Berners-Lee, R; Nielsen, H: Internet draft "Hypertext Transfer Protocol -- HTTP/1.0", URL: ftp://ftp.cs.tu-berlin.de/pub/rfc/draft-fielding-http-spec-00.txt, 11/28/1994

[11] Ghamsari, Majid Salehi: Implementierung von Realzeitzugriff auf multimediale Daten im Rahmen von Multimedia Mail, Diplomarbeit, TU-Berlin, Oktober 1994

[12] ISO/IEC 8824/25: Information technology - Open System Interchange connection - Specification of Abstract Syntax Notation One (ASN.1), Genf, 1989

[13] ISO/IEC 9096: SGML Document Interchange Format (SDIF), Genf, 1990

[14] ISO/IEC 10031-1: Information Technology - Text and Office Systems - Distributed-office-applications Model - Part 1 - General Model (DOAM), Genf, 1991

[15] ISO/IEC 10918: Digital Compression and Coding of Continuous Still Images (JPEG), Genf, 1992

[16] ISO/IEC 11172: Information Technology - Coding of moving pictures and associated audio for digital storage media up to about 1,5 mbit/s (MPEG-1), Genf, 1993

[17] ISO/IEC 13818: Information Technology - Generic coding of moving ictures and associated audio (MPEG-2), Genf, 1994

[18] Jayant, N. S.; Noll, Peter: Digital Coding of Waveforms, Prentice-Hall Int. London, 1984

[19] Krüger, Robert: Modellierung und Implementierung eines elektronischen Lokalisierungsdienstes basierend auf Active Badges, Studienarbeit, TU-Berlin, 1995

[20] Krüger, Robert: Modellierung und Implementierung eines modular erweiterbaren elektorinischen Lokalisierungsdienstes, OKS Diplomarbeit, TU-Berlin, 1995

[21] Küsters, Heiner: Bilddatenkomprimierung mit JPEG und MPEG, Stand- und Bewegtbildkomprimierung, Franzis-Verlag,1995

[22] LeGall, Didier: MPEG: A Video Compression Standard for Multimedia Applications, ACM 4/91 (34/4), 1991

[23] Lempel, A; Ziv, J. : A Universal Algorithm for Sequential Data Compression, IEEE Transactions on Information Theory, May 1977

[24] Maier, Gunther; Wildberger, Andreas: In 8 Sekunden um die Welt, Addison-Wesley, New-York, 1994

[25] Meyer, Jürgen: Konzerption und Implementierung von Benutzeroberflächenelementen für Multimedia Mail UA mit Sicherheitsfunktionalitäten, OKS Diplomarbeit, TU-Berlin, 1995

[26] Markley, Richard W.: Data Communications and Interoperability, Prentice-Hall, 1990

[27] McGilton, Henry: PostScript by Example, Addison-Wesley, Menlo Park, 1992

[28] Milde, Torsten: Videokompressionsverfahren im Vergleich, dpunkt/HVS, 1995

[29] Mitchell, O.; Delp, E., Carlton, S: A new approach to Image Compression, Conference Record IEEE, International Conference Communications,Volume 1, June 1978

[30] Murray, James: Encyclopedia of Graphics File Formats, O'Reilly & Associates, Ney York, 1994

[31] Musmann, H.-G.; Werner, O.; Fuchs, H.: Kompressionsalgorithmen für interaktive Multimedia-Systeme, IT+TI, Oldenbourg-Verlag, 2/1993, pp 4-18

[32] Neidecker-Lutz, B; Ulicheny, R.: Software Motion Pictures, Digital Technical Journal 5(2), 1993

[33] Ranagan, P. Venkat; Vin, Harrick M.: Designing File Systems for Digital Video and Audio, Proc. 13'th ACM symposium on Operating Systems Principles (SOSP'91), Monterey, Operating Systems Review, 10/91 (Vol.25/No.5), 1991

[34] Scheibe, Marco: Audioformate, Semesterarbeit KBS, TU-Berlin, 1993

[35] Steinmetz, Ralf: JPEG, MPEG und DVI, Bild- und Datenkompression, Springer, Berlin, 1993

