Technische Universität Ilmenau
Fakultät für Elektrotechnik und Informationstechnik
Institut für Medientechnik

Diplomarbeit

Konzeption einer Videodatenverteilung im Rahmen des Digitalen Video Projektes (DVP)

von

Florian Meffert

Autor:	Florian Meffert
geboren am:	20.05.1975 in Wolfenbüttel
Matrikel:	95
Studiennummer:	24820
Studiengang:	Elektrotechnik
Studienrichtung:	Elektronische Medientechnik
verantwortlicher Hochschullehrer:	Prof.Dr.-Ing. Karlheinz Brandenburg
betreuender wiss. Mitarbeiter:	Dipl.-Inf. Detlef Streitferdt

ausgegeben am:	15.08.2002
Abgabetermin am:	11.02.2003
Registriernummer:	2181-02D-30

durchgeführt in:	Fakultät für Informatik und Automatisierung Fachgebiet Prozessinformatik

Ilmenau, den 12. Februar 2003

Danksagung

Als erstes möchte ich Herrn Prof. Dr.-Ing. K. Brandenburg danken, ohne dessen Zustimmung diese Arbeit nicht möglich gewesen wäre.

Ein großes Dankeschön geht ebenfalls an meinen Betreuer, Herrn Dipl.-Inf. D. Streitferdt, der mich in fachlicher und organisatorischer Hinsicht bei der Durchführung meiner Diplomarbeit unterstützte.

Herrn Roland Heiber und Herrn Ronny Stephan danke ich für ihre offenen Ohren und kleinen Tipps in speziellen Angelegenheiten.

Inhaltsverzeichnis

Einleitung

Die zunehmende Bedeutung der digitalen Formate für Audio und Video setzt sich in unserer heutigen Kommunikationsgesellschafft immer weiter fort. Die Übertragungsformate für Bild und Ton werden bereits in naher Zukunft vollständig digital sein. So wird die analoge Fernsehtechnik in den nächsten Jahren durch eine digitale Übertragungstechnik, die *Digital Video Broadcasting*-Übertragung (DVB), verdrängt werden. Zudem verschwimmt immer mehr die strenge Unterscheidung zwischen den Komponenten der Unterhaltungselektronik und der PC-Technik. In vielen Haushalten finden sich heutzutage mehrere leistungsfähige PCs, die problemlos in der Lage sind, sowohl als Multimedia-Endgerät oder auch als eine Station zur Verteilung von Multimedia-Daten zu dienen.

Das *Digitale Video Projekt* (DVP) möchte sich diesen Umschwung zur Digitaltechnik zu nutze machen. So steht in dem Projekt die Entwicklung eines Systems zur Be-und Verarbeitung von digitalen Videodaten im Vordergrund. Das Ziel des *Digitalen Video Projektes* ist die Weiterentwicklung eines digitalen Videorekorders, dem *VideoDiskRecorder* (VDR) [1], auf der Basis von handelsüblichen PC-Komponenten, der alle Funktionen eines herkömmlichen Gerätes mit den Vorzügen der Digitaltechnik kombiniert. Einzelne Software-und Hardwarekomponenten sollen hierbei beliebig miteinander kombiniert werden, was zu einer Vielzahl möglicher Anwendungen führen soll.
Eine dieser möglichen Anwendungen soll dabei eine Verteilung der gespeicherten Videodaten des digitalen Videorecorders sein. Die Videodatenverteilung soll über ein lokales Netzwerk, ein Übertragungsmedium welches sich in einem Haushalt befinden kann, an eine bestimmte Anzahl von Endgeräten stattfinden. Demnach soll der digitale Videorecorder zusätzlich die Funktion eines Videoservers einnehmen, der sich sowohl mit dem Übertragungsprozess der Videodaten als auch mit dem Steuerungsprozess der Videodatenverteilung befassen soll.

Im Rahmen dieser Diplomarbeit soll eine Videodatenverteilung sowie das dazu nötige Umfeld recherchiert und bewertet werden. Aufgrund einer anschließenden Analyse verschiedener Übertragungsmedien, soll eine Entscheidung für die Umsetzung und Anpassung einer bestimmten Übertragungsvariante für das *Digitale Video Projekt* getroffen werden. Eine problemlose Integration der Videodatenverteilung in die Umgebung des DVP-Systems soll dabei im Vordergrund stehen. Ziel dieser Arbeit ist die Konzeption und anschließende Umsetzung einer Videodatenverteilung innerhalb des *Digitalen Video Projektes*. Die konzeptionelle Erstellung wird den Entwurf und die softwaremäßige Umsetzung einer nutzbaren Anwendung umfassen.

Kapitel 1

Stand der Technik

In diesem Kapitel werden einige Grundlagen abgehandelt, die zum Verständnis dieser Arbeit und der im Kapitel 3 dargestellten Konzeption notwendig sind.

1.1 Digital Video Broadcasting (DVB)

1.1.1 Historie des Digitalen Video Broadcasting

Zu Beginn der 90-er Jahre entstanden erste Ideen, das klassische analoge Fernsehen durch das digitale Fernsehen abzulösen, vorrangig aufgrund der verschiedenen Entwicklungen in den USA, Japan und Europa. Die immer besser werdende digitale Signalverarbeitung und vor allem die immer leistungsfähigeren Mikrorechner sowie neuartige Komprimierungsverfahren bestärkten die Idee des Umstiegs auf die Digitaltechnik. Als ein Meilenstein in der Geschichte des digitalen Fernsehens ist der Start des *European Digital Video Broadcasting Project* am 10. September 1993 zu sehen. Ziel war es, die kommerziellen wie auch die technischen Interessen gleichrangig zu verfolgen und dass als Ergebnis ein Standard für alle entstehen sollte. Momentan umfasst das *Digital Video Broadcasting Project* hauptsächlich Mitglieder von Firmen und Organisationen, wie z.B. Rundfunk- und Fernsehsender, Kabel- und Satellitenbetreiber, Behörden sowie die Europäische Kommision [2].

Im Laufe der Zeit haben sich die am Anfang gesetzten Entwicklungsziele erheblich verändert, wodurch viele internationale Normen und Standards [3] für das digitale Fernsehen entstanden sind. Unter anderem wurde auch die Spezifikation eines *Application Programming Interface* (API) für die Programmierung

von DVB-konformen Applikationen entwickelt. Die für diese Diplomarbeit verwendete Projektsoftware VDR wird im Rahmen des *Digitalen Video Projektes* (DVP) der Fakultät Prozessinformatik der TU Ilmenau genutzt; sie baut auf der DVB-API auf.

Zusammenfassend bieten die DVB-Lösungen für digitales Fernsehen eine Reihe von Vorteilen:

- Die Erhöhung der Anzahl von Fernsehprogrammen, die über eine bestimmte Sendefrequenz übertragen werden kann.
- Die Option, die Bild- und Audioqualitäten an die jeweiligen Anwendungen anzupassen.
- Die Verfügbarkeit besonders sicherer Verschlüsselungsmethoden für Pay-TV-Dienste.
- Die Verfügbarkeit von Werkzeugen zur Entwicklung und Implementierung neuer Dienste. Dies sind z.B. Daten-Broadcast, Multimedia-Broadcast und *Video-on-Demand*.
- Die Möglichkeit, Funktionalitäten bzw. Dienste von Fernseher, HiFi Anlage und Computer in einem "Multimediaterminal" zu verschmelzen. Dieser Gedanke spielte auch bei der Entwicklung des *Digitale Video Projektes* (DVP) eine große Rolle.

1.1.2 Mögliche Verbreitungswege von DVB

Für die Verteilung und Verbreitung der digitalen Fernsehsignale gibt es heutzutage laut Ziemer [4] im Wesentlichen drei Standards.

1. **DVB-S:** Dieser Übertragungsstandard definiert die Satellitenübertragung im 11/12 GHz-Band, das für unterschiedliche Transponder, Bandbreiten und Leistungen konfigurierbar ist. Diese Form der unidirektionalen Verbreitung wurde als erste Form spezifiziert und in Deutschland eingeführt. Der Systemaufbau des *Digitalen Video Projektes* besteht aus DVB-S Komponenten. Der Verbreitungsweg wird im Abschnitt 1.1.3 näher erläutert, und speziell für diese Diplomarbeit im Kapitel 3 genauer betrachtet.

2. **DVB-C:** Der Übertragungsstandard dient zur Verbreitung der digitalen Signale im Breitbandkabelnetz. Er ist kompatibel zum DVB-S Standard und für 8 MHz-Kabelkanäle ausgelegt. Die Verteilung der Fernsehsignale auf die Kabelkanäle erfolgt mit Hilfe der QAM[1]-Technologie.

3. **DVB-T:** Ein Übertragungsstandard für die terrestrische Funkausstrahlung von digitalem Fernsehen und digitalem Datenrundfunk für 7 bis 8 MHz-Kanäle. In jüngster Vergangenheit wurde durch Pilotversuche in mehreren Großstädten gezeigt, dass das digitale, terrestrische Fernsehen gemäß dem DVB-T Standard sowohl für den stationären als auch mobilen Einsatz geeignet ist. Gegenwärtig werden politische und auch wirtschaftliche Anstrengungen unternommen, einen DVB-T Regeldienst in Deutschland zu etablieren.

1.1.3 Die DVB-S Komponenten und deren Aufgaben

Ein Digitales Video Broadcasting (DVB) Satellitennetzwerk besteht im Wesentlichen aus vier Einheiten, die in Abbildung 1.1 dargestellt werden. Die ersten drei werden als Head-End-Enheiten bezeichnet [5].

1. Die Datenkomprimierungs-Einheit. Innerhalb dieser Einheit wird das bereits in digitaler Form vorliegende Video-/Audio-Material nach dem MPEG-2 Standard komprimiert und als Folge von MPEG-2 Datenpaketen mit fester Länge formatiert.

2. Die Multiplex-Einheit. Diese dient der Bildung eines MPEG-2 Transportstromes, siehe Abschnitt 1.2.1.

3. Die Übertragungseinheit. Mit Hilfe einer QPSK[2]-Modulation werden die Datenpakete für die Übertragung über das physikalische Medium entsprechend aufbereitet.

4. Das Empfangs- oder auch Endsystem führt im Wesentlichen die inversen Operationen durch, die in der Head-End-Einheit durchgeführt wurden.

Für die Steuerung und Konfiguration der Encoder und des Multiplexers wird ein sogenanntes *Network Management System* (NMS) benutzt. Nach dem Abschluss

[1] Quadrature Amplitude Modulation
[2] Quarternary Phase Shift Keying

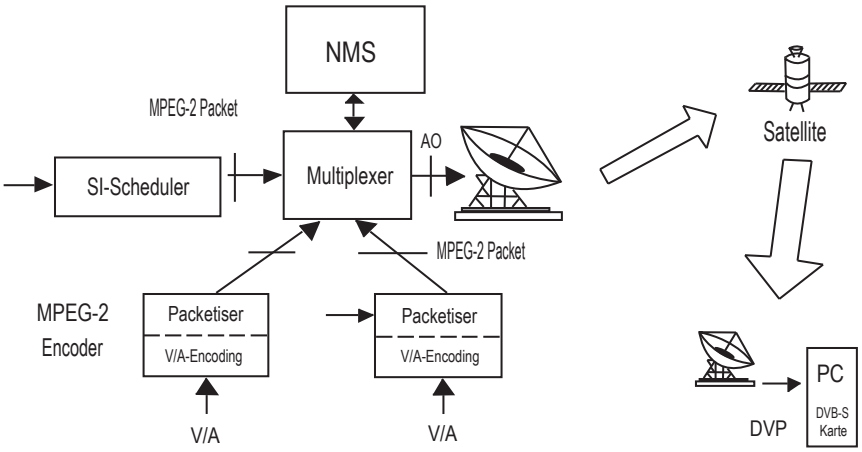

Abbildung 1.1: Zentrale Komponenten eines DVB-Satellitennetzwerkes

der Konfiguration liegen die Anzahl der Video-/Audio- und Datendienste fest und damit auch die Anzahl der Video-/Audio- und Datenkomponenten, die je den einzelnen Dienst bilden. Das NMS generiert daraufhin die notwendigen Steuertabellen, die laut Reimers [5] als *Program Specific Information* (PSI) bezeichnet werden. Die PSI-Tabellen werden als Datenkarussel zyklisch ausgespielt und werden am Empfangsgerät benötigt, um die entsprechenden Video- und Audiokomponenten wieder zusammenzusetzen. Für die Beschreibung der Fernsehdienste stehen weitere Tabellen zur Verfügung. Namen, Inhalt, Dauer, etc. sind in *Service Information* (SI) Tabellen enthalten, die vom Anbieter (z.B. Premiere) an den Netzwerkbetreiber in "Rohform" ausgeliefert werden.
Für die Erzeugung der eigentlichen *Service Information*, die Formatierung, und die Ausspielung wird ein eigenes System, ein sogenannter *Scheduler*, eingesetzt. Die *Service Information* ist notwendig, damit das Empfangsgerät eben diese Zusatzinformation zu dem Kanal mit den jeweiligen Diensten darstellen kann [6].

1.2 Die Verwendung von MPEG-2

Eine der ersten wichtigen Entscheidungen des Europäischen DVB-Projektes war die Auswahl des Komprimierungsstandards MPEG-2 für die Quellenkodierung von Audio- und Videodaten, sowie der MPEG-2-Systemtechnologie für das Erzeugen der unterschiedlichen MPEG-2 Stream-Typen. Da man allerdings praktischen Anforderungen folgen und ökonomische Implementierungen ermöglichen wollte, konnte der Original-MPEG-2-Standard [7] aufgrund seiner Vielfalt nicht verwendet werden. Die Syntax und möglichen Parameter mussten eingeschränkt werden. Die Vorgehensweise und Richtlinien, die für DVB zutreffen und verwendet werden, sind im *Guidelines Document* [8] enthalten.
Im folgenden Abschnitt wird nun kurz auf die Bildung sowie den Inhalt von MPEG-2 Datenpaketen eingegangen.

1.2.1 Die MPEG-2 Stream-Typen

Die kodierten Video-/Audiodaten, auch als *Elementary Streams* (ES) bezeichnet, werden in *Packetized Elementary Streams* (PES) oder Sections[3] verpackt. Diese werden innerhalb des Multiplexers in MPEG-2 Pakete zerlegt. Die einzelnen Pakete werden im Zeitmultiplexverfahren zu einem seriellen Datenstrom zusammengefügt. Bei Satellitennetzwerken wird dieser Datenstrom vor der Sendung mit Redundanzinformation versehen, damit Fehlerschutz- und Korrekturmechanismen genutzt werden können. Die MPEG-2 Stream-Typen können also folgendermaßen aufgeteilt werden [5]:

- *Elementary Stream* (ES): Die einzelnen Video- und Audioströme, komprimiertes Audio- und Videomaterial nach dem MPEG-2 Standard[4] werden auch Elementarströme genannt.

- *Packetized Elementary Stream*(PES): Unter PES versteht man Ströme aus Datenpaketen, welche bis zu 64 kByte groß sein können. Diese Pakete sind unter anderem auch mit einem Zeitstempel versehen, damit eine Synchronisation mit anderen PES (Video-/Audioströme) möglich ist. Weiterhin können diese *Packetized Elementary Streams* zu einem Programm- oder Transportstrom kombiniert werden.

[3] Datenblöcke zur Übertragung von SI, Anwenderdaten oder IP-Paketen
[4] ISO/IEC 13818

- *Program Stream* (PS): Der Programmstrom ähnelt dem MPEG-1 Strom[5], sollte aber in einer Umgebung mit einer niedrigen Fehlerrate eingesetzt werden. Die Pakete des Programmstroms können eine variable Länge aufweisen. Man kann den Zeitstempel dieses Stroms dazu verwenden, eine konstante Ende-zu-Ende Verzögerung zu implementieren.

- *Transport Stream* (TS): Der Transportstrom bündelt die PESs und eine oder mehrere unabhängige Zeitbasen in einem einzelnen Strom. Dieser Strom wurde zur Verwendung für verlustbehaftete oder verrauschte Medien entwickelt. Der Transportstrom ist zur Übertragung digitalen Fernsehens und Videotelefons über Glasfaser, Satellit, Kabel, ISDN, ATM und anderen Netzwerken sowie zur Speicherung auf digitalen Videobändern und anderen Geräten gut geeignet.

- *Multiple Program Transport Stream* (MPTS): Da auf einem Übertragungskanal (Transponder) meist deutlich mehr Bandbreite (ca. 38 Mbit/s beim DVB-S Standard) zur Verfügung stehen, als ein einzelner Dienst benötigen würde, können mehrere Dienste und damit mehrere Transponderströme auf ein und demselben Übertragungskanal im Zeitmultiplex-Verfahren übertragen werden.

1.2.2 Der Transport Stream

Der Schwerpunkt wird auf das MPEG-2 Transportsystem gelegt, da im weiteren Verlauf dieser Arbeit nur der MPEG-2 Transportstrom für die Verteilung von Videodaten relevant ist. Auf ihn wird nun näher eingegangen.

Die PESs werden innerhalb des Multiplexers in Pakete konstanter Länge zerlegt, die am Ausgang einen sogenannten MPEG-2 Transportstrom ergeben. Ein MPEG-2 TS-Paket ist immer 188 Byte lang und besteht aus einem 4 Byte langen Header (Kopf), einem Adaption Field und dem Payload. Im Letzteren sind die eigentlichen Nutzdaten enthalten. Obwohl die einzelnen kodierten Videoströme aufgrund ihres Inhaltes eine unterschiedliche Bandbreite benötigen, weist der am Ausgang des Multiplexers vorhandene Strom eine konstante Bandbreite auf.

[5] ISO/IEC 11172

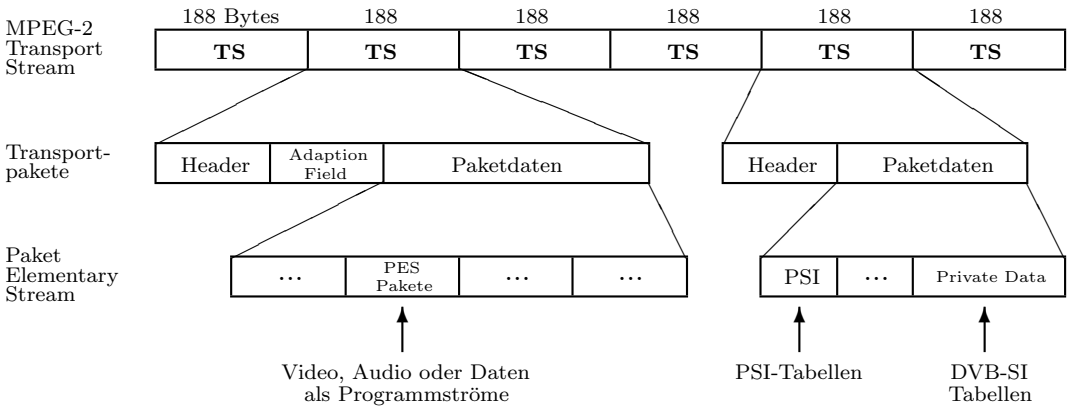

Abbildung 1.2: Inhalte eines MPEG-2 Transportstromes

Für die Übertragung von TS-Paketen spielt der Header mit seinen Komponenten eine wichtige Rolle, wie z.B. zur Synchronisierung des Empfangssystems. Außerdem kommt dem *Program Identifier* (PID)-Parameter eine besondere Bedeutung zu. Da Video-, Audio- und Datenpakete in einem Zeitmultiplexverfahren gebildet und übertragen werden, ist es notwendig, eine Referenz zu einer Einheit herzustellen, damit eine Filterung und ein Zusammenbau am Empfänger möglich ist. Diese Referenz bildet der PID-Parameter. Der PID-Parameter belegt 13 Bit innerhalb des 32 Bit langen Headers.

Ein DVB konformer Transportstrom besteht laut Reimers [6] also aus einem Strom von MPEG-2 TS-Paketen, deren Inhalt sich aus Video-/Audiopaketen, privaten Sections sowie SI- und PSI-Tabellen zusammensetzt.

1.3 Grundlagen der Netzwerktechnik

Der Informationsaustausch zwischen einzelnen Computersystemen in heterogenen Netzen wird in einem Modell abstrahiert, das die Basis für die Entwicklung von Protokollen zur Datenübertragung darstellt. In dieser Arbeit steht insbesondere die Datenübertragung von digitalem Video im Vordergrund, so dass hauptsächlich die hierfür notwendigen Grundlagen betrachtet werden.

1.3.1 Das OSI-Referenzmodell

Das *Open Systems Interconnection* Modell (OSI) wurde 1974 von der *International Organization for Standardization* (ISO) entwickelt und beschreibt die Kommunikation zwischen Systemen in einem offenen Schichtenmodell. Die verschiedenen Bereiche werden auf sieben separate Schichten verteilt, die klar voneinander abgegrenzt sind, so dass Änderungen bei den Standards einer Schicht keinen Einfluss auf Standards anderer Schichten haben. OSI definiert Dienste (siehe 1.3.2) und Funktionen der Schichten, die durch unterschiedliche Protokolle erfüllt werden können. Die folgende Darstellung gibt einen Überblick über die einzelnen Schichten und ihre Reihenfolge:

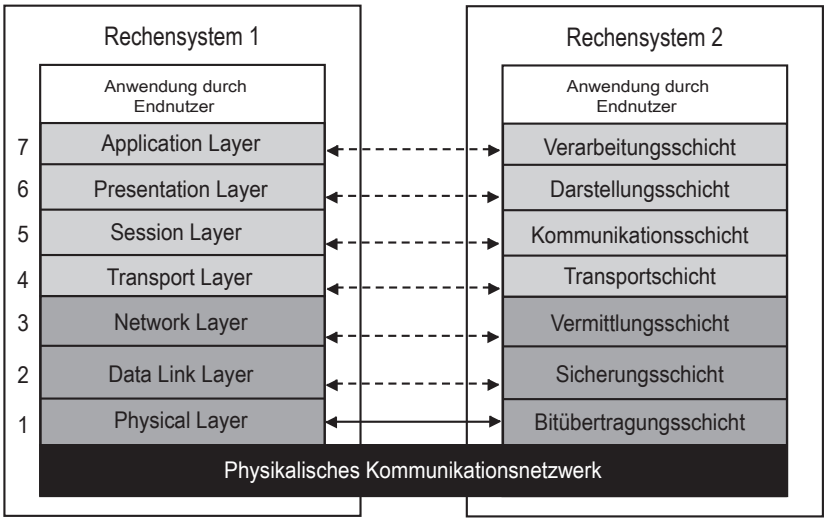

Abbildung 1.3: Schichten des OSI-Referenzmodells

Die Schichten 1-3 betreffen Protokolle für die Kommunikation zwischen unmittelbar benachbarten Einrichtungen auf einer Übertragungsstrecke. Die Schichten 4-7 betreffen Protokolle für die Kommunikation von Endpunkt zu Endpunkt (Sender und Empfänger) [9].

