ABSCHLUSSBERICHT

DFN Gigabit-Media-Dienste für kooperative Postproduktion von Film und Video (GigaMedia)

1.11.1999 – 31.10.2001

Projektleitung:

Dr. Ralf Schäfer, Heinrich-Hertz-Institut für Nachrichtentechnik Berlin GmbH
Einsteinufer 37, 10587 Berlin
Tel.: (030) 31002-560, Fax: (030) 3927200, E-Mail: schaefer@hhi.de

Beteiligte Einrichtungen:

das Werk
Oslerwaldstraße 10, 80805 München
Thomas Tannenberger
Tel. (089) 368148-0, E-Mail: tanne@das-werk.de

Fraunhofer Forschungsinstitut für offene Kommunikationssysteme (FOKUS)
Kaiserin-Augusta-Allee 31, 10589 Berlin
Dr. Georg Carle
Tel.: (030) 34637149, E-Mail: carle@fokus.gmd.de

Heinrich-Hertz Institut für Nachrichtentechnik Berlin GmbH (HHI)
Einsteinufer 37, 10587 Berlin
Dr. Ralf Schäfer
Tel.: (030) 31002-560, E-Mail: schaefer@hhi.de

Institut für Rundfunktechnik (IRT)
Floriansmühlstr. 60, 80939 München
Markus Berg
Tel. (089) 32399-279, E-Mail: berg@irt.de

Inhalt

1 Zusammenfassung

Ziel des GigaMedia-Projektes war die Untersuchung der Nutzung von Anwendungen zur kooperativen Postproduktion von Film und Video in der Gigabit-Testbed-Infrastruktur des DFN. Dazu wurden Anwendungs- und Netzinfrastruktur-Komponenten zur Unterstützung von Multimedia-Diensten mit selektierbaren Dienstqualitäten über das Netz realisiert und erprobt. Diese Komponenten wurden zum Aufbau einer verteilten Produktionsumgebung bei den Partnern *das Werk* (München), *Institut für Rundfunktechnik* (München) und *Heinrich-Hertz-Institut* (Berlin) eingesetzt. Von *FhI FOKUS* (Berlin) wurden auf die Anwendungsszenarien anpassbare Kommunikationsdienste zur Verfügung gestellt. Darüber hinaus wurde das Projekt bei der messtechnischen Auswertung unterstützt.

Das Anwendungsgebiet der kooperativen Postproduktion stellt sehr hohe Anforderungen an die Datenrate und die Qualität der Übertragung, da im Produktionsbereich vorwiegend ohne Komprimierung gearbeitet wird. Für die Übertragung von Videomaterial in TV-Auflösung im weit verbreiteten SDI-Format wird eine Datenrate von 270 Mbit/s benötigt, d.h. für die Übertragung eines Spielfilms fällt ein Datenvolumen von ca. 1,5 Tbit an. Verwendet wird beispielsweise eine D1-Kassette, sie kann bei einer Aufzeichnungsrate von ca. 40 MByte/s je nach Länge mehr als 30 GByte (12 min. Video) bzw. mehr als 150 GByte Daten (1 h Video) fassen. Bei HDTV-Auflösung (die für Filmqualität benötigt wird) versechsfacht sich das Datenaufkommen. Der Austausch von hochwertigem Videomaterial geschieht heute zumeist mittels Magnetbändern, die per Kurier versendet werden, wobei jedoch Latenzzeiten von mehreren Stunden bis zu mehreren Tagen eintreten.

Neben der hohen Datenrate und der damit verbundenen Echtzeitfähigkeit ist es vor allen Dingen auch die Möglichkeit, teure Geräteinfrastruktur dezentral zu nutzen, die eine Kopplung von Produktionsstandorten attraktiv macht. Typische Anwendungen sind z.B. die Nutzung von zentralen Filmabtastern, die verteilte Nachbearbeitung, die Komprimierung von Filmmaterial, das Fernsteuern von Aufzeichnungsgeräten sowie die effiziente und ökonomische Übertragung der dabei anfallenden Daten.

Stellvertretend für eine ganze Reihe von Möglichkeiten, hochratige Übertragungstechniken zur verteilten Produktion zu nutzen (*siehe Kapitel 3, „Konzepte und Anwendungen“*) wurde im Rahmen des Projektes in Zusammenarbeit mit dem Produktionshhaus *das Werk* ein beispielhafter Ablauf entworfen, gerätetechnisch aufgebaut und im Betrieb erprobt (*siehe Kapitel 8, „Demonstrator“*). Kernstück des Demonstrators bilden vom *IRT* entwickelte SDI-ATM-Adapter, welche eine Übertragung von unkomprimierten SDI-Videoströmen über ATM-Netzwerke ermöglichen (*siehe Kapitel 5, „SDI über ATM“*). Vom *HHI* wurde als exemplarisches Postproduktionstool ein off-line Video-Formatkonverter entwickelt, der in den Demonstrator integriert wurde (*siehe Kapitel 7, „Formatkonversion als verteiltes Postproduktions-Tool*“). Für eine effektive und ökonomische Nutzung der Übertragungsbandbreite wurde von *FOKUS* das Konzept des „adaptiven Streamings“ entwickelt (*siehe Kapitel 6, „Adaptives Streaming“*).

2 Anforderungen an die verteilte Postproduktion

2.1 Einleitung

Der zunehmende Einsatz Computer-basierter Tools in der Film- und Fernsehproduktion und der Programmverbreitung führt zu einem dramatischen Wechsel in den Arbeits-Workflows der beteiligten Produktionseinrichtungen. Dadurch wird auch die Verwendung digitaler (Hochgeschwindigkeits-)Netze zwischen den Standorten der Rundfunkanstalten und/oder Produktionshäuser steigen. Das kann dazu führen, dass Produktionszeiten verkürzt werden, was wiederum zu einer Kostenreduktion führt, aber nur, wenn die neue Infrastruktur (computer-basierte Produktionstools in Zusammenarbeit mit lokalen und Weitverkehrs-Computernetzen) zur Kostenreduktion beiträgt.

Die in den neuen Produktions- und Postproduktionsszenarien verwendeten Netze müssen nicht nur Audio/Video-Signale, sondern auch andere Dienste wie IP-Daten und Metadaten transportieren. Außerdem müssen sie zusätzliche Services wie Videokonferenzen, Fernsteuerung von Produktions-Einrichtungen etc. unterstützen.

2.2 Anforderungen

Speziell für die verteilte Produktion von Film und TV müssen die verwendeten Netze rigide Anforderungen erfüllen. Die höchsten Anforderungen, die vom Rundfunk an die Netztechnik gestellt werden, lassen sich aus der Übertragung von Echtzeit-Video/Audio-Daten ableiten. Dort sind neben geringen Laufzeiten und extrem kleinen Fehlerraten vor allem möglichst minimale Laufzeitschwankungen (Jitter/Wander) und Unterbrechungsfreiheit gefordert. Weiterhin muss das Netz zuverlässig verfügbar sein, Punkt-zu-Punkt und Punkt-zu-Multipunkt Verbindungen sowie Hoch-geschwindigkeits-Filetransfer zwischen Video/Audio Servern ermöglichen. Ein einfaches Management für das gesamte Rundfunknetz inklusive Adapter ist ebenfalls wichtig.

Zahlreiche Untersuchungen des *IRT* ergaben, dass ATM zur Zeit die einzige Technologie ist, die die harten Anforderungen der Rundfunk-Echtzeitanwendungen erfüllen kann. ATM garantiert die Einhaltung der geforderten Dienstgüte (Quality of Service, QoS) für die gesamte Dauer einer Verbindung, bietet eine geringe Laufzeit bei der Übertragung, führt nur zu geringem Jitter. ATM ermöglicht so Echtzeitübertragungen und File-Transfer über ein (Standard-Telekommunikations)-Medium und ist daher optimal für Rundfunkanwendungen geeignet. Aus diesem Grunde führte das *IRT* aufwendige Test- und Demonstrationsinstallationen durch (*siehe Kapitel 5*), die die Möglichkeit von Multiservice-Netzwerkplattformen (inklusive verteilter Produktion/Postproduktion) basierend auf ATM untersuchten und aufzeigten.

3 Konzepte und Anwendungen

In enger Zusammenarbeit mit *das Werk* wurde ein Produktionsszenario entwickelt, welches einen für die Postproduktion üblichen Arbeitsablauf darstellt. Auf Basis dieses Szenarios wurden für verschiedene Teilaufgaben Konzepte entwickelt, welche in Form von Einzeldemonstrationen aufgebaut und getestet wurden.

3.1 Produktionsszenario

Als typisches Produktionsszenario wird der Vorgang von der Filmabtastung des Rohmaterials bis zur Bereitstellung des geschnittenen Materials auf einem Videoserver beschrieben (*siehe Abb. 1*):

Das entwickelte Film-Negativ (35mm oder 16mm) wird zunächst vom Kopierwerk am Basisstandort angeliefert. Es erfolgt eine sogenannte Einlicht-Abtastung des Negativs auf Video (Positiv). Das Ziel dieses Arbeitsschrittes ist es das gesamte vorhandene Filmmaterial für den Video-Schnitt verfügbar zu machen. Ein Laufzeitenverhältnis zum Endprodukt von 10:1 bis zu 30:1 ist hierbei durchaus normal.

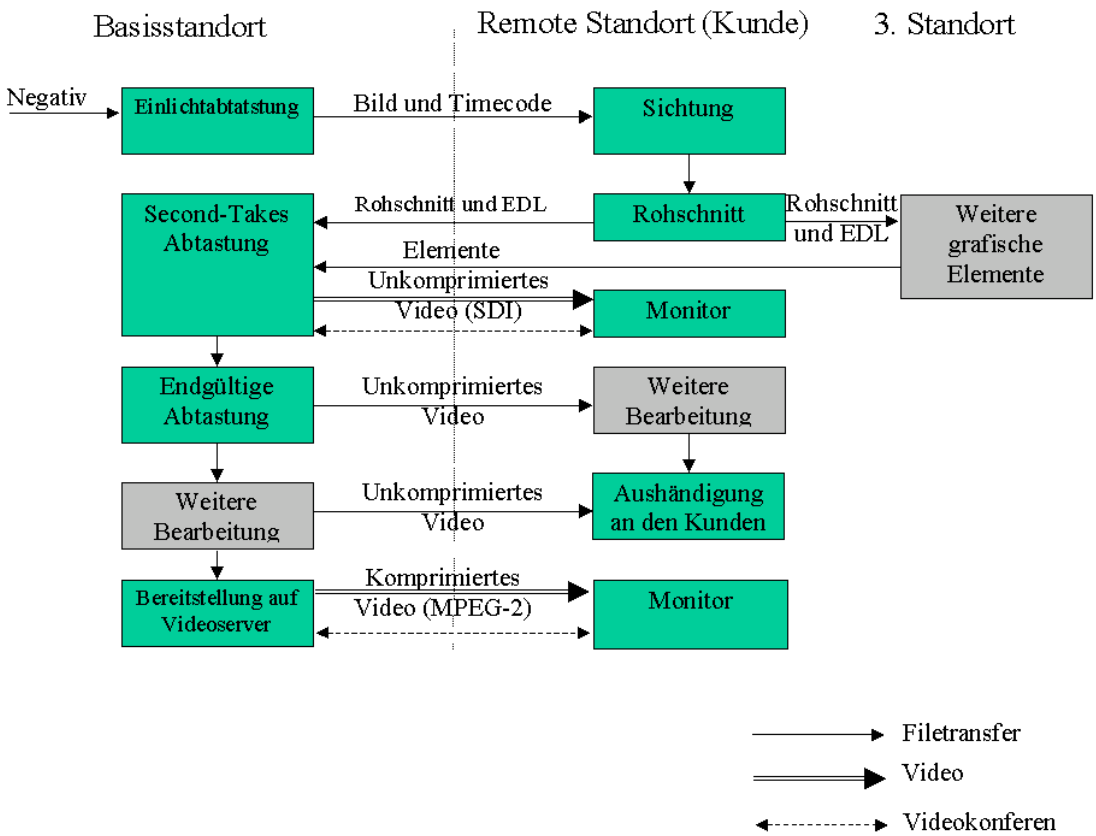

Abb. 1: Produktionsszenario

Im zweiten Schritt wird das abgetastete Videomaterial über die WAN Verbindung an den Remote-Standort übertragen. Eine Verkopplung von Bildinformation und Timecode ist für diese Aufgabe zwingend erforderlich. Eventuell könnte das Material noch im Basisstandort mit Hilfe eines kompatiblen Video-Schnittsystems erfasst werden und dann per Filetransfer als geschlossenes Projekt an den Remote-Standort übertragen werden. In diesem Arbeitsschritt würde man dann auch Tondaten, die beim

Dreh auf Film meistens auf DAT aufgezeichnet werden, bereits mit dem Bildmaterial synchronisieren und diese gemeinsam zum Remote-Standort übersenden. Das Videomaterial wird dann vom Kunden gesichtet und der Rohschnitt wird an einem Offline Editing-System erstellt. Das Ergebnis dieses Rohschnittes ist eine EDL, und eine Rohschnittversion als Videosequenz, die dann zurück an den Basisstandort übertragen wird. Eventuell werden anhand des Rohschnittes an einem dritten Standort zusätzliche grafische Elemente gefertigt und an den Basisstandort übertragen. Nun erfolgt anhand der EDL und des Rohschnittes (der zur Kontrolle auch am Basisstandort vorliegt) die Selected-Takes Abtastung. In diesem Arbeitsschritt werden nur die im Rohschnitt vorhandenen Abschnitte des Gesamt-Negativs auf Video abgetastet, wobei nun die endgültige Wirkung der Bilder (Bildeindruck) im Vordergrund steht. Der Kunde ist über Tele-Konferenz und SDI-ATM-SDI Verbindung "live" an diesem Vorgang beteiligt.

Die Übertragung der unkomprimierten Videodaten zum Kunden ist in diesem Falle sehr wichtig, da die Qualität des Ergebnisses ausschlaggebend ist. Nachdem der Bildeindruck festgelegt wurde, wird nun das Negativ endgültig auf Video abgetastet. Hierbei wird das Negativ Material vom Film-Abtaster über ein Farbkorrektur-System auf Video-Band übertragen. Das Video Material wird dann entweder am Basisstandort weiter bearbeitet (Kombination mit 3D Elementen, etc.) oder zur Weiterverarbeitung an den Remote-Standort übertragen.

In einem Compositing-System werden die einzeln gedrehten Sequenzen zu neuen kombiniert und an einem Online Editing-System die endgültige Sequenz erstellt. Der fertige Videoclip wird über die WAN Verbindung an den Remote-Standort übertragen und dem Kunden ausgehändigt. Dies erfolgt ohne Qualitätsverlust über den SDI-ATM Adapter.

3.2 Remote und Joint Viewing

Um Zwischenergebnisse bei der verteilten Film- und Fernsehproduktion zu überprüfen, muss das Material auf einem Videoserver abgelegt werden. Am Produktionsablauf beteiligte Personen wie Regisseure, Art-Directoren, Cutter, 3D-Animatoren an unterschiedlichen Standorten können das Material sichten und bewerten. In dem oben beschriebenen Produktionsablauf wird das fertige Material auf einen Videoserver übertragen, von dem das Material (im Falle eines Werbespots) Sendeanstalten, Filmproduktion und Kunden verfügbar gemacht werden kann. Zum Remote und Joint Viewing muss ein Videoserver bereitgestellt werden auf dem das Material abgelegt wird. Die am Produktionsablauf beteiligten Personen können das Videomaterial mittels eines Clients betrachten. Die Client-Software ermöglicht eine Steuerung des Abspielvorgangs.

3.3 Video Streaming und Remote Viewing

Beim Remote Viewing fordern mehrere Nutzer Videoströme von einem Videoserver an. Dieser „Video on Demand“-Dienst ermöglicht ein entferntes Abspielen mit Steuerungsfunktionen wie Anhalten, Abspielen, Zurück- und Vorspulen usw. Die Anwendung ist auf der Basis von internationalen Standards realisiert. Die Inhalte werden auf einem Server gespeichert und können auf einem entfernten Client über das Gigabit-Testbed unter Echtzeitbedingungen abgespielt werden. Die Anwendung erlaubt die Auswahl von verschiedenen Videoinhalten.