[36] Steinmetz, Arndt; Hemmje, Matthias: Konzeption eines digitalen Videoeditiersystems auf Basis des internationalen Standard ISO/IEC 11172 (MPEG-1)

[37] Pan, Davis Yen: Digital Audio Compression, Digital Technical Journal, 5(2), März 1993

[38] Pusch, Herwart: Design and implementation of a global reference mechanism for data objects, GMD Fokus, Berlin, 1994

[39] Waller, Marco: Design and implementation of a Phone Mail Server for the PCSS, Studienarbeit, TU-Berlin, 1995

Internet-RFC (*Request For Comments*):

[40] Hicks, G: User FTP documentation, RFC 412, November 1972 *
[41] White, J: Host-dependent FTP parameters, RFC 480, März 1973 *
[42] Postel, Jon: Revised FTP reply codes, NWG/RFC 640, JBP, Juni 1975
[43] Lieb, J: CWD command of FTP, RFC 697, Juli 1975 *
[44] Postel, J:File Tranfer Protocol IEN 149/RFC 765, ISI, Juni 1980
[45] Crocker, David H.: Standard for the Format of ARPA Internet Text Messages, RFC 822, August 1982
[46] Borenstein, N.: The text/enriched MIME Content-type, RFC 1563, Januar1994

Die offiziellen Kopien der RFCs können via FTP von *nic.ddn.mil* bezogen werden ,sofern nicht mit einem Stern (*) gekennzeichnet.

Anhang

A. Installationshinweise

A.1 Player/Recorder

Die im Rahmen dieser Arbeit entstandene Software läßt sich einfach auf dem Betriebsystem Solaris und Linux installieren. Vorausgesetzt ist die Installation der folgenden Software-Pakete:

- **Tcl-7.4** (URL: ftp://ftp.cs.berkeley.edu/pub/tcl/tcl7.4*.tar.gz)
- **TK 4.0** (URL: ftp://ftp.cs.berkeley.edu/pub/tcl/tk4.0*.tar.gz)
- **Tix 4.0beta2-beta5** (URL: ftp.cis.upenn.edu/pub/ioi/Tix4.0b*.tar.gz)
- **GCC 2.5.8-2.6.3**
- **SunVideo-Entwicklerkit** (SUNWits/Graphics-sw/xil, wird mit der SunVideo-Karte mitgeliefert)

Das Software-Archiv **mmtools-1.0.tar.gz** wird an eine geeignete Stelle kopiert, dekomprimiert und dearchiviert.

```
mkdir /tmp/mmtools
cp mmtools-1.0.tar.gz /tmp/mmtools
cd !$
gunzip *.gz
tar xvf *.tar
```

Anschließend muß das **Makefile** editiert werden. Die Konfigurationszeilen für das entsprechende Betriebsystem sollten entkommentiert und die Pfade zu den Tcl-, Tk-, Tix- und SunVideo-Include und Library-Verzeichnissen sollten überprüft und eventuell angepaßt werden. Der Name des Rechners der als Message-Store verwendet werden soll, sollte eingetragen werden. Die Installations-Directory und die Installations-Library sollten richtig eingetragen werden. Dann:

```
make solaris
```

oder

```
make linux
```

Dann sollten noch die voreingestellten Konfigurationsdateien **mmpcssrecordcap**, **mmpcssplaycap** und **mmpcssmimetype**, die durch den **make**-Aufruf vom doc-Verzeichnis in das Installationsverzeichnis kopiert wurden, kontrolliert bzw. korrigiert werden.