Da sich diese Diplomarbeit mit der Datenübertragung in Netzwerken befasst, spielt die Vermittlungsschicht (Network Layer) sowie die Transportschicht (Transport Layer) mit ihren Protokollen eine besondere Rolle. Die Vermittlungsschicht (Network Layer) ist für die Einrichtung, den Betrieb und die Auslösung von Netzverbindungen zwischen offenen Systemen verantwortlich. Routing, Interpretieren von Adressen und optimale Wegewahl der Verbindung gehören ebenfalls dazu. Darüberhinaus ist das Multiplexen von Verbindungen auf die Kanäle der einzelnen Teilstrecken zwischen den Netzknoten eine weitere Aufgabe dieser Schicht.
Die Transportschicht ist für den Ende-zu-Ende Datentransport zuständig. Der Beginn und das Ende einer Datenübermittlung sowie der Datenfluss werden gesteuert, Nachrichten werden segmentiert und zusammengefasst. Weitere Aufgaben der Schicht 4 sind die Fehlerbehandlung, Datensicherung, die Zuordnung zwischen symbolischen und physischen Geräteadressen und die Optimierung des Informationsweges. Als Bindeglied zwischen den netzabhängigen Schichten 1-3 und den völlig netzunabhängigen Schichten 5-7 stellt sie den höheren Schichten eine netzunabhängige Schnittstelle zur Verfügung. Dadurch werden laut Steinmetz [10] Anwendungen unabhängig vom jeweils verwendeten Netztyp.

Die Integration multimedialer Datenströme verlangt neben einer äußerst leistungsfähigen Netzinfrastruktur vor allem auch geeignete Kommunikationsprotokolle. Eine Betrachtung und Analyse der jeweilig nötigen Protokolle bei unterschiedlichen Übertragungsmedien folgt im Kapitel 2.3. Im nächsten Abschnitt wird ein kurzer Überblick der verschiedenen Protokolle aufgezeigt.

1.3.2 Dienste und Protokolle

Ein netzwerkfähiges Multimediasystem erlaubt den Datenaustausch diskreter und kontinuierlicher Videodaten zwischen einer Anzahl von Geräten. Diese Kommunikation erfordert geeignete Dienste und Protokolle für die Datenübertragung [11].

- Ein Dienst stellt seiner jeweiligen Anwendung eine Menge von Operationen zur Verfügung. Zusammengehörige Dienste werden zu einzelnen Schichten zusammengefasst (siehe OSI-Modell). Damit bietet eine solche Schicht der jeweiligen darüberliegenden Schicht ihre Dienste an. Diese beschreiben das

Verhalten der Schicht und ihre Dienstelemente, den *Service Data Units* (SDUs).

- Ein Protokoll besteht aus einer Anzahl von Regeln, die zwischen Partnerinstanzen einer Schicht auf Rechnern gelten, welche miteinander kommunizieren. Ein Protokoll beinhaltet das Format (die Syntax) und die Bedeutung der auszutauschenden Dateneinheiten, den Protokolldateneinheiten (*Protocol Data Units* PDUs). Die Partnerinstanzen auf den verschiedenen Rechnern kooperieren, um einen Dienst zu erbringen.

Als Überblick einzelner Protokolle der jeweiligen Schichten folgt eine Auflistung bestimmter Protokolle in der nächsten Tabelle:

	Schicht	**Beispiele**
7	Anwendungsschicht	Web-Browser, Mail-Programm
6	Darstellungsschicht	ASCII, HTML, XML, MIME
5	Steuerungsschicht	HTTP, FTP, POP3, SMTP
4	Transportschicht	**TCP**, **UDP**, SPX, NetBeui
3	Vermittlungsschicht	IP, IPX, X.25, T.70, T.90NL
2	Sicherungsschicht	PPP, X.75, LAP, HDLC, T.30
1	Bitübertragungsschicht	IEEE 802, ATM, V.110

Tabelle 1.1: Beispiele einiger Protokolle

Wichtige Protokolle für den weiteren Verlauf und das Konzept dieser Arbeit sind die Protokolle: TCP und UDP. Sie werden zu einem späteren Zeitpunkt im Abschnitt 2.4.4 noch genauer betrachtet.

1.3.3 Die Netztopologie

Unter Topologie versteht man hier die physikalische Struktur des Netzwerkes. Sie ist das architektonische Bild von vorhandenen Geräten und Verbindungen, welche die Konfiguration des Netzes darstellt. Die laut Chimi [12] am häufigsten verwendeten Arten von Topologien sind: Bus, Stern, Ring. Die Erläuterung folgt anhand des Beispiels von Ethernet Netzwerken.

Bus-Topologie:

Bei einem als Bus aufgebauten Netzwerk ziehen alle Rechner an einem Strang: Ein einziges Kabel verläuft vom ersten bis zum letzten der maximal 30 Rechner (Thin Ethernet, auch Cheapernet genannt). Diese Technik wurde als IEEE 802.3 genormt und eignet sich im Prinzip bis zu einer Datenrate von 10 MBit/s, einige Speziallösungen schaffen auch 20 MBit/s.
Die maximale Länge vom einen bis zum anderen Ende des Kabels darf bei "Thin Ethernet" 190 m nicht überschreiten, da sonst einerseits die Kollisions-Erkennung wegen zu großer Signallaufzeiten nicht mehr funktioniert, andererseits aber auch die Kabeldämpfung zu groß wird. Der Mindest-Abstand zweier Rechner beträgt 0,5 m. Wenn ein solches Koaxkabel-Netz einmal nicht funktioniert, ist häufig ein Kabelproblem die Ursache.
Heutzutage findet diese Art der Topologie sowie die Benutzung von Koaxialkabel in einem Netzwerk kaum noch Verwendung.

Stern-Topologie:

Bei einer Stern-Topologie benutzt man normalerweise einen zentralen *Hub* mit RJ45-Buchsen[6] als Knoten. An einen *Hub* können eine Anzahl von Rechnern mit jeweils einem eigenen Kabel angeschlossen werden, wobei *Hubs* für Netzwerke ab 3 Rechnern benötigt werden. Die Verbindung erfolgt über ein massefreies, symmetrisch verdrilltes TP-Kabel (Twisted Pair).
Gegenüber der Bus-Technologie hat diese Art der Verkabelung den Vorteil, dass der Ausfall eines Kabelsegments nicht gleich das gesamte Netzwerk lahmlegt. Ferner sind außer den herkömmlichen 10 MBit/s (10BaseT) damit höhere Datenraten möglich: Die *Fast-Ethernet*-Netze (100BaseTX) schaffen 100 MBit/s, das *Gigabit*-Netz (1000BaseT) sogar bis 1 GBit/s zu übertragen.
Hubs besitzen meist 4, 6, 8 oder 16 RJ45-Anschlüsse für STP-Kabel (*Shielded Twisted Pair*) oder UTP-Kabel (*Unshielded Twisted Pair*). Letztere sollte man nicht zu dicht an anderen Kabeln verlegen, da sie störanfälliger sind. Einige Hubs besitzen zusätzlich noch einen BNC-Anschluß, damit sie mit einem auf der Bus- oder Ring-Topologie basierenden Netz-Segment verbunden werden können, das beispielsweise die einzelnen Stockwerke eines Gebäudes miteinander verbindet.

[6] gleich den ISDN-Anschlüssen

Ring-Topologie:

Im PC-Bereich ist die Ring-Topologie vor allem in Zusammenhang mit Glasfaser-Netzen bekannt (*Fiber Distributed Data Interface* = FDDI, standardisiert durch ANSI X3T9 und ISO 9314). Bis zu 1000 Rechner-Stationen sind ringförmig miteinander verbunden, und die Entferung kann rund 200 km betragen, weit mehr als bei den anderen Topologien. Ein Ring-Protokoll (Token-Ring) dient zur Vermeidung von Datenpaket-Kollisionen: Ein spezielles Datenpaket, das "Token", wird zyklisch von einer Station abgesandt und durchläuft den Ring im Uhrzeigersinn. Jeder Rechner, der etwas zu senden hat, kann an dieses Token seine Nutzdaten anhängen.
Das Kabel braucht natürlich nicht wirklich ringförmig verlegt zu werden. Hin- und Rückleitung können in einem einzigen Strang untergebracht werden, so daß sich der Netz-Aufbau ganz ähnlich gestaltet wie etwa beim Verbinden mehrerer ISDN-Geräte an einem Bus.

Aus Kostengründen ist die Bestimmung einer optimalen Topologie eine wichtige Designaufgabe, insbesondere, wenn die geographische Ausdehnung des Netzes groß ist. In der Regel wird es selten möglich sein, dass jeder Knoten physisch mit jedem anderen verbunden werden kann.

1.3.4 Übertragungsmodi von Datenpaketen

Netzwerke sind paketorientierte Verbindungen. Die Daten werden als kleine, zerlegte Pakete verschickt und anschließend beim Empfänger wieder zusammengesetzt. In der Literatur werden die Übertragungsmodi von Datenpaketen in drei Arten unterteilt [10]. Hier wird speziell eine Betrachtung für die Übertragung von Videodatenpakete durchgeführt:

- **Asynchron:** Die Pakete sollen so schnell wie möglich den Empfänger erreichen, dabei besteht jedoch keinerlei zeitliche Beschränkung. Dieser Modus ist für Streaming Video nicht brauchbar, da er völlig unberechenbar ist. Jedes Paket nimmt einen beliebigen Weg, und somit können die Pakete in ihrer Reihenfolge durcheinander gewürfelt werden. Es kann außerdem vorkommen, dass der Empfänger eine Zeit t gar keine Daten und im nächsten Moment viele Pakete nacheinander erhält. Diese Videodaten kann der Empfänger höchstwahrscheinlich aufgrund seiner eingeschränkten Bandbreite

nicht annehmen. Dies könnte einen Stau verursachen und das Video stocken lassen.

- **Synchron:** Bei dieser Übertragungsart ist eine maximale Ende-zu-Ende-Verzögerung [11] definiert. Diese Zeit wird nie überschritten. Jedoch können Pakete beliebig zu früh eintreffen und müssen daher in Folge zwischengespeichert werden.

- **Isochron:** Die maximale sowie minimale Ende-zu-Ende-Verzögerung sind definiert. Pakete kommen exakt innerhalb eines gewählten Zeitfensters an. Damit wird erreicht, dass die nötige Puffergröße beim Empfänger eingeschränkt werden kann.

Für kontinuierliche Daten wie z.B. Streaming Video ist der isochrone Übertragungsmodus ideal geeignet, da die Daten rechtzeitig den Empfänger erreichen und niemals zu viele Daten zwischengespeichert werden müssen.
Leider lässt sich dieser Modus in Netzwerken nur schwer oder gar nicht realisieren, weswegen Streaming-Protokolle einen synchronen Modus voraussetzen. Eine weitere Betrachtung und Erläuterung der Streamingverfahren und Streaming-Protokolle folgen im Abschnitt 2.4.

Kapitel 2

Videodatenübertragung

Wie bereits in der Einleitung erwähnt wurde, soll in dieser Arbeit eine Analyse der möglichen Verteilung von Videodaten in einem Haushalt durchgeführt werden. Die Idee, die dieser Verteilung zugrunde liegt, kann in dem Schlagwort "Video-on-Demand" zusammengefasst werden. Abbildung 2.1 stellt dabei das Grundprinzip dar. Die im Rahmen dieser Diplomarbeit relevanten Teile sind die graumarkierten Server- und Client-Komponenten sowie der Übertragungsweg zwischen ihnen.

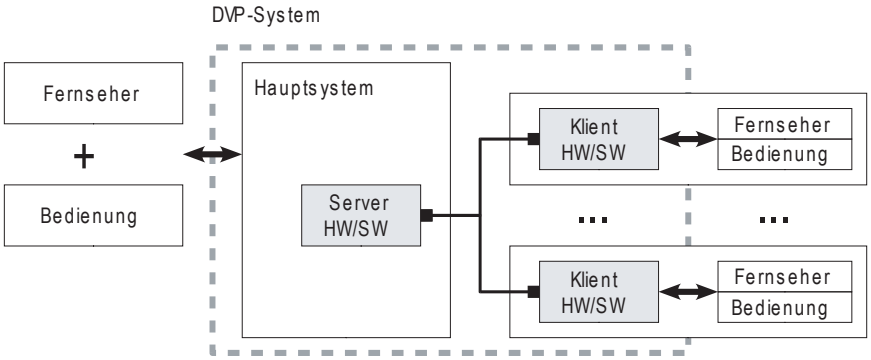

Abbildung 2.1: Grundprinzip des DVP-Systems

Nach einigen grundlegenden Betrachtungen folgt im Abschnitt 2.3 eine Analyse der verschiedenen Übertragungsmedien zwischen dem Hauptsystem (Server) und dem entfernten System (Client). Weiterhin wird ein nötiges Übertragungsverfahren von digitalem Video im Abschnitt 2.4 analysiert.
Im letzten Teil dieses Kapitels wird die konkretisierte Aufgabenstellung dieser Diplomarbeit dargestellt. Sie dient als Ausgangspunkt für das zu erarbeitende Konzept, welches ausführlich in Kapitel 3 beschrieben wird.

2.1 Warum der Einsatz lokaler Netzwerke?

Ein wichtiges Argument für eine lokale Vernetzung, auch als *Local Area Network* (LAN) bezeichnet, ist der Zugang zu einer Vielfalt von Informationen und Daten. In unserem Falle sichert die lokale Vernetzung einem privaten Nutzer die Verfügbarkeit von Videodaten an unterschiedlichen Nutzungsorten. Hinzu kommt, dass die Möglichkeiten der Archivierung von Daten und deren Austausch sich in den vergangenen Jahren verbessert hat. Die persönliche Videoarchivierung ist mittlerweile für die Heimanwendung erschwinglich, Formate wie MPEG-1 oder MPEG-2 lösen die analoge Technik ab. Die Speicherung von AV-Inhalten sowie die Verbreitung optischer Medien, wie die DVD, lässt die Einrichtung zentraler Server sinnvoll erscheinen. Warum sollen Speichermedien transportiert werden, wenn ein Netzwerk von verschiedenen Stellen aus den Zugriff auf eine eigene komplette Sammlung von Video- und Audioobjekten ermöglichen kann.
Ein weiteres Argument für eine lokale Vernetzung ist zudem die gemeinsame Nutzung von Hardwareressourcen. Grundsätzlich ist damit eine Reduktion der Beschaffungskosten und des Platzbedarfs möglich. Es ist jedoch abzuwägen, welche Zusatzkosten die Netzwerk-Hardware verursacht. Außerdem steigt mit der Vernetzung die Systemkomplexität und damit der Aufwand für die Entwicklung solcher Systeme. Jedoch bietet eine Vernetzung der Systeme die Möglichkeit der Betreuung und Wartung der einzelnen Komponenten. Es sollte darauf geachtet werden, dass das System mit seinen Komponenten anwenderfreundlich und übersichtlich bleibt und keinen eigenen Administrator benötigt. Natürlich gibt es auch Argumente gegen die Vernetzung. So ist die Entwicklung und vor allem die Installation eines Client/Server-Systems aufwändiger, als die einer Anzahl von einzelnen Mediengeräten. Vor allem die Netzwerk-Hardware verursacht in diesem Fall Kosten.

Somit müssen bei einem Entwurf und dem Betrieb breitbandiger lokaler Netzwerke für die Übertragung von Videodaten alle Argumente sorgfältig betrachtet werden, bevor die Umsetzung eines privaten lokalen Netzwerkes in Erwägung gezogen wird. Als einen wichtigen Vorteil bietet die Vernetzung des DVP-Systems z.B. die Option eines gemeinsamen Bedienkonzeptes für alle Funktionen des Systems. Das *Digitale Video Projekt* soll somit ein schlüssiges Gesamtkonzept mit einer Anzahl von Funktionalitäten bieten, eine Verschmelzung vieler einzelner Elektronikgeräte und Softwarekomponenten.

2.2 QoS-Parameter für die Übertragung

Die digitale Echtzeitübertagung von Audio- und Videodaten stellt bisher noch nicht dagewesene Anforderungen an die Rechenleistung von Computerprozessoren und die Eigenschaften der Übertragungsnetze. Digitales Video stellt dabei bei weitem die höchsten Ansprüche an die Übertragung: es benötigt eine große Bandbreite, es muss mit minimaler Verzögerung übertragen werden und toleriert keine hohe Fehlerrate. Auch nach der Komprimierung und Codierung (z.B. nach MPEG-2 Standard), benötigt eine Videoübertragung in annehmbarer Qualität immer noch 4 bis 10 MBit/s.
Verzögerungen bei Videodaten sind deshalb kritisch, weil die Einzelbilder mit einer Rate von 25 Bilder pro Sekunde dargestellt werden müssen. Echtzeitübertragungen vertragen keine Schwankungen der absoluten Verzögerungszeiten der einzelnen *Frames*[1], wie sie aber bei den meisten verfügbaren Netzen üblicherweise auftreten. Kommunikationsnetze sind selten fehlerfrei. Einzelne Bits können verfälscht werden (Bitverfälschung) oder es gehen sogar ganze Pakete verloren, was sich wiederum nachteilig auf die Übertragung auswirken kann, da das komprimierte Videosignal wegen seiner zuvor entfernten örtlichen und zeitlichen Redundanz besonders störanfällig ist.

Die vom Übertragungsnetzwerk zur Verfügung gestellte Leistung hängt von verschiedenen Parametern, wie garantierte Mindestdatenrate, der maximalen Verzögerungszeit, Spezifikation der maximalen Verlustrate (Bitfehlerrate) und Verzögerungsschwankungen (Netzwerkjitter) ab, die auch als *Quality of Service*-Parameter (QoS) bezeichnet werden. Viele der bestehenden Netzwerke garantieren dem Benutzer keine konstanten QoS-Parameter, da diese beliebig zeitlich variieren können. Um eine zuverlässige und effektive Videoübertragung über ein lokales Netzwerk anbieten zu können, ist eine Untersuchung der QoS-Anforderungen von kodierten Videosignalen notwendig [13].

2.2.1 Datenübertragungsrate

Gängige Videokodierverfahren sind üblicherweise verlustbehaftet. Bei jedem verlustbehafteten Videokodierverfahren ist im allgemeinen die durchschnittliche Bitrate der kodierten Daten proportional zur dekodierten Darstellungsqualität. Bei

[1] Einzelbilder

einer DCT[2]-basierten Videokodierung (z.B. MPEG-2) tritt der Qualitätsverlust beim Quantisieren auf. Je gröber die Quantisierung ist, desto schlechter wird die Bildqualität nach der Kodierung. Somit ist die Quantisierungsauflösung ein Schlüsselparameter für die Kontrolle der Videobitrate.

Bei einer Videoübertragung wird ein Kanal mit fester Datenrate benutzt, weswegen das kodierte Signal mit einer konstanten Datenrate (CBR-*Constant Bit Rate*) übertragen werden muss. Dies bringt jedoch Probleme mit sich, da die kodierten Videodaten hochgradig schwankende Datenraten besitzen. Damit diese Videodaten in einem Übertragungskanal mit konstanter Bandbreite untergebracht werden können, werden die Daten vor der Übertragung in einem Pufferspeicher zwischengelagert und mit konstanter Rate ausgelesen. Somit glättet man kurzzeitige Änderungen aus. Um auch längerfristige Schwankungen auszugleichen, müssten größere Pufferspeicher verwendet werden. Da dies aus Gründen der Verzögerungszeit meist nicht machbar ist, wird eine Rückkopplung zum Quantisierer verwendet. Durch diese Methode ist die Qualität des dekodierten Videosignals nicht konstant, welche sich in Abhängigkeit der Datenmenge am Ausgang des Quantisierers ändert.

Paketvermittelte Netzwerke können die Übertragung mit veränderlicher Datenrate (VBR-*Variable Bit Rate*) unterstützen. Dieses Verfahren bietet einige Vorteile gegenüber dem der konstanten Bitrate. Auf DCT-Basis enkodierte Videodaten liefern eine konstante Bildqualität beim Empfänger, da die Quantisierungsschritte immer konstant gehalten werden können. Allerdings kann es auch hier zu Problemen kommen, wenn beispielsweise viele Szenenwechsel oder schnelle Bewegungsabläufe die momentane Datenrate derart stark anwachsen lassen, dass die ausgehandelte maximale Datenrate im Übertragungskanal überschritten wird. Es müssen also auch hier Vorkehrungen getroffen werden, dass erstens die Bildqualität in solch einem Fall nicht leidet und zweitens trotzdem eine effiziente Nutzung der vorhandenen Ressourcen stattfindet.

[2] *Discrete Cosine Transform*

2.2.2 Fehler und Verluste

Die in der Praxis gebräuchlichen Übertragungsnetzwerke verfälschen die zu übertragenden Daten, indem sie Informationen verändern oder verlieren. Sogar eine relativ niedrige Fehlerrate oder ein sehr geringer Datenverlust kann Auswirkungen auf die dekodierten Sequenzen haben. Kompressionsalgorithmen entfernen viele Redundanzen aus dem vorhandenen Signal. Das führt umgehend zu Problemen, wenn das kodierte Videosignal verfälscht wird, viel mehr als dies beim unkodierten Signal der Fall wäre.
Aus diesem Grund kann ein kodiertes Videosignal nur eine sehr niedrige Fehlerrate tolerieren bevor die Qualität merklich abfällt. Laut Sojka [14] kann eine Fehlerwahrscheinlichkeit von mehr als 10^{-6} (ein Fehler auf 10^6 Bits) schon einen sichtbaren Qualitätsverlust bedeuten.

2.2.3 Verzögerungen

Aufgrund der bei der Kodierung notwendigen Pufferung von *Frames* kann es zu merklichen Verzögerungen kommen. Diese bewegen sich im Bereich von Millisekunden. Weiterhin können Verzögerungen durch das Übertragungsmedium Netzwerk entstehen. Diese können konstant und voraussagbar (z.B. bei ISDN) oder aber variable (z.B. in einem IP-Netzwerk) sein. Auch bei der Dekodierung werden durch Pufferungen Verzögerungen ausgelöst. Diese Werte bilden zusammen eine Gesamtverzögerungszeit, die bei einer qualitativ hochwertigen Videoübertragung einen bestimmten Wert nicht überschreiten darf [14]:

- Bei einer Zweiweg-Videodatenübertragung (Duplex) beispielsweise ist die Gesamtverzögerungszeit sehr kritisch. Beträgt diese mehr als 300 ms, wird eine ”natürliche” Konversation schwierig.