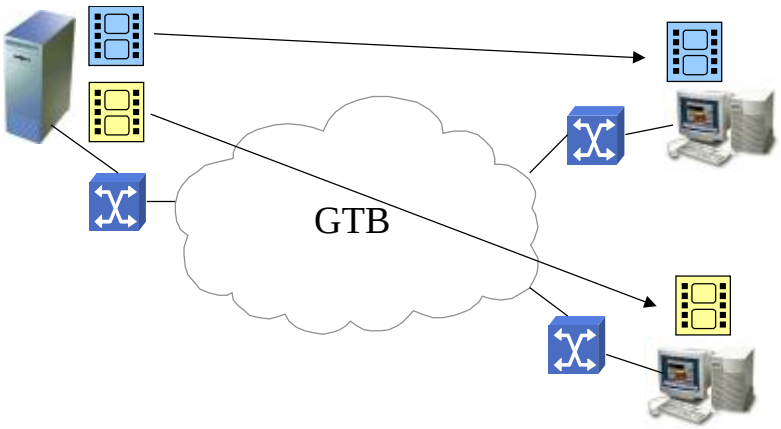

Abb. 2: Video Streaming and Remote Viewing Szenario

Abb. 2 zeigt das Demonstratorszenario für Remote Viewing. Bei *das Werk* wurde ein Video Server aufgebaut. Hierbei handelte es sich um ein Bildspeichersystem, welches direkt mit einem Filmabtaster gekoppelt war. Abgetastete Sequenzen konnten so direkt von den Remote Viewing Clients bei *FOKUS*, *IRT* und beim *HHI* abgerufen werden. Die Videosequenzen wurden über das Gigabit Testbed übertragen. Während der Übertragung wurden verschiedene Messinstrumente eingesetzt, um die Qualität der Übertragung zu bestimmen.

3.4 Video Server und Joint Viewing

Beim Joint Viewing Szenario betrachten zwei oder mehr Personen an verschiedenen Orten die gleiche Videoübertragung. Dabei kann die Steuerung des Abspielvorgangs entweder zentral (einer der Teilnehmer kontrolliert die Übertragung) oder dezentral (jeder kann die Übertragung kontrollieren) erfolgen. In *Abb. 3* wird das exemplarische Szenario für Joint Viewing dargestellt. Bei *FOKUS* wurde ein Videoserver aufgebaut. Das Videomaterial wurde mittels eines Remote Viewing Clients bei *FOKUS* und *HHI* vom Server abgespielt. Der Abspielvorgang wurde hierbei vom *FOKUS* Video Client aus gesteuert. Zwischen *FOKUS* und *HHI* lief während des Abspielvorgangs eine Videokonferenz, bei der über das Videomaterial diskutiert wurde.

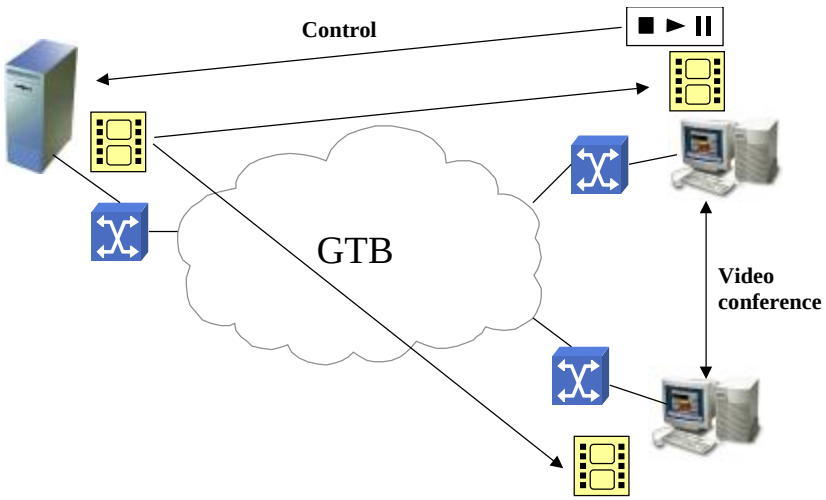

Abb. 3: Video-Server and Joint Viewing Szenario

4 Netz- und Geräte-Infrastruktur

Diese Kapitel beschreibt den Aufbau der Projektinfrastruktur von GigaMedia. Damit in engem Zusammenhang steht auch die Planung und Durchführung von Tests und Messungen in dieser Infrastruktur. Dabei handelt es sich in der Aufbauphase um Funktionstest zum Aufbau der Infrastruktur, der Schwerpunkt liegt aber während der Projektlaufzeit auf der unterstützenden Messung der Netzperformance und dem Nachweis der Funktionsfähigkeit und Leistungsfähigkeit der im Projekt GigaMedia entwickelten Ansätze.

4.1 Das GigaMedia-Netz im Gigabit-Testbed

Die vier Standorte der Projektpartner des Projekts GigaMedia sind über die das Gigabit-Testbed (GTB) des DFN miteinander verbunden gewesen. Je zwei Partner befinden sich in Berlin und München. Die Weitverkehrsverbindung erfolgte über das GTB, in den Städten wurden die lokalen Wissenschaftsnetze verwendet.

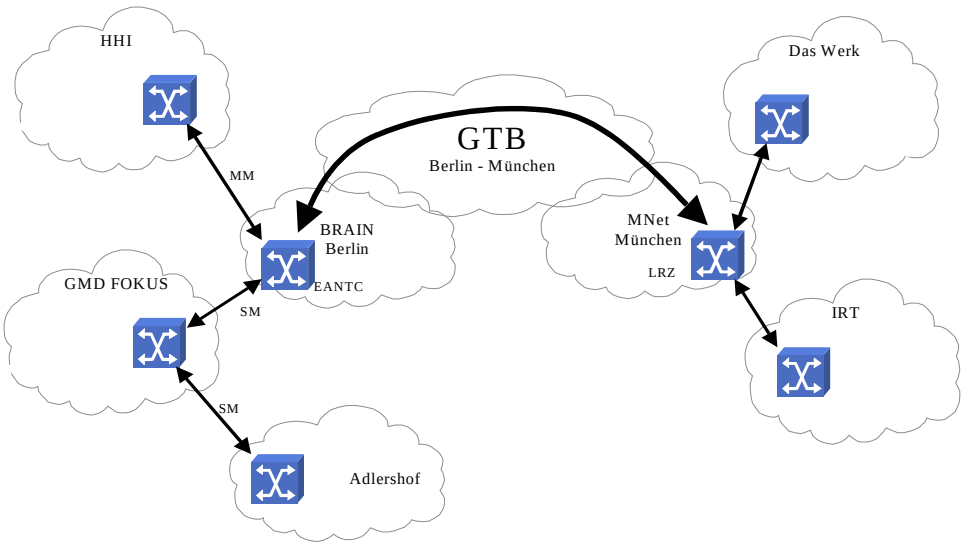

Abb. 4: Die GigaMedia-Standorte im Gigabit-Testbed (GTB) des DFN

An den Standorten der Projektpartner mussten verschiedene Endsysteme angeschlossen werden, die hier nach Standard-Endsystemen und Spezialendsystemen aufgeteilt sind:

Standard-Endsysteme

- Videokonferenz (PC / Win NT, SUN / Solaris)
- Videostreaming (PC / Win2000, Linux)

Spezialsysteme

- SDI/ATM-Adapter zur hochratigen Videoübertragung
- Ferngesteuerte Koppelfelder und MAZ
- Messgeräte zur Leistungsmessung

Die Infrastruktur im Projekt GigaMedia basiert auf Marconi (FORE) Switches. Das GTB basierte auf ATM mit PNNI, als Protokoll wurde in GigaMedia CLIP (Classical IP over ATM) verwendet. Etwa zehn Standard-Endsysteme in Form von PCs oder Workstations waren neben zwei SDI/ATM Adaptern im Einsatz. Zusätzlich wurde das GigaMedia-Netz mit vier Testsystemen instrumentiert.

Die folgende Abbildung zeigt die grundlegende Struktur des Projektaufbaus im Anschlussbereich der Partner. Dargestellt ist die beispielhafte Struktur für eine Applikation, es soll keine technische Realisierung bei einzelnen Projektpartnern dargestellt werden.

Jeder Standort ist mit dem gleichen Videokonferenz-System auf einen Steuer-PC ausgestattet. Das Videokonferenz-System wird um die verteilte GigaMedia-Applikation (also grundsätzlich die verteilte Steuerung von Video-Equipment) erweitert. Damit übernimmt das Videokonferenzsystem (z.B. Netmeeting von Microsoft) die Verwaltung der Konferenzschaltung, die eigentliche GigaMedia-Applikationen werden über Webserver gesteuert und können diese Infrastruktur nutzen.

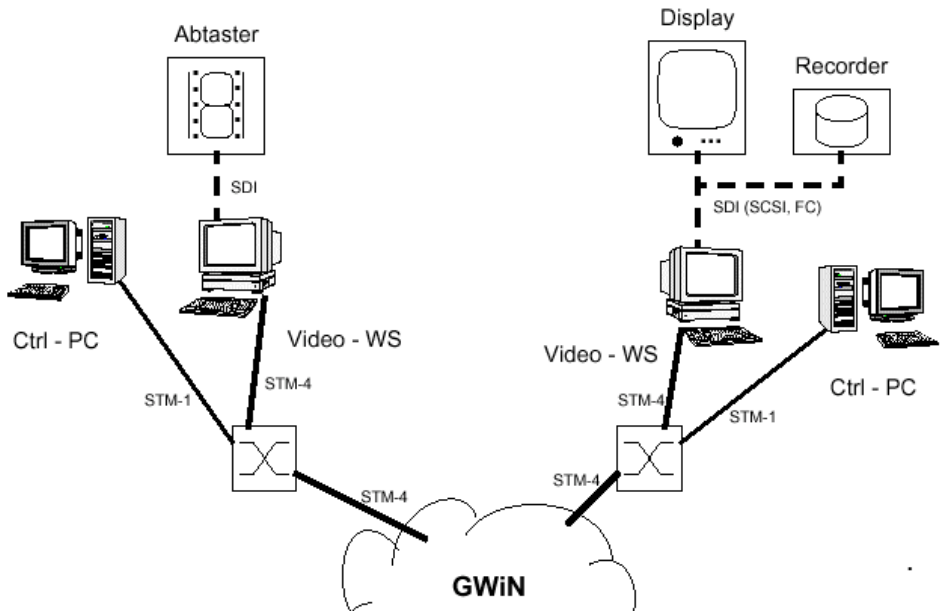

Abb. 5: Infrastruktur der Projektpartner in GigaMedia

Zusätzlich kann ein Group Server für die Verwaltung gemeinsamer Daten (Kalender für die Ressourcenverwaltung) in einem Produktionsszenario genutzt werden.

Die Standorte am WiN des DFN werden über einen Switch angeschlossen. Zur Reduzierung der Anschlusskosten bei einzelnen Partnern, die bisher kein ATM hatten (in GigaMedia der Partner *das Werk*), konnte auch ein bestehender Anschluss-Switch über eine Dark-Fiber-Verbindung mitgenutzt werden.

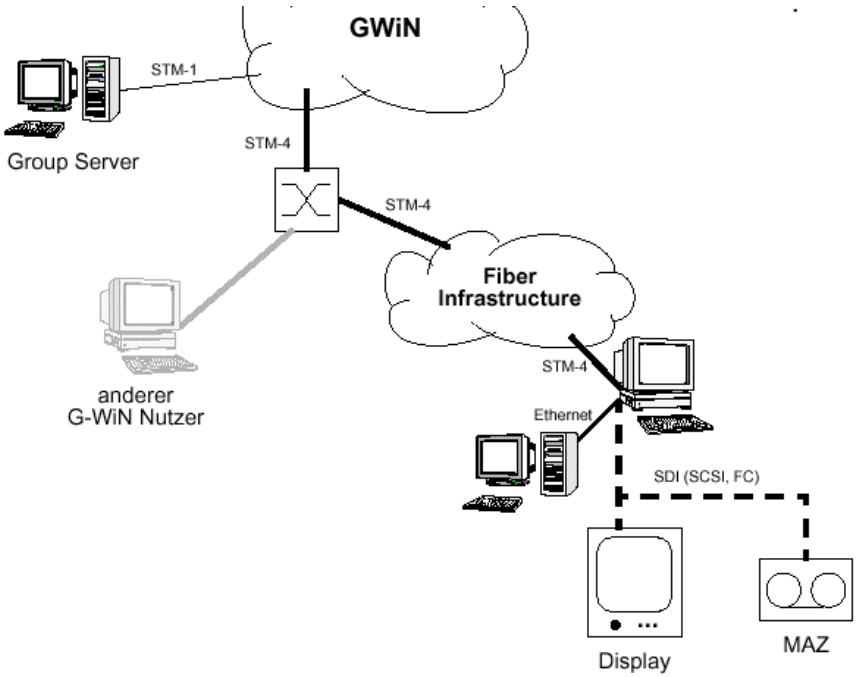

Abb. 6: Anschluss eines Projektpartners über Dark-Fiber

Die obigen Darstellungen stellen nur die prinzipielle Anbindung der GigaMedia-Applikationen dar. Natürlich wurde die lokale Infrastruktur bei den Projektpartnern mit für GigaMedia berücksichtigt oder genutzt. So verfügten *FOKUS* und *IRT* bereits über ATM-Infrastrukturen an ihren Standorten.

4.2 Messungen in GigaMedia

Zur Unterstützung des GigaMedia-Projekts wurden verschieden Messungen durchgeführt und eine einfache verteilte Messplattform aufgebaut. Ziele dieser Arbeiten waren dabei:

- Unterstützung beim Aufbau des Testnetzes
- Gewinnung von Verkehrsprofilen
- Darstellung der möglichen QoS im Netz

Während der Aufbau des Testnetzes mit den Messgeräten und den Erfahrungen vom *IRT* und von *FOKUS* durchgeführt wurde, standen bei dem Arbeitspaket der Messunterstützung zwei wesentliche Punkte im Vordergrund:

- Aufbau von Lastgeneratoren

  Die Steuerung der QoS über Konfiguration des Netzes oder über Signalisierung sowie der Mechanismus des adaptive Streaming läßt sich nur unter bestimmten Bedingungen im Netz zeigen. Wird der Verkehr in verschieden QoS-Klassen eingeteilt und diese Klassen im Netz verschieden priorisiert, so läßt sich die Funktion diese Mechanismus nur in einem belasteten Netz zeigen. Ist das Netz

nicht sehr stark belastet, so werden alle Pakete übertragen, egal welcher Klasse sie angehören.

Zum Einsatz kamen sowohl eigene Systeme von *FOKUS* als auch andere Systeme, wie z.B. der Smartbits-Lastgenerator oder freie Software. Entscheidend war neben der adäquaten Generierung des Verkehrs die Möglichkeit der Steuerung des Systems während der Messungen.

- Aufbau einer verteilten Testplattform
  Um in einem komplexen Szenario wiederholbare Entwicklungs-, Tests- und Evaluationsergebnisse zu bekommen, wurde eine steuerbare verteilte Messplattform aufgebaut. In der Plattform wurden die Instrumente der obigen beiden Arbeitspunkte zusammengefasst und unter eine gemeinsame Kontrolle gestellt.

  Die folgende Abbildung zeigt eine typische Konfiguration schematisch: Verschiedene Klienten(C) rufen von einem Server (S) Videosequenzen ab. Für die Messung wird dabei nur ein Videostrom betrachtet (zu C_1), die Videoströme der anderen Klienten konkurrieren als Hintergrundströme um Ressourcen des Servers (S). Im Netz konkurriert der Videostrom mit anderen Datenströmen, die durch Lastgeneratoren (L) erzeugt werden. Über einen Monitor (M) kann der Transportstrom des Videos aufgezeichnet und beurteilt werden.

  Die Aufgabe der verteilten Mess- und Testplattform ist es, vor und während der Messung in kontrollierter Weise komplexe Bedingungen einstellen zu können. D.h. das sich während des Abspielens eines Videos z.B. die Last ändern soll, um Effekte auf die Qualität oder die Protokollmechanismen sichtbar zu machen.

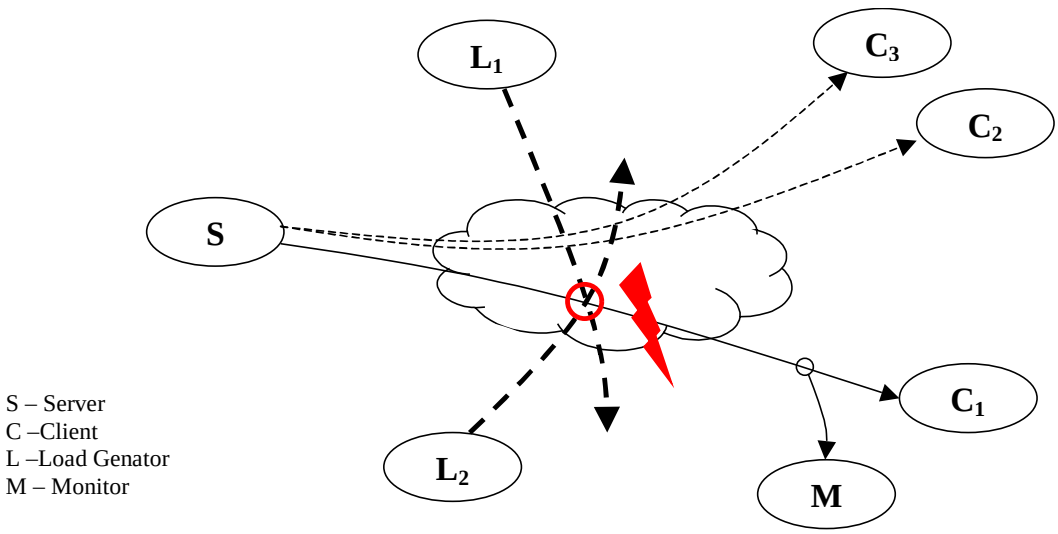

Abb. 7: Schematische Szenario für QoS-Messungen

4.3 Messwerkzeug MediaProbes

Ausgehend von dem oben dargestellten Szenario für QoS-Messungen wird in diesem Abschnitt die Architektur und die ersten Implementierungen dieser Plattform, genannt MediaProbes, vorgestellt.