A.2 Message-Store

Zur Installation des Message-Stores sollte wie folgt vorgegangen werden:

Installation eines FTP-Servers (siehe: **man ftpd**) mit anonymen Zugriff z. B. unter /usr/local/ftp
Installation eines PCSS Unterverzeichnisses, z. B.

mkdir /usr/local/ftp/pcss
chmod 755 /usr/local/pcss

Installation von Unterverzeichnissen im PCSS-Baum für jeden Message-Store bzw. PCSS Benutzer. Die Namen des Benutzerverzeichnis muß mit seiner Unix-Systemkennung übereinstimmen.

mkdir /usr/local/ftp/pcss/phade
chmod 1775 /usr/local/ftp/pcss/phade

Der VISA- bzw. PhoneMail-User muß in die Gruppe des FTP-Verzeichnisses aufgenommen werden, um Schreibberechtigung auf die Benutzerverzeichnisse zu erhalten.

Ein HTTP-Server muß auf dem gleichen Computer kompiliert und installiert werden, das root-Verzeichnis des Servers muß dabei das gleiche des FTP-Servers sein, so daß eine URL zu einer Benutzernachricht z.B. wie folgt angegeben werden kann:

http://rechnername/pcss/pha/au1012951255.mp2

HTTP- und FTP-Server sollten ausschließlich zum Betrieb des Message-Stores benutzt werden, die Benutzung von zeitaufwendigen CGI-Scripten vom HTTP-Server sollte bei der Konfiguration unterbunden werden.

B. Softwarebeschreibung

Dieser Anhang soll die genaue Verwendung der entstandenen Funktionen und Applikationen erläutern, um die Integration dieser in einem anderem Umfeld zu erleichtern, sowie die Konfiguration zur dokumentieren.

B.1 MMTOOLS

Im folgenden werden die Aufrufparameter der entstandenen Multimedia-Applikationen aufgelistet (gleiche Parameter bei verschiedenen Applikationen werden dabei nur einmal erklärt):

Allen Applikationen kann ein Dateinname sowohl in relativer oder absoluter, als auch in URL-Schreibweise übergeben werden. Dieser wird dann jeweils als Dateiname zur Speicherung der eingelesenen oder eingegebenen Daten verwendet oder bezeichnet die zur Darstellung zu bringende Datei. Die Darstellung einer Datei aus dem Message-Store kann nur über das Protokoll HTTP erfolgen, die Angabe der Protokollart HTTP in der URL und die Angabe des Rechnernamens ist vorgeschrieben.

MMTEXT

-simple/intermediate/complex Startet das Interface entsprechend den verschiedenen Komplexitätsstufen. Vorgabewert ist "complex". Ein komplexes Interface enthält dabei alle Menüs (File und Help), ein simples Interface enthält ausschließlich das Text-*widget* und einen Exit-Knopf zum Beenden der Applikation.

-path <dir> Mit diesem Parameter kann das Anfangverzeichnis innerhalb des Verzeichnisbaumes angegeben werden. Als Vorgabe wird sonst das aktuelle Verzeichnis verwendet.

-mimetype <mime-type> Hier kann der MIME-Typ der geladenen Datei explizit voreingestellt werden. Ein testweises Parsieren des Datei*headers* entfällt.

-recorder Das Verhalten einer als *recorder* gestarteten Applikation unterscheidet sich teilweise von der einer als *player* gestarteten, z.B. wird normalerweise bei der Angabe eines Dateinamens das Vorhandensein der Datei überprüft. Bei einer als *recorder* gestartenen Applikation wird aber davon ausgegangen, daß der übergebene Dateiname nur zur Speicherung eines noch einzugebenen Textes benutzt werden soll. Weiterhin wird davon ausgegangen das ein eingegebener Text auch unbedingt gespeichert werden soll.

-viewer Durch diesen Parameter wird z.B. auf eine explizite Nachfrage zur Speicherung verzichtet.

-geomstr <x-y string> Dieser Parameter kann zur Angabe der X/Y-Bildschirmposition der zur startenden Applikation benutzt werden. Sein Format ist '**+x+y**'.

MMPICS

Die Angabe der Parameter **-simple/intermediate/complex**, **-path**, **-mimetype**, **-recorder**, **-viewer** und **-geomstr** ist auch bei dieser Applikation möglich. Ein als 'simple' deklariertes Interface wird entsprechend MMTEXT nur mit einem Exit-Knopf dargestellt, ein Einlesen eines neuen Bildes ist also nicht möglich.