- Bei einer Einweg-Videodatenübertragung (Simplex) ist hingegen die Variation der Verzögerungszeiten kritischer. Ihre Größe ist proportional zu der Größe des Eingangspufferspeicher des Dekoders.

2.3 Analyse der Übertragungmedien

Als Übertragungsmedium können eine Anzahl von Möglichkeiten in Betracht gezogen werden, die sich in einem Haus anbieten. Für eine zukünftige Hausvernetzung könnte also jede der im Bild 2.2 dargestellten Netzstrukturen in Erwägung gezogen werden.

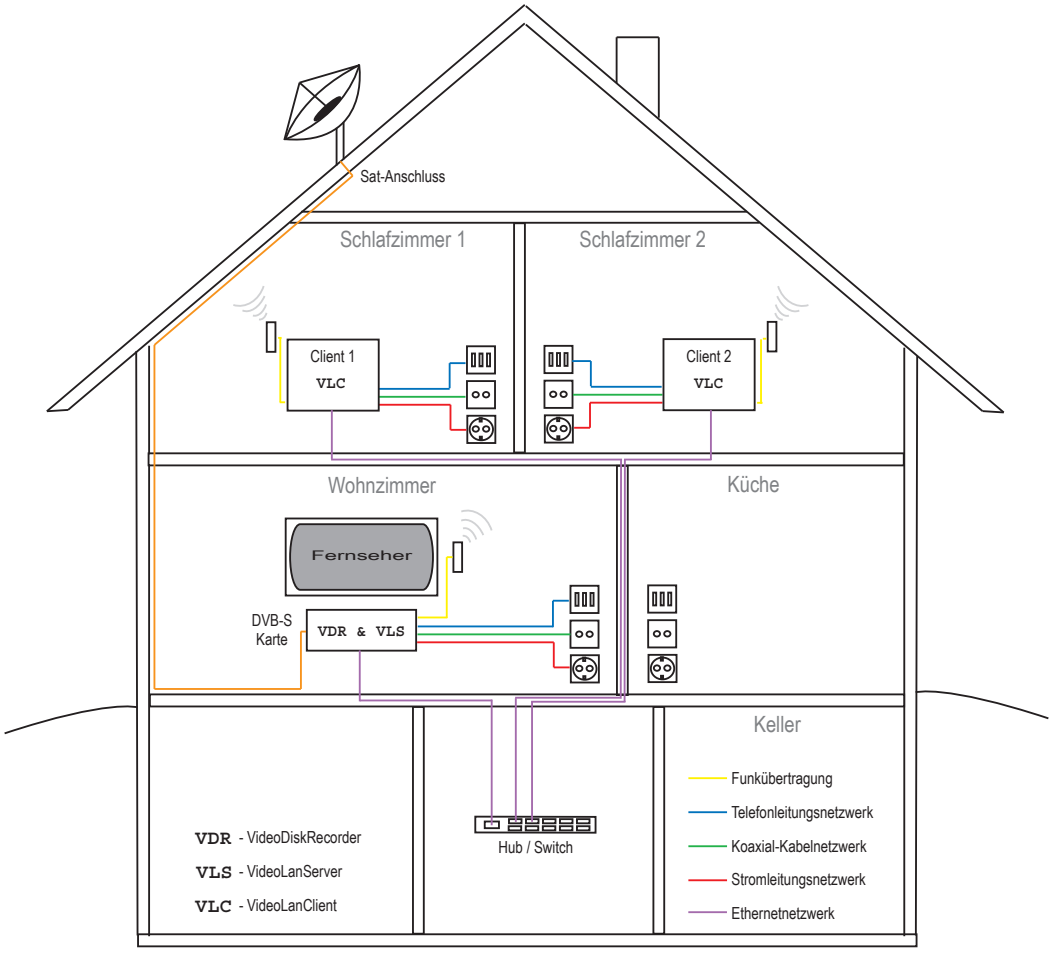

Abbildung 2.2: Das ”Vernetzte Haus” nach dem DVP-Prinzip

Ein wesentliches Kriterium für die Planung eines vernetzten Hauses ist jedoch die Frage des Installationsaufwands und der Robustheit. So kann man erwägen, ob nicht auf vorhandene Leitungen wie Antennenkabel, Telefon-Hausanlage oder Stromversorgungsnetz zurückgegriffen werden kann. Die Verlegung neuer Leitungen wie z.B. Ethernet-Netzwerkkabel ist in vielen privaten Haushalten aufwändig und teuer. Ein drahtloses System wie die Funkübertragung wäre daher vorzuziehen, um mit minimalem Installationsaufwand ein Netzwerk aufzubauen.

Doch die wichtigste Voraussetzung, die ein Übertragungsmedium erfüllen muss, ist eine ausreichende Datenrate. Dies spielt vor allem für die Übermittlung breitbandiger Signale, in diesem Fall Videodaten, eine große Rolle. Die Tabelle 2.1 zeigt die Datenraten unterschiedlicher Medienströme. Die wesentlichen Anforderungen für die Bandbreite ergeben sich aus der Datenrate der Videoströme.

Anwendung	**typische Datenrate**
Steuerungsfunktion, Fernbedienung	100 bit/s
Sprachübertragung	64 kbit/s
Stereoton komprimiert (MP3)	192 kbit/s
Stereoton unkomprimiert	1,411 Mbit/s
Videodaten "VHS-Qualität"	1 Mbit/s
Videodaten DVB (MPEG-2)	3...9 Mbit/s
Videodaten DV	25 Mbit/s

Tabelle 2.1: Datenrate verschiedener Anwendungen

Man bedenke jedoch, dass dies nur die Datenraten *eines* Videostromes sind. In einem Server-System mit mehreren Clients wie dem DVP-System, muss aber zu bestimmten Zeiten mit einer Spitzenauslastung gerechnet werden, falls eine Anzahl von Nutzern mehrere Videoströme von dem Server anfordern. Die Datenrate ist demnach mit der maximalen Zahl der unabhängigen Nutzungen zu multiplizieren. Dies kann eine Größenordnung von 50 bis 60 Mbit/s erreichen.

In den folgenden Unterkapiteln werden nun verschiedene Übertragungsmöglichkeiten, die physikalische Schicht eines Netzwerkes (Host-to-Host), analysiert. Dies wird vor allem in Anbetracht einer ausreichenden Datenrate für das MPEG-2 Format geschehen, welches ausschlaggebend für das *Digitale Video Projekt* ist.

2.3.1 Stromversorgungsnetz als Übertragungsweg

Die Nutzung des in jedem Haus vorhandenen 230-V-Niederspannungsnetzes zur Übertragung digitaler Daten wäre eine naheliegende Lösung. In jedem Raum sind ausreichend viele Steckdosen vorhanden, um Signale in das Netz einzuspeisen oder Signale aus dem Netz zu empfangen, somit wäre eine ideale lokale Netzstruktur vorhanden. Die technische Entwicklung hat heute den Stand erreicht, dass diese Art der Datenverteilung über die sogenannte *Powerline* zuverlässig möglich ist.

Die dazu notwendigen PLC-Modems (*Powerline Communication*) stehen inzwischen zur Verfügung.
Möchte man die Stromleitungen nun als lokales Netzwerk verwenden, müssen erst die Vor- und Nachteile für eine Datenübertragung genauer betrachtet werden. Sie werden in folgender Tabelle zusammengefasst:

***Powerline Communication* (PLC)**	
Vorteile:	**Nachteile:**
· gute Flächendeckung im Haus	· Störfaktoren, elektromagnetische Wellen
· gleichzeitige Lieferung von Strom und Kommunikation	· fehlende internationale Standarisierung
· einfache Vernetzung von PC´s (im Versuchsstadium)	· nur für kurze Entfernung ohne Zwischenverstärker, 350 m
	· teure PLC-Modems nötig
	· geringe Datenrate

Tabelle 2.2: Vor- und Nachteile von PLC

Die größte Einschränkung der *Powerline*-Übertragung ist jedoch momentan die Datenrate. Nach neueren Untersuchungen sehen die Entwickler die absolute Datenrate im besten Fall bei etwa 2 Mbit/s und im schlechtesten Fall bei 400 kBit/s. Hinzu kommt, dass Datenraten von mehreren Mbit/s Frequenzbänder von einigen MHz erfordern. Mit höheren Frequenzen auf der ungeschirmten Stromleitung steigt auch das Emissions-Problem und die Gefahr der Störung von Funknetzen. Will man also breitbandige Signale in einem Stromnetz übermitteln, sind besondere Regelungen und neue Zulassungsgrundlagen zur Vermeidung von EMV-Problemen (Elektromagnetische Verträglichkeit) erforderlich [15].

Für das Übertragen digitaler Videodaten scheidet dieses Verfahren somit vorerst aus. Zwar ist ein Rückkanal bei der *Powerline Communication* realisierbar, doch dieses kann nicht die fehlende Datenrate des *Downstreams*[3] ersetzen.

[3]Datenstrom zum Empfänger

2.3.2 Telefonleitung als Übertragungsweg

Mit der Einführung der *Digital Subscriber Line*-Technik (DSL) 1999 wurde es möglich, digitale Signale auch auf herkömmlichen Kupferleitungen zu übertragen. Es entstanden verschiedene Nutzungsarten der Technik (HDSL, SDSL, ADSL, VDSL), mit Trägerfrequenzen, auf denen paket- oder zeilenorientierte Protokolle aufgesetzt werden, um digitale Signale zu übertragen.
Ursprünglich bezeichnet ein Verfahren der skalierbaren xDSL-Familie ein Modem, das für Basic Rate ISDN eingesetzt wird. Ein solches Modem überträgt Daten in beiden Richtungen simultan - es arbeitet also duplex. Der Multiplex- und Demultiplexvorgang des Datenstromes in zwei B-Kanäle und einem D-Kanal findet in angeschlossenen Terminals statt. xDSL verwendet Echokompensation, um Sende- und Empfangssignal an beiden Enden zu trennen. DSL-Modems können je nach Verfahren eine Bandbreite von 0 bis 52 MBit/s verwenden, schließen aber damit die gleichzeitige Verwendung der Leitung für den üblichen Telefondienst POTS (*Plain Old Telephone Service*) aus [16].

	HDSL	**SDSL**	**ADSL**	**VDSL**
max. Datenrate Empfang	1,544 Mbit/s	2,048 Mbit/s	bis 8 Mbit/s	bis 52 Mbit/s
max. Datenrate Senden	1,544 Mbit/s	2,048 Mbit/s	bis 1 Mbit/s	bis 2,3 Mbit/s
max. Leitungslänge	bis 4 km	bis 3 km	bis 5,5 km	Datenrate bei 0,3 km
benötigte Adernpaare	2	1	1	1
Frequenzbereich	0-240 kHz	0-378 kHz	0-1 MHz	0-30 MHz
POTS im Basisband	Nein	Nein	Ja	Ja
Anwendung	Ersatz für T1-Leitungen	WAN/LAN	Internet Intranet	WAN/LAN Multimedia
Verfügbarkeit in Deutschland	Verfügbar	Verfügbar	Verfügbar	Testphase

Tabelle 2.3: Verschiedene DSL-Techniken

In der Tabelle 2.3 wird ein kurzer Überblick über die unterschiedlichen DSL-Technologien gegeben. In Hinblick auf die Datenraten der einzelnen Technologien lässt sich nach Auswertung der Tabelle eine Schlussfolgerung für eine Videoverteilung im Sinne des DVP-Prinzips ziehen: Für das Übertragen der MPEG-2 Daten käme nur VDSL (*Very high bit rate Digial Subscriber Line*), eine Weiterentwicklung von ADSL (*Asymmetric Digital Subscriber Line*), in Frage.
VDSL bietet die nötige Datenrate und lässt sogar passive Netzabschlüsse zu, wodurch mehrere VDSL-Modems an eine Leitung angeschlossen werden können, was wiederum günstig für eine optimale Netzwerktopologie ist. Ein Rückkanal ist bei VDSL vorhanden, denn es gibt einen Up- und Downstream. Die Trennung des Hin- und Rückkanals übernimmt das *Frequency Division Multiplexing* (FDM).
Leider befindet sich die Entwicklung geeigneter VDSL-Modems noch in der Testphase, und es sind noch keine einheitlichen Systeme auf dem Markt vorhanden. Während der Untersuchungen zu dieser Diplomarbeit, wurde zwar von der Firma Siemens das Produkt *Attane Solution for Video over DSL* [17] vorgestellt, wobei dort aber nur von einer Datenrate bis zu 13 Mbit/s die Rede ist.

Unter Berücksichtigung dieser Umstände kommt eine Videoverteilung über ein Telefonnetzwerk im Haushalt für das *Digitale Video Projekt* nicht in Frage.

2.3.3 Kupfer-Koaxialkabel als Übertragungsweg

Neben der Stromleitung und dem Telefonkabel liegt in den meisten Haushalten eine weitere Leitung: Das Kupfer-Koaxialkabel für Kabel-TV. Internet-Anwendungen über das Fernsehkabelnetz werden bereits jetzt angeboten (über Set-Top-Boxen). Hierfür werden spezielle Kabelmodem benötigt. Bei der breitbandigen Datenübertragung über das Fernsehkabel gibt es jedoch grundlegende Probleme, die vor allem für das DVP-System ungünstig sind: Diese Art des Übertragungsweges ist noch nicht rückkanaltauglich. Das bedeutet, dass für Steuerungsinformationen andere Wege gefunden werden müssen, wie herkömmliche Telefon- oder Funkverbindungen.
Außerdem fehlt eine internationale Standardisierung. In Amerika sind bereits Kabelmodem nach dem DOCSIS-Standard (*Data Over Cable Service Interface Specification*), z.B das Elsa Kabelmodem mit einem *Downstream* von 43 Mbit/s, im *Upstream*[4] bis zu 10,24 Mbit/s, für eine bidirektionale Datenübertragung auf

[4]Rückkanal zum Sender

dem Markt, welche sich aber in Europa noch nicht durchsetzen konnten. Das *Digital Audio Visual Council* (DAVIC) und die DVB-Plattform in Europa arbeiten deshalb an einem europäischen Standard [18].

Solange sich noch kein einheitlicher Standard entwickelt hat, wird diese Art des Transportmediums für breitbandige Videodaten keinen großen Anklang finden. Im Moment sind jedoch der fehlende Rückkanal und die geringe Auswahl der Kabelmodem auf dem Markt dafür ausschlaggebend, dass sich das Kupfer-Koaxialkabel für eine Client/Server-Anwendung nicht eignet.

2.3.4 Funk als Übertragungsweg

Die Tabelle 2.4 beschreibt die bestehenden Funkverfahren, die für WLAN´s (*Wireless Local Area Network*) verwendet werden, in einer Übersicht. Die typischen Reichweiten variieren abhängig von den gewählten Leistungs- und Modulationsparameter der jeweiligen Verfahren.

Verfahren	**maximale Datenrate**	**typische Reichweite**	**Frequenzbereich**
DECT	1 Mbit/s	300 m	1880...1900 MHz
Bluetooth	1 Mbit/s	1...100 m	2400...2483,5 MHz
HomeRF	10 Mbit/s	10...100 m	2400...2483,5 MHz
IEEE 802.11b	2...16 Mbit/s	10...100 m	2400...2483,5 MHz
HiperLan/2	2...54 Mbit/s	10...30 m	5150...5725 MHz

Tabelle 2.4: Eigenschaften drahtloser Netzwerke

Wegen der hohen Datenraten sind insbesondere der vom *European Telecommunication Standards Institute* (ETSI) definierte HIPERLAN/2 (*HIgh PErformance Radio Local Area Network*) Standard und eine Weiterentwicklung des von IEEE definierten IEEE-802.11-Standards, der IEEE 802.11a Standard, für eine Übertragung von Videodaten relevant. Die restlichen Verfahren können für eine Videodatenverteilung im Sinne des *Digitalen Video Projektes* vernachlässigt werden.
Beide Verfahren bieten hohe Datenraten von bis zu 54 Mbit/s im 5 GHz Frequenzbereich. Es können dabei stationäre, portable oder sich bewegende Geräte drahtlos vernetzt werden. Die Bitübertragungsschicht und ein Teil der Sicherungsschicht (der MAC-Sublayer, *Media Access Control*) wurden dazu neu entwickelt. Beide Standards unterstützen spontane Vernetzung, sowie Kommunikation über

Access Points (AP) [19]. Doch leider sind momentan keine kommerziellen Produkte für diesen hohen Frequenzbereich auf dem Markt erhältlich. Nur der zu einem früheren Zeitpunkt entwickelte IEEE 802.11b Standard im 2,4 GHz ISM Band, bietet bereits eine kommerzielle Produktlösung an, z.B. *Telia Homerun*. Leider erreicht dieses Verfahren nur eine Datenrate von bis zu 16 Mbit/s.

Bei Funkübertragungen von digitalen Signalen besteht außerdem kein definierter Schutz vor Störungen. Neben der Wechselwirkung mit anderen Funkdiensten, deren Anzahl beständig wächst, hat insbesondere Störstrahlung einen erheblichen Einfluss. Aufwändige Verfahren der Modulation und Kanalcodierung müssen eine zuverlässige Datenübertragung sicherstellen. Die Übertragungseffizienz in den einzelnen Frequenzbändern ist laut Brakensiek [19] begrenzt: Erreichbare Störabstände werden durch die relativ geringe Sendeleistung, Dämpfungseinflüsse und durch Mehrwegausbreitung beschränkt. Der Einfluss von Hindernissen wie Wänden, Möbeln und Personen ist kritisch und nimmt mit der Betriebsfrequenz zu.

Durch die bis jetzt fehlende Fertigstellung der HiperLAN/2-Spezifikation kann somit eine Videoübertragung per Funk für das DVP außer acht gelassen werden.

2.3.5 Ethernet als Übertragungsweg

Die Ethernet-Technik zum Aufbau lokaler Netzwerke ist für unterschiedliche physikalische Medien (Koax-Kabel, Twisted Pair, faseroptisch) standardisiert [20]. Abhängig vom jeweils verwendeten Standard erlaubt es Übertragungsdatenraten von 10 Mbit/s bis 10 Gbit/s. Dabei zeichnet es sich dadurch aus, dass wegen der großen Verbreitung und der vorhandenen Erfahrung das Netzwerk robust und preiswert ist. Anfallende Kosten wären je nach gewähltem Standard und Aufbau der Netzwerkstruktur: der Installationsaufwand (nachträgliche Kabelverlegung im Haushalt) sowie die nötige Netzwerktechnik (Hub, Netzwerkkarten). Die hohe Datenrate machen das Ethernet natürlich für eine Übertragung von Echtzeit-Videodaten besonders interessant.

Jedoch besitzt das Ethernet durch die fehlende Bereitstellung von *Quality of Service* (siehe Abschnitt 2.2) einige Schwächen, an deren Beseitigung derzeit an vielen Stellen gearbeitet wird.

Auf die Funktionsweise der Datenübertragung sowie die verschiedenen Netztopologien von Ethernet-Netzwerken wurde bereits ausführlich in dem Abschnitt 1.3 eingegangen.

Für eine Client/Server-Anwendung nach dem DVP-Prinzip (Abbildung 2.1) ist somit Ethernet als Übertragungsmedium am idealsten geeignet. Es stellt als einzige, momentan verfügbare Möglichkeit die erforderlichen Datenraten von digitalem Video zur Verfügung und ermöglicht zugleich einen idealen Rückkanal zur Übermittlung bestimmter Steuerbefehle an den Server. Es reicht dabei schon die breitbandige, preiswert verfügbare 100MBit-Ethernet-Vernetzung aus. Die kostspielige Glasfaser-Vernetzung zur Übertragung mehrerer Gigabyte wäre für eine Anwendung im Privathaushalt überdimensioniert. Eine optimale Netzwerktopologie im "Vernetzten Haus" (Abbildung 2.2) wäre die Stern-Topologie.

2.4 Streaming von Videodaten

Nach der Analyse unterschiedlicher Übertragungswege stellt sich die Frage, wie man am besten eine größere Menge unterschiedlicher Videodaten an eine Anzahl von Clients (Empfänger der Videodaten) in einem Netzwerk übertragen kann. Für diesen Zweck eignet sich das Streaming von Videodaten, auf das nun näher eingegangen wird.

2.4.1 Warum Streaming?

Streaming ist eine Technik, die es ermöglichen soll, multimediale Inhalte in Echtzeit über ein Netzwerk von einem Server auszuspielen. Es existiert eine Softwarekomponente auf dem Server, der sogenannte *Videoserver*. Er liest die Videodaten von seiner Festplatte und überträgt sie über ein spezielles Netzwerkprotokoll zum *Videoplayer* eines Empfängers. Der multimediale Inhalt soll dabei nicht bereits komplett auf den abspielenden Rechner kopiert worden sein, sondern direkt während des Ladevorganges abgespielt werden. Schon abgespielte Teile des Video/Audio-Streams können nach ihrer "Konsumierung" wieder gelöscht werden.
Streaming kann im weitesten Sinne die Voraussetzung des *Video-on-Demand*-Prinzips erfüllen, in dem z.B. eine sich auf der Festplatte des *Videoservers* befindene Filmdatei zeitversetzt von mehreren Clients angefordert und zu den jeweiligen Clients gestreamt werden kann. Die einzige Beschränkung hierbei ist der Aufbau des Server-Rechners (Betrachtung in Kapitel 3).

Die Vorteile des Streamings können wie folgt zusammengefasst werden:

- Das Verfahren bietet kürzere Verzögerungen zwischen Anfragen und Abspielen einer ausgewählten Filmdatei.
- Es herrscht eine kontrollierte Bandbreite im Netz, wodurch mehrere Nutzer gleichzeitig möglich sind.
- Man hat eine garantierte Dienstegüte, welches ein "Ruckeln" des Videomaterials verhindert.
- Bietet einen besseren Schutz vor ungewollten Kopien.
- Durch Streaming entsteht ein geringer Speicherverbrauch im *Client*.

Natürlich gibt es auch Einschränkungen die bei der Streaming-Technik auftreten, welche folgende sind:

- Bestimmte Funktionalitäten eines *Videoplayers* sind nur sehr aufwändig zu realisieren, so z.B. Bildsuchlauf, Zeitlupe, sowie Vor- und Rücklauf.

- Es ist selten eine Kommunikation zwischen verschiedener Streamingsoftware möglich. Die Player/Server-Kommunikation ist vom Hersteller abhängig und verwendet häufig proprietäre Produkte und Protokolle.