Die Architektur des verteilten Testsystems MediaProbes beruht auf einer Reihe von Vorüberlegungen, um im Projekt eine möglichst einfache Umsetzung zu erreichen. Das Projekt GigaMedia hatte nicht die Entwicklung eines Testtools zum Ziel, sondern die Entwicklung und Evaluation von Applikationen. Dementsprechend einfach sollten unterstützende Arbeiten ausfallen. Auch werden Details der Implementierung vorgestellt, die nützlich für die Änderung oder Erweiterung des Testsystems durch Partner sein können. Das Tool selbst wird von *FOKUS* über eine Webseite mit detaillierten Beschreibungen zur Installation vorgehalten. Für die reine Nutzung des Tools sind die hier dargestellten Details nicht nötig.

Grundsätzlich muss das Tool so **einfach** wie möglich aufgebaut sein, da Tests und Messungen nur unterstützende Funktion im Projekt GigaMedia haben. Das Ziel war sowohl die Unterstützung von Applikationsentwicklern (z.B. durch eine einfache Möglichkeit, auf entfernte Ressourcen zuzugreifen) wie auch die Durchführung von Tests und Messungen in reproduzierbarer Form. Zu beachten war dabei, dass die Anwendungen in einer heterogenen Umgebung laufen werden, d.h. es kamen sowohl PCs (für die Videokonferenz-Anwendung) als auch verschiedene Workstations (für Videoserver oder Schnittsysteme) zum Einsatz. Mehr noch, es sollte auch für neue Systeme ein Migrationspfad in die verteilte Testumgebung existieren. Dabei ging man davon aus, dass das Testsystem parallel zu den Endsystemen des GigaMedia-Projekts installiert wird. Um die Kosten zu senken, sollte die Testplattform direkt auf diesen Endsystemen laufen. Weiterhin sollte das Testsystem sowohl einfach mit den Applikationen auf diesen Endsystemen verknüpft werden können, als auch die Möglichkeit bieten, Standart-Testsoftware (wie z.B. netperf) zu verwenden. Gerade die letzte Anforderung sparte viel Zeit und führte zu reproduzierbaren und vergleichbaren Ergebnissen.

Ein weiterer praktischer Aspekt war, dass die Projektarbeit mit einem Testwerkzeug **sichtbar** gemacht werden kann. Viele interne Mechanismen in Applikationen lassen sich nicht direkt beobachten oder treten nur unter bestimmten Voraussetzungen auf, z.B. bei Überlast im Netz. Das Testtool sollte auch in Demonstrationen eingesetzt werden können, um diese Mechanismen sichtbar zu machen oder die verschieden Netzzustände bei der Nutzung einer Applikation einzustellen. Auch hierbei war es nötig, dass vorhandene Software und die Applikationen (so sie denn ohne graphische Oberflächen steuerbar sind) in dem Testtool als eine Art Framework eingebaut werden konnten.

Auch sollte das System **fair** sein. Mit dem Testsystem wurde ein System aufgebaut, dessen Test- und Messpunkte (PCO - points of control and observation) über die ganze Infrastruktur verteilt sind und damit auch auf Rechnern **aller** Projektpartner and den vier Standorten. Neben Sicherheitsaspekten, die noch weiter unten angesprochen werden, sollte grundsätzlich allen Projektpartnern der gleiche Zugriff auf diese Infrastruktur ermöglicht werden.

4.3.1 Architektur

Wie schon angesprochen, sollte das Testsystem parallel zu den anderen Anwendungen auf allen Endsystemen des Projekts GigaMedia laufen. Damit waren dann alle verteilten Tests möglich und es können Aussagen über die Ende-zu-Ende-Qualität gemacht werden, die ein wichtiger Parameter ist. Eine der einfachsten Möglichkeiten, ein Endsystem mit zusätzlicher Funktionalität im Bereich der Datenübertragung auszustatten, ist die Verwendung von Web-Technologie. Ein Web-Browser als Client-Komponente gehört auf allen Endsystemen zur Grundausstattung und ist meist schon installiert. Als Server-Komponente stehen Web-Server für alle Betriebsysteme auch als freie Software zur Verfügung. Das liegt u.a. auch daran, dass ein Web-Server nicht sehr kompliziert aufgebaut sein muss. Eine wichtige Funktion, die in den verwendeten Web-Servern aber vorhanden sein muss, ist die Möglichkeit, externe Programme auszuführen. Beispiele für diese Mechanismen sind CGI und PHP. Diese Möglichkeit der Ausführung von externen Programmen kann genutzt werden, um eigene Programme einzubinden oder auch nur Wrapper zu schreiben, die Standard-Programme (wie z.B. ping oder netperf) aufrufen.

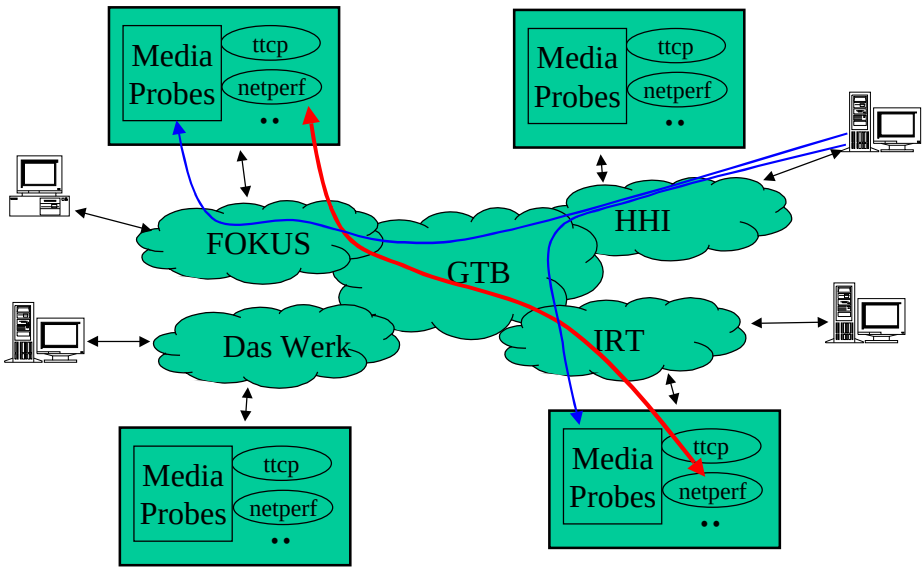

Abb. 8: Architektur von MediaProbes und Einsatzszenario

Die Abbildung zeigt das Prinzip. Über eine Web-Seite werden einzelne Server angesprochen, die dann ihrerseits jeweils eine Funktion starten. Also z.B. wird auf einem Web-Server ein Sender auf einem anderen Web-Server ein Empfänger zu Testzwecken gestartet. Eine Messung wird durchgeführt und das Ergebnis auf weiteren Web-Seiten präsentiert.

Die Vorteile dieser Art der Implementierung liegen auf der Hand: Das System ist einfach zu benutzen und einfach zu implementieren. Auch hat man von vornherein strukturierte Ergebnisse in einer Art graphischen Oberfläche. Sehr wichtig ist auch die Plattformunabhängigkeit der Implementierung, die natürlich an ihre Grenzen stößt, wenn externe Komponenten genutzt werden sollen. Insgesamt wird mit der Web-Technologie der Aspekt der Verteilung und der Kontrolle von entfernten

Komponenten wie von einem Framework übernommen und kann sofort genutzt werden. Andere Mechanismen, wie die Programmierung einer Testumgebung in Java, erfordern mehr Aufwand in der Programmierung und mehr Kenntnisse bei der Installation auf den Endsystemen. Gerade das Installieren auf den Endsystemen sollte von jedem Partner getrennt vorgenommen werden und erfordert somit eine getrennte Einarbeitung, die bei komplexen Vorgängen nicht mehr tragbar wird.

Die Nachteile einer solchen Implementierung sind hauptsächlich in der Einfachheit des Systems begründet. Aufgrund der einfachen Mechanismen fehlt eine direkte Kontrolle der entfernten Komponente bei der Fernsteuerung. Es kann nur eine entfernte Funktion parametrisiert und gestartet werden, eine direkte Kontrolle während der Laufzeit ist nicht vorgesehen. Hierbei kann es zu Problemen mit Statusmeldungen kommen oder das Abbrechen von falschen Funktionsaufrufen kann nicht gewährleistet werden. Da es sich aber um ein einfaches System für Experten handelt, kann man mit diesen Nachteilen leben. Wichtiger ist der Aspekt der Sicherheit, auf den hier kurz eingegangen werden soll.

Die Sicherheitsaspekte resultieren aus den gegensätzlichen Anforderungen, die an die GigaMedia-Infrastruktur gestellt werden. Einerseits handelt es sich um Projektinfrastruktur, auf die jedes Projektmitglied zugreifen können soll. Anderseits handelt es sich zumindest teilweise um Rechner in den Netzen von verschiedenen Organisationen und um Rechner, die in Produktionsumgebungen eingebunden sind. Zudem sind CGI-Applikationen auf einem Web-Server ein potentielles Sicherheitsloch. Gelöst werden kann das Problem der Sicherheit nur durch ein Reihe von Maßnahmen. Einerseits sollte ein bekannter und gut dokumentierter Web-Server genutzt werden. Es kann aber auch ein Web-Server genutzt werden, der von dem beteiligten Projektpartner bevorzugt wird und z.B. von einer zentralen Systemadministration gemanagt wird. Ein auf diese Weise ausgesuchter Server kann jeweils auf den neusten Stand der Sicherheit gebracht werden. Der Server selbst sollte so konfiguriert werden, dass er nur aus dem GTB erreichbar ist. Weitere Konfigurationen sind für Zugriffsrechte und Rechte für die Ausführung von CGIs nötig. Diese Konfigurationshinweise sind Teil der Installationsanleitung von *FOKUS*.

Da CGIs inhärent unsicher sind (es können beliebige Funktionen innerhalb des Programms ausgeführt werden, bei Programmierfehlern können sogar unerwünschte Funktionen von einem Benutzer ausgeführt werden), hilft nur die Auslieferung dieser Programme in Source-Code, die dann bei den Projektpartnern compiliert werden. Anhand des Source-Codes kann der Projektpartner entscheiden, ob er das Programm laufen lassen will oder nicht. Programme, die aus einer dritten, als sicher erachteten Quelle stammen (z.B. netperf) können natürlich auch als Binärdatei eingespielt werden. Die Entscheidung liegt bei dem jeweiligen Projektpartner.

4.3.2 Implementierung

Als Web-Server wurde Apache1.3.12 ausgewählt. Dieser Server ist frei erhältlich (auch für Windows in einer stabilen Beta-Version) und soweit verbreitet, dass es genügend Sicherheitshinweise und Sekundärliteratur gibt. Für die Implementierung von externen Funktionen wurde auf den CGI-Mechanismus zurückgegriffen, der im Server enthalten ist. PHP bietet Vorteile in der Leistungsfähigkeit, muss jedoch extra installiert werden, was in GigaMedia zu aufwändig war.

Die externen Programme werden mit GNU GCC compiliert. Dieser Compiler ist für fast alle Plattformen erhältlich, für Windows im Paket CYGWIN inklusive einer UNIX-ähnlichen Ausführungsumgebung. Durch diese Wahl muss in den Sourcen nur zwischen Standard-Unix und dem recht ähnlichen CYGWIN unterschieden werden, was auf einfache Weise die Programmierung portabler Programme ermöglicht.

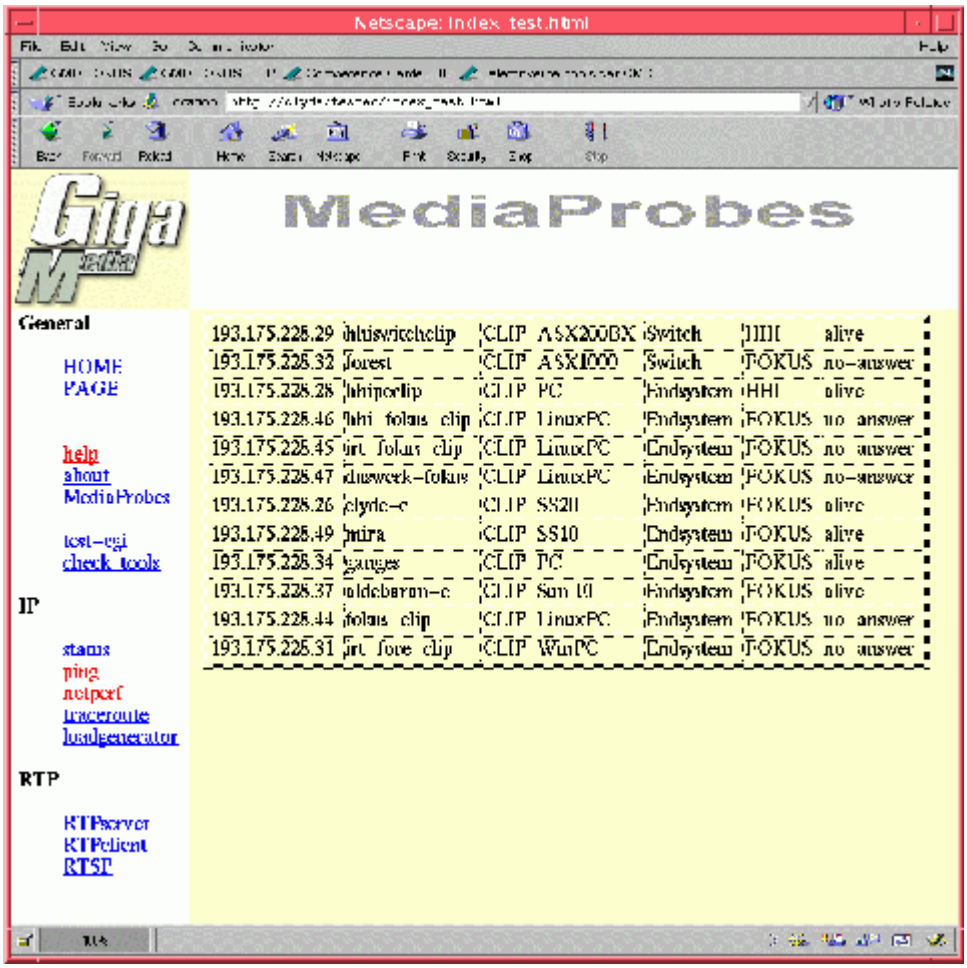

Abb. 9: Darstellung der GigaMedia-Infrastruktur in MediaProbes

Eine Besonderheit ist bei der Nutzung von Servern (z.B. netserver, der Server zu netperf) zu beachten. Hier gibt es eine Reihe von möglichen Optionen, den Server zu starten. Einerseits kann ein Server als Daemon bereits auf der Maschine vorliegen und wird z.B. vom Internet-Daemon gestartet. Das ist z.B. bei vielen Maschinen von *FOKUS* der Fall, da die *FOKUS*-Maschinen im TAM-Labor fast alle netserver installiert haben. In diesem Fall braucht der Server nicht über das Testsystem MediaProbes gestartet werden. Ein Server kann aber auch als normales User-Programm gestartet werden, in diesem Fall kann es aber passieren, das sich das Programm des Servers nach dem Abbau einer Verbindung beendet. In diesem Fall muss der Benutzer den Server über das Testsystem MediaProbes für jede Messung erneut starten. Der Fall des Daemons ist natürlich bequemer, erfordert aber eine Umkonfiguration des Endsystems mit Root-Rechten. Der Fall des User-Programms ist einfacher zu installieren und zu starten (keine Root-Rechte erforderlich, wenn der

benutzte Port > 1024 ist). Auch kann man auf diese Weise den gleichen Port mit verschiedenen Servern nacheinander nutzen. Das kann Vorteile haben, wenn man verschiedene Serverimplementierungen auf dem „well-known„ Port für diese Applikation laufen lassen will, z.B. weil die Umkonfiguration des Ports nicht funktioniert oder ein Testsystem den „well-known„ Port erwartet.