MMAUDIO

Die Angabe der Parameter **-simple/intermediate/complex**, **-path**, **-mimetype**, **-recorder**, **-viewer** und **-geomstr** ist auch bei dieser Applikation möglich. Ein als 'simple' deklariertes Interface wird nur mit einem Stop/Exit-Knopf und einem Lautstärken- bzw. Aufnahmeemfindlichkeitsregel dargestellt. Ein als 'intermediate' deklariertes Interface erlaubt die wiederholte Wiedergabe und Aufnahme einer Audiodatei, sowohl das Audioformat als auch der (voreingestellte) Dateiname zur Speicherung kann nicht geändert werden. Weiterhin können folgende Audio-spezifische Parameter übergeben werden:

-volume Ein Interface zur Einstellung der Lautstärke wird gestartet.

-bigbuttons Das jeweilige Interface wird mit vergrößerten Knöpfen dargestellt. Dies ermöglicht die Benutzung der Applikation mit Hilfe von Touchscreens.

-headphone Die Ausgabe des Audiosignal erfolgt nicht über den internen Lautsprecher, sondern über den Kopfhörerausgang.

-linein Die Aufnahme des Audiosignals erfolgt nicht über ein angeschlossenes Mikrofon, sondern über eine Verbindung zu einer externen Audioquelle (z.B. Tonband).

MMSAVE

Als Parameter können nur **-path** und **-version** angegeben werden. Letzterer zeigt die Versionsnummer der Applikation an, die sonst im *About*-Dialog jeder Applikation dargestellt wird. MMSAVE enthält jedoch keine Menüs.

B.2 LIBRARIES

Im folgenden soll die grundlegende Funktionsweise und die Aufruf- und Rückgabeparameter der in den Libraries LIBMMPCSS und LIBMMACCESS enthaltenden Tcl-Funktionen und -Prozeduren genau vorgestellt werden:

LIBMMPCSS

int MMpcss_viewer_capable (char *MIMETYPE)	Überprüft, ob global oder benutzerspezifisch für den gegebenen MIME-Typen **MIMETYPE** eine *viewer*-Applikation spezifiziert ist. Gibt den Wert 1 zurück, wenn dies zutrifft, ansonsten den Wert 0.
int MMpcss_recorder_capable (char *MIMETYPE)	Überprüft, ob global oder benutzerspezifisch für den gegebenen MIME-Typen **MIMETYPE** eine *recorder*-Applikation spezifiziert ist. Gibt den Wert 1 zurück, wenn dies zutrifft, ansonsten den Wert 0.
char * MMpcss_get_viewer (char *MIMETYPE)	Wertet aus, welche *viewer*-Applikation global oder benutzerspezifisch für den gegebenen MIME-Typen **MIMETYPE** spezifiziert ist. Gibt den Aufrufstring oder NULL bei keiner spezifizierten Applikation zurück.
char * MMpcss_get_recorder (char *MIMETYPE)	Wertet aus, welche *recorder*-Applikation global oder benutzerspezifisch für den gegebenen MIME-Typen **MIMETYPE** spezifiziert ist. Gibt den Aufrufstring oder NULL bei keiner spezifizierten Applikation zurück.
int MMpcss_execute_player (char *MIMETYPE, char *FILE)	Startet im Hintergrund die für den gegebenen MIME-Typen **MIMETYPE** spezifizierte *viewer*-Applikation mit der Datei oder URL **FILE**. Gibt den Wert 1 bei erfolgreichem Start einer Applikation zurück, ansonsten den Wert 0.
char * MMpcss_execute_recorder (char *MIMETYPE, char *FILE)	Startet im Vordergrund die für den gegebenen MIME-Typen **MIMETYPE** spezifizierte *recorder*-Applikation mit der Datei oder URL **FILE**. Weiterhin wird der Dateiname der aufgenommenen Datei vom X11-Server-Cutbuffer 0 und die Länge der Aufnahme in Sekunden vom Cutbuffer 1 gelesen, bevor die Funktion nach Beendigung der *recorder*-Applikation zurückkehrt. Gibt den Dateinamen zurück, bei keiner erfolgreichen Aufnahme den String "0".
int MMpcss_get_duration ( )	Gibt die Länge der letzten Aufnahme in Sekunden zurück.
char * MMpcss_tmpnam (char *MIMETYPE)	Erzeugt einen eindeutigen Dateinamen zur Speicherung einer Datei im Message-Store passend zum gegebenen MIME-Typen **MIMETYPE**. Gibt den Dateinamen zurück.