2.4.2 Ein Anwendungsgebiet: *Video-on-Demand*

Relevant für diese Arbeit ist die Anwendung von Streaming für das *Video-on-Demand*-Prinzip. Bei dieser Anwendung hat ein Nutzer die Möglichkeit, einen laufenden Datenstrom anzufordern und den Inhalt zu konsumieren. Ermöglicht man dem Nutzer darüber hinaus, den Beginn der Übertragung der Videodatei zu wählen, so wird dies als *Video-on-Demand* (VoD) bezeichnet. Allen VoD-Systemen ist gemein, dass die hardwarebedingten Ressourcen des Videoservers (I/O-Kapazität und Netzwerkbandbreite) eine maximale Anzahl parallel ausspielbarer Datenströme ermöglichen und dass diese Datenströme nach unterschiedlichen Algorithmen aufgeteilt werden müssen. Dabei unterscheidet man zwischen Unicast, Multicast und Broadcast-Verbindungen von Rechnern in IP-Netzwerken, wobei eine Unicast-Verbindung für *Video-on-Demand* Zwecke am idealsten geeignet ist [21].

Existierende VoD-Systeme, welche die verfügbaren Ressourcen direkt auf die Nutzer aufteilen, werden dabei in der Literatur allgemein als nutzerbasiert (*User-Centered*) bezeichnet [22]. Darunter fallen im Netzwerk z.B. alle Verfahren, die einen festgelegten Datenstrom pro Nutzer (Unicast) bereitstellen. Hingegen werden bei den datenbasierten Verfahren (*Data-Centered*) die verfügbaren Ressourcen auf ein Media-Objekt verteilt. Dabei werden bei den serverseitig initiierten Karussellverfahren (*Server-Push*) die verfügbaren Ressourcen in K logische Kanäle aufgeteilt und im einfachsten Fall zeitversetzt übertragen [23]. Alle komplexen Karussellverfahren, die bei gleicher Datenrate am Server gegenüber dem zeitversetzten Abspielen die Wartezeit (Zugriffszeit) zum Beginn einer Videodatei deutlich reduzieren, setzen einen empfängerseitigen Datenspeicher voraus, welches auch als *Prefetching* bezeichnet wird.

Neben den unidirektionalen *Server-Push*-Systemen können datenbasierte Verfahren auch von den Empfängern initiiert sein (*Client-Pull*). Dabei werden im Unterschied zu den *Push*-Systemen nur dann Datenströme übertragen, wenn diese von einem Empfänger auch konsumiert werden. In der Betrachtung eines *Batching* VoD-Sytems von Dan und Sitaram [23] werden Anfragen von Empfängern für eine bestimmte Videodatei zu einer Gruppe zusammengefasst und dann per Multicast übertragen.

2.4.3 Anforderung an ein Netzwerk für Streaming

Bei Streaming ist eine gute Qualität des multimedialen Inhalts, wenige Aussetzer beim Empfänger und vor allem eine geringe Verzögerung zwischen Senden und Abspielen erwünscht. Die Zeitspanne, die zur Übertragung von Videodaten über das Netzwerk benötigt wird, kann, sollte aber nicht sehr groß sein. Datenströme werden innerhalb des Netzwerkes in einzelne Pakete unterteilt. Es besteht die Möglichkeit, dass Pakete in falscher Reihenfolge beim Empfänger ankommen (Jumbling), in diesem Fall ist bei einer Echtzeitübertragung eine Reorganisation der empfangenen Pakete ohne großen Zeitaufwand von Nöten.

Das Netzwerk sollte konstant genug Bandbreite für den zu übertragenden Datenstrom garantieren, da dieser in Echtzeit übertragen werden soll. Die Verzögerung zwischen dem Aussenden und dem Abspielen eines *Streams* sollte minimal sein. Diese Verzögerung ist abhängig von der Latenz, die sich aus Ausbreitungsverzögerung (Entfernung/Lichtgeschwindigkeit), der Übertragungsverzögerung (Paketgröße/Bandbreite) und der Wartezeit aufgrund der Zwischenspeicherung in eventuellen Zwischenstationen, ergibt.
Die Übertragungsverzögerung kann durch eine Minimierung der Datenstromgröße oder eine Maximierung der Bandbreite optimiert werden. Die Maximierung der Bandbreite ist zu bevorzugen, da Minimierung der Datenstromgröße meist auf Komprimierung beruht, welche bei multimedialen Inhalten in den meisten Fällen Qualitätseinbußen zur Folge hat. Eine Wartezeit innerhalb der Switches z.B. kann nur durch Reservierung bestimmter Ressourcen optimiert werden.
Um *Jitter*[5] präventiv entgegenzuwirken, sollten Puffer für Daten angelegt werden, dies ist vor allem bei *Video-on-Demand*-Anwendungen von Nutzen. Die Abschätzung des *Jitters* sollte möglich sein, um die Puffergröße anzupassen [24].

[5] Schwankungen

2.4.4 Verschiedene Streamingprotokolle

Protokolle sind gewissermaßen Regeln, die die Kommunikation zwischen einem Server und Client steuern (siehe Abschnitt 1.3.2). Dies gilt auch für das Streaming. Es gibt verschiedene Protokollschichten, die aufeinander aufbauen. Aus Platzgründen werden hier die beiden für diese Diplomarbeit wichtigen Protokolle UDP und TCP genauer betrachtet.

2.4.4.1 UDP und TCP

Das *User Datagram Protocol* (UDP) ist ein verbindungsloses Transportprotokoll, das keinerlei Unterstützung von Fehlerkorrektur oder Identifikation von verlorengegangenen Paketen unterstützt. Es sendet die Pakete auf dem schnellsten Weg in Richtung des Empfängers, ohne zu garantieren, dass diese dort in der richtigen Reihenfolge ankommen. Bei Nutzung des UDP Protokolls werden verlorene Pakete ignoriert. So kann es vorkommen, dass einzelne Bruchstücke nicht den Empfänger erreichen, aber eine kontinuierliche Wiedergabe trotz Paketverlust möglich ist.

Das *Transmission Control Protocol* (TCP) ist ein sicheres, verbindungsorientiertes Protokoll, welches garantiert, dass alle gesendeten Pakete beim Empfänger in richtiger Reihenfolge ankommen. Diese Garantie resultiert aus der Eigenschaft, dass TCP die Paketübertragung bei Verlust wiederholt bis der Client den Empfang des Paketes quittiert hat. TCP ermöglicht kein Verwerfen einzelner Pakete und bestätigt auch nicht ihr Ankommen.
Diese Eigenschaft von TCP sind für viele Anwendungen äußerst wichtig, für Streaming-Anwendungen jedoch eher schädlich, da Übertragungsengpässe bei der Nutzung von TCP zur Unterbrechung der kontinuierlichen Wiedergabe des Echtzeit-Inhalts führen kann.

Was bedeutet dies für die Übertragung von Echtzeit-Videodaten?

- **TCP** bietet also keine guten Voraussetzungen für den Transport zeitabhängiger Daten. Dieses Protokoll ist zur Übertragung von Datenströmen effizient, solange die Daten nicht an eine Zeitgrenze gekoppelt sind.

- **UDP** ist dagegen für den Transport von Audio- und Videodaten eher geeignet. Es arbeitet verbindungslos. Durch den kurzen Paket-Kopf entsteht ein gutes Nutzdaten/Paketlängenverhältnis [11].

Zusammenfassend bietet die folgende Tabelle einen konkreten Vergleich der beiden Transportprotokolle:

	UDP	**TCP**
Vorteile:	· Geringer Bandbreitenverlust · Keine Wartezeit durch Datenverlust	· Kein Datenverlust
Nachteile:	· Möglicher Datenverlust · Keine Staukontrolle	· Hohe Verzögerung · Hoher Datenüberschuß
Fazit:	Geeignet für die Echtzeit-Videoübertragung	Nicht geeignet

Tabelle 2.5: UDP kontra TCP

2.4.4.2 Die *Realtime*-Protokolle

Für Videodaten, welche in Echtzeit übertragen werden sollen, definierten die *Internet Engineering Task Force* (IETF), die Arbeitsgruppe *Audio and Video Transport* (AVT) und das Unternehmen *RealMedia* einige neue Protokolle [25]:

- **RTP** (*Realtime Transport Protocol*):
  RTP ist ein "nicht-zuverlässiges", verbindungsloses Protokoll und ist für die eigentliche Video- und Audiodatenübertragung gedacht. Da das RTP ein Protokoll der Anwendungsschicht (1.3.1) ist, kann es nicht direkt auf dem *Internet Protocol* (IP) basieren, sondern ist auf ein Protokoll der Transportschicht UDP angewiesen. Es kann die ankommenden Daten typisieren (entscheiden, ob sie zum Video- oder Audiostream gehören), und nummeriert die Pakete beim Versand. So kann der Decoder diese in die richtige Reihenfolge bringen und muss sie nicht in der eintreffenden Reihenfolge verarbeiten. RTP bietet keine Möglichkeiten der Ressourcenreservierung oder der Kontrolle des *Quality of Service* (2.2).

- **RTCP** (*Realtime Transport Control Protocol*):
  Dieses Protokoll kann man als Ergänzung zum RTP betrachten, weil damit die Medien- und Steuerungsinformationen zwischen Client und Server übertragen werden. Durch das RTCP Protokoll wird eine Bestimmung des *Quality of Service* realisiert.

- **RSVP** (*Resource Reservation Protocol*):
  Dieses Protokoll ermöglicht die Reservierung des Datenweges zwischen dem Client und dem Server, wodurch alle Daten vom Server zum Client denselben Weg nehmen. Außerdem kann durch das RSVP eine Mindestbandbreite für den Übertragungskanal angegeben werden, welche freigehalten werden soll. Dies muss von Zeit zu Zeit wiederholt werden, damit nicht mehr benötigte Reservierungen die Bandbreite nicht zu lange unnötig einschränken.

- **RTSP** (*Realtime Streaming Protocol*):
  RTSP vereinigt RTCP und RTP. Der Zugriff findet meist durch kleine Beschreibungsdateien statt, welche per HTTP übertragen werden. Meist wird durch RSVP die Verbindung reserviert.

2.5 Präzisierte Aufgabenstellung

Nach der Betrachtung der Grundlagen in Kapitel 1 sowie der Analyse verschiedener Übertragungsmedien und der Bestimmung eines idealen Übertragungsverfahrens für Videodaten, die sich für ein lokales Netzwerk im Haushalt eignen, lässt sich mit den sich daraus ergebenen Rahmenbedingungen und Eckpunkten in den folgenden Kapiteln die präzisierte Aufgabenstellung dieser Diplomarbeit bearbeiten.

Laut konkretisierter Aufgabenstellung soll eine Konzeption zur Videodatenverteilung vorgelegt werden. Das zu erarbeitende Konzept soll einen grundlegenden Bestandteil des *Digitalen Video Projektes* bilden und auf dessen Systemkomponenten aufbauen. Durch die Integration der vorgelegten Konzeption in das *Digitale Video Projekt* soll eine Videodatenverteilung von dem Hauptsystem, einem PC mit DVB-Karte und der Softwarekomponente VDR, an einen Client-PC realisiert werden. Für die Konzeption wird von einer Netzwerkarchitektur eines lokalen Netzwerkes in einem Einfamilienhaushalt (siehe Abbildung 2.2) ausgegangen. Es wird davon ausgegenangen, dass in der Konzeption einer Videodatenverteilung mindestens 2 und maximal 5 Clients auf den *VideoDiskRecoder*-PC zugreifen werden und die Videodaten dementsprechend verteilt werden müssen.
Anhand des Analyseergebnisses aus Abschnitt 2.3 kann eine Entscheidung hinsichtlich des Übertragungsweges für die Konzeption und die spätere Umsetzung getroffen werden:
Von einem PC des *Digitalen Video Projektes*, der die Funktionen eines Videoservers übernehmen soll, sollen auf Befehl eines Nutzers Videodaten über ein 100 Mbit Ethernet-Koaxkabel an den Client-PC gesendet werden. Es soll die Kommunikation zwischen den Systemen, besonders eine Steuerung des Server-Systems, für mögliche Anwendungsfälle betrachtet werden. Weiterhin soll in dem Verteilungskonzept ein geeignetes Protokoll für die Übertragung von MPEG-2 Daten vorgelegt werden.

Kapitel 3

Konzept zur Videodatenverteilung

Im folgenden Kapitel werden die wichtigsten Anforderungen der Videodatenverteilung im *Digitalen Video Projekt* ausgearbeitet und ein geeigneter Lösungsansatz aufgezeigt.

3.1 Vorüberlegungen

Es wird ein Konzept für den gesamten Vorgang einer Videodatenverteilung im *Digitalen Video Projekt* erstellt. Der Ablauf kann wie folgt grob umschrieben werden: Es wird zu jedem Client der, über ein lokales Ethernet-Netzwerk mit einem Server-PC verbunden ist, eine festgelegte Verbindung (*Unicast*-Verbindung) zur Übertragung von Datenpaketen aufgebaut.
Im vorherigen Kapitel wurde bereits von einem Client/Server-Aufbau gesprochen. Wünschenswert ist es, eine Videodatenverteilung in einer Umgebung wie in Abbildung 2.2 dargestellt, zu erzielen. In Betracht gezogen wird dazu ein Client/Server-Ansatz für das DVP in Betracht gezogen, der im Verlauf dieses Kapitels erläutert wird. Die gesamte Steuerung der Videodatenverteilung soll dabei aus der Projektsoftware VDR ermöglicht werden. Dies führt zu einer Weiterentwicklung der Software.
Für eine Videodatenverteilung im Rahmen des *Digitalen Video Projekt* ergeben sich demnach folgende Fragen:

- Wie soll die Rechnerarchitektur erweitert werden, um eine Videodatenverteilung zu ermöglichen?

- Welche Anforderungen werden an einen Server-PC gestellt?
- Welche Anforderungen werden an einen Client-PC gestellt?
- Wie wird eine Kommunikation zwischen dem Client und dem Server realisiert? Wie kann eine Steuerung des Servers aussehen?
- Welche Softwarekomponenten werden benötigt, um MPEG-2 Daten über ein Netzwerk zu versenden/empfangen?
- Welche Protokolle sollen die Übertragung der Videodaten regeln?
- Wie sieht ein typischer Ablauf der Videodatenanfrage eines Clients aus?

3.2 Betrachtung der Rechnerarchitektur

Bestimmte Hardware- und Softwarekomponenten sind im DVP-Rechner unumgänglich, damit er seine Funktion als digitaler Videorecorder erfüllen kann. Dazu zählen die zwei DVB-Karten, der Linux DVB-Treiber, die Softwarekomponente *VideoDiskRecorder* VDR (siehe Abschnitt 4.2.1) und zur Steuerung des VDRs eine Infrarotschnittstelle mit zugehöriger Software LiRC (*Linux Infrared Remote Control*). Diese Komponenten wurden durch Analyse existierender Komponenten als am besten für das DV-Projekt geeignet herausgefiltert. Sie können als Grundaufbau des DVP-Server-PC betrachtet werden. Dazu gehören auch die Satellitenschüssel und ein Fernseher.

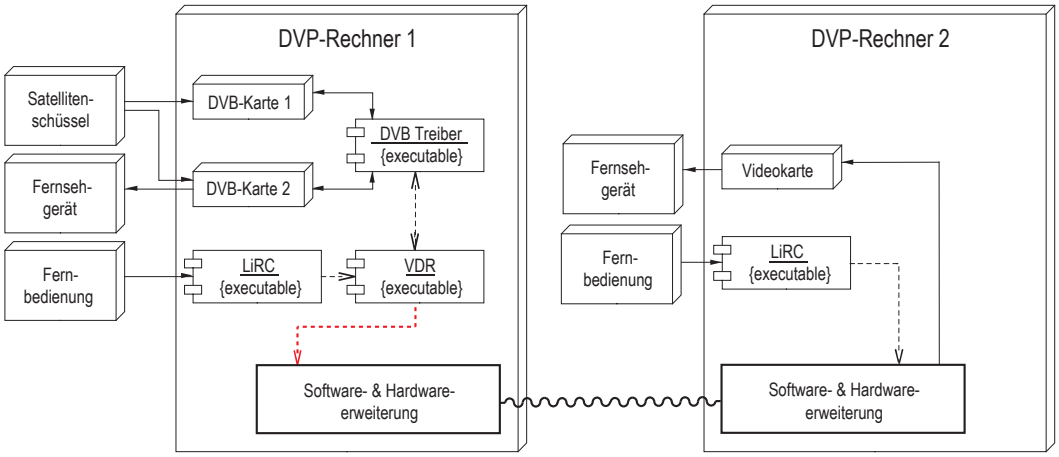

Abbildung 3.1: Der Grundaufbau im DV-Projekt

Das Komponentendiagramm 3.1 zeigt den vorhandenen Aufbau der Projektrechner. Zu sehen ist bereits eine Erweiterung der Rechnerarchitektur. Wird diese Erweiterung als eine Einheit betrachtet, die sich aus bestimmten Aufgabenbereichen zusammensetzt, muss erörtert werden, welche Funktionen oder Aufgaben einzelne Teilbereiche übernehmen oder zusichern sollen.
Bei der Recherche und Analyse einer Videodatenübertragung (siehe Kapitel 2) kam man zu dem Ergebnis, dass bei einer Videodatenübertragung eine *Quality of Service*, eine Datenverarbeitung und der eigentliche Übertragungsprozess zugesichert werden müssen. Die DVP-Rechner müssen somit, um eine Videodatenverteilung im *Digitalen Video Projekt* zu realisieren, durch geeignete Software- und Hardwarekomponenten erweitert werden (siehe Abschnitt 3.4).

Neben dem Aufbau der einzelnen Rechner ist die Bestimmung einer geeigneten Netzwerkstruktur interessant. Verschiedene Topologien wurden im Abschnitt 1.3.3 erläutert. Für die Verteilung im Sinne dieser Arbeit sichert alleine die Stern-Topologie, bei einem realistischen Ansatz mit bis zu 5 Clients, eine volle Datenrate zu. Bei einer Datenverteilung im Privat-Haushalt mit einer durchschnittlichen Familiengröße von 4 Personen, ist der Einsatz von 5 Clients (Fernsehapparaten) ausreichend und bietet somit eine optimale Lösung. Es wird auch keine teuere Netzwerktechnik benötigt.

	Hub	***Switch***
Vorteile:	· Preis	· besitzt 'Intelligenz' · Datenpakete werden nur an die angegebene Zieladresse verteilt
Nachteile:	· keine 'Intelligenz' · Datenpakete werden an alle Clients verteilt	· Preis (nur geringfügig teurer)
Fazit:	ungeeignet für eine Videodatenverteilung	geeignet für eine optimale Videodatenverteilung

Tabelle 3.1: *Hub* kontra *Switch* für eine Stern-Topologie

Für die Auslegung des lokalen Netzwerkes bei maximal 5 Clients ist demnach der Einsatz eines *Switches* am idealsten. Der konkrete Aufbau einer Stern-Topologie kann der Abbildung 2.2 entnommen werden.

3.3 Der Client/Server-Ansatz

Zunächst ist es wichtig, auf die Bedeutung von Client/Server-Systemen einzugehen. Es stellt sich die Frage, was die Anforderungen der einzelnen Systeme sind, wie sich deren Aufbau gestalten und wie die Kommunikation zwischen dem Client und dem Server geregelt werden soll.

Vereinfacht ausgedrückt sind Client/Server-Systeme zwei Prozesse. Der eine bietet eine Dienstleistung an (Server), der andere nimmt eine Dienstleistung in Anspruch (Client). Das System sollte Transparenz besitzen, d.h. jeder Teil sollte sich nur auf seine eigentlichen Aufgaben konzentrieren. Ein weiterer wichtiger Aspekt für Client/Server-Systeme ist die Verteilung der Daten. Die Zentralisierung der Videodaten bietet den Vorteil einer besseren Übersicht der vorhandenen Daten und minimiert somit überflüssige doppelte Dateien auf den verteilten Systemen. Jedoch bringt die Zentralisierung auch Probleme mit sich, da es laut Literatur nur einen *Single Point of Failure* gibt [10]. Sollte es zu einem Problem mit dem Server kommen, sind alle Clients betroffen. Ein Ausfall eines Clients hingegen beeinträchtigt das System in keiner Weise.

Weiterhin ist es wichtig, dass sich ein Client/Server-System einfach erweitern lassen soll. Diese Skalierbarkeit sollte sowohl für die Clients, als auch für den Videoserver im DVP gegeben sein. Um Skalierbarkeit zu gewährleisten, soll aber auch eine Interoperabilität möglich sein. Bei der Anwendung im DVP soll es nicht zu Problemen führen, wenn unterschiedliche Hard- und Softwarekomponenten verwendet werden. Dies ist vor allem erwünscht, falls sogar eine Integration zweier Softwarekomponenten durchgeführt werden muss. Es sollten aber auch bei der Zusammenarbeit zwischen Betriebssystemen wie Microsoft oder Linux keine Probleme auftreten. Der Server-Rechner im DVP verwendet das Linux Betriebsystem, wobei den Client-PCs die Wahl eines Betriebsystems ermöglicht werden soll. Die Interoperabilität stellt insbesondere an die Softwarekomponenten des Client-PCs erhöhte Ansprüche.

3.3.1 Anforderung an den Server

Im DVP-System soll der Server-PC die Aufgaben eines DVB-Receivers, eines digitalen Videorecorders zur Speicherung und Bereitstellung der MPEG-2 Videodaten und durch seinen Anschluss an ein lokales Netzwerk die Aufgabe der Verteilung von Videodaten übernehmen. Im Vergleich zum Client werden im Rahmen des *Digitalen Video Projektes* demnach unterschiedlich hohe Anforderungen an den Server gestellt. Anhand des Aufbaus der beiden Projektrechner wird dies bereits an dem Komponentendiagramm 3.1 verdeutlicht.
Das System soll im DVP wichtige Anforderungen bewältigen können.

1. Die Dekodierung der Satellitensignale. Je nach DVB-Karte geschieht dies durch eine Software oder wird direkt von der Hardware übernommen.

2. Die Ressourcen eines Videorecorders, d.h. es muss ausreichender Speicherplatz für die Speicherung von Videodaten bereitgestellt werden.

3. Eine automatisierte Verwaltung der einzelnen Clients. Es sollte eine einfache Erweiterung des System, z.B. das Hinzufügen eines Clients, möglich sein.

4. Das sogenannte *Request Handling*, die Steuerung des Servers von den Clients aus. Eine Anzahl von Steuerbefehlen werden an den Server gesendet, die er bearbeiten muss (siehe Abschnitt 3.3.3).

5. Eine *Quality of Service* zur Verfügung stellen (siehe Abschnitt 2.2). Je mehr Streams an die Clients verteilt werden, um so höher werden die Anforderungen an die Festplatten, den Rechner-Bus und vor allem an die Netzwerkkarte. Der Server muss die geforderte Bandbreite zu jedem Client garantieren.