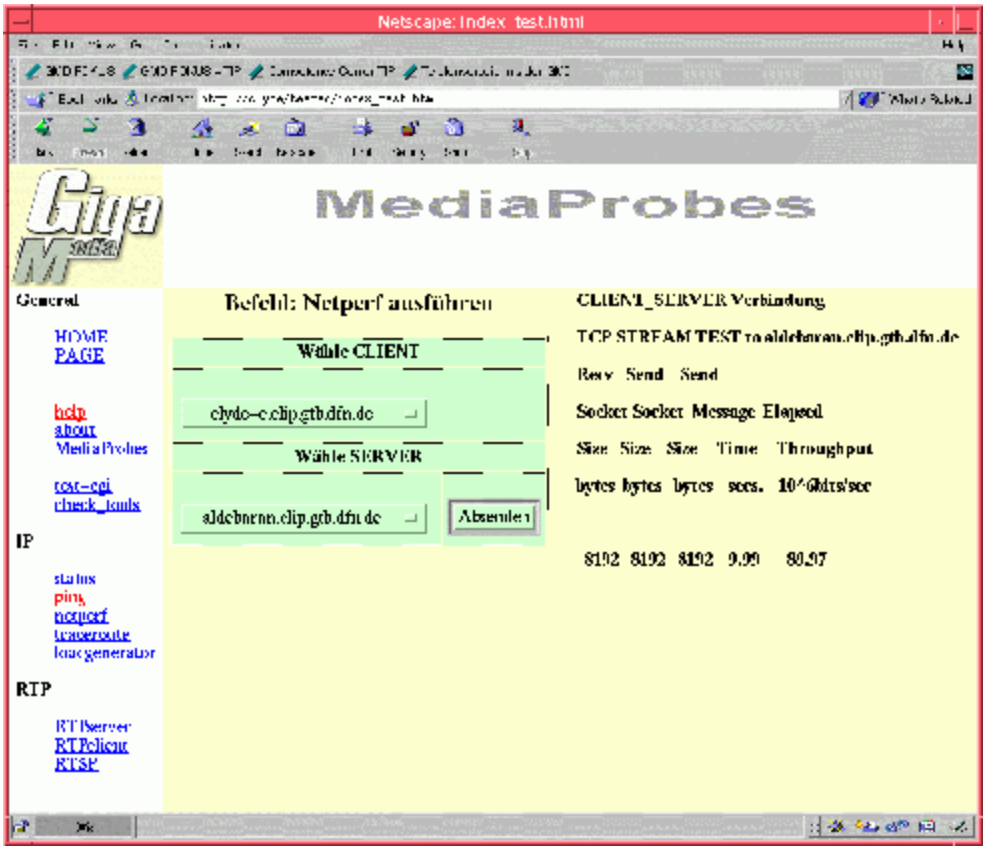

Abb. 10: Ausführung von ´netperf´unter GigaMedia

Die Testsoftware von *FOKUS* ist in drei Distributionen aufgeteilt:

- Original Apache Web Server
- Die Web-Seiten und Skripte für MediaProbes (gmed-tip-xx.zip)
- Zusätzliche Tools aus anderen Quellen (z.B. netperf)

Diese Aufteilung ermöglicht eine logische Trennung der Komponenten auf verschiedenen Plattformen. Die Apache-Distribution ist plattformabhängig und kann durch einen eigenen Web-Server des Projektpartners ersetzt werden. Die eigentliche MediaProbes-Distribution ist dagegen plattformunabhängig und der eigentliche Kern des Systems, d.h. hier ist die gesamte Logik des Test- und Messsytems MediaProbes zu finden. Das Testsystem wiederum stützt sich auf externe Messprogramme, die in einer eigenen Distribution ausgeliefert werden. Diese Distribution ist wiederum plattform-abhängig (insbesondere kann nicht jeder Nutzer Sourcen unter Windows compilieren) und wurde aus Gründen der Einfachheit von *FOKUS* zusammengestellt.

5 SDI über ATM

5.1 Einleitung

Die direkte Kopplung von Produktionsbetrieben über 270-Mbit/s-DSK-Technik (DSK = Digitale Serielle Komponenten, Übertragung über Schnittstelle SDI = Serielles Digitales Interface) gewinnt zunehmend an Bedeutung. Ein wesentlicher Grund ist neben der uneingeschränkten Bild- und Tonqualität (embedded audio) die **geringe Signalverzögerung** bei der Echtzeitübertragung. Besondere Sendeformen (z.B. Diskussionsrunden im Studio mit zugeschalteten Teilnehmern in weit entfernten Studios) sowie die vernetzte Produktion, bei der Echtzeit-Interaktionen über weite Entfernungen eine gewichtige Rolle spielen, können nur sehr eingeschränkt eingesetzt werden, sobald nennenswerte Signalverzögerungen auftreten. Dies ist z.B. der Fall bei bitratenreduzierenden Codecs, die typische Verarbeitungszeiten von 200 ms bis zu 800 ms aufweisen.

Dank des zur Zeit rapide zunehmenden Ausbaus von optischen Netzen mit extrem hohen Übertragungskapazitäten im Wettbewerb der Netzbetreiber sowohl in Deutschland als auch in Europa, wird die direkte unkomprimierte Kopplung von Produktionsbetrieben zu ökonomischen Bedingungen in nächster Zukunft Realität werden. Andererseits werden die verfügbaren Bitraten im Gigabitbereich für lokale Netze ebenfalls die unkomprimierte Übertragung von TV-Signalen bei hohen Qualitätsanforderungen z.B. im Klinikbereich (Telemedizin) fördern. Erste Anfragen für derartige Anwendungen liegen bereits vor.

Die vom *IRT* europa- und weltweit vorgestellten Entwicklungsszenarien für die Vernetzung von Produktionsbetrieben sowie die auf nationalen und internationalen Messen und in Zusammenarbeit mit deutschen und europäischen Rundfunkanstalten (z.B. Hessischer Rundfunk - ARD Sternpunkte und Europäische Rundfunkunion in Genf – EBU) gezeigten Demonstrationen der im Rahmen des GigaMedia-Projektes untersuchten Technik haben vielfältige Reaktionen von Produktionsbetrieben und von Firmen bezüglich des möglichen Einsatzes dieser Techniken hervorgerufen.

Dazu mussten Applikationen entwickelt und Netzinfrastrukturen aufgebaut und untersucht werden, die Multimedia-Anwendungen mit wählbarer Dienstgüte (QoS, Quality of Service) unterstützen. Das *IRT* hat während der Dauer des GigaMedia-Projekts einige solcher Feldtests durchgeführt, wobei insbesondere darauf geachtet wurde, dass die strengen Anforderungen, die Rundfunkapplikationen an Netzwerke richten, eingehalten wurden.

Die im Folgenden aufgeführten Untersuchungen wurden auf ATM-Netzen durchgeführt, da ATM zur Zeit die einzige Technologie ist, die die harten Anforderungen von Rundfunk-Echtzeitanwendungen erfüllen kann.

Durch flexible ATM-Netzzugänge können neben der Videoübertragung auch andere Dienste wie Filetransfer, Video-Konferenz etc. über die gleiche Strecke genutzt werden. Die Wirtschaftlichkeit der TV/Filmproduktion kann so erhöht werden, da nicht mehr in jedem Studio alle u.U. investitions- und personalintensiven technischen Einrichtungen vorgehalten werden müssen, das Produktionsequipment kann auf die einzelnen Standorte verteilt aufgebaut sein und gemeinsam genutzt werden.

5.2 Technologie

Der in den Tests und Vorführungen eingesetzte SDI/ATM-Adapter überträgt die Signale der im digitalen Studio üblichen Standard-Schnittstelle SDI (Serial Digital Interface) transparent über ein ATM Netz. Dabei können Signale mit uneingeschränkter Bild- und Tonqualität über 270-MBit/s-DSK-Technik zwischen Produktionsbetrieben oder Außenstellen transportiert werden.

Die technischen Eigenschaften des Adapters lauten:

- Übertragung eines SDI Signals über ATM-Netze mit Standard-ATM-Technologie (quasi AAL1)
- Höchste Qualität und minimale Verzögerung, da keine Video- oder Audio-Kompression eingesetzt wird (Laufzeit über 2 Geräte <300µs)
- Mapping eines SDI-Signals (270 MBit/s) in Standard-ATM-Technologie (STM-4, 622MBit/s)
- Transparente Übertragung eines SDI-Signals
- Ideal geeignet für verteilte Videoproduktion
- Sehr gut geeignet für hochauflösende Telemedizin-Anwendungen (hohe Farbtiefe) sowie für Telerobotik-Anwendungen (geringe Signal-Verzögerung)

Die subjektiven Tests (IBC 2000 und SYSTEMS 2000) mit geschultem TV-Personal ergaben, dass die Technik durch kurze Laufzeiten und höchste Qualität den Produktionsansprüchen der Rundfunkanstalten auch für interaktive Live-Übertragungen entspricht.

Qualitätstests im Labor zeigten, dass die Übertragungen auch über längere Zeiträume stabil und fehlerfrei liefen. Durch den Einsatz eines FEC (Forward Error Correction) können, falls erforderlich, Übertragungsfehler korrigiert werden, wobei jedoch in heutigen glasfaserbasierten Netzen Bitfehler und Zellverluste extrem unwahrscheinlich sind.

5.3 Tests und Demonstrationen

Im Jahr 2000 wurden vom *IRT* anlässlich der Messen IBC 2000 (Amsterdam) und SYSTEMS 2000 (München) aufwendige Demonstrationsinstallationen durchgeführt, die eine Multiservice-Netzwerkplattform für Rundfunkapplikationen beinhalteten und Szenarien der verteilten Produktion über ATM-Netze zeigten.

IBC 2000

Die Vorführungen im Rahmen der IBC 2000 zeigten eine große Anzahl rundfunkspezifischer Applikationen, die über ein Weitverkehrsnetz (622 Mbit/s ATM) zwischen dem *IRT* in München und der Messe in Amsterdam liefen. Dieses Szenario der vernetzten Produktion, Post-Produktion, Zuspielung und Verteilung über eine einzige Netzinfrastruktur bietet den Rundfunkanstalten optimalen Einsatz von Arbeitsressourcen.

„Highlight“ der Demonstration war die Übertragung eines unkomprimierten digitalen Videosignals (270 Mbit/s, SDI) über einen im *IRT* entstandenen „SDI over ATM“-Adapter. So kann das SDI-Signal über ein öffentliches oder privates ATM-Netz mit garantierter Dienstgüte und minimaler Laufzeit übertragen werden. Verteilte Post-

Produktion und verteilte Studios sind nur zwei mögliche Anwendungen für dieses Szenario.

Weitere Applikationen der Demonstration beinhalteten:

- Zuspielung und Verteilung von Programmsignalen (MPEG-2 4:2:2) über die gleiche ATM-Verbindung mit garantierter QoS.
- Verteilung von DVB-Strömen für MHP (Multimedia Home Platform) und DVB-Monitoring Anwendungen.
- Schneller File-Transfer mit dem ATP-Protokoll, einer nativen ATM-Anwendung zum Erreichen der optimalen Übertragungsgeschwindigkeit.
- Lineare Übertragung von Audio-Programmen (AES/EBU).
- Server-Verbindungen über Weitverkehrsnetze (WAN) mit SCSI und/oder Fibre Channel SAN (Storage Area Network).
- Video- und Audio-Streaming von einem Windows Media Server mit UDP-Paketen über ein WAN-Intranet.
- ATM-Verschlüsselung.

Abb. 11 zeigt den Aufbau der Demonstration.

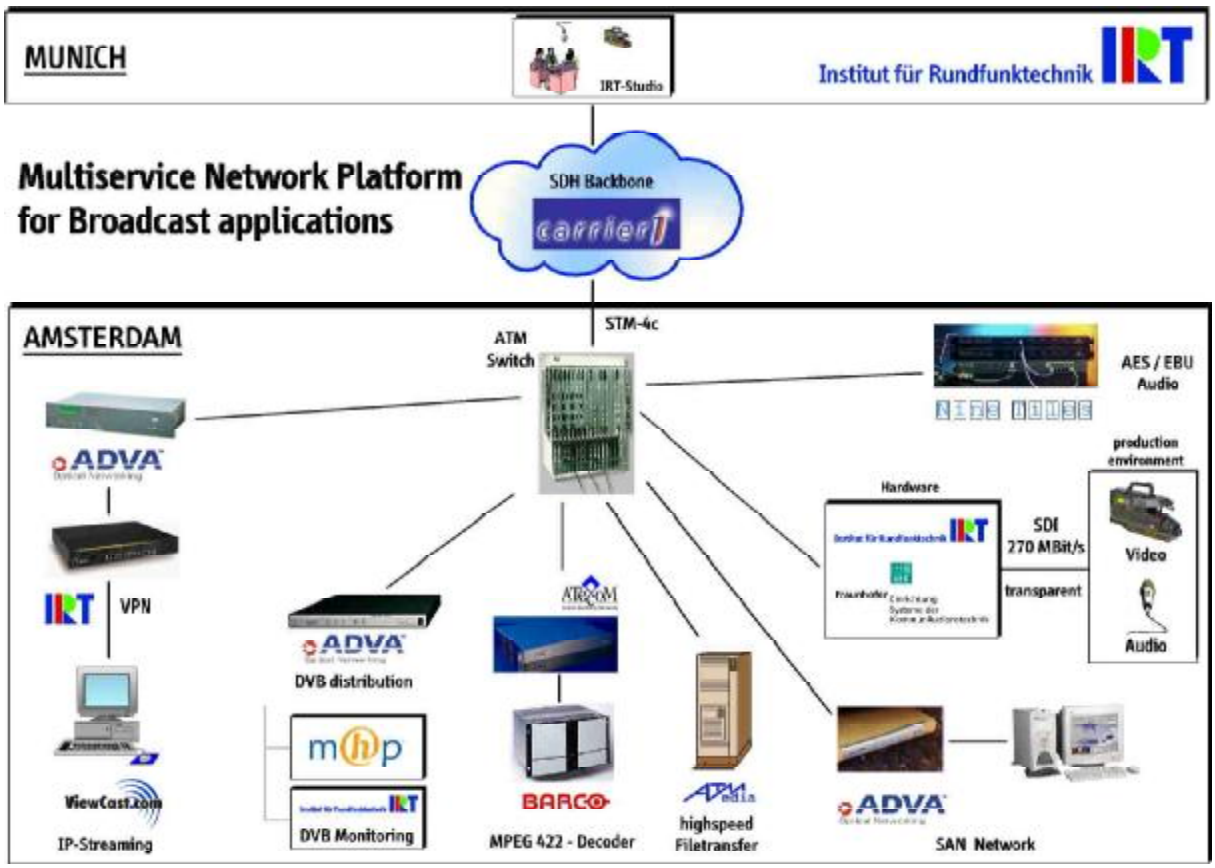

Abb. 11: Aufbau IBC 2000

Sämtliche Anwendungen konnten problemlos nach Bereitstellung der STM-4-Verbindung aufgesetzt werden und liefen während der Messe fehlerfrei. Als besonders beeindruckend wurde von den Fachbesuchern die quasi verzögerungsfreie Übertragung des SDI-Signals zwischen München und Amsterdam angesehen, da man

damit echte Interviews ohne lästige Laufzeiten und die damit verbundenen Probleme durchführen kann.

SYSTEMS 2000

Auf der SYSTEMS 2000 wurde, ähnlich wie bei der IBC, eine WAN-Verbindung zwischen dem *IRT* und dem Messestand in München-Riem aufgebaut. In diesem Fall wurde jedoch eine Dark Fiber mit WDM (Wavelength Division Multiplex) als Transportschicht genutzt. Auf dieser Glasfaser standen drei Wellenlängen zur Verfügung, wovon zwei für jeweils ATM-622 Mbit/s und eine Wellenlänge für ATM-155 Mbit/s verwendet wurden.

Auch hier wurden ähnliche Applikationen wie bei der IBC 2000 vorgeführt.

Zusätzlich zu den IBC-Anwendungen wurde noch vorgeführt, wie das ATM-Netz des *IRT* vom Messeort aus fernkonfiguriert werden kann. Die Applikationen und den Netzaufbau zeigt *Abb. 12*.