int MMpcss_get_config_viewer (char *MAIN_MIMETYPE)	Überprüft, welcher Komplexitätsgrad für den Haupt-MIME-Typ **MAIN_MIMETYPE** global bzw. benutzerspezifisch für *viewer* konfiguriert ist. Gibt den Komplexitätswert zurück oder einen leeren String, wenn keiner spezifiziert ist.
int MMpcss_get_config_recorder (char *MAIN_MIMETYPE)	Überprüft, welcher Komplexitätsgrad für den Haupt-MIME-Typ **MAIN_MIMETYPE** global bzw. benutzerspezifisch für *recorder* konfiguriert ist. Gibt den Komplexitätswert zurück oder einen leeren String, wenn keiner spezifiziert ist.
int MMpcss_config_player (char *MAIN_MIMETYPE, char *COMPLEXITY)	Setzt den gegebenen Komplexitätsgrad **COMPLEXITY** für alle Einträge des Haupt-MIME-Typs **MAIN_MIMETYPE** im benutzerspezifischen *viewer*-Konfigurationsfile. Gibt den Wert 1 bei erfolgreiche Neukonfiguration zurück, ansonsten 0.
int MMpcss_config_recorder (char *MAIN_MIMETYPE, char *COMPLEXITY)	Setzt den gegebenen Komplexitätsgrad **COMPLEXITY** für alle Einträge des Haupt-MIME-Typs **MAIN_MIMETYPE** im benutzerspezifischen *recorder*-Konfigurationsfile. Gibt den Wert 1 bei erfolgreiche Neukonfiguration zurück, ansonsten 0.
int MMpcss_Init (Tcl_Interp *INTERP)	Initialisiert die Funktionen der Library LIBMMPCSS und gibt die angebotenen Funktionen dem aktuellen Tcl-Interpreter INTERP zur weiteren Verwendung bekannt. Gibt immer den Wert TCL_OK (1) zurück.

LIBMMACCESS

int MMaccess_http_connect (char *URL, char *MIMETYPE)	Öffnet die durch eine URL angegebene Datei via HTTP zum weiteren Lesen. **URL** ist eine HTTP-Adresse in URL-Schreibweise. **MIMETYPE** ist ein Pointer auf ein mindestens 30 Zeichen langes Feld. In dieses Feld wird der via HTTP-Protokoll bekanntgegebene MIME-Typ der Datei zurückgegeben. Gibt einen Filedescriptor bzw -1 bei Fehler zurück.
long MMaccess_http_file_length ( )	Gibt die via HTTP-Protokoll bekanntgegebene Länge der zuletzt geöffneten Datei züruck.
char * MMaccess_http_mime_type ( )	Gibt den via HTTP-Protokoll bekanntgegebenen MIME-Typ der zuletzt geöffneten Datei zurück.
char * MMaccess_http_get_line (fd)	Liest eine Zeile einer ASCII-kodierten Datei via HTTP. Als Trennzeichen werden sowohl Carriage-Return als auch Line-Feed oder deren Kombination erkannt. **fd** ist der von der connect-Funktion zurückgelieferte Filedescriptor. Gibt einen Pointer auf die gelesene Zeile oder NULL bei Fehler zurück.