Möchte man den Ansprüchen der unterschiedlichen Anforderungen genügen, muss die Hardware des Server-PCs für die geforderten Belastungen ausgelegt werden. Dies gilt besonders für die Netzwerkkarte. Sie muss bei einem maximalen Zugriff von 5 Clients eine Datenrate garantieren, so dass jeder Client einen kontinuierlichen Stream ohne Paketverluste oder Fehler erhält. Tritt der Fall auf, dass zu einem Zeitpunkt alle 5 Clients einen 5 Mbit/s paketierten Videostream vom Server anfordern, muss die Netzwerkkarte folgenden Datendurchsatz garantieren:

$$5Mbit/s * 5Clients \Rightarrow 25Mbit/s \tag{3.1}$$

Es ist deshalb ratsam, eine Stream-Begrenzung von einem Stream pro Client einzuführen, damit jedem Client ein fehlerfreier Stream zugesichert werden kann. Insofern sollte auch die benötigte Netzwerktechnik (Kabel, *Switch*) des lokalen Netzwerkes für diese Belastung ausgelegt sein.
Neben der Netzwerkkarte stellen jedoch die Festplatten und die Busgeschwindigkeit des Rechners eine viel größere Einschränkung dar. Möchte jeder Client eine auf der Festplatte gespeicherte Filmdatei übertragen bekommen, wird auch von der Festplatte und dem Rechner-Bus ein Datendurchsatz von 25 MB/s verlangt. Für diesen Zweck sind Festplatten mit hohen Umdrehungen (7200 U/min) und den gewünschten Zugriffszeiten von unter 9 ms nötig. Sie sollten außerdem eine Pufferspeicherung von ca. 4-8 MB besitzen, um einen kontinuierlichen Datenfluss sicherstellen zu können.
Diese Probleme könnten auch mit speziellen Festplatten-Systemen oder schnellen *SCSI*-Festplatten, wie z.B. einer *Ultra-2-SCSI* Festplatte mit einer Geschwindigkeit von 40 MB/s, behoben werden. Ein Beispiel für ein Festplatten-System wäre das *RAID*-System [26], das häufig in kommerziellen, professionellen Videoservern der Fernsehsender verwendet wird.

Möchte man einen Server ausreichend dimensionieren, müssen demnach mehrere Faktoren in Betracht gezogen werden. Eine feste Einschränkungsgröße bildet dabei der Rechner-Bus B des Videoservers, da dieser nicht bei der Dimensionierung verändert werden kann. Die variablen Größen, welche beim Aufbau eines Servers dementsprechend ausgelegt werden können, setzen sich aus der Prozessorleistung P, der RAM-Taktfrequenz R und der Zugriffszeit einer Festplatte Z zusammen.

$$100\% \cdot Datenrate = (10\% \cdot P + 10\% \cdot R + 40\% \cdot B + 40\% \cdot Z) \cdot Rechenlast \quad (3.2)$$

Geht man von einer erwartenden Datenrate von maximal 25 Mbit/s aus, so können sich, laut der allgemein verfügbaren Quelle [27], die prozentualen Anforderungen an einzelnen Komponenten des Server-Rechners wie in Gleichung 3.2 zusammensetzen.

3.3.2 Anforderung an den Client

Der geplante Aufbau eines Clients im *Digitalen Video Projekt* sollte einer Set-Top-Box ähneln. Er muss für den Nutzer einfach zu installieren und zu bedienen sein. Eindeutige Aufgabenbereiche des Client-PCs sollen in dem Fall vorgegeben und definiert sein. Der Anwender soll den PC nur für das Abspielen der Videodaten nutzen; er sollte sich nicht mit einer umfangreichen Konfiguration und Wartung des Empfängergerätes befassen müssen. In Zukunft soll z.B. mit Hilfe des *Dynamic Host Configuration Protocol* (DHCP) der Server eine automatische Erkennung eines Client-PCs in dem Netzwerk durchführen. Somit soll eine aufwendige Administrations des DVP-Systems vermieden werden.

Die Zusammensetzung und der Aufbau der Clients sollen im Vergleich zum Server keine hohen Produktionskosten verursachen. Die Aufgaben der Komponenten bestehen hauptsächlich darin, einen paketierten MPEG-2 Transportstrom zu empfangen, zu verarbeiten und an den Fernseher weiter zu leiten. Die Anforderung an die Netzwerkkarte ist hier ein kontrollierter Prozess. Jeder Nutzer eines Clients soll auf einen Stream begrenzt werden, so dass eine Netzwerkkarte maximal 4-9 Mbit/s verarbeiten muss. Diese Begrenzung ist bei einem 5 Client-System notwendig, da es sonst zu erhöhten Belastungen des Videoservers führt, und dieser den Clients keine garantierte Datenrate mehr zusichern kann.

Die Prozessorleistung soll im Client-PC für die Aufgaben der Software reserviert werden. Falls eine MPEG-Dekodierung nötig ist, sollte diese von der Hardware übernommen werden. Bei der Darstellung eines Streams (max 9 Mbit/s, DVD) wird, laut einer allgemein verfügbaren Quelle [27], eine Prozessorleistung eines Pentium III mit 450 MHz vorausgesetzt. Auf dem Client-PC können unterschiedliche Softwarekomponenten installiert werden. So z.B. eine Softwarekomponente, die das Zusammensetzen der einzelnen, empfangenen Datenpakete der Netzwerkkarte garantieren muss, und das Abspielen des Gesamtdatenstroms (mit einem *Player*). Das Komponentendiagramm 3.1 zeigt weitere optionale Komponenten auf. Um den Client-PC anwenderfreundlich zu gestalten, soll eine Software zur Steuerung einer Infrarotschnittstelle installiert werden, um den Client-PC per Fernbedienung bedienen zu können.

3.3.3 Kommunikation zwischen Client und Server

Der Datenaustausch vom Server-PC in Richtung Client dient der Übertragung der paketierten MPEG-2 Daten. Der Ablauf dieses Prozesses wird im Abschnitt 3.5 genau erklärt.
Für einen Client/Server-Ansatz jedoch ist es von großer Bedeutung, auch eine Verbindung vom Client zum Server hin zu ermöglichen, die auch über den Datenübertragungskanal realisiert wird. Diese Verbindung wird als Rückkanal bezeichnet, über den eine interaktive Kommunikation zwischen den beiden Systeme realisiert werden kann. Ein Client ist so in der Lage, durch das Senden bestimmter Daten oder Befehlsstrukturen, Operationen auf dem Server aufzurufen. Abbildung 3.2 stellt einen simplen Kommunikationsablauf zwischen Client und Server dar.

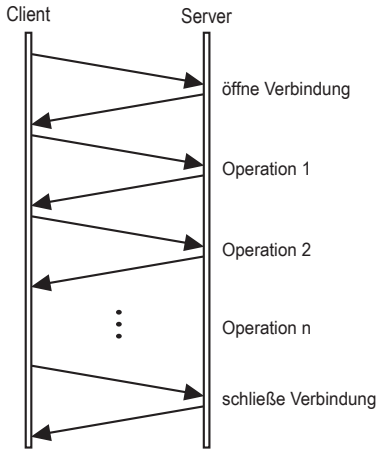

Abbildung 3.2: Kommunikation zwischen Client und Server

Der Rückkanal wird in der Konzeption einer Videodatenverteilung für die Übermittlung der Steuerbefehle an den Server verwendet. Diese Befehle werden benötigt, um eine Verteilung der Videodaten im DVP zu regeln und um den Abspielvorgang der Streams auf dem Client zu steuern. Einem Anwender sollte eine Auswahl an vorgegebenen Befehlen zur Verfügung stehen, mit denen er sich z.B. einen gewünschten Stream vom Server abrufen kann.
Es sollen dabei drei Gruppen von Steuerbefehlen bestehen. Eine Befehlsgruppe soll den Abspielvorgang der Streams bestimmen, die nächste Befehlsgruppe soll für das Auslesen bestimmter Verzeichnisse des Servers verwendet werden, und die letzte Befehlsgruppe wird für eine zusätzliche Serverkonfiguration vom Client aus benötigt.

Die folgende Tabelle weist Beispiele rudimentärer Befehle der drei Befehlsgruppen auf, die ein System bereitstellen sollte:

Stream-Steuer Gruppe	**Funktion**
`start stream`	Baut Verbindung auf, startet einen Stream
`stop stream`	Unterbricht den Stream samt Verbindung
`pause stream`	Hält Stream an, Verbindung bleibt bestehen
`resume stream`	Fährt mit der Übertragung fort
Stream-Auswahl Gruppe	
`select stream input`	Auswahl der Streaming-Quelle (Datei, DVB)
`list available files`	Dateiverzeichnis anzeigen lassen
`list available channels`	Sender eines DVB-Pakets anzeigen lassen
Server-Setup Gruppe	
`add client`	Hinzufügen eines neuen Clients
`delete client`	Entfernen eines bestehenden Clients
`setup client`	Einstellung des Clients (IP-Adresse, Port)

Tabelle 3.2: Klassifizierung der Steuerbefehle

Natürlich stellt die Tabelle 3.2 nur ein Grundgerüst der verschiedenen Befehle dar. So müssen im konkreten Fall einer Videodatensteuerung vom Nutzer bestimmte Variablen, Datei- oder Sendernamen den jeweiligen Befehlen hinzugefügt werden. Ein Beispiel eines vollständigen Befehls könnte folgendermaßen aussehen:

`start` [Filmdatei/Sender] [Quelle] [Ziel]

`start RTL DVB Client1`

Es sollten auch bestimmte Befehlsgruppen zu einem Befehl zusammengefasst werden können. Anhand des Beispiels wird dies verdeutlicht. Mit dem Befehl `start` wird eine Verbindung zum Server aufgebaut, ein Sender namens `RTL` von der Stream-Quelle `DVB` ausgewählt, und an die Zieladresse `Client 1` übertragen.
So kann durch eine Kombination der drei Befehlsgruppen eine schnelle, vereinfachte Steuerung der Videodaten realisiert werden.

3.4 Erweiterung der DVP-Rechnerarchitektur

Der Client/Server-Ansatz wird auf den Grundaufbau des DV-Projektes übertragen. Dadurch entsteht ein Server-PC, der seine bestimmten Aufgaben erfüllen muss (siehe Abschnitt 3.3.1), und ein Client-PC, der wiederum seine speziellen Aufgaben bewältigen soll (siehe Abschnitt 3.3.2).
Um nun eine Übertragung der auf dem Server gespeicherten Videodaten, den Inhalt einer DVD oder der einzelnen MPEG-2 Transportströme einer DVB-Karte an mehrere Clients zu ermöglichen, bietet sich durch Analyse existierender Verfahren das *Streamen* von Videodaten an (siehe Abschnitt 2.4). Für diesen Zweck soll eine geeignete Softwarekomponente, in der Abbildung 3.3 als VideoStreaming bezeichnet, für das server-seitige Senden und das client-seitige Empfangen der Videodaten in die Rechnerstruktur integriert werden. Besonders wichtig ist dabei die Frage nach dem geeigneten Protokoll, um die 188 Byte langen MPEG-2 Datenblöcke über das lokale Netzwerk zu verteilen. Dieses Problem wird in Abschnitt 3.5 genauer betrachtet.

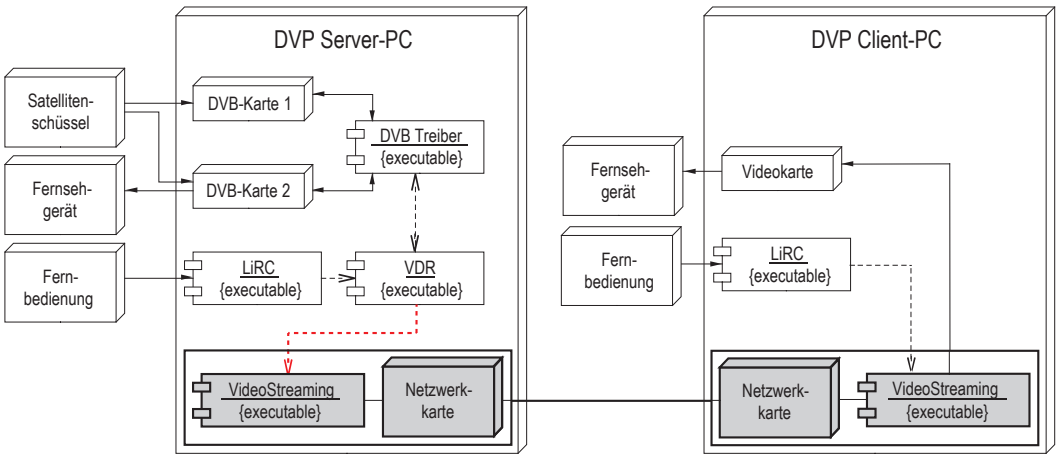

Abbildung 3.3: Vorgeschlagene Erweiterung der Rechnerarchitektur

Um schließlich eine Verteilung der Datenpakete auf physikalischer Ebene zu realisieren, soll als Hardwarekomponente eine Ethernet-Netzwerkkarte in die Rechnerarchitektur integriert werden.
Der Aufbau der Projektrechner soll demnach durch die grau markierten Komponenten, eine Streaming-Komponente und eine Netzwerkkarte, erweitert werden, damit eine Verteilung der Videodaten im DVP ermöglicht wird.

Die Basis des *Digitalen Video Projektes* bildet die VDR-Softwarekomponente. Im Mittelpunkt steht dabei ihre Weiterentwicklung. Für die Konzeption wird geplant, die Steuerung der Videodatenverteilung im gesamten DVP-System aus dem Bedienungsumfeld des digitalen Videorecorders zu ermöglichen. Deswegen soll auf dem DVP Server-PC ein Zusammenhang zwischen der VDR-Software und der VideoStreaming-Komponente geschaffen werden, um diese Verteilung der Videodaten aus dem Umfeld des VDRs zu kontrollieren (roter Pfeil in Abbildung 3.3). Es soll jede Softwarekomponente ihre Grundfunktionen beibehalten, die durch eine Interaktion oder Integration der beiden Softwarekomponenten nicht beeinträchtigt werden darf. Die nötige Verarbeitung der MPEG-2 Daten zur Übertragung über das lokale Netzwerk soll auf dem Server-PC und dem Client-PC jeweils durch die VideoStreaming-Komponente übernommen werden.

3.5 Der MPEG-2 Transportstrom über IP

3.5.1 Verarbeitung im Server-PC des DVP

Viele Multimedia-Streaming-Anwendungen benutzen zum Transport der Echtzeit-Videodaten über IP das UDP Protokoll. Jedoch besitzt dieses Protokoll ein paar Einschränkungen, siehe Abschnitt 2.4.4.
Zur Verbesserung des Verhaltens von UDP bei der Videodatenübertragung soll in dieser Arbeit eine Erweiterung in Betracht gezogen werden. Dies kann z.B. durch das Hinzufügen eines speziellen Zeitstempels geschehen, um die Übertragungseigenschaften des UDP Protokolls zu verbessern.
Im DVP soll ein Protokoll der höheren Schicht, das auf UDP aufbaut verwendet werden. Es soll eigene Mechanismen zur Verfügung stellen, die für die Paketierung der Daten und deren späteren Zusammensetzung im Client-PC sorgen. Unter der Nutzung des *Realtime Transport Protocol* (RTP) soll dies möglich gemacht werden.

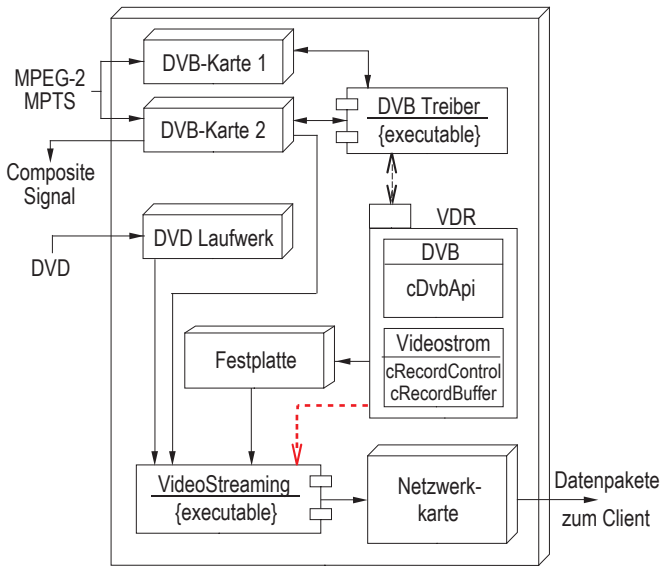

Abbildung 3.4: Ablauf im Server-PC

Die Abbildung 3.4 stellt die Zusammenhänge der einzelnen Komponenten sowie die Wege der MPEG-2 Daten in dem Server-PC des *Digitalen Video Projektes* dar. Zuerst sollen die verschiedenen Quellen, denen ein MPEG-2 Transportstrom entnommen werden kann, betrachtet werden. In der Menüauswahl des VDRs stehen dem Nutzer drei Quellen zur Verfügung:

1. Das Streamen der durch die Videorecorder-Software VDR gespeicherten MPEG-2 Daten auf der Festplatte.
   Die aufgenommen Videodaten werden in ein vorher definiertes Verzeichnis der Festplatte gespeichert. Durch die cDvbApi-Klasse erhält VDR den Zugriff auf den Linux DVB-Treiber, der wiederum die DVB-Karten anspricht. Anhand der Klasse cRecordControl wird der Speicherungsprozess des VDRs gesteuert. In Verbindung mit cRecordBuffer werden die gewünschten MPEG-2 Daten in Form eines Transportstromes in das Verzeichnis der Festplatte übertragen. Die VideoStreaming-Softwarekomponente soll nun ausgewählte Aufnahmen aus dem Verzeichnis an die entsprechenden Clients streamen. Bei 5 möglichen Clients besteht keine Ressourcenbeschränkung auf dem Server-PC solange jeder Nutzer einen Film anfordert.
   Belastung an das Netzwerk: pro MPEG-2 Stream => 3 bis 4 Mbit/s

2. Zusätzlich das Streamen eines MPEG-2 Transportstomes, der direkt dem jeweiligen DVB Device entnommen wird.
   Da im Server-PC zwei DVB-Karten vorhanden sind, bietet diese Möglichkeit eine günstige Erweiterung des Streaming-Angebots. Die VDR-Software nimmt durch ihre Benutzung ein DVB-Device in Anspruch. Das jeweilig Device wird dadurch blockiert. Ein Streamen des blockierten DVB-Device ist in dem Fall nicht mehr möglich, und es kann nur, unter Verwendung des Linux DVB-Treibers, ein Transponder-Paket der zweiten DVB-Karte gestreamt werden. Jeder der fünf Clients ist somit in der Lage, einen Sender eines DVB-Paketes problemlos zu empfangen, unabhängig davon, was auf dem Server-PC geschaut wird.
   Belastung an das Netzwerk: pro MPEG-2 Stream => 3 bis 4 Mbit/s

3. Das Streamen einer DVD.
   Dieses Streaming-Angebot kann nur begrenzt zur Verfügung gestellt werden und nimmt zugleich einen hohen Anteil der verfügbaren Bandbreite der Netzwerkkarte in Anspruch. Nur ein Client kann diesen Stream anfordern. Der Zugriff auf das Laufwerk wird dadurch für die anderen Nutzer gesperrt.
   Belastung an das Netzwerk: DVD MPEG-2 Stream => 6 bis 9 Mbit/s

Die Schnittstelle zwischen den Quellen und der Netzwerkkarte muss somit die VideoStreaming-Komponente bilden. Sie muss die MPEG-2 Daten für eine Übertragung über das LAN entsprechend verarbeiten und soll den Vorgang regulieren.

In dieser Arbeit wird eine Ergänzung des üblichen UDP/IP Protokollturmes mit dem RTP Protokoll vorgeschlagen. Die Eigenschaften der einzelnen Protokolle können dem Abschnitt 2.4.4 entnommen werden.
Die in das RTP eingebundene MPEG-2 Datenpakete werden durch die Streaming-Software fortlaufend nummeriert (Sequenznummer) und mit Zeitinformationen (Timestamps) versehen. Anhand dieser Information ist es der Softwarekomponente des Clients möglich, Paketverluste zu erkennen oder Daten vor ihrem Abspielen für den Nutzer zu puffern, um deren Reihenfolge, die während der Übertragung möglicherweise verändert wurde, wiederherzustellen oder die Varianz in den Übertragungszeiten (*Jitter*) auszugleichen. Der konkrete Aufbau eines *Real-time Transport Protocols* kann der RFC 2250 entnommen werden [28].

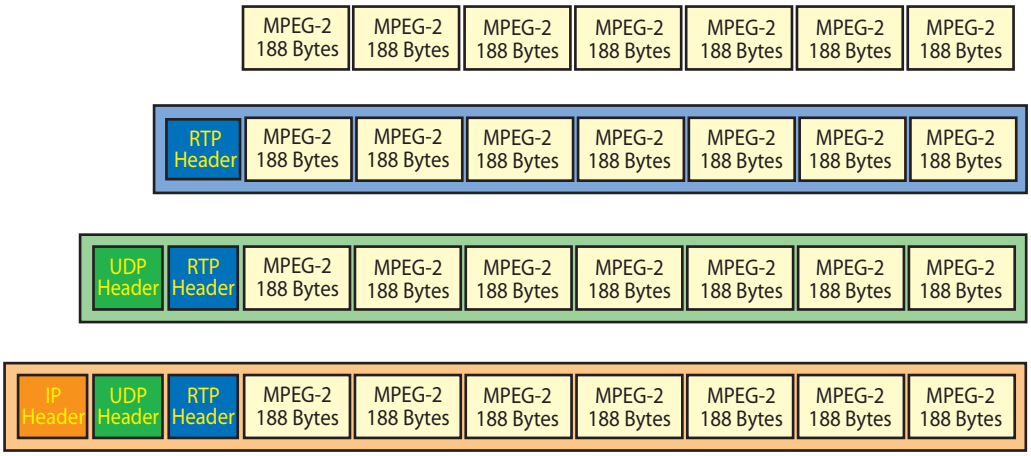

Abbildung 3.5: Verarbeitung der MPEG-2 TS-Pakete

Der Ablauf, MPEG-2 Pakete mittels des RTP/UPD/IP Protokollturmes über Ethernet zu verschicken (Abbildung 3.5), wird kurz verdeutlicht.

1. Es werden 7 MPEG-2 Datenpakete a 188 Bytes in ein RTP-Paket eingebunden. Diese Anzahl der Datenpakete ist wegen der maximalen Ethernet-Frame Länge von 1526 Bytes begrenzt, und während der gesamten Übertragung festgelegt. Die im RTP-Header enthaltene Zeitreferenz beträgt 90 kHZ und wird von dem 90 kHz PCR-Zähler (*Program Clock Reference*) des MPEG-2 Dekoder übernommen. Jedem Paket wird eine Sequenznummer zugewiesen.