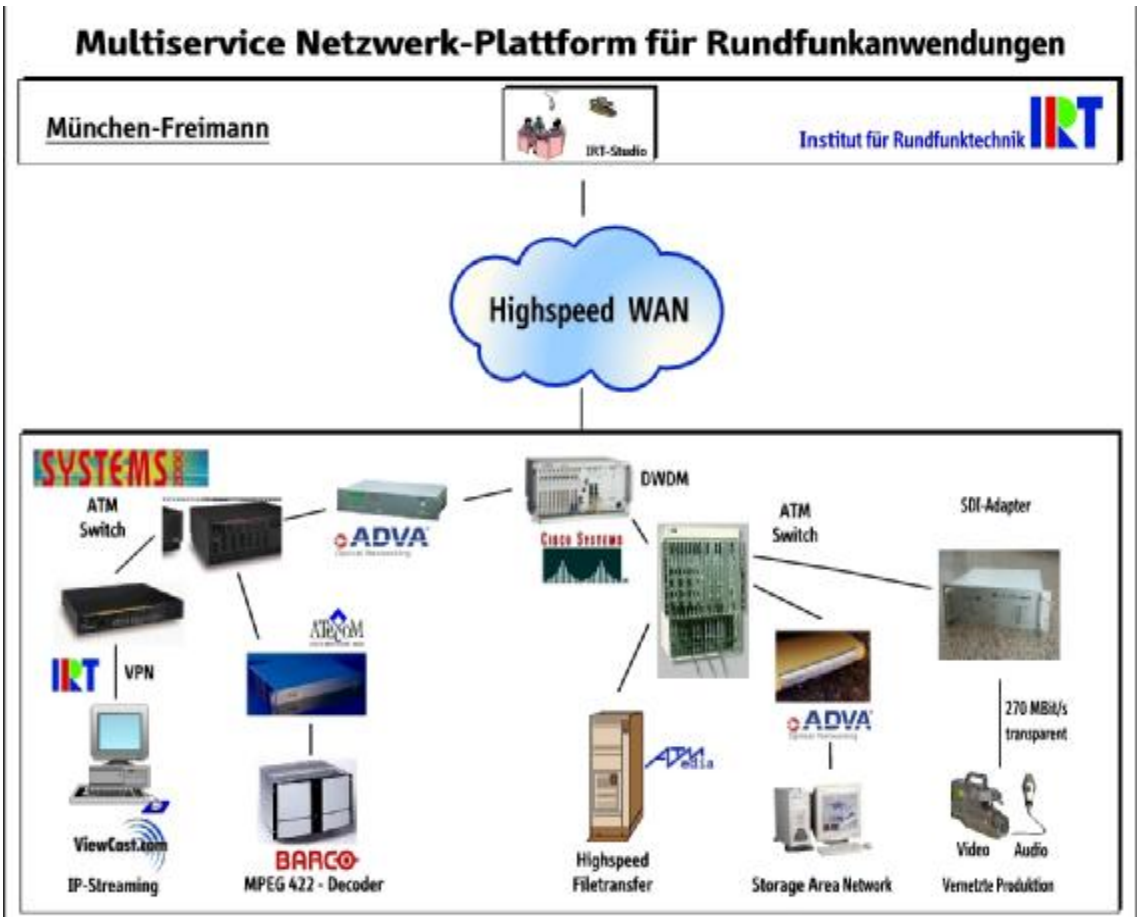

Abb. 12: Aufbau SYSTEMS 2000

5.4 Fazit

Mit Hilfe von ATM ist es heute möglich, Szenarien der verteilten Produktion, Post-Produktion, Zuspielung und Verteilung optimal abzubilden und so eine sehr gute Ausnutzung der Arbeitsressourcen von Produktionsfirmen und Rundfunkanstalten zu erreichen. ATM ist zur Zeit die einzige Technologie, die den rigiden Anforderungen

der Rundfunkanwendungen gerecht wird und Echtzeitdienste und File-Transfer über ein einziges (Standard-Telekommunikations)-Medium ermöglicht.

6 Adaptives Streaming

6.1 Einleitung

Die Übertragung von Videoströmen über IP Netze wird mit wachsenden Bandbreiten zunehmend attraktiver. Denkbare Anwendungen sind zum Beispiel: Video on Demand, Tele-Teaching, Video Fernüberwachung und Internet TV. Die benötigte Bandbreite für hochqualitative Videoströme ist allerdings noch recht groß, so benötigt man für DVD-Qualität MPEG-2 mit einer Datenrate von 8Mbit/s. Problematisch ist die Überlast, die im Netz auftritt und je nach Transportprotokoll zu Fehlern im Video (UDP) oder zu Hängern führt (TCP). Eine garantierte Dienstqualität [Diffserv] löst das Übertragungsproblem. Allerdings steigen dann die Übertragungskosten, denn im Gegensatz zu einem Best Effort Dienst der in Zukunft vermutlich überall über eine Flat Rate abgerechnet wird, ist es zu erwarten, dass ein garantierter Dienst eine volumenbasierte Abrechnung hat. Um diese Übertragungskosten auf das Minimum zu reduzieren, wurde im Rahmen dieses Projektes ein Adaptives Streaming Verfahren entwickelt. Dabei wurde von folgenden Randbedingungen ausgegangen:

- Ein Best Effort Dienst ist billig (Flat Rate).
- Garantierte Dienste sind teuer (volumenbasierte Abrechnung).
- Internet Service Provider (ISPs) oder Firmen haben längerfristige Verträge untereinander, die die Übertragung von verschiedenen Dienstklassen umfassen.
- Es werden ausschließlich nicht interaktive Anwendungen betrachtet, die ein gewisses Verzögerungsbudget ermöglichen.

Die folgenden Ziele sollen erreicht werden:

- Der garantierte Dienst soll so sparsam wie möglich genutzt werden.
- Der Best Effort Dienst soll so gut wie möglich genutzt werden, ohne andere Nutzer des Dienstes zu benachteiligen.
- Die gestreamten Videos sollen eine hohe Qualität (DVD) haben.
- Es soll keine Qualitätseinbußen bei der Übertragung, also keine Verluste und Hänger auf Anwendungsebene, geben.

Abb. 13 erläutert die grundlegende Idee des adaptiven Streaming. Bei einer festen Reservierung wird eine bestimmte Bandbreite für die Zeitdauer der Übertragung reserviert. Die reservierte Bandbreite muss der Spitzenrate der Videoübertragung entsprechen, um eine Übertragung mit maximaler Qualität zu realisieren. Da bei MPEG der zu übertragene Datenstrom eine variable Rate haben kann, dessen Spitzenrate weit über der durchschnittlichen Rate liegt, bleibt möglicherweise viel Bandbreite ungenutzt oder muss weiter vergeben werden. Im Optimalfall hat der MPEG Datenstrom eine konstante Rate, so dass die gesamte reservierte Rate auch genutzt wird. Bei einer Reservierung muss aber ggf. für jedes reservierte Byte bezahlt werden, es entstehen weit mehr Kosten als bei einem Best Effort Dienst.

Bei einer Übertragung über einen billigen Best Effort Dienst variiert die verfügbare Bandbreite entsprechend der Netzauslastung. Dies führt dazu, dass bei hoher Auslastung die verfügbare Bandbreite nicht mehr ausreicht, um das Video zum Empfänger zu übertragen. Beim adaptiven Streaming werden die Videodaten sowohl über einen Best Effort Dienst als auch über einen garantierten Dienst übertragen. Dabei wird versucht, soviel Daten wie möglich über den elastischen Best Effort Dienst zu übertragen. Reicht die hier verfügbare Bandbreite für den Videostrom nicht

aus, wird der Rest der benötigten Bandbreite über einen garantierten Dienst übertragen. Da sich die verfügbare Bandbreite des Best Effort Dienstes ständig ändert und nicht vorhersagbar ist, muss die reservierte Bandbreite immer an die aktuelle Situation angepasst werden, so dass es beim Empfänger nicht zu Puffer Unter- oder Überläufen kommt. Wie beim Smoothing [FeRe97] muss der Empfänger Schwankungen der Bandbreite durch eine Abspielverzögerung und einen Puffer ausgleichen.

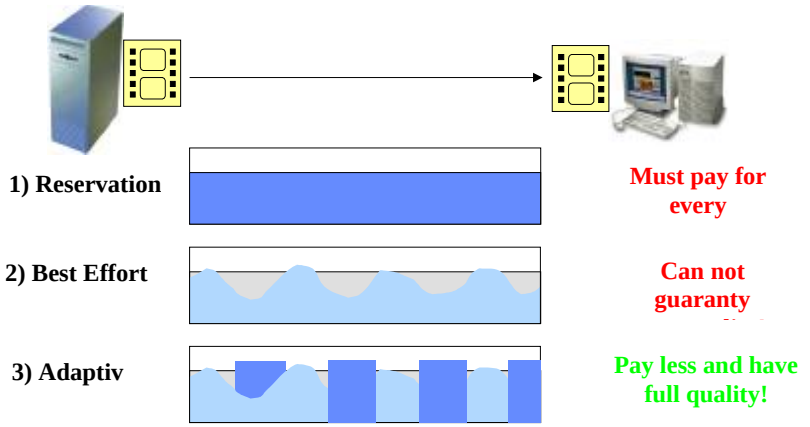

Abb. 13: Adaptive Streaming

Eine Videoübertragung mit garantierter Dienstqualität (garantiertes Streaming) erfolgt durch eine feste Reservierung im Netz. Dies garantiert eine bestimmte Bandbreite, ist aber unelastisch und teuer. Außerdem wird ein Dienstanbieter nur einen gewissen Teil seiner Bandbreite als garantierten Dienst anbieten, so dass die Anzahl der gleichzeitigen Nutzer eingeschränkt ist. Ein Best Effort Dienst ist ein preiswerter Dienst der jedoch keine Garantien gibt. Deshalb muss zur Übertragung ein zuverlässiges Protokoll benutzt werden. Um andere Nutzer dieses Dienstes nicht zu beeinträchtigen muss das Protokoll außerdem TCP-friendly [MaFlo97] sein. Adaptives Streaming ist eine Kombination beider Dienste in Verbindung mit Prefetching, d.h. Daten werden bereits im Voraus übertragen und in einem Puffer am Empfänger zwischengespeichert. Adaptives Streaming erhöht den Multiplexgewinn im Netzwerk und verringert die Übertragungskosten im Vergleich mit einem rein auf Reservierung basierenden Streaming. Außerdem können mehr Nutzer gleichzeitig den Dienst nutzen. Aus den oben genannten Zielen lassen sich die folgenden Anforderungen an eine Implementierung herleiten:

- MPEG-2 (DVD Qualität) Streaming (RTP)
- Client Feedback (RTCP)
- RTSP Session Kontrolle
- Flexible, erweiterbare Implementierung
- Zuverlässiger, TCP-friendly Transport Mode
- Geringer Aufwand bei der Konfiguration (keine Datenbank etc.)
- Integrierbar mit Messplattform

Zu Beginn der Arbeiten im GigaMedia Projekt und auch bis heute unterstützt keine der frei verfügbaren (Open Source) Lösungen hochqualitatives MPEG-2 Video- und Audiostreaming. Kommerzielle Produkte dagegen benutzen häufig nicht standardisierte bzw. undokumentierte Protokolle, Erweiterungen wie die Integration

von Adaptivem Streaming sind grundsätzlich nicht möglich und ggf. muss sogar spezielle Hardware gekauft und eingesetzt werden. Auch sind nahezu alle dieser kommerziellen Anwendungen nicht besonders flexibel einsetzbar und nur schwer integrierbar. Sowohl die Flexibilität als auch die Erweiterbarkeit sind problematisch, wenn die Anwendung nicht im Quellcode verfügbar ist.

Deshalb wurde entschieden, im Rahmen des GigaMedia Projektes einen Prototypen für einen MPEG-2 Streaming Server und Client für das Linux-Betriebssystem zu entwickeln. Linux wurde gewählt, weil es Vorteile bei der Entwicklung eigener Software bietet. Gleichzeitig wird es immer populärer und ist eine stabilere Plattform als Windows. Um hochratiges MPEG-2 abspielen zu können, wird eine Stradis MPEG-2 Decoder Karte eingesetzt, die semi-professionellen Ansprüchen genügt (4:2:2 bis 50Mbit/s).

6.2 Verwandte Arbeiten

Das Real Time Streaming Protocol (RTSP) ist im RFC2326 [RFC2326] definiert und dient der Steuerung der Übertragung von Daten mit Echtzeitanforderungen, z.B. von On-Demand Audio-/Video-Daten. RTSP handelt dabei die Übertragungsparameter einer aufzubauenden Verbindung aus, die eigentliche Datenübertragung wird von anderen Protokollen erledigt (insbesondere durch das RTP Protokoll [RFC1889]).

Das MPEG-2 Videoformat ist ein von der ISO standardisiertes Dateiformat, das Video- und Audiodaten komprimiert in digitaler Form beschreibt [ISO93]. Es werden verschiedene Qualitätsstufen (Anzahl der Zeilen, Samples, Frames usw.) über sogenannte Levels und Profiles unterstützt [ATM98]. Die gebräuchlichste Qualitätsstufe ist „Main Level at Main Profile“ (ML@MP), sie hat 30 Frames/s und eine maximale Videodatenrate von 15 Mbit/s.

Die Übertragung über das Netz geschieht mit Hilfe des Real-Time Transport Protocol (RTP), welches die Übertragung von Audio- und Videodatenströmen in Realzeit beschreibt [RFC1889], es ist von der Beschaffenheit der Audio- und Videodaten unabhängig. Das RTP Protokoll wird vom RTP Control Protocol (RTCP) erweitert, das sowohl Funktionen zur Session-Kontrolle sowie Feedback vom Client zum Server über die Übertragungsqualität umfasst [RFC1889]. Die Nutzung des RTP Protokolls mit MPEG2-Daten ist in [RFC1890], [RFC2250] und [RFC2343] definiert. Dort ist hauptsächlich definiert, wie die einzelnen MPEG-2-Stromarten in RTP-Pakete gekapselt werden müssen.

Der Ansatz eine Mischung aus reserviertem Dienst und Best Effort Dienst für die Übertragung eines Videos zu benutzen wird ebenfalls in [RaTh98] postuliert. Es wird allerdings nicht beschrieben, wie eine solche Übertragung realisiert werden kann, d.h. wie ein Videostrom auf die beiden Dienste verteilt ohne Verlust der Qualität übertragen werden kann. Es wird lediglich eine Netzwerkarchitektur beschrieben, die Layered Videoübertragung unterstützt. Dabei können die einzelnen Layer jeweils über einen reservierten oder einen Best Effort Dienst übertragen werden. Es wird nicht beschrieben, wie eine optimale Aufteilung der Layer erfolgen kann.

Das hier vorgestellte Adaptive Streaming ist vom Ansatz her dem sogenannten Smoothing recht ähnlich. Beim Smoothing wird versucht, ein Video mit variabler Bitrate (MPEG) mit möglichst geringer Varianz der Übertragungsrate zu senden. Anhand der Videodaten und der Größe des Playout-Puffers am Empfänger kann ein Bandbreitenplan berechnet werden, um die Varianz bei der Übertragung zu minimieren [SZKT96]. Hierbei wird davon ausgegangen, dass die benötigte

Bandbreite stets zu Verfügung steht. In [ReTo99] wird untersucht, wie sich das Smoothing optimieren lässt, wenn der Dienstanbieter nicht den gesamten Pfad vom Videoserver bis zum Playout-Puffer kontrolliert. Es wird eine Technik vorgestellt, um Video mit variabler Rate effizient auf einer Teilstrecke zu übertragen. Verschiedene Smoothing Verfahren werden in [FeRe97] verglichen. Alle Verfahren funktionieren nur mit bereits vorhandenen Videodaten, können also nicht für Live-Video eingesetzt werden. [RSFT97] beschreibt ein Verfahren, dass auch bei Live-Übertragungen von Videos mit variabler Bitrate eingesetzt werden kann.

[VFJF99] gibt einen Überblick über die vorhandenen Techniken im Bereich Adaptive Streaming auf Anwendungsebene. Es werden sowohl Layered Encoding, Adaptive Forward Error Correction (FEC) als auch Smoothing Verfahren beschrieben. Ein dem Adaptiven Streaming im Gigamedia Projekt ähnliches Verfahren wird nicht aufgeführt. [RNKA97] beschreibt einen adaptiven Video-Streaming Service, der die Qualität des Videos entsprechend der Übertragungsqualität optimiert. Ein Algorithmus wählt die Video Adaption, welche die beste visuelle Qualität für eine bestimme Dienstqualität des darrunterliegenden Netzwerkes liefert.

6.3 Adaptive Übertragung

Hier wird die adaptive Übertragung zwischen einem Sender und einem Empfänger betrachtet (siehe *Abb. 13*). Bei einer Videoübertragung muss der Playout-Puffer am Empfänger beachtet werden. Es darf keinen Unterlauf und Überlauf des Buffers geben. Wie beim Smoothing wird vorausgesetzt, dass der Sender die Puffergröße *B* kennt. Grundsätzlich versucht der Algorithmus den Pufferfüllstand immer in Puffermitte zu halten[1]. Ein fast leerer Puffer ist gefährlich, da dann ein Puffer Unterlauf droht. Ein Unterlauf kann nur durch einen ausreichenden Füllstand verhindert werden. Ein zu voller Puffer bedeutet, es wird möglicherweise garantierte Bandbreite verschenkt. Der Sender kann einen Pufferüberlauf immer verhindern, weil er die Puffergröße kennt. Ist die reservierte Bandbreite bereits auf 0 und der Füllstand ist immer noch oberhalb der oberen Schwelle so muss er die Senderate über den Best Effort Dienst drosseln. Eine Änderung der reservierten Bandbreite erfolgt, wenn der Puffer Füllstand eine obere oder eine untere Schranke über bzw. unterschreitet oder die letzte Änderung länger als *Dt* zurückliegt.