char * MMaccess_http_get_buffered_line (fd)	Liest eine Zeile einer ASCII-kodierten Datei via HTTP. Intern wird jedoch paketweise in einen Buffer gelesen (Performanzsteigerung). Als Trennzeichen werden sowohl Carriage-Return als auch Line-Feed oder deren Kombinationen erkannt. **fd** ist der von der connect-Funktion zurückgelieferte Filedescriptor. Gibt einen Pointer auf die gelesene Zeile oder NULL bei Fehler zurück.
int MMaccess_http_eof ( )	Überprüft das Ende einer via HTTP geöffneten Datei. **fd** ist der von der connect-Funktion zurückgelieferte Filedescriptor. Gibt den Wert 1 zurück, wenn das Ende der Datei errecht ist, ansonsten den Wert 0.
long MMaccess_http_save (char *URL, char *FILE)	Lädt die als **URL** angegebene Datei und speichert diese unter dem angegebenen Dateinamen **FILE** ab. Gibt die Länge der gespeicherten Datei in Bytes zurück oder den Wert 0 bei einem Fehler.
int MMaccess_http_close ( )	Schließt die Verbindung zum HTTP-Server. Gibt den Rückgabewert der C-Funktion *close* zurück.
char * MMaccess_resolve_url (char *URL)	Gibt eine modifizierte Version der übergebenen **URL** in Anhängigkeit vom URL-Typen zurück: *file-URL*: Umwandlung in einen absoluten, lokalen Dateinamen. *ftp-URL*: Umwandlung in einen Dateinamen relativ zum root-Verzeichnis des FTP-Servers. *http-URL*: Keine Modifikation.

char * MMaccess_ftp_store (char *FILE, char *USER)	Speichert die als **FILE** angegebene Datei in dem PCSS-Verzeichnis des Benutzers **USER** ab. USER ist dabei der beim FTP-Zugriff bekannte Unix-Benutzername. Gibt eine komplette URL auf die gespeicherte Datei oder den Wert 0 bei Fehler zurück.
int MMaccess_ftp_delete (char *URL, char *USER)	Entfernt die unter der **URL** im Message-Store gespeicherte Datei des Benutzers **USER**. Gibt immer den Wert 1 zurück.
int MMaccess_Init (Tcl_Interp *INTERP)	Initialisiert die Funktionen der Library LIBMMACCESS und gibt die angebotenen Funktionen dem aktuellen Tcl-Interpreter INTERP zur weiteren Verwendung bekannt. Gibt immer den Wert TCL_OK (1) zurück.

LIBMMAUDIO

void MMaudio_audio_open ( )	Öffnet das entsprechende Audio*controldevice* exklusiv zur Nutzung mit den weiteren Funktionen. Diese Funktion sollte von Tcl-Programm nach MMaudio_Init () aufgerufen werden. Gibt keinen Wert zurück.
void MMaudio_set_play_volume (int VOLUME)	Stellt die Lautstärke auf den gegebenen Wert **VOLUME** ein (zwischen 1 und 100). Gibt keinen Wert zurück.
void MMaudio_set_record_volume (int VOLUME)	Stellt die Aufnahmeempfindlichkeit auf den gegebenen Wert **VOLUME** ein (zwischen 1 und 100). Gibt keinen Wert zurück.
void MMaudio_set_line_out ( )	Lenkt die Audioausgabe auf LineOut um. Gibt keinen Wert zurück.
void MMaudio_set_line_in ( )	Lenkt die Audioausgabe auf LineIn um. Gibt keinen Wert zurück.
int MMaudio_get_paused_state ( )	Überprüft, ob das Audio*device* auf Pause gestellt ist. Gibt den Wert 1 zurück, wenn dies zutrifft, ansonsten 0.
int MMaudio_get_play_volume ( )	Gibt den aktuell eingestellten Lautstärkewert zurück.
int MMaudio_get_record_volume ( )	Gibt den aktuell eingestellten Wert der Aufnahme-empfindlichkeit zurück.
int MMaudio_toggle_audio_pause (int PAUSE)	Setzt den Pause-Wert des Audio*devices* auf den übergebenen Wert **PAUSE** (0 oder 1). Gibt immer den Wert TCL_OK zurück.
int MMaudio_analyze_au (char *FILE)	Überprüft die übergebene Datei **FILE**, ob es sich um eine Audiodatei mit einem Sun-Audio*header* handelt. Gibt den Wert -1 bei einer Sun-Audiodatei mit einer nicht unterstützten Kodierung, den Wert -2 bei einem Fehler bzw. wenn es sich um keine Sun-Audiodatei handelt, ansonsten einen anderen Wert.
int MMaudio_analyze_wav (char *FILE)	Überprüft die übergebene Datei **FILE**, ob es sich um eine Audiodatei mit einem WAV-Audio*header* handelt. Gibt den Wert -1 bei einer WAV-Audiodatei mit einer nicht unterstützten Kodierung, den Wert -2 bei einem Fehler bzw. wenn es sich um keine WAV-Audiodatei handelt, ansonsten einen anderen Wert zurück.
int MMaudio_analyze_mpeg (char *FILE)	Überprüft die übergebenen Datei **FILE**, ob es sich um eine Audiodatei mit einem MPEG-Audio*header* handelt. Gibt den Wert -1 bei einer MPEG-Audiodatei mit einer nicht unterstützten Kodierung, den Wert -2 bei einem Fehler bzw. wenn es sich um keine MPEG-Audiodatei handelt, ansonsten den Wert 0.