2. Im UDP Header sind eine Quelle- und eine Ziel-Portadresse enthalten. Wegen unterschiedlicher UDP Portadressen können die 7 MPEG-2 Pakete aus

unterschiedlichen Streams (Eingangsquellen) stammen. Fordert ein Client mehrere Streams an dieselbe IP Adresse an, werden die Streams anhand der unterschiedlichen Portnummern nachher getrennt und dementsprechend zusammengesetzt.

3. Das UDP Paket wird samt seines kompletten Inhaltes in ein IP Paket eingebunden. Durch die Hinzufügung des IP-Headers werden dem Datenpaket Quelle- und Ziel-Adresse gegeben. Diese Information befähigt die Routers und Switches, die Streams an die richtige Adresse (Client) zu senden.

4. Das End-Paket samt den benötigten Übertragungsvariablen wird über das lokale Netzwerk geschickt.

3.5.2 Verarbeitung im Client-PC des DVP

Die Verarbeitung der paketierten MPEG-2 Daten gestaltet sich im Client des DVPs sehr viel einfacher als im Server-PC. Dies wird alleine schon aus den Anforderungen an die einzelnen Systeme deutlich (3.3.1 und 3.3.2). Die Aufgabe des Clients liegt dabei hauptsächlich in der Zusammensetzung der übertragenen Datenpakete.

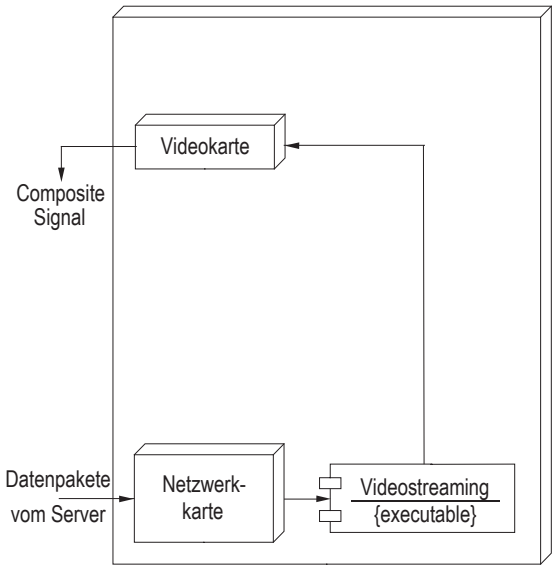

Abbildung 3.6: Ablauf im Client-PC

Die Aufgabe der VideoStreaming-Softwarekomponente besteht aus einer Auswertung der einzelnen *Header*-Informationen des RTP/UDP/IP Protokollturmes der

jeweiligen an der Netzwerkkarte eintreffenden Pakete. Anhand dieser Informationen folgt eine Aneinanderreihung der zusammenhängenden MPEG-2 Pakete, um am Ende einen einzelnen Gesamtdatenstrom erzeugen zu können. Dieser kann mit Hilfe eines *Players* angeschaut werden.
Folgende Fälle können dabei eintreten:

1. In dem Paketdatenstrom, der an die Netzwerkkarte eintrifft, sind nur die MPEG-2 Daten eines Filmes/Senders enthalten. Die Adressierung der Datenpakete soll durch den Server festgelegt und an einen definierten UDP-Port des Client-PCs übertragen werden.
   In diesem Fall müssen die MPEG-2 Inhalte aus den RTP/UDP eingebundenen Datenstrom extrahiert werden. Anhand des Timestamps und der Sequenznummer im RTP Header erfolgt eine kontollierte Zusammensetzung der einzelnen Pakete. Zugleich soll der Netzwerk-Jitter aus jedem Transportstrom entfernt werden. Dabei bleibt nur noch ein minimaler PCR Jitter über.

2. In dem Datenpaketstrom, der an der Netzwerkkarte eintrifft, sind MPEG-2 Daten unterschiedlicher Filme/Sender enthalten, die vom Server an verschiedene UDP-Ports des Clients gesendet werden.
   In diesem Fall wird bereits anhand der UDP Header-Information eine Sortierung und Aufteilung der zusammenhängenden MPEG-2 Daten geschehen. Auch hier sorgt RTP für eine Reduzierung des Netzwerk-Jitters und zusätzlich für ein kontrolliertes Zusammensetzen der MPEG-2 Pakete.

3.6 Mögliche Anwendungsfälle

Für den Ablauf einer Videodatenverteilung im DVP werden zwei bestimmte Anwendungsfälle betrachtet, die bei einer Bedienung des konzipierten DVP-Systems durch einen Nutzer des Systems auftreten können. Als Vorlage sollen dabei die vorgeschlagenen Befehlsgruppen samt ihren jeweiligen Steuerbefehlen dienen. Diese können der Tabelle 3.2 in dem Abschnitt "Kommunikation zwischen Client und Server" entnommen werden.

Use Case 1:

Ein Nutzer hat zu Beginn die Auswahl das System zu starten, falls es schon läuft es zu beenden oder eine Kanalumschaltung der TV-Sender durchzuführen. Zugleich wird im die Möglichkeit geboten, mittels dem `select`-Befehl der Stream-Auswahl Gruppe sich die gespeicherten Filme (Movies) anzeigen zu lassen, die sich auf der Festplatte des Server-PCs befinden. Aus diesen möchte er einen bestimmten Stream auswählen.

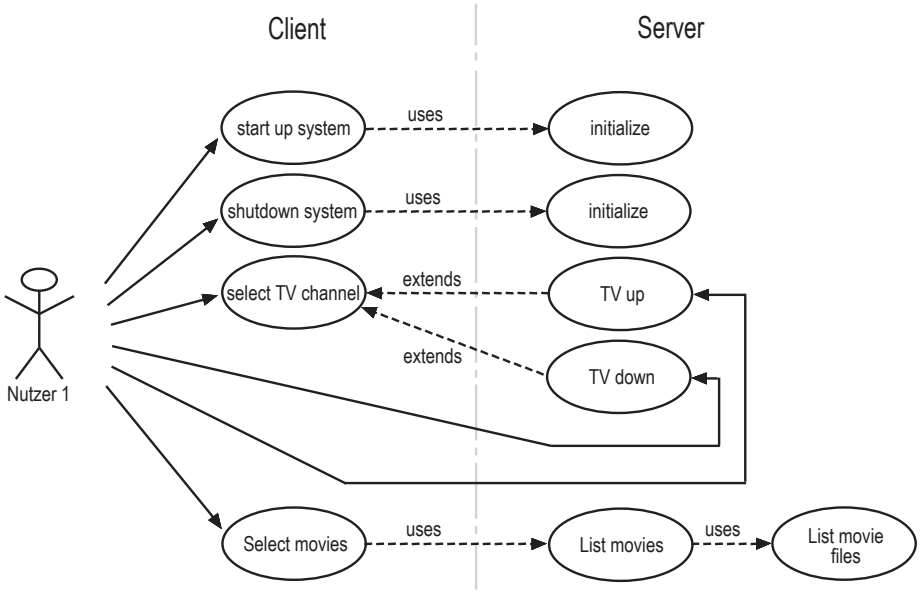

Abbildung 3.7: Use Case 1

Use Case 2:

In diesem Fall wird dem Nutzer die Wahl eines Steuerbefehls aus der Stream-Steuer Gruppe (Start-, Stop-Befehl, usw.) oder der Stream-Setup Gruppe (Add-, Delete-Befehl) angeboten. Der anschließende Steuervorgang des Severs ist bei jedem Steuerbefehl identisch.

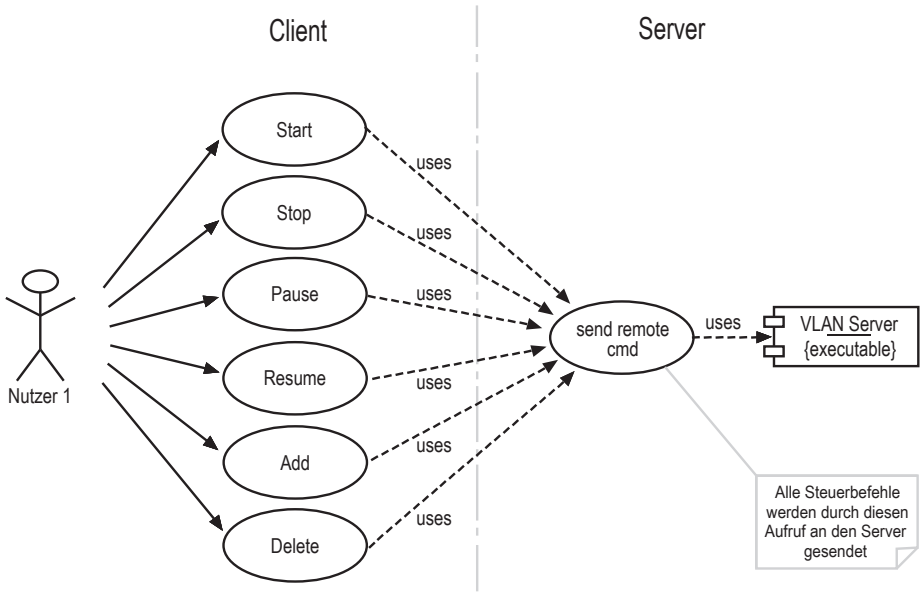

Abbildung 3.8: Use Case 2

Kapitel 4

Prototypische Realisierung

4.1 Das Umfeld

Für eine prototypische Realisierung der Videodatenverteilung im DVP müssen zunächst ein paar Voraussetzungen erläutert werden.

- Nach intensiver Recherche geeigneter Lösungen für das DVP wurde beschlossen, für die Realisierung im Sinne dieser Arbeit auf eine bestehende Streaming-Software *VideoLAN* [27] der *École Centrale Paris* zurückzugreifen. Die Software ist eine vollständige *Open-Source*-Streaming-Lösung unter Linux. Sie erfüllt ideal alle Bedingungen, die an beide VideoStreaming-Softwarekomponenten auf dem Server-PC und dem Client-PC in der Konzeption im Kapitel 3 gestellt wurden. Die Streaming-Software setzt sich aus einem *VideoLan Server* und einem *VideoLAN Client* zusammen.
  Beide Komponenten werden jeweils im Abschnitt 4.2 näher erklärt.

- Ein wichtiger Anteil dieser Arbeit ist die Weiterentwicklung der Projektsoftware VDR. Die Steuerung einer Videodatenverteilung soll in diese Software integriert werden. Leider ist es nicht möglich, die gewünschte Software auf einem Client-PC ohne DVB-Karte zu verwenden, so wie der Aufbau des Client-PCs im Kapitel 3 konzeptioniert wurde. Das *On-Screen-Display*-Menü des VDRs ist auf den DVB-Treiber und den damit verbundenen TV-Ausgang der DVB-Karte angewiesen. Für diesen Zweck wird der Steuerungsprozess der Videodatenverteilung auf den Server-PC verlegt. Der Client-PC wird zunächst nur als Empfänger der gesendeten Videostreams angesehen. Eine ausführliche Erklärung der Umsetzung folgt im Abschnitt 4.3.

4.2 Aufbau der Rechnerarchitektur im DVP

Die Abbildung 4.1 stellt den konkreten Aufbau der Projektrechner dar. Auf die für diese Arbeit relevanten Komponenten wird nun näher eingegangen.

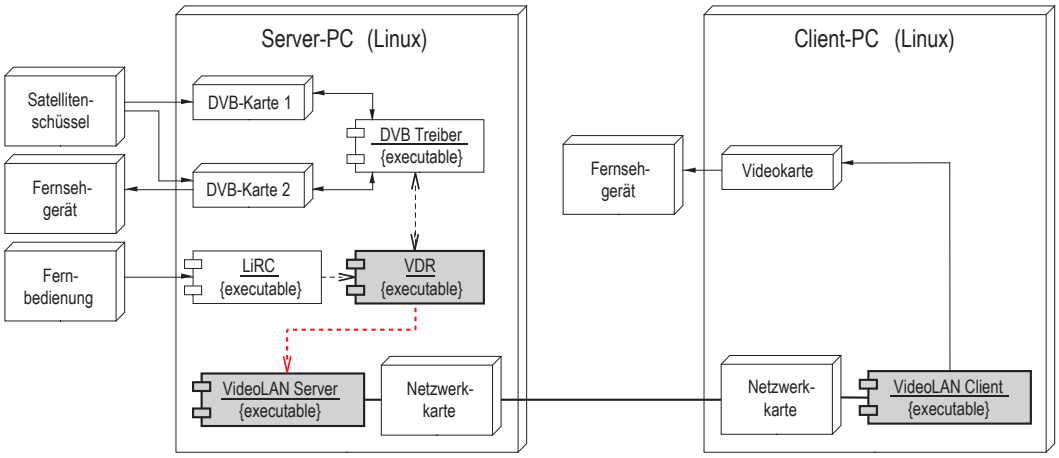

Abbildung 4.1: Rechnerarchitektur im DV-Projekt

4.2.1 Die Projektsoftware VDR

Der *VideoDiscRecorder* (VDR) ist ein *Open-Source*-Projekt für den Empfang, Aufzeichnung und Wiedergabe von digitalen Satellitenprogrammen (DVB-S). Die Steuerung orientiert sich an aktuellen VHS-Recordern und erfolgt, je nach Konfiguration, über eine Fernbedienung oder Tastatur und das *On-Screen-Display* (OSD) auf dem Fernseher.

Von bisherigen Video-Recordern setzt sich VDR durch eine Reihe von Komfortfunktionen ab. Da VDR auf einem normalen Linux-PC läuft, lassen sich die MPEG-2-Aufzeichnungen wahlweise direkt auf dem gegebenen Rechner bearbeiten oder über eine Netzwerkverbindung an andere Rechner übertragen. Zum lokalen Schneiden braucht es keine zusätzliche Software, VDR bietet eine eigene Schnittfunktion. Außerdem unterstützt die Lösung den von vielen Digitalsendern ausgestrahlten elektronischen Programmführer (EPG) und lässt sich optional sogar im *Time-Shift*-Modus[1] bearbeiten. Hierzu benötigt der Server-Rechner eine zweite DVB-Karte, wie in Abbildung 4.1 skizziert. Die VDR-Software wurde von Klaus Schmidinger [1] entwickelt und ist im Internet frei erhältlich.

[1] Erlaubt das Zurückspulen und Betrachten einer laufenden Aufzeichnung

4.2.2 Die Serverarchitektur

Hardware:
Die Anforderungen an den Server im *Digitalen Video Projekt* wurden im Abschnitt 3.3.1 bereits geklärt. Ein leistungsstarker *Pentium 4* Prozessor mit 1,5 GHz Taktfrequenz, 512 MB Hauptspeicher und eine 40 GB große Festplatte zur Speicherung der Videodaten bilden demnach die Grundlage dieses Server-PCs. Als Netzwerkkarte wurde eine 100 Mbit Ethernet-Karte gewählt. Der Server enthält zwei DVB-Karten der Marke *TechnoTrend*, die durch den *Siemens* DVB-Treiber angesprochen werden. Eine der DVB-Karten besitzt einen TV-Ausgang, um die Ausgabe des Composite Fernsehsignals zu realisieren.
Für eine prototypische Realisierung ist die Hardware somit ausreichend dimensioniert, was durch die Lastbetrachtung an den Server in Abschnitt 3.3.1 bestätigt wird.

Software:
Als VideoStreaming-Softwarekomponente auf dem Server-PC wurde der *VideoLan Server* (VLS) [27] verwendet. Er erfüllt alle Voraussetzungen, die an eine VideoStreaming-Komponente gestellt werden, und bietet folgende Funktionen:

- Er besitzt die Fähigkeit, verschiedene MPEG-2 Datenquellen (DVB-S Karte, Festplatte, DVD) auszulesen und die MPEG-2 Daten wie im Abschnitt 3.5.1 dargelegt zur Übertragung über das lokale Netzwerk zu verarbeiten.

- Er hat die gesamten Funktionalitäten eines Servers. Es können Übertragungskanäle zu mehreren Clients festgelegt und Nutzerprofile mit bestimmten Zugriffsrechten erstellt werden.

Der *VideoLan Server* benötigt eine spezielle Konfiguration mittels zweier Konfigurationsdateien. Die nötige Konfiguration für den in der Realisierung verwendeten *VideoLan Server* ist dem Anhang A zu entnehmen.
Die VLS-Softwarekomponente erfüllt alle Voraussetzungen, die für einen Einsatz auf dem Server-PC notwendig sind. Sie ermöglicht einen Zugriff mehrerer Clients und reguliert den gesamten Verteilungsvorgang der Datenpakete über das lokale Netzwerk.
Auf dem DVP Server-PC ist das SuSE Linux (Version 8.0) Betriebssystem installiert.

4.2.3 Die Clientarchitektur

Hardware:

Die Anforderungen an einen Client wurden bereits im Abschnitt 3.3.2 betrachtet. Trotzdem wurden im Client-PC ein *Pentium 4* Prozessor mit 1,5 GHz Taktfrequenz, 512 MB Hauptspeicher und eine 40 GB große Festplatte eingebaut, da der Client-Rechner noch anderweitig im *Digitalen Video Projekt* genutzt wird und nicht nur als Client verwendet werden soll. Er besitzt eine 100 Mbit Ethernet-Netzwerkkarte sowie zwei DVB-Karten. Für den Verlauf der Arbeit, speziell für die Realisierung, wird jedoch von einem Grundaufbau ohne DVB-Karten, wie er im Abschnitt 3.2 dargestellt ist, ausgegangen.

Software:

Als VideoStreaming-Softwarekomponente auf dem Client-PC wurde der *VideoLan Client* (VLC) [27] verwendet. Für die Diplomarbeit ist seine Funktion, die Möglichkeit aus einzelnen Datenpaketen (siehe Abschnitt 3.5.2) einen Gesamtdatenstrom zu bilden, besonders wichtig. Der *VideoLAN Client* (VLC) ist auch ein plattformunabhängiger Multimedia-Player und kann DVDs, VCDs, MPEG, DivX Formate sowie den MPEG-2 Transportstrom einer DVB-S Karte unter Linux und Windows abspielen.
Die VLC-Softwarekomponente erfüllt alle Voraussetzungen, die für einen Einsatz auf dem Client-PC notwendig sind. Sie kombiniert ideal den nötigen Verarbeitungsvorgang der gesendeten Datenpakete mit den Funktionen eines *Players*. Es wird somit keine weitere Softwarekomponente auf dem Client benötigt.
Auf dem DVP Client-PC ist das SuSE Linux (Version 8.0) Betriebssystem installiert.

4.2.4 Die Netzwerkarchitektur

Die Netzwerkarchitektur ist für die Realisierung der Videodatenverteilung im *Digitale Video Projekt* vorgegeben. Sie gleicht nicht der gewünschten Architektur der Konzeption dieser Diplomarbeit (siehe Abbildung 2.2). Beide Projektrechner sind an dem Netzwerk der Fakultät Prozessinformatik der TU Ilmenau angeschlossen. Der Übertragungsweg zwischen dem Server-PC und dem Client-PC wird jedoch für die prototypische Realisierung vereinfacht und als eine direkte Verbindung angesehen. Bei verschiedenen Streaming-Tests zwischen den Projektrechnern traten keine durch die Netzwerkstruktur verursachten Engpässe auf.

4.3 Umsetzung der Videodatenverteilung

Nachdem bereits die Rechnerarchitektur für die exemplarische Realisierung dargestellt und eine Erläuterung der dafür relevanten Softwarekomponenten VDR, VLS und VLC durchgeführt wurde, wird nun auf die Umsetzung der Videodatenverteilung im DVP eingegangen. Sie setzt sich aus einem Übertragungsprozess und einem Steuerungsprozess zusammen. Besonders wichtig ist dabei das Zusammenspiel der Komponenten *VideoLAN Server* und VDR.

Wie bereits erwähnt, erfüllt die Streaming-Lösung *VideoLAN* [27] alle Bedingungen der Konzeption, die im Abschnitt 3.5 an eine Übertragung von MPEG-2 Daten über ein lokales Netzwerk gestellt wurden. Darüberhinaus besitzt der *VideoLAN Server* den vollen Funktionsumfang eines üblichen Servers, wie Nutzerverwaltung und Datenverwaltung. Er wird deshalb in den Systemaufbau integriert und zusätzlich für eine Verwaltung der Clients konfiguriert. Die für diese Arbeit relevante Konfiguration ist dem Anhang A zu entnehmen. In der Konfiguration wird dem Server die

- Verbindungsart $\Longrightarrow$ eine Unicast-Verbindung zum Client,
- Zieladresse des Clients $\Longrightarrow$ Netzwerk Adresse und Port,
- Deklaration der Stream-Quellen $\Longrightarrow$ DVB, Filmdatei und DVD

vorgegeben. Der Übertragungsprozess wird demnach durch den *VideoLAN Server* (VLS) übernommen und durch den *VideoLAN Client* (VLC), der auf dem Client-PC integriert wird, abgerundet.

Besonders relevant ist jedoch der Steuerungsprozess der Videodatenverteilung im DVP. Die Steuerung des *VideoLAN Servers* soll aus der *On Screen Display*-Menüführung des VDRs geregelt werden. Zu diesem Zweck wurde das Plugin `vls` konzipiert, modelliert und anschließend in die bestehende Softwarekomponente VDR integriert. Mit Hilfe des Plugins `vls` wird eine Kommunikation zwischen dem Server und dem digitalen Videorecorder VDR ermöglicht. Dieser wird durch das Plugin sozusagen zu einem "Client" umfunktioniert.
Durch das Plugin `vls` wird somit eine Integration der Softwarekomponenten VDR und VLS erreicht und eine serverseitige Steuerung der Videodatenverteilung für die prototypische Realisierung im *Digitalen Video Projekt* erzielt.

4.4 Die Programmierung

Die Programmierung des Plugins `vls` erfolgte in der Programmiersprache C++. Sie wurde durch die bestehende Struktur der VDR-Software vorgegeben. Die Programmiersprache C++ ist objektorientiert und bietet den Vorteil einer Vererbung von Objekten und Klassen der VDR-Software für das Plugin `vls`.