Die in einem Intervall *Dt* übertragenen Daten *a(Dt)* werden dabei über einen Best Effort und einen garantierten Dienst übertragen: $a(Dt)=a_{be}(Dt)+a_g(Dt)$.

$a_{be}(\Delta t)$ ändert sich dabei in jedem Intervall abhängig von der Auslastung des Netzwerkes und damit von der Verlustrate. In drahtgebundenen Netzen entstehen Verluste nahezu ausschließlich durch Überläufe der Warteschlangen in den Routern. Deshalb muss $a_g(Dt)$ so angepasst werden, dass es zu keinem Pufferüber- oder unterlauf kommt. Gleichzeit soll natürlich die Menge des über den garantierten Dienst übertragenen Daten möglichst klein sein: $\min\left(\sum a_g(t)\right)$

Außerdem muss die Variabilität von a_g möglichst gering sein, um die für garantierten Dienst verfügbare Bandbreite möglichst gut ausnutzen zu können: $\min\left(\mathrm{var}\left(a_g\right)\right)$

[1] Eine Verallgemeinerung des Verfahrens würde versuchen, den Pufferstand auf einem Mittelwert θ zu halten ($0<\theta<1$), bei dem die gewünschte maximale Wahrscheinlichkeit zum Pufferunterlauf eingehalten wird und gleichzeitig eine möglichst preiswerte Übertragung gewährleistet ist.

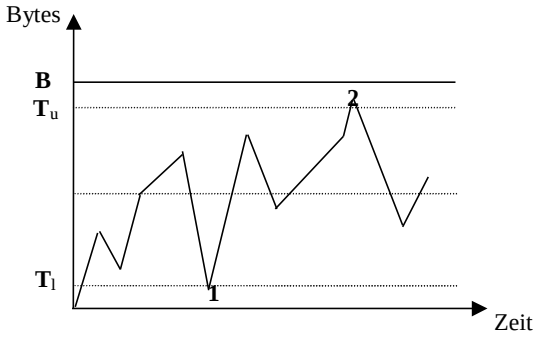

Abb. 14: Playout-Puffer Füllstand

Abb. 14 verdeutlicht die Funktionsweise anhand eines beispielhaften Pufferverlaufs. Am Punkt 1 wird die untere Schranke unterschritten, d.h. zu diesem Zeitpunkt muss die Reservierung erhöht werden, um den Pufferunterlauf zu vermeiden. Im Punkt 2 kann die Reservierung verringert werden, da der Puffer nahezu voll ist. Im folgenden bezeichnet A(t) die tatsächliche Übertragungsfunktion und S(t) die optimale Übertragungsfunktion. Das Ziel zum Zeitpunkt *t+Dt* ist wie folgt: $A^* = \sum_{1}^{t \cdot fps} fi + 0{,}5 \cdot B$

Dabei ist t_{la} die Zeit, die der Algorithmus zur Anpassung von *A(t)* an *S(t)* hat. Um unnötig schnelle Änderungen zu vermeiden und damit die Varianz von a_g zu verringern muss $t_{la} >= Dt$ sein. Natürlich muss gelten $t_{la} = min(t_{la}, T\text{-}t)$. Für die Menge der im nächsten Intervall zu übertragenen Daten gilt dann:

$$a(\Delta t) = A(t + \Delta t) - A(t) = \frac{A^* - A(t)}{t_{la}} \quad \text{mit } a(Dt) = a_g(Dt) + a_{be}(Dt)$$

$a_{be}(\Delta t)$ ist natürlich im Voraus nicht bekannt. Deshalb wird $a_{be}(\Delta t)$ auf der Basis des bisherigen Durchsatzes im letzten Intervall Δt gemittelt: $\sim a_{be} = \frac{\sum_{i=1}^{t} a_{be}(i)}{\Delta t}$

Somit ergibt sich für die zu reservierende Bandbreite $a_g(Dt)$:

$$a_g(\Delta t) = \frac{A^* - A(t)}{t_{la}} - \sim a_{be}(\Delta t)$$

Für t_{la} gilt hier: $t_{la} = \min\left(t_{la}, \frac{\sum frames}{fps} - t\right)$

Die Reservierung wird nicht in festen Intervallen geändert, sondern nur dann wenn die obere oder untere Pufferschwelle über bzw. unterschritten wird. Das bedeutet, es kann nicht mehr garantiert werden, dass Änderungen der garantierten Bandbreite in Intervallen *Dt* erfolgen. Hier ergibt sich die Frage, wie schnell sich eine Reservierung in der Realität ändern lässt. Eine Alternative ist allerdings eine vertraglich festgelegte quasi-statische Reservierung, wobei nur für das benutzte Volumen bezahlt wird. Mit Diffserv lässt sich das tatsächlich genutzte Volumen durch das Markieren von Paketen mit dem entsprechenden Diffserv Codepoint auf Basis von einzelnen Paketen ändern. Am Anfang der Übertragung muss der Puffer natürlich gefüllt werden. Dazu wird das Abspielen am Client erst gestartet, wenn der Puffer zur Hälfte gefüllt ist oder eine festgelegte maximale Startup-Verzögerung erreicht wird.

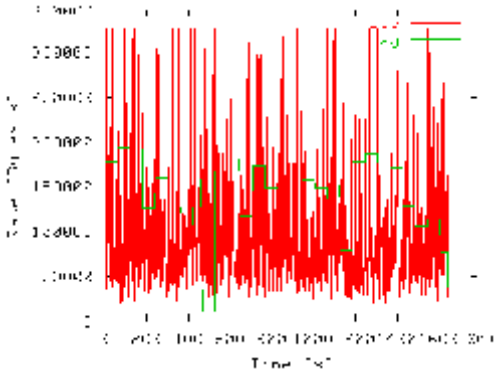

Abb. 15: Adaptives Video Streaming

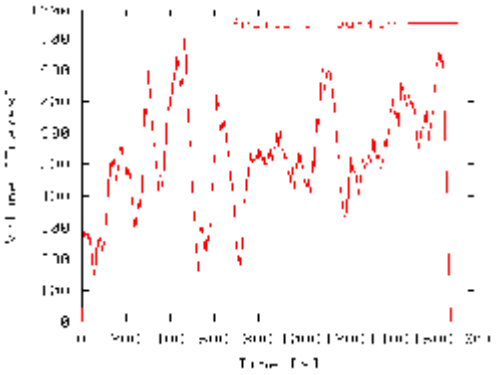

Abb. 16: Anzahl Frames im Puffer

Der entwickelte Algorithmus wurde zunächst mit Hilfe einer analytischen Simulation untersucht. Dabei wird in Intervallen von 1 Sekunde der Durchsatz des elastischen Best Effort Dienstes neu anhand des Modells aus [PFTK98] berechnet. Die Rate des garantierten Dienstes wird anhand des Algorithmus aus diesem Abschnitt berechnet. Bei der Simulation wird angenommen, dass die Verlustrate des garantierten Dienstes gleich 0 ist. Für den Best Effort Dienst wird die Verlustrate mit einer Exponential-verteilung berechnet [ZhPS00], [BoGa98]. Für die Simulation des adaptiven Video Streamings werden Tracefiles von existierenden MPEG Videos verwendet [Wuerz98].

Abb. 15 zeigt adaptives Video Streaming des Terminator Videos [Wuerz98] bei einer mittleren Verlustrate von 0,5%. Die Puffergröße am Empfänger beträgt 8Mbyte (t_l=500.000, t_h=7.500.000), die Startup-Verzögerung 10 Sekunden. Die Simulation ergibt CF von 0,6 und PCF von 0,85, d.h. die Videoübertragung lässt sich trotz unverminderter Qualität für 60% der Kosten realisieren. *Abb. 16* zeigt die Anzahl der Frames im Puffer des Empfängers über der Zeit bei dieser Übertragung. Eine ausführliche Beschreibung der Simulationsergebnisse findet man in [GMED_D12].

6.4 Design und Implementierung des Video Servers

6.4.1 Eigenschaften

Die von FOKUS im Rahmen des GigaMedia Projektes entwickelte Video Server Implementierung verfügt über die folgenden Eigenschaften:

- MPEG-2 über RTP (Transport/Program Streams)
- RTCP und RTCP Fast Feedback
- RTCP Anwendungs-spezifische Nachrichten (Pufferfüllstand, ...)
- Basis RTSP Implementierung (unterstützt Play, Pause, Stop)
- Text-basiertes Interface zur RTSP Protokoll Engine ermöglicht GUI Flexibilität, es wurden ein Java GUI, ein Web-basiertes GUI, sowie ein Shell Script Interface implementiert
- MPEG-2 über RTP kann ohne RTSP benutzt werden (Sender/Receiver Mode mit Hilfe einer interaktiven Shell Steuerung)
- RTP Transport über UDP und TCP
- Adaptive Streaming mit RTP über TCP
- Flexibles RTSP Server-seitiges “Backend” ermöglicht die Steuerung von externen (z.B. rtptools, MAZ) als auch eingebetteten Anwendungen (MPEG-2)
- Erweiterbarkeit durch modularen Aufbau (C++, Libraries)

- Hochratiges MPEG-2 kann wiedergegeben werden, da ein semi-professioneller MPEG-2 Hardware-Decoder verwendet wird

6.4.2 Entwurf

Der Client dekodiert den empfangenen MPEG2-Strom mit Hilfe einer Hardware MPEG-2 Decoder-Karte der Firma Stradis [Strad00]. Die Stradis-Karte ist eine semi-professionelle PCI MPEG-2-Decoder-Karte und kann MPEG-2 Videos im High Profile (Samplerate 4:2:2) oder im Main Profile (Samplerate 4:2:0) mit bis zu 50 Mbit/s dekodieren. Durch den modularen Aufbau des Clients ist es aber möglich, den MPEG-2-Strom auch auf eine andere Art zu dekodieren, z.B. mit einem Software-Decoder. Dazu muss nur das nötige Decoder Objekt implementiert werden.

Das Design ist in *Abb. 17* dargestellt. Der Benutzer gibt im Graphischen User Interface (GUI) an, welche Medien von welchem Server er abspielen möchte (Play Kommando). Ist man bereits im Abspielmodus, so kann man die Wiedergabe unterbrechen (Pause) oder Beenden (Stop). Die GUI gibt diese Anforderung an den RTSP-Client weiter. Der Client baut eine Verbindung zum Server auf und sendet ein RTSP-Request, d. h. eine Anforderung im Format des RTSP Protokolls. Das Modul, welches im Server für die Verarbeitung von RTSP-Kommandos zuständig ist, empfängt den Request und veranlasst den Server, das gewünschte Kommando auszuführen. Im Fall von Play fängt der Server mit der Wiedergabe an, indem er die angegebene MPEG-Datei einliest und diese gemäß dem Real-Time Transport Protokoll (RTP) an die Zieladresse sendet. Auf der Client-Seite empfängt der MPEG-Client die Daten und gibt sie über die Stradis-Decoder-Karte auf dem Bildschirm aus.

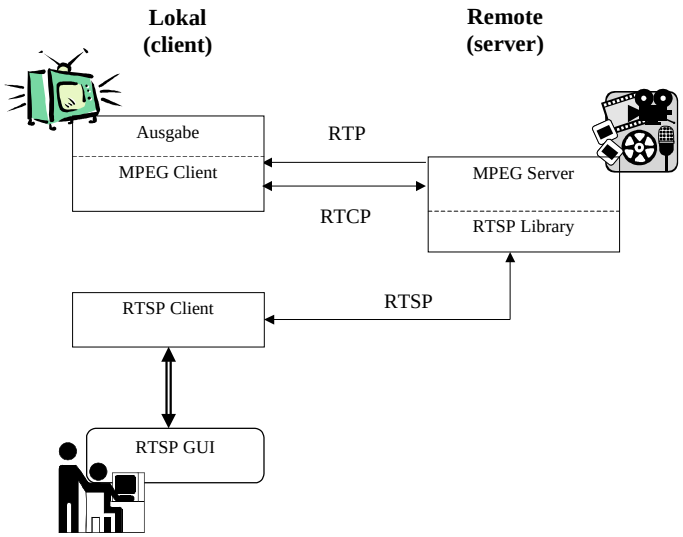

Abb. 17: Video-Server Design

Client und Server kommunizieren zusätzlich über das RTCP Protokoll. Hauptsächlich sendet der Client Feedback Information an den Server. In diesen Informationen ist die momentane am Empfänger gemessene Qualität (Loss, Jitter) enthalten, sowie weitere Zustandsinformationen des Clients (z.B. Pufferfüllstand) die am Server für den Adaptive Streaming Algorithmus benötigt werden.

Bei der Entwicklung der RTSP Video-Steuerung wird ein sehr flexiblen Ansatz verwendet. Die RTSP Implementierung ist nicht ein monolithischer Code-Block der in den Video-Server integriert ist. Auch die direkte Steuerung, in der Regel durch eine Grafisches User-Interface (GUI) beim Client, ist von der RTSP Implementierung getrennt. Der Video-Server teilt dem RTSP Modul über ein Interface (Capability-

Interface) mit, über welche Fähigkeiten er verfügt und das RTSP Modul handelt dann aufgrund dieser Informationen eine Verbindung mit dem Client aus. Nach erfolgreichem Verbindungsaufbau werden die vom Client ausgelösten Aktionen wie PLAY, PAUSE, STOP über ein weiteres Interface, dem Aktions-Interface, an den Video-Server zurückgegeben, damit dieser eine Übertragung mit den ausgehandelten Parametern starten, stoppen, etc. kann. Auf der Seite des Clients gibt es zwischen GUI und RTSP Implementierung ein text-basiertes Interface.

6.4.3 Implementierung

Das RTSP Protokoll Modul ist in C implementiert. Der Parser wurde mit Hilfe der im Compilerbau bewährten Tools flex [flex] und yacc [yacc] generiert. Server- und Client-RTSP Modul können zur Zeit entweder als eigenständige Programme erzeugt oder als Library in Projekte eingebunden werden. Als eigenständiges Programm kann die Implementierung genutzt werden, um bestehende Applikation zu steuern. Wenn die Applikation als Source Code vorliegt, kann die RTSP Steuerung direkt in die Applikation integriert werden.

Im Server-Modus gestartet, erwartet das RTSP Modul TCP-Verbindungen auf einem bestimmten Port, etabliert eine Verbindung im Falle eines Client-Requests und tritt in den RTSP Protokoll-Dialog ein. Der Server handelt daraufhin mit dem Client die Übertragungs-Parameter einer folgenden Streaming-Verbindung aus. Welche Parameter dabei jeweils für Client und Server akzeptabel sind, hängt von den Daten ab, die dem Client- und Server-RTSP-Modul beim Start über das Capability-Interface übergeben worden sind. Das Capability-Interface ist durch eine Anzahl von Übergabe-Parametern auf der Client- bzw. Server-Seite realisiert. Das Aktion-Interface besteht aus Funktionen, die beim Auftreten der RTSP-Kommandos aufgerufen werden und ihrerseits wieder externe Programme oder interne Funktionen aufrufen, die die entsprechenden Aktion realisieren. Im Client wird jede Aktion durch Eingaben an das RTSP Kommando-Interface initiiert. Das Kommando-Interface ist mit Named Pipes realisiert. Eine Pipe ist dabei der Kanal für Kommandos zum Client-RTSP-Modul und die zweite Pipe ist der Antwort-Kanal für eventuelle Fehlermeldungen etc. Momentan entspricht die RTSP Implementierung im wesentlichen den MUST Forderungen des RFC 2326 für eine minimale Implementierung.

Sowohl der MPEG-2 Server als auch der Client sind in C++ unter Linux implementiert. Die wichtigsten Klassen sind in *Abb. 18* dargestellt. Beim Server kann der Dateiparser, Klasse `MpegReader`, MPEG-2-Transportströme einlesen und bearbeiten. Der Encoder überwacht die Einhaltung der vom Parser errechneten Verzögerung (Klasse `MpegEncoder`). Für den Versand und Empfang sind die Klassen `IPNetwork` bzw. die Basisklasse `Network` zuständig. Beim Server können mehrere Sessions (Klasse `Session`) gleichzeitig laufen, d. h. es können gleichzeitig Daten aus verschiedenen (oder gleichen) MPEG-2-Dateien an verschiedene Clients gesendet werden.