int MMaudio_Init (Tcl_Interp *INTERP)	Initialisiert die Funktionen der Library LIBMMAUDIO und gibt die angebotenen Funktionen dem aktuellen Tcl-Interpreter INTERP zur weiteren Verwendung bekannt. Das Audio*controldevice* muß explizit geöffnet werden. Gibt immer den Wert TCL_OK (1) zurück.

LIBMMFILE

long file_size (char *FILE)	Gibt die Länge der gegebenen Datei **FILE** in Bytes zurück.
int file_copy (char *SOURCE, char *TARGET)	Kopiert die gegebene Datei **SOURCE** auf eine neue Datei namens **TARGET**. Gibt den Wert 1 bei erfolgreichem Kopiervorgang, ansonsten den Wert 0 zurück.
int file_remove (char *SOURCE)	Löscht die Datei **SOURCE**. Gibt den Rückgabewert der C-Funktion *unlink* zurück.
int file_rename (char *SOURCE, char *TARGET)	Benennt die Datei **SOURCE** in den Namen **TARGET** um. Gibt den Rückgabewert der C-Funktion *rename* zurück.
int MMfile_Init (Tcl_Interp *INTERP)	Initialisiert die Funktionen der Library LIBMMFILE und gibt die angebotenen Funktionen dem aktuellen Tcl-Interpreter INTERP zur weiteren Verwendung bekannt. Gibt immer den Wert TCL_OK (1) zurück.

Zur Einbindung dieser Libraries in andere Tcl/Tk-Applikationen und dem Aufruf der Funktionen via Tcl, muß ein applikationsspezifischer Tcl/Tk-Interpreter (wish) erzeugt werden. Dazu wird ein C-Hauptprogramm geschrieben (ein Beispiel ist in der Tcl-Distribution enthalten), dieses wird übersetzt und mit den Tcl/Tk- und Tix- sowie den hier vorgestellten Libraries ge*linkt*. Mit diesem speziellen Interpreter wird dann das eigene Tcl/Tk-Programm ausgeführt.

In die *Main*routine des Hauptprogrammes müssen je nach Bedarf folgende Zeilen zur Initialisierung der MM-Libraries eingefügt werden:

```
extern int MMpcss_Init ();
extern int MMaccess_Init ();
extern int MMaudio_Init ();
extern int MMfile_Init ();

if (MMpcss_Init (interp) == TCL_ERROR) return (TCL_ERROR);
if (MMaccess_Init (interp) == TCL_ERROR) return (TCL_ERROR);
if (MMaudio_Init (interp) == TCL_ERROR) return (TCL_ERROR);
if (MMfile_Init (interp) == TCL_ERROR) return (TCL_ERROR);
```