4.4.1 Aufbau und Struktur des Plugins `vls`

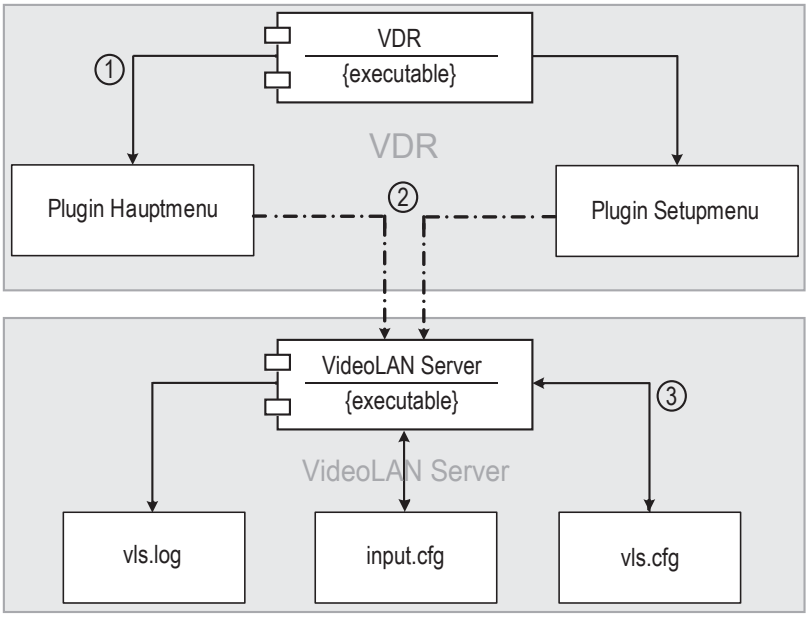

Abbildung 4.2: Ablauf der Anwendung

1. In die VDR Menüführung gelangt man in die Struktur des Plgins (siehe Abbildung 4.3). Sie werden beim Start der VDR-Software initialisiert.
2. Die Steuerbefehle werden aus der neuen, durch das Plugin `vls` entstandenen, Menüumgebung des VDRs an die Socket des *VideoLAN Servers* gesendet.
3. Abfrage der Konfigurationsdateien zur Überprüfung der Steuerbefehle.

Am Anfang der Anwendung steht die VDR-Menüseite, in die ein neuer Menüpunkt für das Plugin `vls` integriert wurde. Aus diesem Startmenü gelangt man zum Hauptmenü des Plugins `vls`. Dort stehen dem Nutzer verschiedene Optionen zur Auswahl (siehe Abschnitt 4.4.2), aus deren möglichen Entscheidungen eine Zusammensetzung der unterschiedlichen Steuerbefehle des *VideoLAN Servers* entsteht. Diese Steuerbefehle werden aus dem Plugin direkt an den *VideoLAN*

Server übergeben, der die gewünschten Befehle ausführen soll. Zuerst überprüft jedoch der Server, ob die gewünschten Optionen des Nutzers ausführbar sind. Dies geschieht anhand einer Abfrage der beiden Konfigurationsdateien vls.cfg und input.cfg in denen sich wichtige Informationen wie Zieladressen (IP-Adresse und Portnummer) sowie andere Umgebungsvariablen der jeweiligen Clients befinden (siehe Anhang A).

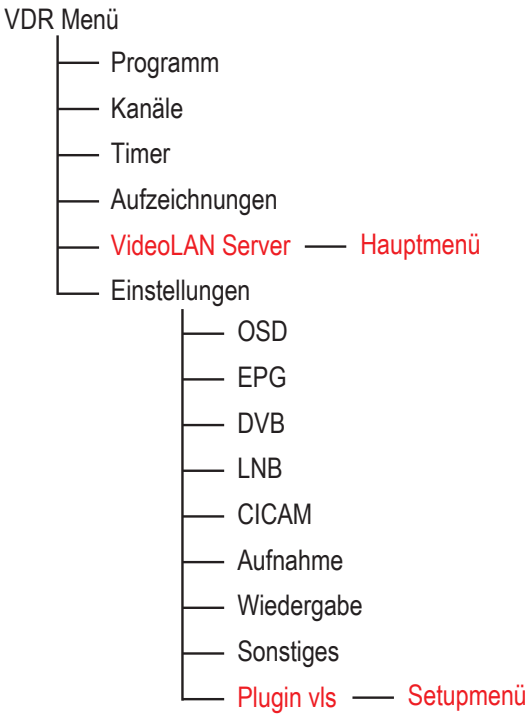

Abbildung 4.3: VDR-Menüführung mit dem Plugin `vls`

Im VDR-Menü besteht die Möglichkeit, in den Untermenüpunkt VDR-Einstellung zu wechseln. Wie in der Baumstruktur 4.3 zu sehen ist, wurde auch hier ein neuer Menüpunkt für das Plugin `vls` integriert, von dem aus man in das Plugin-Setupmenü gelangt. Dort steht dem Nutzer eine zusätzliche Konfiguration des *VideoLAN Servers* zur Verfügung (siehe Abschnitt 4.4.3). Konfigurationsbefehle werden direkt an den Server geschickt, wodurch es dem Nutzer ermöglicht wird, eine Konfiguration des *VideoLAN Servers* während des laufenden Prozesses durchzuführen. Die eigentlichen Konfigurationsdateien vls.cfg und input.cfg können nur im heruntergefahrenen Modus des Servers bearbeitet werden. Nach Veränderung der Dateien muss der Server neu gestartet werden, damit die Änderungen in Kraft treten. In der Abbildung 4.2 wird das Zusammenspiel der durch das Plugin erweiterte VDR-Software und dem *VideoLAN Server* verdeutlicht.

Die Tabelle 4.1 weist die einzelnen programmierten Dateien des Plugins sowie eine kurze Beschreibung deren Funktionsweise auf. Die für den Steuerungsprozess relevanten programmierten Dateien socket.c und socket.h des Plugins `vls` werden im Anhang B ausführliche dargestellt.

Datei	Funktionsweise
vls.h vls.c	das Grundgerüst des Plugins, bindet die Programmabläufe des Plugins `vls` in die VDR-Struktur ein
menu.h menu.c	Festlegung der Funktionsabläufe des Hauptmenüs, Zusammensetzung der Steuerbefehle für den *VideoLAN Server*
setup.h setup.c	zusätzliche Konfiguration des *VideoLAN Servers* aus der Bedienungsumgebung des Plugins heraus
socket.h socket.c	Regelung der Kommunikation zwischen dem *VideoLAN Server* und dem Plugin, Übertragung der Steuerbefehle
i18n.h i18n.c	Regelung einer Sprachauswahl für die *On-Screen-Display* Menüführung des Plugins (Englisch/Deutsch)
Makefile	wird zum Compilieren des Plugins `vls` benötigt
vls.log	Logfile speichert alle Vorgänge des *VideoLAN Servers*
vls.cfg	Konfigurationsdatei des *VideoLAN Servers*
input.cfg	Quellen-Konfiguration des *VideoLAN Servers*

Tabelle 4.1: Beschreibung der programmierten C++ Dateien

4.4.2 Das Hauptmenü

Dies ist die zentrale *On-Screen-Display*-Seite des Plugins `vls`. Hier kann eine Auswahl der möglichen Optionen verschiedener Streaming-Quellen (DVB, Filmdatei oder DVD) und bestimmten Streaming-Zielen (Client) getroffen werden. Der Nutzer ist somit in der Lage, einen Stream an den gewünschten Client zu leiten.
Im unteren Teil des Hauptmenüs werden spezielle Funktionstasten, die sich auch auf handelsüblichen Fernbedienungen befinden, dargestellt. Anhand der vorgegebenen Tastenbelegung im Plugin `vls` kann der Nutzer durch Betätigung der jeweiligen Funktionstaste der Fernbedienung (rote, grüne, gelbe und blaue Taste) einen ausgewählte Streams starten oder stoppen. Jede Aktion des Nutzers wird im *On-Screen-Display* des VDRs durch eine dementsprechende Nachricht bestätigt. Wurde ein Stream gestartet, wird dem Nutzer ermöglicht, zum VDR-Startmenü zu wechseln, um andere Funktionalitäten des VDRs zu nutzen. Zu jedem Zeitpunkt der Anwendung besteht die Möglichkeit, in das Hauptmenü des Plugins `vls` zurückzukehren, um einen laufenden Stream zu beenden oder einen neuen zu starten.

Abbildung 4.4: Screenshot des `vls`-Hauptmenüs

4.4.3 Das Setupmenü

Dies ist die *On-Screen-Display*-Seite des Plugins `vls` zur zusätzlichen Konfiguration des *VideoLAN Servers*. Wie bereits erwähnt wurde sie eingerichtet, um dem Nutzer die Konfiguration des Servers aus dem Bedienungsumfeld des *VideoDiskRecorder* (VDR) zu ermöglichen, ohne dass der Nutzer den Server herunterfahren und neu startet muss. Über das Plugin-Setupmenü können derzeit Veränderungen der Clientverwaltung des *VideoLAN Servers* vorgenommen werden. Es können bestehende Clients aus der Konfiguration des Servers entfernt oder neue Clients mit ihrer jeweiligen IP-Adresse und Portnummer der Konfiguration des Servers hinzugefügt werden. Für eine Videodatenverteilung steht die Clientverwaltung im Vordergrund. Zur Erweiterung können zu einem späteren Zeitpunkt zusätzliche Plugin-Einstellungen implementiert werden.
Auch hier besteht die Möglichkeit, jederzeit zwischen dem Setupmenü des Plugins und den vorhandenen OSD-Menüs des VDRs zu wechseln.

Abbildung 4.5: Screenshot des `vls`-Setupmenüs

4.4.4 Die Steuerung des *VideoLAN Servers*

Um die Kommunikation zwischen dem VDR und dem *VideoLAN Server* zu realisieren, wird für das Plugin ein Ansatz auf einer sogenannten niederen Ebene gewählt, die Socket-Programmierung [29]. Ein Socket ist ein *Peer-to-Peer* Kommunikationsendpunkt. Mit der Hilfe von Sockets wird die vorhandene Steuerung des *VideoLAN Servers* ersetzt. Normalerweise bietet dieser Server nur eine Bedienung über eine Telnet-Eingabe. Diese Bedienung ist sehr anwenderunfreundlich und wurde deswegen umgangen.

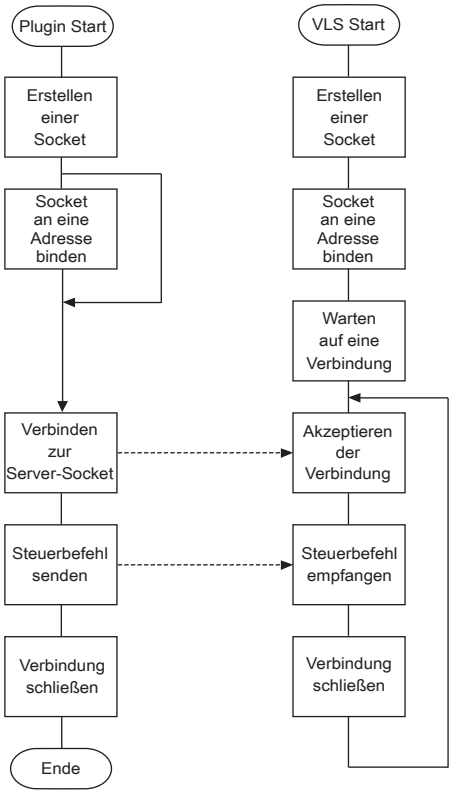

Abbildung 4.6: Ablauf eines Steuerungsprozesses

Um einen korrekten Kommunikationsablauf zwischen dem VDR und dem *VideoLAN Server* und den damit zusammenhängenden Steuerungprozesses zu garantieren, wurde das Flussdiagramm 4.6 erstellt. Das Diagramm verdeutlicht, welche Schritte bei der Socket-Programmierung für das Plugin `vls` beachtet werden mussten. Der daraus entstandene Quellcode zur Erstellung einer Client-Socket ist dem Anhang B zu entnehmen.

Für den Verbindungsaufbau mit dem *VideoLAN Server* im *Digitalen Video Projekt* wird eine Stream-Socket verwendet. Die Stream-Socket benutzt das *Transmission Control Protocol* (TCP) als Schnittstelle, welches sich um die Datenstromkontrolle und Aufrechterhaltung der Verbindung zum Videoserver kümmert. So wird garantiert, dass die gesendeten Befehle aus dem VDR an den *VideoLAN Server* ohne Fehler, Duplikation und in richtiger Reihenfolge ankommen. Die Befehle an den Server werden zum Versenden in einen *Buffer* verpackt. Geht ein *Buffer* verloren, wird der Client (in diesem Fall VDR) veranlasst, die Übertragung zu wiederholen.

Die Zusammensetzung oder Erstellung der Steuerbefehle des *VideoLAN Servers* ist fest definiert [27] und wird in dieser Arbeit als vorgegeben betrachtet. Sie bestehen aus einem Grundgerüst, in das verschiedene Variablen (*Strings*) eingefügt werden. Die Variablen werden durch den Nutzer in dem Haupt- und Setupmenü des Plugins ausgewählt und erhalten so ihren jeweiligen Inhalt. Anhand der konkreten Beispiele wird die Zusammensetzung der beiden relevanten Steuerbefehle der prototypischen Realisierung erläutert.

Der `start`-Befehl:

$$\text{Grundgerüst} \Longrightarrow \texttt{start}\ [\text{Filmdatei/Sender}]\ [\text{Ziel}]\ [\text{Quelle}]$$

Der Nutzer hat sich für den Fernsehsender [Pro7] der Quelle [DVB] entschieden und möchte dies an das Ziel [Client1] übertragen bekommen. Mittels des Kommandos write() wird der Inhalt des *Buffers* an die Socket des Servers übertragen.

$$\textit{Buffer} \Longrightarrow \texttt{start}\ \text{Pro7 Client1 DVB}$$

Der `stop`-Befehl:

$$\text{Grundgerüst} \Longrightarrow \texttt{stop}\ [\text{Filmdatei/Sender}]\ [\text{Ziel}]$$

Der Nutzer bricht den Stream des Fernsehsenders [Pro7] an das Ziel [Client1] ab. Auch in diesem Fall wird der Inhalt des *Buffers* mit dem Kommando write() an die Socket des Servers übertragen.

$$\textit{Buffer} \Longrightarrow \texttt{stop}\ \text{Pro7 Client1}$$

4.4.5 Darstellung auf dem Client

Zuletzt folgt eine Darstellung der Streams auf dem Client-PC. Hierzu wird wie bereits erwähnt die vorhandene Softwarekomponente *VideoLan Client* verwendet (siehe Abschnitt 4.2.3). Sie ist von dem Datenstrom abhängig, der ihr zur Verfügung gestellt wird. Es ist nicht möglich aus dem Bedienungsumfeld des *VideoLan Clients* (siehe Abbildung 4.7) den *VideoLan Server* anzusteuern, um einen gewünschten Stream zu starten. In diesem Fall nimmt der Client DVP-Rechner für den Anwender nur die Funktions eines *Players* ein und man ist demnach von der serverseitigen Steuerung der Videodatenverteilung abhängig.
Zuletzt wurde noch zusätzlich eine Steuerung des VideoLAN Servers von einem Client-Rechner im DVP-System aus realisiert. Durch die im folgendem Abschnitt erläuterte clientseitige Steuerung, wird dem Nutzer des DVP-Systems eine größere Bedienungsauswahl im *Digitalen Video Projekt* geboten.

Abbildung 4.7: Screenshot des *VidoeLAN Clients*

4.5 *Video-on-Demand* im DVP

Mit Hilfe des Plugins `vls` wurde eine serverseitige Steuerung der Videodatenverteilung aus dem Umfeld des *VideoDiskRecorders* realisiert. Um jedoch auch eine *Video-on-Demand* Umgebung (siehe Abschnitt 2.4.2) für den Systemaufbau des *Digitalen Video Projektes* zu erschaffen, wurde zusätzlich eine rudimentäre Steuerung auf dem Client-PC integriert, die eine clientseitige Steuerung des *VideoLAN Servers* ermöglicht, genannt `VLS Steuerung`.

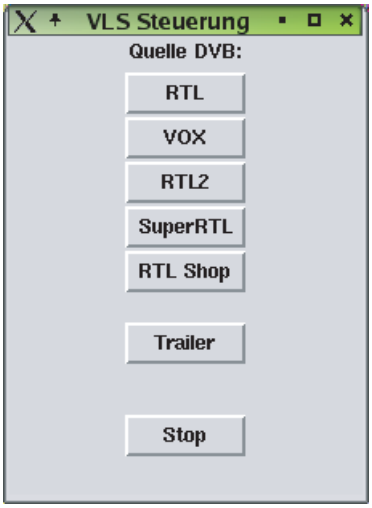

Abbildung 4.8: Screenshot der *VidoeLAN Server* Steuerung

Zu diesem Zweck wurde mittels Tcl/TK, der *Tool command language* (Tcl) und ein auf Tcl basierendes *Toolkit* (TK) für X11 von Linux, eine grafische Oberfläche erstellt, die in der Abbildung 4.8 zu sehen ist. Tcl ist als Interpretersprache konzipiert, die eine *Shell* zur Ausführung der Tcl-Skripte benötigt. Die eigentliche Programmierung der Shell diente nur zur Implementierung der reinen Funktionalität der clientseitigen Steuerung des *VideoLAN Servers*. Durch die Betätigung eines *Buttons* in der grafischen Oberfläche wird der eigentliche Steuerungsprozess initialisiert. Der Steuerungsprozess wird wie bei der serverseitigen Steuerung des *VideoLAN Servers* durch die Socket-Programmierung realisiert, auf die im Abschnitt 4.4.4 bereits näher eingegangen wurde.
Zum Betrachten der Streams wird weiterhin der *VideoLAN Client* benötigt. Durch die Kombination des *VideoLAN Clients* mit der grafischen Steueroberfläche ist eine übersichtliche Steuerung der Videodaten von jedem Client im DVP-System möglich, und bietet zusätzlich eine *Video-on-Demand* Lösung.

Kapitel 5

Ergebnisse

Unter Berücksichtigung der Aufgabenstellung weist das Kapitel 1 den Stand der Technik auf, der für den Verlauf dieser Diplomarbeit nötig ist. Dabei werden insbesondere die relevanten Themenbereiche des *Digital Video Broadcasting* (DVB) und die Grundlagen der Netzwerktechnik bearbeitet.
Im Kapitel 2 werden die laut Aufgabenstellung geforderte Recherche und anschließende Analyse einer Videodatenverteilung durchgeführt. Zu diesem Zweck werden mögliche, bestehende Übertragungsmedien in einem Haushalt für eine Übertragung von Videodaten analysiert. Ausschlaggebenden Kriterien waren dabei die Bereitstellung einer nötigen Datenrate zur Übertragung mehrerer MPEG-2 Datenströme, sowie eine mögliche Einsetzbarkeit für die spätere Umsetzung im DVP. Die Ethernet-Netzwerktechnik erfüllte alle Bedingungen und wurde für den Einsatz im DVP ausgewählt.

Übertragungsweg	**max. Datenrate**	**Verfügbarkeit**
Stromversorgungs-netz (PLC)	bis 2 Mbit/s	- auf dem Markt erhältlich - teures Modem benötigt
2 Ader-Kupfer-leitung (Telefon)	bis 52 Mbit/s VDSL	- befindet sich noch in einer Testphase
Kupfer-Koaxial-Kabel	bis 43 Mbit/s	- in Europa kein einheitlicher Standard vorhanden
Funkübertragung	bis 54 Mbit/s HiperLAN 2	- Spezifikation noch nicht fetiggestellt
Ethernet-Netzwerk	je nach Vernetzung bis in den Gigabyte-Bereich	- günstige bis teure Lösungen einer Vernetzung erhältlich

Tabelle 5.1: Ergebnisse der Analyse

Zusätzlich wurde im Kapitel 2 ein geeignetes Verfahren für die Übertragung von Videodaten über ein lokales Ethernet-Netzwerk bestimmt. Das Streaming-Verfahren soll für die Übertragung der Videodaten im DVP verwendet werden.
Aus den Ergebnissen des zweiten Kapitels ergeben sich die Eckpunkte für die Konzeption und die spätere Umsetzung einer Videodatenverteilung. Es wurde eine Konzeption zur Videodatenverteilung innerhalb des *Digitalen Video Projektes* vorgelegt. Das erarbeitete Konzept baut auf den bestehenden Systemkomponenten auf. Durch die Integration der vorgelegten Konzeption wurde eine Videodatenverteilung von dem DVP-Hauptsystem (Server-PC) an verteilte Systeme (Client-PCs) realisiert. Es wurde eine Komponentenerweiterung für das DVP-System entwickelt, um einen Datenübertragung zu ermöglichen.

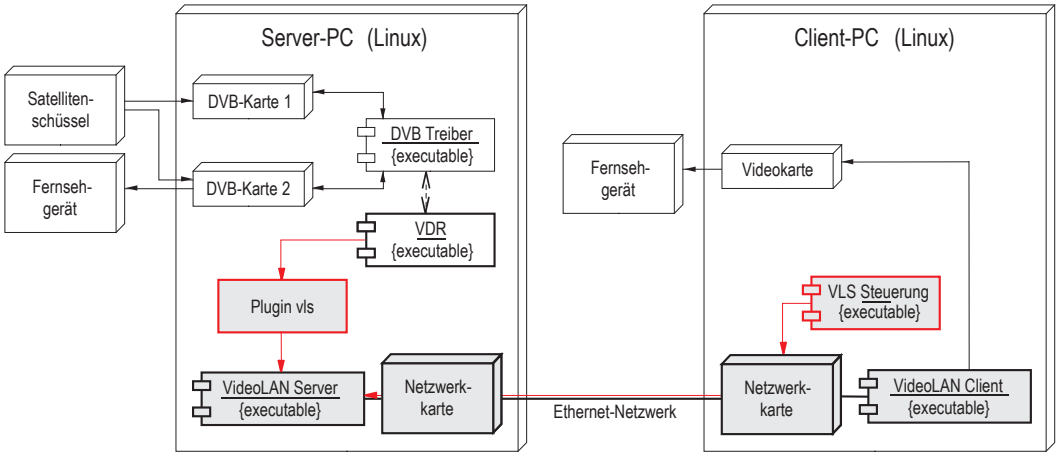

Abbildung 5.1: Ergebnis der erweiterten Rechnerarchitektur

Die Erweiterung ist in der Abbildung 5.1 gekennzeichnet. Zwecks einer besseren Verarbeitung der MPEG-2 Daten für die Übertragung, wurde der UDP/IP Protokollturm durch das RTP Protokoll ergänzt. Der Vorgang Videodatenverteilung wurde durch eine Steuerung der Videodaten abgerundet. Die Konzeption wurde in einer anschließenden prototypischen Realisierung in den Aufbau des DVP-Systems integriert.
Das Ergebnis dieser Integration ist ein abgeschlossener Vorgang einer Videodatenverteilung im DVP. Dieser wird aus dem Bedienungsumfeld, dem *On-Screen-Display*-Menü des digitalen Videorecorders VDR, gesteuert. Die Steuerung wurde mittels dem programmierten Plugin `vls` realisiert. Der Steuerungsprozess der Videodaten, die Kommunikation zwischen dem VDR und dem *VideoLAN Server*, wurde im Plugin `vls` mit Hilfe der Socket-Programmierung umgesetzt. Das er-

stellte Plugin `vls` fügt sich in die Struktur der Softwarekomponente VDR ein, und passt sich der bestehenden Bedienungsumgebung, Menüführung, an. Der Übertragungsprozess der Videodaten wurde durch die Integration der Streaming-Lösung *VideoLAN*, bestehend aus einem *VideoLAN Server* und einem *VideoLAN Client*, gelöst.
Zusätzlich zur serverseitigen Steuerung wird eine clientseitige Steuerung des *VideoLAN Servers* realisiert. Das eigenständige, ausführbare Programm `VLS Steuerung` beruht auf dem Prinzip der Socket-Programmierung und ist im Client-PC der Abbildung 5.1 abgebildet.