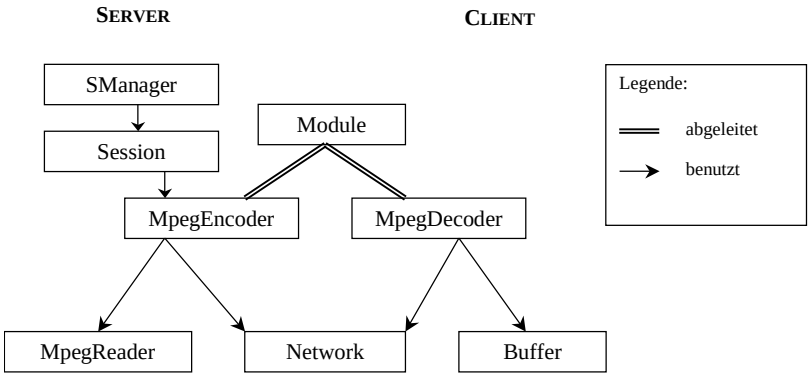

Abb. 18: MPEG-2 Server/Client Klassen

Dabei laufen die Sessions in einzelnen Threads. Die Threads sind mit der POSIX Thread Bibliothek implementiert. Die Klasse `SManager` verwaltet die einzelten Sessions, d. h. sie erstellt und löscht sie. Im Client werden die empfangenen UDP Pakete gleich nach ihrer Ankunft gepuffert (Klasse `Buffer`). Es wird ein FIFO-Puffer konstanter Größe verwendet, der mit Hilfe eines Queue-Containers der Standard Template Library (STL) implementiert ist. Das Auslesen der Pakete aus dem Puffer und das anschließende Dekodieren geschieht in einem separaten Thread (Klasse `MpegDecoder`). Der gleichzeitige Zugriff des Empfangs- und des Dekoderthreads auf die Puffer-Datenstruktur wird mit einem Mutex verhindert. Das Dekoder-Timing wird hier von der Stradis-Dekoderkarte erledigt, da die Karte Schreibzugriffe sperrt, falls zu viele MPEG-2-Daten vorliegen.

Es gibt drei verschiedene Frontends für den RTSP Client. Das Java Frontend ist für den normalen Betrieb und zu Demonstrationszwecken gedacht. Das zweite Frontend ist ein web-basiert, d.h. es ermöglicht die Fernsteuerung eines Clients über das HTTP Protokoll, es wurde mit Perl/CGI realisiert. Das dritte Frontend basiert auf Shell-Skripten, die mit dem RTSP Modul kommunizieren. Es ermöglicht die Realisierung von automatischen Clienten durch Shell Skripte z.B. zum Testen des Servers oder zur Durchführung von Versuchen.

Der Adaptive Streaming Algorithmus wurde in den MPEG-2 Video Server integriert. Der MPEG-2 Video Client wurde so erweitert, das er regelmäßig spezielle Anwendungs-spezifische RTCP Nachrichten an den Server sendet. Damit der Server schnell genug reagieren kann, wurde ein RTCP Fast Feedback Mechanismus mit konfigurierbarem Intervall implementiert. Die RTCP Feedback Nachrichten vom Client enthalten den momentanen Pufferfüllstand und den vom Client gemessen Durchsatz in beiden Klassen. Der Sender errechnet seinerseits den für die Übertragung nötigen Durchsatz anhand der Videodaten. Der Sender passt die Senderate an, so dass der Pufferfüllstand immer möglichst auf dem Zielfüllstand ist. Der Sender weist außerdem jedes zu sendende Paket einer der beiden Klassen zu. Er passt die Markierungsrate entsprechend dem Durchsatz in der Best Effort Klasse und dem momentanen Pufferfüllstand am Empfänger an. Der Algorithmus besteht aus zwei unabhängigen Teilen, nämlich der Flusskontrolle und dem adaptiven Markieren:

1. Durch die RTCP Feedback-Pakete wird der Pufferfüllstand vom Sender überwacht und mit der Soll-Füllrate (i. A. 50%) verglichen. Ist Δn die Differenz des Ist- und Sollfüllstandes (in Bytes) und Δt die Zeit zwischen zwei RTCP Paketen, dann wird die momentane Senderate s modifiziert: $s' = s + \frac{\Delta n}{\Delta t}$, wobei s' nicht kleiner als 0 werden darf. Damit werden die im Puffer fehlenden Daten nachgesendet

bzw. die Senderate wird verkleinert, so dass die überflüssigen Daten im Puffer abgearbeitet werden.

2. Senden wir über die normale Senderate *s* hinaus, dann werden Pakete mit „Best Effort“ markiert. Dazu wird das Verhältnis $r = \frac{\Delta n / \Delta t}{s}$ gebildet. *r* gibt an, mit welcher Wahrscheinlichkeit Pakete als Best Effort markiert werden (für $r > 1$ setzt man $r = 1$ und $r < 0$ setzt man $r = 0$).

Die Zuweisung der Pakete zu den Klassen wird nicht direkt vom Server ausgeführt. Stattdessen wird das RTP m-bit [RFC1889] benutzt um die Klasse des Paketes zu signalisieren. Ist das m-bit auf 1 gesetzt, soll das Paket in der garantierten Dienstklasse übertragen werden. Ist das m-bit auf 0 gesetzt, soll das Paket in der Best Effort Klasse übertragen werden. Es wurde ein NetFilter Klassifizierer [NetFilt01] entwickelt, der RTP Pakete über UDP oder TCP anhand von Payload, SSRC oder m-bit klassifizieren kann. Dieser Klassifizierer läuft im Linux Kernel und klassifiziert alle Pakete entsprechend der Konfiguration. Anhand der Klassifikation wird dann der Diffserv Code Point (DSCP) mit dem Linux Diffserv Tool tc [LinDiff01] gesetzt. Um eine fehlerfreie Übertragung zu realisieren, werden RTP Pakete als TCP Strom übertragen. Dies erfolgt mit dem in [UDPT] vorgeschlagenen Verfahren. In dem entwickelten Demonstrator werden TCP Retransmissions immer über die garantierte Klasse übertragen, um eine maximale obere Grenze für die Verzögerung zu haben.

6.4.4 Demonstratorszenarien

Es wurden vier verschiedene Demonstratorszenarien realisiert:

- MPEG-2 Video Streaming
- MAZ Steuerung
- Steuerung existierender Tools (am Beispiel rtptools)
- MPEG-2 Adaptive Video Streaming

Beim MPEG-2 Video Streaming wird die Übertragung von MPEG-2-Videos vom Empfänger ferngesteuert. Dazu gibt man im GUI die Adresse des Servers und die abzuspielende Datei im Format von RTSP an, z.B. `rtsp://master.ailab.fokus.gmd.de/riox.mpeg` und drückt PLAY. Um die Wiedergabe anzuhalten bzw. fortzusetzen drückt man PAUSE bzw. PLAY. Mit STOP kann man die Wiedergabe stoppen. *Abb. 20* zeigt die Bildschirmausgabe.

Bei der Steuerung einer MAZ werden RTSP Kommandos vom Client an den Server gesendet. Der RTSP-Server gibt die empfangenen und verarbeiteten Kommandos an die MAZ weiter. Die MAZ wird in diesem Szenario durch eine Stradis-Karte simuliert, das Videomaterial ist eine MPEG-2 Datei. PLAY, STOP und PAUSE Kommandos werden an die Stradis-Karte weitergeleitet. Die Ausgabe erfolgt über das SDI Interface der Karte, so dass mit Hilfe des SDI über ATM Adapters ein SDI Strom ferngesteuert werden kann.

Aufgrund des modularen Aufbaus kann die entwickelte RTSP Implementierung auch dazu genutzt werden, bereits existierende Tools zur Videoübertragung zu steuern. In diesem Demonstratorszenario benutzen wir eine Video Datei, die aus einer RTP-Übertragung erzeugt wurde und geben dieses mit Hilfe von rtpplay [rtptools] wieder.

Für das Adaptive Streaming wird zur Bereitstellung der unterschiedlichen Dienstklassen Diffserv [Diffserv] benutzt. Das Diffserv Netzwerk besteht aus einem Edge Router und einem Core Router. Der Edge Router klassifiziert und markiert die Pakete, d.h. der DSCP wird gesetzt. Der Core Router forwardet die Pakete

entsprechend dem DSCP mit unterschiedlicher Priorität. Es werden zwei Klassen benutzt: Best Effort und Expedited Forwarding (EF) [Diffserv]. Der EF Dienst garantiert die vorher ausgehandelte Bandbreite. Die nutzbare Bandbreite des Best Effort Dienstes variiert mit der Netzauslastung. Der Edge Router klassifiziert die beiden TCP Ströme als Best Effort und EF Klasse.

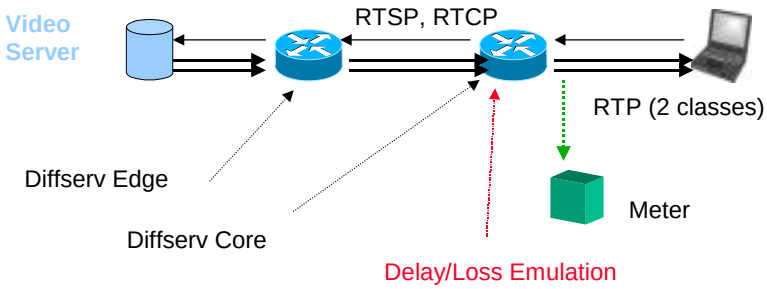

Abb. 19: Adaptive Streaming Demonstration Netzwerk

Abb. 19 zeigt den logischen Aufbau des Demonstrationsszenarios. Ein Client fordert Videos per RTSP Protokoll von einem Server an. Der Server sendet das Video mittels RTP über TCP zum Client und der Client sendet während der Übertragung regelmäßige Feedback Informationen per RTCP an den Server. Der Server teilt das Video entsprechend dem Adaptive Streaming Algorithmus auf 2 Klassen auf. Der Videostrom läuft über 2 Diffserv fähige Router. Der Edge Router markiert die Pakete mit dem entsprechenden DSCP und im Diffserv Core Router werden die Pakete entsprechend dem DSCP gescheduled. Die Diffserv Router sind beide Linux Router mit aktiviertem Diffserv [LinDiff01]. In unserem Demonstratorszenario wird eine Verlustrate in der Best Effort Klasse emuliert.

Abb. 20: MPEG-2 Streaming Demo

Abb. 21: Adaptive Streaming Demo

Abb. 21 zeigt die Ausgabe während der Demonstration des Adaptiven Streaming. Während das Video übertragen wird, misst ein Meter im Netzwerk den momentanen Durchsatz (oben links) und das kumulative Volumen (oben rechts) in beiden Klassen. Die garantierte Klasse ist rot, die Best Effort Klasse grün dargestellt. Die momentane Verlustrate im Netzwerk wird ebenfalls angezeigt (unten links). Außerdem wird (unten rechts) das momentane Verhältnis vom garantierten Volumen zum gesamten Volumen angezeigt. Im Falle einer einfachen Reservierung wäre das immer 1. In *Abb. 21* sind es knapp über 0.14 also 14%. Die Ersparnis im Falle einer volumen-basierten Abrechnung beträgt also fast 86%.

6.5 Zusammenfassung und Ausblick

In diesem Kapitel wurde eine Motivation für den Einsatz eines adaptiven Übertragungsverfahrens gegeben. Ein Algorithmus für die adaptive Video-Übertragung wurde hergeleitet. Es wurden die Ergebnisse einer ersten Simulation präsentiert. Die Simulation zeigt, dass adaptive Übertragung einen großen Gewinn bringt, wenn der Durchsatz der Best Effort Verbindung im Vergleich zur benötigten Bitrate groß genug ist. Da durch adaptives Streaming weniger garantierte Bandbreite verbraucht wird, die garantierte Bandbreite bei einem Dienstanbieter aber stets eingeschränkt ist, kann auch eine größere Anzahl von Dienstnutzern unterstützt werden.

Es wurde ein leichtgewichtiger, flexibler experimenteller MPEG-2 Video Server und Client entwickelt. Dieser Video Server unterstützt RTSP, sowie RTP/RTCP über UDP oder TCP und kann MPEG-2 Transport- oder Program-Ströme übertragen. Die Implementierung der RTSP Steuerung entspricht den MUST Forderungen des RFC 2326. Die Steuerung lässt sich flexibel sowohl für externe (MAZ) als auch eingebettet Anwendungen (MPEG-2 Video Server) einsetzen. Der entwickelte prototypische MPEG-2 Videoclient und –server lässt sich über die RTSP Implementierung steuern. Die Decodierung der Videos auf Seite des Clients erfolgt mit der Stradis MPEG-2 Decoderkarte.

Aufgrund der flexiblen und modularen Implementierung kann das RTSP Modul auch genutzt werden, um andere vorhandene Programme wie z.B. die RTP-Tools [rtptools] zu steuern. Die Trennung von RTSP Modul und Benutzer Interface ermöglicht eine einfache Integration von verschiedenen Interfaces. Es wurden drei verschiedene Interfaces mit unterschiedlicher Zielsetzung entwickelt. Ein Java GUI als normales Benutzer-Interface, ein web-basiertes GUI zur Fernsteuerung eines Clients per HTTP und ein Skrip-basiertes Interface zum Testen und zur Erstellung von automatischen Clients. Das entwickelte Adaptive Streaming Verfahren wurde implementiert und in den Video Server integriert. Der Algorithmus wurde in einer Diffserv Testbed Umgebung getestet und es wurde bereits eine erfolgreiche Demonstration durchgeführt.

In der Zukunft soll die Implementierung weiter verbessert und in weiteren Projekten eingesetzt werden. Insbesondere ist es geplant, die entwickelte Streaming Lösung mit einer Messplattform zu integrieren, um automatisch Streaming Tests und Qualitätsuntersuchungen durchführen zu können. Der Adaptive Streaming Algorithmus soll weiter verbessert werden. Außerdem sind noch weitere Erweiterungen denkbar, um einen flexibleren Einsatz zu gewährleisten. Zunächst soll die momentan in der Entwicklung befindliche Software MPEG-2 Decoder Implementierung für Linux integriert werden, so dass eine Wiedergabe auch ohne teure Hardware erfolgen kann. Darüber hinaus sind weitere Erweiterungen denkbar, z. B. die Verbesserung des Playout-Puffers oder die Unterstützung von MPEG-2 Elementary Streams. Neben kleineren noch zu beseitigenden Instabilitäten der Software muss noch die Interoperabilität mit anderen RTSP Implementierungen getestet und hergestellt werden. Hier sollen Tests mit dem Apple Darwin Streaming Server [Apple01] durchgeführt werden.

7 Formatkonversion als verteiltes Postproduktions-Tool

7.1 Einleitung

Betrachtet man die Entwicklung auf dem Gebiet der digitalen Film- und Videotechnik, so sind heutzutage rasante Fortschritte zu erkennen. Hierbei sind die größten Neuerungen in der Übertragungstechnik auf der einen, und der Darstellungstechnik auf der anderen Seite zu beobachten. Immer weiter optimierte Kompressionsverfahren und der verstärkte Aufbau hochratiger Datennetze (wie das dem Projekt zugrundeliegende GTB) bilden die Grundlage für digitale Bildübertragung, welche schon allein vom Datenvolumen wesentlich höhere Anforderungen an das Übertragungsmedium stellt als beispielsweise reine Audioübertragung.

Im Bereich der Darstellungsgeräte gibt es eine Fülle von Techniken die über kurz oder lang die „althergebrachte“ Kathodenstrahlröhre und den Filmprojektor ablösen werden. Neue Entwicklungen gibt es sowohl im Bereich der Bildschirme als auch bei Projektionsgeräten. Ihnen gemein ist die Tatsache, dass diese Geräte Bilder letztendlich digital verarbeiten und dementsprechend am besten mit digitalen Bilddaten gespeist werden.

Als Begleiterscheinung entsteht zu fast jeder neuen Technik – sowohl bei der Übertragung als auch der Darstellung – ein neues digitales Datenformat. Die große Anzahl vorhandener Formate unterscheidet sich hauptsächlich in bezug auf zwei Merkmale: Die Bildauflösung und die Bildwechselfrequenz. Hinzu kommt die Tatsache, dass Bilddaten, die aus der analogen Fernsehtechnik in digitale Formate übertragen werden, normalerweise im sogenannten Halbbildformat[2] vorliegen.

Um nun Produktions-, Übertragungs- und Darstellungstechnik verschiedener Formate problemlos miteinander koppeln zu können muss also zwangsläufig eine sogenannte Formatkonversion durchgeführt werden.