Kapitel 6

Ausblick

6.1 Allgemeiner Ausblick

Die Be- und Verarbeitung digitaler Videodaten und die PC-Technik werden in Zukunft immer häufiger kombiniert werden. Der Einsatz digitaler Videorecorder auf der Basis von PC-Komponenten gewinnt vor allem im Heimbereich immer mehr an Bedeutung, da die dafür benötigte, leistungsfähige PC-Technik auch für den Privatanwender erschwinglicher wird.
Mit der Entwicklung vom Ein- zum Mehr-PC-Haushalt und der Erschaffung lokaler Netzwerke wird eine Verteilung von Multimedia-Daten im privaten Haushalt immer interessanter. Sowohl Telekommunikations- als auch Datennetze können in Zukunft für die Übertragung der durch den digitalen Videorecorder gespeicherten Audio-/Videodaten eingesetzt werden. Die Bedeutung lokaler breitbandiger Netzweke wird vermutlich in den kommenden Jahren stark zunehmen und erheblichen Einfluss auf den Umgang privater Nutzer mit elektronischen Medien haben. Derzeit ist die Anwendung im privaten Umfeld jedoch noch relativ gering.

6.2 Ausblick für die Anwendung

Das lokale Netzwerk des DVP-Systems sollte in Zukunft selbstorganisierend sein. Neue Elemente müssen sich selber anmelden und sich mit dem bereits existierenden Netzwerk verbinden. Dabei darf die Funktion verbundener Netzwerkelemente nicht beeinträchtigt werden. Für den Nutzer bedeutet das im Idealfall, dass er nach dem Kauf einer DVP-Client/Server-Lösung, diese einfach auspackt, aufbaut und einschaltet. Nach einer automatischen Selbstinstallation der Geräte sollte das

System funktionsbereit sein. Fällt ein Endgerät (Client) aus, so kann es einfach entfernt, ersetzt oder repariert werden, während der Rest des Systems nicht unterbrochen werden soll.
Im Rahmen einer Weiterentwicklung des Plugins `vls`, könnte der Funktionsumfang ausgebaut werden. Dies bezieht sich besonders auf die Darstellung der gespeicherten Filme und anschließende Aktualisierung der Konfigurationsdateien des *VideoLAN Servers*. Für diesen Zweck wäre die Integration einer Datenbank in das *Digitale Video Projekt* sinnvoll.
Ein wichtiger Punkt in der heutigen Kommunikations- und Datenübertragungstechnik ist die Sicherheit einer Anwendung. Dies trifft auch für die Übertragung der Videodaten über das lokale Netzwerk im DVP zu und speziell auf die Kommunikation mit dem Client und dem *VideoLAN Server*. Im Rahmen einer Weiterführung dieses Projektes könnten daher zusätzliche Sicherheitskonzepte, die Datensicherheit während der Übertragung betreffend, entworfen und realisiert werden. Dies schließt vor allem eine Login-Oberfläche in der Bedienungsumgebung des VDRs für den *VideoLAN Server* ein sowie die Vergabe bestimmter Rechte an die Nutzer des DVP-Systems.
In Zukunft wäre auch eine Funktionserweiterung des *VideoLAN Clients* für die optimale Verwendung im DVP sinnvoll. Dies beinhaltet z.B. die Integration der konzipierten `VLS Steuerung` in die grafische Oberfläche des *VideoLAN Clients*. Zusätzlich wäre eine Steuerung des Client-PC über eine Fernbedienung wünschenswert, um das Endgerät anwenderfreundlich zu gestalten.

Kapitel 7

Zusammenfassung

Diese Arbeit befasst sich mit der Konzeption und Umsetzung einer Videodatenverteilung im *Digitalen Video Projekt*, vom theoretischen Entwurf bis hin zur praktischen Umsetzung. Dabei wird zunächst ein Überblick über das DVB und den Stand der heutige Netzwerktechnik gegeben, die dazu auf ihre einzelnen Komponenten, wie Datenprotokolle, Netztopologien und bestehende Übertragungsmodi, reduziert und untersucht werden. Aus den gewonnen Erkenntnissen über die Netzwerktechnik und der anschließenden Analyse der Übertragungsmedien wird eine Entscheidung für die Umsetzung und Anpassung einer bestimmten Übertragungsvariante für das DVP getroffen.
Das Konzept dieser Arbeit orientiert sich dabei an die Anforderungen einer Videodatenverteilung. Der Verteilungsvorgang setzt sich aus einem Übertragungs- und einem Steuerungsprozess zusammen. Der Übertragungsprozess selbst wird mit Hilfe der frei verfügbaren Software VideoLAN aufgebaut. Der Steuerungsprozess wird anhand des erstellten Plugins `vls` aus der Umgebung des digitalen Videorecorders geregelt. Mit dem entstandenen Client/Server-System im Umfeld des *Digitalen Video Projektes* werden die Vorzüge der digitalen Vidoetechnik und der Ethernet-Netzwerktechnik kombiniert. Durch den Komponentenaufbau der einzelnen System-Rechner ist das DVP-System für eine Videodatenverteilung zukunftsorientiert ausgelegt. Die System kann durch die Integration neuer Softwarekomponenten in seinem Funktionsumfang einfach erweitert werden.
Mit dieser Arbeit ist es einem Nutzer nun möglich, ein System zur Verteilung von Videodaten aus den untersuchten Software- und Harwarekomponenten in Kombination mit dem erstellten Plugin `vls` und der clientseitigen Steuerung `VLS Steuerung` aufzubauen, das ideal in einem privaten, lokalen Netzwerk eingesetzt werden kann.

Anhang A

Konfiguration des *VideoLanServers*

Nach der Installation des *VideoLanServers* auf dem Server-Rechner muss eine Konfiguration der Software erfolgen. Dies geschieht anhand der vls.cfg und input.cfg Konfigurationsdateien. Um eine problemlose Funktionalität des *VideoLanServers* zu garantieren, müssen diese sich laut Angaben der Entwickler [27] in dem Verzeichnis /usr/local/etc/videolan/vls des Linux-Betriebssystems befinden.

A.1 vls.cfg

```
//VLS Konfigurations Datei

//Umgebungskonfiguration
BEGIN "Vls"
  LogFile       = "vls.log"               // Log Datei
  ScreenLog     = "enable"                // Log auf Console
  SystemLog     = "disable"               // Log zum Systemlog
END

// Sicherheit:
// Bei "Groups" wird dir Rechteverteilung der Befehle festgelegt.
// Jeder Nutzer welcher zur Gruppe 'master" gehört, hat alle Rechte.
// Jeder der zur Gruppe 'monitor' gehört, hat nur "Lese-Rechte".
BEGIN "Groups"
  monitor       = "help|browse|logout"
  master        = "help|browse|start|stop|shutdown|logout|config|program|input|channel|show"
END

// Benutze "mkpasswd" um ein 'Encrypted Password' zu generieren.
BEGIN "Users"
```

```
  monitor       = "3BcKWoiQnOvi6:monitor"  // Password ist 'monitor'
  bozo          = "JKg2TpPerilnw:master"   // Password ist 'bozo'
END

// Telnet Administration
BEGIN "Telnet"
// Domain        = "Inet6"                 // Inet4 oder Inet6, Default ist Inet4
   LocalPort     = "9999"                  // Port Deklaration
END

// Stream Quellen deklarieren
BEGIN "Inputs"
   local1       = "local"                 // Localer Input
   dvb          = "dvb"                   // Video Input für DVB Karte
END

// Input Konfiguration
BEGIN "local1"
  ConfigPath    = "/usr/local/etc/videolan/vls"
END

// Video Input (DVB) Konfiguration
BEGIN "dvb"
  DeviceNumber  = "0"                     // /dev/ost/dvr<i>

  Frequency = "12188000"                  // Frequenz in kHz des Transponders
  Polarization  = "1"                     // 0 oder 1
  SymbolRate    = "27500000"              // Symbol Rate

  DiSEqC    = "0"                         // DiSEqC Adresse des verwendeten LNBs
  LNB_Lof1  = "9750000"                   // Locale Frequez des 'Lower LNB band'
  LNB_Lof2  = "10600000"                  // Locale Frequenz des 'Upper LNB band'
  LNB_SLof  = "11700000"                  // Switch Frequenz des LNBs

  SendMethod   ="0"                       // 0 - Sendet alle Pids
                                          // 1 - Sendet nur MPEG2 Daten
END

// Kanal (Ausgang) deklarieren
BEGIN "Channels"
      localhost        = "network"
      client1          = "network"
      client2          = "network"
//  multicast      = "network"
//  localfile      = "file"
END

// Kanal Konfiguration
BEGIN "localhost"                         // Der Client ist auf dem selben Rechner wie Server
  DstHost = "127.0.0.1"
  DstPort = "1234"
END
```

```
BEGIN "client1"                          // Client1, Unicast
  DstHost = "141.24.205.107"             // IP-Adresse des Client-PCs im DVP
  DstPort = "1234"                       // Port
END

BEGIN "client2"                          // Client2, Unicast
  DstHost = "141.24.205.107"             // selbe IP-Adresse wie "client1"
  DstPort = "1235"                       // anderer Port
END


//BEGIN "multicast"                      // Multicast Beispiel
//  Type    = "multicast"
//  TTL     = "1"                        // Time To Live
//  DstHost = "239.2.12.42"              // Multicast Adresse
//  DstPort = "1234"                     // Destination Port
//END

//BEGIN "localfile"                      // Beispiel um Stream in eine Datei zu schreiben
//  FileName = "stream.ts"
//  Append   = "no"                      // Überschreiben falls schon vorhanden
//END

// Kommandos festlegen die beim Start des Server ausgeführt werden
// 'Commands' wie in der Telnet Eingabe.
//BEGIN "LaunchOnStartUp"
//  command1 = "start DLD s local1 --loop" // Beispiel Datei streamen
//  command2 = "start 12020 1 dvb"         // Beispiel DVB streamen
//END
```

A.2 input.cfg

```
// VLS Localer Input Konfiguration

// Haupteinstellung
BEGIN "Input"
  FilesPath = "/data/movies"       // Verzeichnis der Aufnahmedateien des VDR
  ProgramCount = "3"               // Anzahl der Recordings
END


// Hier ein Beispiel für eine Programmkonfiguration
// --- Format:
// BEGIN "program_number"
//   Name     = "program_name"
//   Type     = "type"             Kann Mpeg1-PS, Mpeg2-PS, Mpeg2-TS, oder Dvd sein
//   FileName = "path"             Lege hier Pfad fest, wenn dort eine Datei gestreamt werden soll
//   Device   = "device"           Lege hier Device fest, wenn DVD gestreamt werden soll
// END


BEGIN "1"
  Name     = "DLD"
  FileName = "DLDVDDEMO.VOB"
  Type     = "Mpeg2-TS"
END

BEGIN "2"
  Name     = "Fish"
  FileName = "fish.mpg"
  Type     = "Mpeg2-TS"
END

BEGIN "3"
  Name     = "MD"
  FileName = "mdintro.mpg"
  Type     = "Mpeg2-TS"
END

// Beispiel DVD
//BEGIN "4"
//  Name     = "film"
//  Device   = "/dev/cdrom"
//  Type     = "Dvd"
//END

// Beispiel DVD ausgelagert auf der Festplatte
//BEGIN "5"
//  Name     = "matrix"
//  Device   = "/mnt/data/matrix/VIDEO_TS"
//  Type     = "Dvd"
//END
```

Anhang B

Die Socket-Programmierung

Ein ausgesuchtes Quelltext-Beispiel des programmierten Plugins `vls`, um eine *Server*-Steuerung vom *Client* aus zu realisieren. Es wird somit die einzig vorhandene Steuerung (Telnet-Administration) des *VideoLanServers* umgangen.

socket.h:

```
#ifndef VDR_VLS_SOCKET_H
#define VDR_VLS_SOCKET_H

# include <stdio.h>
# include <stdlib.h>
# include <string.h>
# include <unistd.h>
# include <sys/types.h>
# include <sys/socket.h>
# include <netinet/in.h>
# include <netdb.h>
# include <arpa/inet.h>

#define PORTNUM 9999                // Portnummer des VideoLAN Servers
#define HOSTNAME "localhost"        // IP-Adresse (Server und Client(VDR) auf einem Rechner)

# define oops(msg) { perror(msg) ;  exit(1) ; }


int send_telnet_command(char* cmd);


#endif // VDR_VLS_SOCKET_H
```

socket.cpp:

```
# include "socket.h"

int send_telnet_command(char* cmd)
  {
       struct sockaddr_in bba ;
       struct hostent *hp ;

       int s, i=0 ;
       char buffer[200000], command[100];

       buffer[i] = command[i] = '\0';

// Erschaffen der Netzwerk Adresse:
       memset(&bba, 0 , sizeof(bba)) ;
       bba.sin_family = AF_INET ;       // benutze TCP/IP Protokol
       hp = gethostbyname(HOSTNAME) ;

       if (hp == NULL) oops ("no such computer");
       memcpy(&bba.sin_addr, hp->h_addr, hp->h_length) ;
       bba.sin_port = htons(PORTNUM) ;

// Zum Server verbinden:
       s = socket (AF_INET, SOCK_STREAM, 0) ;
       if (s == -1) oops ("socket") ;
       if (connect (s, (struct sockaddr*) &bba, sizeof(bba)) != 0) oops ("connect") ;

// Login- und Willkommen-String vom Server auslesen:
       i = read (s, buffer, 2000) ;
       buffer[i] = '\0' ;

// Login String senden:
       write(s, "bozo\r\n", 6);

// Password-Prompt vom Server abholen:
       i = read (s, buffer, 2000) ;
       buffer[i] = '\0' ;

// Password-String senden:
       write(s, "bozo\r\n", 6);

// Command Prompt vom Sever abholen:
       i = read (s, buffer, 2000) ;
       buffer[i] = '\0' ;

// Ausführbares Kommando zum Server schicken (zur Bereinigung):
       write(s, "stop client1\r\n", 14);

// Command Prompt vom Server abholen:
       i = read (s, buffer, 2000) ;
       buffer[i] = '\0' ;
```

```
// Eigentlicher Befehl-String zum Server schicken:
// Beispiel eines Strings: "start 12020 client1 dvb"
      write(s, command, sizeof(command));

// Prompt wieder abholen:
      i = read (s, buffer, 2000) ;
      buffer[i] = '\0' ;

// Logout-String senden
      write(s, "logout\r\n", 8);

// Logout-Bestätigung abholen:
      i = read (s, buffer, 2000) ;
      buffer[i] = '\0' ;

      return 0;

}
```

Abkürzungsverzeichnis

ADSL	Asymmetric Digital Subscriber Line
AP	Access Points
API	Application Programming Interface
ATM	Asynchronous Transfer Mode
AVT	Audio and Video Transport
CBR	Constant Bit Rate
DAVIC	Digital Audio Video Council
DCT	Discrete Cosine Transformation
DECT	Digital Enhancement Cordless Telecommunication
DHCP	Dynamic Host Configuration Protocol
DOCSIS	Data Over Cable Service Interface Specification
DSL	Digital Subscriber Line
DVB	Digital Video Broadcast
DVD	Digital Versatile Disc
DVP	Digitale Video Projekt
EMV	Elektromagnetische Verträglichkeit
EPG	Electronic Program Guide
ES	Elementary Stream
ETSI	European Telecommunication Standards Institute
FDM	Frequency Division Multiplexing
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISDN	Integrated Services Digital Network
ISM	Industrial Scientific Medical
LAN	Local Area Network
MAC	Media Access Control
MPEG	Motion Picture Expert Group

MPTS	Multiple Program Transport Stream
OSD	On Screen Display
NMS	Network Management System
OSI	Open System Interconnection
PCR	Program Clock Reference
PES	Packetized Elementary Stream
PID	Program Identifier
PLC	Powerline Communication
POTS	Plain Old Telephone System
PSI	Program Specific Information
PS	Program Stream
QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
QPSK	Quarternary Phase Shift Keying
RSVP	Resource Reservation Protocol
RAID	Redundant Array of Inexpensive Disks
RAM	Random Access Memory
RFC	Requests For Comments
RTCP	Realtime Transport Control Protocol
RTP	Realtime Transport Protocol
RTSP	Realtime Streaming Protocol
SI	Service Information
SCSI	Small Computer System Interface
Tcl	Tool command language
TK	Toolkit
TCP	Transmission Control Protocol
TS	Transport Stream
UDP	User Datagram Protocol
VBR	Variable Bit Rate
VDR	Video Disc Recorder
VDSL	Very high bit rate Digital Subscriber Line
VLC	VideoLan Client
VLS	VideoLan Server
VoD	Video on Demand
WLAN	Wireless Local Area Network

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] Klaus Schmiedinger. *Video Disc Recoder-Homepage.* 2002. http://www.cadsoft.de/people/kls/vdr.

[2] A.Reich M.Lenz. *Digitales Fernsehen in der Praxis.* Franzis, Feldkirchen, 2 edition, 1999.

[3] DVB Forum. *Standards and Specifications of DVB.* 2002. http://www.dvb.org/dvbtechnology/standspecs.html.

[4] A. Ziemer. *Digitales Fernsehen-Eine neue Dimension der Medienvielfalt.* R.v.Decker, Heidelberg, 1994.

[5] Ulrich Reimers. *Digitale Fernsehtechnik: Datenkompression und Übertragung für DVB.* Springer, Berlin, 2 edition, 1997.

[6] Ulrich Reimers. *Digital Video Broadcast (DVB). The International Standard for Digital Television.* Springer, Berlin, 2 edition, 2000.

[7] International standards organisation. information technology-genereic coding of moving pictures and associated audio information (mpeg2). *International Standard ISO/IEC*, IS 13818:ISO IEC 1, 1996.

[8] Implementation guidelines for the use of mpeg-2 systems, video and audio in satellite and cable broadcasting applications. *European Telecommunication Standards Institute*, ETSI Technical Report 154, 1994.

[9] Margarete Payer. *Computervermittelte Kommunikation: OSI-Open Systems Interconnection Model.* 1999. http://www.payer.de/cmc/cmcs03.htm.

[10] Ralf Steinmetz. *Multimedia Technologie.* Springer, Berlin, 3 edition, 2000.

[11] J.Garcia-Luna-Aceves F.Kuo, W.Effelsberg. *Multimedia Communications: Protocols and Applications.* Prentice Hall, USA, 1998.

[12] Emmanuel Chimi. *High-Speed Networking. Konzepte, Technologien, Standards*. Hanser, München, 1998.

[13] H. Gebhard K.Schröder. *IP-basierte Video-Kommunikation*. Veröffentlichung, Universität Dortmund, 1999.

[14] Reinhard Sojka. *IP Packet Delay Variation*. Master-Thesis, Mai, 2002.

[15] Holger Philipps. *Hausinterne Stromversorgungsnetze als Übertragungsweg für hochratige digitale Signale*. Dissertation, Technische Universität Braunschweig, 2001.

[16] Michael Kraemer. *Architektur und Technik kommunaler Telekommunikationsnetze*. Diplomarbeit, Universität Kaiserslautern, 1996.

[17] Siemens. *Attane Solution for Video over DSL*. 2002. http://w4.siemens.de.

[18] Stefan Hofmeir. Daten-highways für die letzte meile. *Funkschau*, 5, 1999.

[19] Jörg Brakensiek. *Ein Beitrag zu dezentralen, funkbasierten Inhouse-Netzen für breitbandige Kommunikation*. Dissertation, Universität Dortmund, 2000.

[20] IEEE. Standards for local and metropolian area networks-overview and architecture. *IEEE Standard*, 802, 1990.

[21] H. Gebhard K.Schröder. Audio-/video-streaming über ip. *FKT Heft 1+2*, ISSN 1430-9947:24–34, 2000.

[22] D. Towsley L.Gao. Supplying instantaneous video-on-demand services using controlled multicast. *Proceedings of IEEE*, Multimedia Computing Systems´99, 1999.

[23] D. Sitaram A.Dan. Scheduling policies for an on-demand video server with batching. *IBM Research Division*, T.J., 1994.

[24] M.Breitenbach M.Bossert. *Digitale Netze*. G.G. Teubner, Leipzig, 1999.

[25] Thomas Bauer. *Webbasiertes skalierbares Videoinformationssystem (WESVIN)*. Diplomarbeit, Technische Universität Ilmenau, 1999.

[26] Hartware.net. *RAID: Grundlagen und Systeme*. 2002. http://www.hartware.de/report.html?id=369.

[27] VideoLan. *OpenSource Video streaming solution for every OS.* 2002. http://www.videolan.org.

[28] RFC 2250. *RTP Payload Format for MPEG1/MPEG2 Video.* 2002. http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc2250.html.

[29] Richard Stevens. *Programmieren von UNIX-Netzwerken.* Hanser, München, 2 edition, 2000.

Erklärung

Hiermit erkläre ich, dass ich diese Arbeit selbständig durchgeführt und abgefasst habe. Quellen, Literatur und Hilfsmittel, die von mir benutzt wurden, sind als solche gekennzeichnet.

Ilmenau, den 12. Februar 2003

Thesen der Diplomarbeit

1. Für die Videodatenübertragung sind nicht alle Netzwerke tauglich.
2. Nicht alle Netzwerkprotokolle sind gleich gut für eine Videodatenübertragnung geeignet.
3. Die Übertragung eines Echtzeit-Videostreams über ein Ethernet-Netzwerk ist möglich.
4. Eine Videodatenverteilung wird anhand eines Client/Server-Systems in der Umgebung des *Digitalen Video Projektes* ermöglicht.
5. Für das *Digitale Video Projekt* sind ein Hin- und Rückkanal notwendig.
6. Mit dem Plugin `vls` kann eine Verteilung der Videodaten im *Digitalen Video Projekt* gesteuert werden.
7. Das Plugin `vls` bietet eine Funktionserweiterung der DVP-Systemsoftware VDR.
8. Die Softwarekomponente *VideoLAN* in Kombination mit dem Plugin `vls` hat sich für den Aufbau der Anwendung bewährt.
9. Durch die Applikation `VLS Steuerung` wurde das *Video-on-Demand*-Prinzip erfüllt.