7.2 Einsatz innerhalb der Postproduktion

Formate im Bereich der digitalen Postproduktion beschränken sich heutzutage meistens auf klassische Video-, bzw. Filmformate (von Standard TV über HDTV bis hin zu 4k-Film-Formaten), während im Bereich der Display-Techniken eher „klassische“ Computerformate vorherrschen (VGA bis QXGA). Selbst im Hinblick auf die Darstellung von Video entwickelte 16:9 Formate wie Wide-XGA (1366×768) finden in der digitalen Postproduktion (noch) keine Anwendung.

Postproduktionseinrichtungen (z.B. bestimmte Effektgeräte) sind normalerweise auf eine kleine Palette von Video-, bzw. Filmformaten beschränkt. Liegt das zu bearbeitende Material in einem anderen Format vor, so können diese nicht eingesetzt werden. Auch steigen die Kosten für den Einsatz von Postproduktionsgeräten überproportional zur verwendeten Bildauflösung. Hier stellt sich dann die Frage, ob eine Verarbeitung in einer niedrigeren Auflösung für ein bestimmtes Projekt nicht ausreichend ist.

---

[2] Um eine Erhöhung der Bildwechselfrequenz bei gleichbleibender Datenrate/Übertragungsbandbreite zu erreichen, wurde das sogenannte Zeilensprungverfahren entwickelt. Mit der Kamera werden hierbei abwechselnd nur die geraden, bzw. ungeraden Zeilen eines Vollbildes abgetastet. Es entstehen sogenannte Halbbilder.

Durch den Einsatz von Video-Formatkonversion wird es möglich, verschiedenste Postproduktionseinrichtungen für ein Projekt zu nutzen.

7.3 Grundlagen

Ziel der Formatkonversion ist es, Videomaterial bestimmter (beliebiger) Formate in z.B. für verschiedene Displays geeignete Formate zu wandeln. Mit Format ist in diesem Fall die örtliche und zeitliche Auflösung (Größe der Einzelbilder und Bildwechselfrequenz) gemeint. Die Konversion gliedert sich in drei Teilaufgaben: Die Zwischenzeilenfilterung, die Zwischenbildfilterung und die Bildskalierung.

Die Aufgabe der Zwischenzeilenfilterung (engl.: deinterlacing) ist es, Bildfolgen im Halbbildformat (engl.: interlaced) in Vollbildsequenzen (engl.: progressive, non-interlaced) zu wandeln. Es sind verschiedene Methoden denkbar, die im Bezug auf den zu erwartenden Bildinhalt optimiert werden können. So sind bei der Darstellung von Benutzeroberflächen von Computern eher langsam bewegte Objekte mit hohen Ortsfrequenzen zu erwarten, während bei der Wiedergabe von TV-Sequenzen aufgrund von Kameraunschärfe häufig schnelle Objekte mit geringeren Ortsfrequenzen vorhanden sind.

Im Gegensatz dazu werden im Bereich der Zwischenbildfilterung normalerweise keine eigentlichen Filter eingesetzt, sondern eine Wandlung wird lediglich in der Form durchgeführt, dass einzelne Bilder der Sequenz mehrfach dargestellt oder ausgelassen werden, um eine Verringerung, bzw. Erhöhung der Bildwechselfrequenz zu erreichen. Dies wird in dem Augenblick problematisch, wenn die Bilder in Form von Halbbildern vorliegen. Hier hat eine Wiederholung zur Folge, dass die Halbbilder mit falscher Rasterlage angezeigt werden.

Die Bildskalierung dient der Änderung der Größe der Einzelbilder. Die zu erzielende Bildqualität hängt hier stark von der Komplexität des verwendeten Filters ab. Auf der anderen Seite kann ein Skalierungsfilter Bildstörungen, die in einer der beiden vorgeschalteten Stufen entstehen nicht ausgleichen. Es ist daher darauf zu achten, dass bereits bei der Zwischenzeilenfilterung die bestmögliche Bildqualität erreicht wird. Wird als Ausgabeformat wieder ein Halbbildformat gewünscht, so kann das Zeilensprungverfahren im Anschluss (oder in Kombination) mit der Skalierung durchgeführt werden.

Selbst wenn ein Halbbildformat in ein anderes konvertiert werden soll, ist es notwendig, über den Schritt der Zwischenzeilenfilterung zunächst Vollbilder zu erzeugen, und diese dann weiterzuverarbeiten, um bestmögliche Wandlungsqualität zu garantieren.

Hochqualitative Video-Formatkonversion stellt hohe Ansprüche an Rechenleistung und Speicheraufwand. So werden die qualitativ besten Ergebnisse mit sogenannter bewegungskompensierender Format-Konversion erzielt, bei der die Bewegung eines jeden Bildpunktes innerhalb eines Bildes berechnet und bei der Filterung berücksichtigt wird.

7.4 HiCon Video-Format-Konverter

Im Heinrich-Hetz-Institut wird zurzeit ein bewegungskompensierender Video-Formatkonverter als Ein-Chip-Lösung namens „HiCon“ integriert. Basis für diese Entwicklung bildet ein Software-Referenzmodell welches auf einer Windows-Plattform implementiert wurde. Mit Hilfe dieser Software lassen sich die verschiedenen Betriebsabläufe des zukünftigen Konverter-Chips simulieren, bzw.

Ergebnisse der Hardware-Simulation evaluieren. Aufgrund gesteigerter Rechenleistungen im PC-Bereich lässt sich das Referenzmodell aber auch als eigenständige Applikation für die Video-Formatkonversion einsetzen. An eine Konversion in Echtzeit, wie sie mit dem Konverter-Chip möglich sein wird, ist allerdings (noch) nicht zu denken. In der aktuellen Version wird eine Verarbeitungszeit von ca. 1:50 bei TV-Material erreicht, d.h. für ein Video von einer Minute Dauer werden ca. 50min Verarbeitungszeit gebraucht.

Wie bereist erwähnt, stellt die HiCon-Software ein Arbeitsmodell eines Formatkonverters dar, bei dem sämtliche Betriebsparameter von außen kontrolliert werden können (*siehe Abb. 22*). Für den Einsatz als Postproduktions-Tool ist die Applikation in dieser Form nicht geeignet, da sie zuviel Kenntnis über die Arbeitsweise voraussetzt. Für das GigaMedia-Projekt wurde daher der Funktionsumfang reduziert und für die möglichen Konfigurationen wurden die optimalen Betriebsparameter ermittelt. Somit können auch ohne spezielle Vorkenntnisse bestmögliche Ergebnisse erzielt werden.

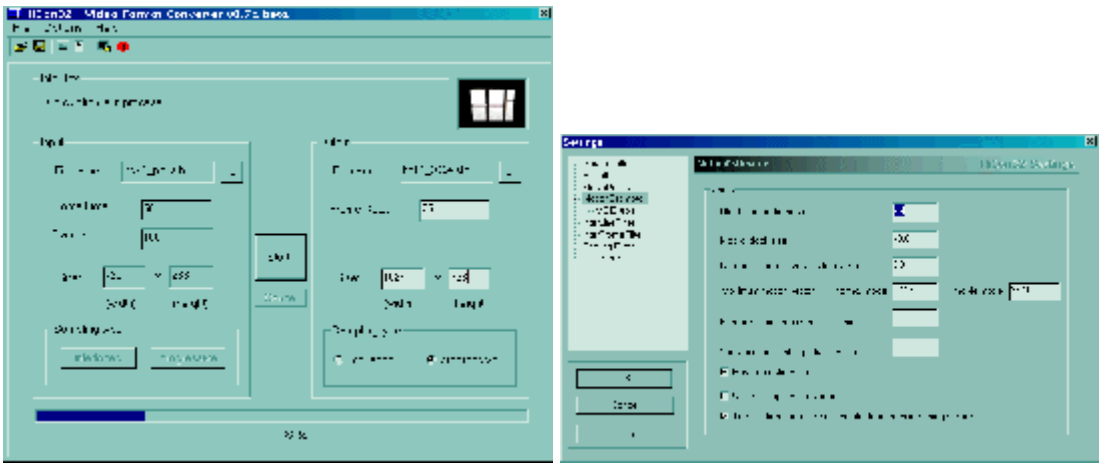

Abb. 22: Grafische Oberfläche der HiCon32-Applikation

Bei HiCon handelt es sich um einen bewegungskompensierenden Video-Formatkonverter. Die berechneten Bewegungsinformationen werden benötigt, um in den Interpolationsfiltern auch Bildinformationen aus angrenzenden Bildern verwenden zu können.

Beim De-Interlacing können auf diese Weise die Zwischenzeilen nahezu fehlerfrei interpoliert werden, da auf vorhandene Zeilen in benachbarten Bildern bewegungsrichtig zugegriffen wird. Einfache Verfahren, die statisch auf entsprechende Zeilen zugreifen erzeugen auf diese Weise Bildfehler durch Auszahnungen an schnell bewegten vertikalen Kanten und Moiree-Störungen in Bereichen mit hoher örtlicher Auflösung. Ein Beispiel hierzu ist in *Abb. 23* dargestellt.

a) statische Interpolation b) HiCon bewegungskompensierend

Abb. 23: Vergleich von De-Interlacing-Verfahren

Bei der Wandlung der Bildwechselfrequenz findet eine echte Zwischenbild-interpolation statt, d.h. es werden fehlende Zwischenbilder komplett neu erzeugt. Die gängigste (und einfachste) Methode, die Bildrate zu erhöhen, besteht darin, einzelne Bilder zu wiederholen. Dies macht sich aber in der Videosequenz als störendes ruckeln bemerkbar. Nur wenn die Zwischenbilder bewegungsrichtig interpoliert werden, bleibt eine fließender Bewegungsablauf gewährleistet. HiCon verfügt darüber hinaus über eine MovieMode-Kompensation[3], bei der ein 2:2-Pulldown zunächst rückgängig gemacht wird und im Anschluss Zwischenbilder neu generiert werden. Auf diese Weise kann der Bewegungseindruck von Filmsequenzen (z.B. auf DVD) nachträglich noch deutlich verbessert werden.

Abgeschlossen wird eine Konversion durch eine hochqualitative Skalierung. Hierbei wird ein separierbares Filter eingesetzt, bei dem horizontale und vertikale Skalierungsverhältnisse frei gewählt werden können.

[3] Bei der Konversion von Filmmaterial (24Hz, progressiv) nach TV (50Hz, interlaced) wird üblicherweise der sogenannte MovieMode (auch „2:2 Pulldown“ genannt) verwendet. Hierzu wird der Film schneller abgespielt (25Hz) und ein Vollbild wird auf zwei Halbbilder verteilt. Folge davon ist ein deutlich erkennbares Ruckeln in schnell bewegten Szenen.

8 Demonstrator

Stellvertretend für eine ganze Reihe von Möglichkeiten, hochratige Übertragungstechniken zur verteilten Produktion zu nutzen, wurde im Rahmen des Projekts in Zusammenarbeit mit dem Produktionshaus *das Werk* ein beispielhafter Ablauf entworfen, gerätetechnisch aufgebaut und im Betrieb zwischen *IRT, FOKUS* und *HHI* erprobt.

Ein Kunde gibt einem Produzenten den Auftrag, Werbespots für ihn zu produzieren. Diese Werbespots sollen in Supermärkten auf Plasma-Bildschirmen mit sehr hoher Qualität wiedergegeben werden. Die Plasmabildschirme haben eine XGA-Auflösung und eine progressive Abtastung (1024×768 Bildpunkte, progressiv 1:1, 60Hz), weshalb eine direkte Einspeisung von TV-Signalen (720×576 Bildpunkte, Zeilensprung 2:1, 50 Hz) nicht möglich ist (*siehe auch Abschnitt 7.1*). Andererseits soll die Produktion aus Kostengründen im TV-Format erfolgen.

Der Produzent kann die gesamte Produktion nicht vollständig anfertigen, da ihm die teuren Post-Produktionseinrichtungen fehlen. Er hat aber eine Postproduktions-Firma als Geschäftspartner, mit der er über das GTB vernetzt ist. Ebenso ist der Kunde an das GTB angeschlossen, so dass der Produzent sich entschließt, eine verteilte Produktion über das Netz hinweg vorzunehmen, weil dies Kosten und Zeit spart.

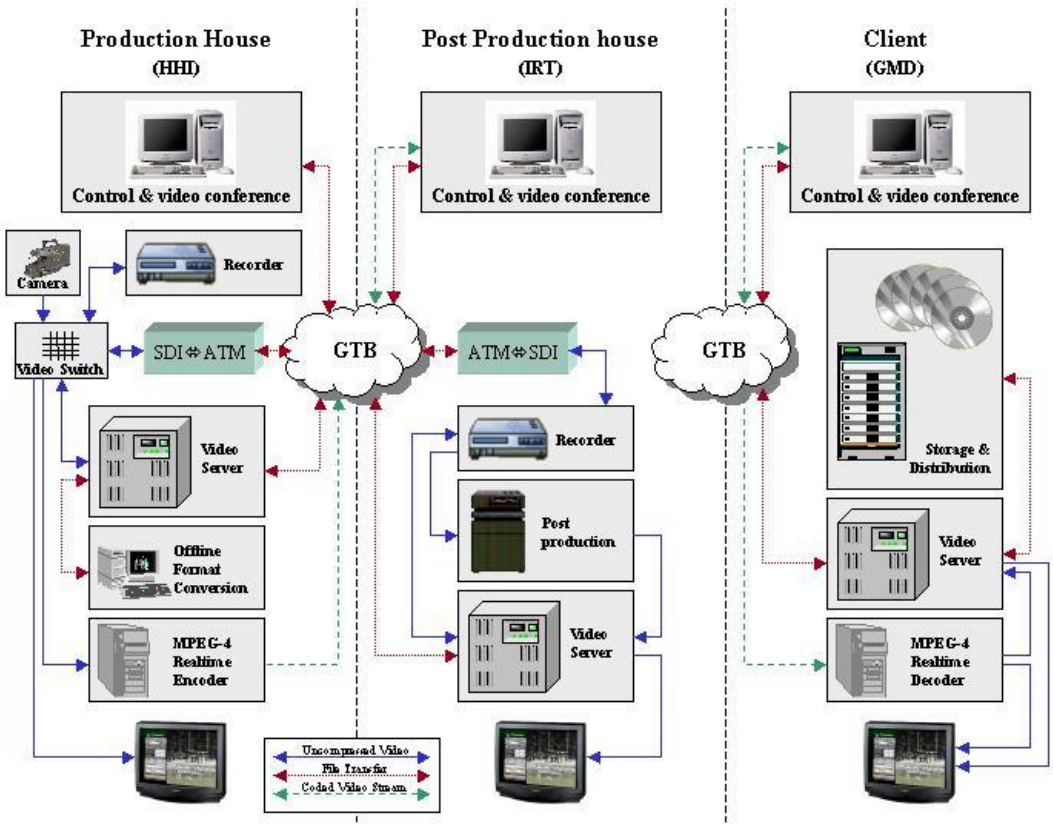

Abb. 24: Netzstruktur und Gerätepark für die verteilte Produktion eines Werbefilms

8.1 Geräte- und Netzinfrastruktur

Der für die gesamte Produktion an den drei Standorten benötigte Gerätepark ist in *Abb. 24* dargestellt.

Die Bildspeichersysteme beim Produzenten und Post-Produzenten sowie der Video-Switch, die off-line Format-Konversion und der MPEG-4 Echtzeit-Encoder beim

„Produzenten“ sind über das Netz fernsteuerbar und können daher von jedem Standort aus (fern-)bedient werden. Die Echtzeit-Videoübertragung im SDI-Format geschieht mit Hilfse von SDI-ATM-Adaptern, die vom *IRT* entwickelt wurden, mit 270 Mbit/s über das Netz hinweg. Der File-Transfer von verarbeiteten Videodaten geschieht zwischen den Video-Servern an allen drei Standorten.

8.2 Beschreibung des Web-Interfaces

Die Hard- und Softwarekomponenten werden im Produktionshaus über einen Video-Server gesteuert, der gleichzeitig als Web-Server im GTB agiert. Auf diese Weise können Webseiten abgerufen werden, welche über aktive Inhalte eine Steuerung einzelner Komponenten ermöglichen.

Videosequenz-System

Das Web-Interface für die Steuerung der Hardware-Komponenten arbeitet sequenzbasiert, d.h. es wird immer eine Videosequenz ausgewählt mit der eine bestimmte Aktion ausgeführt wird. Eine zentrale Rolle kommt hierbei dem Videosequenz-System (VSS), einem festplattenbasierten Bildspeichersystem, zu. Für das VSS wurde grundsätzlich die Funktionalität eines Videorekorders nachgebildet. So gibt es eine Webseite zum Abspielen und eine zum Aufnehmen von Sequenzen. In *Abb. 25* ist die Wiedergabesteuerung dargestellt. Neben den Tasten „Play“, „Pause“ und „Stop“ kann zusätzlich der Modus der Schleifendarstellung gewählt werden. Sequenzen werden grundsätzlich in einer Endlosschleife wiedergegeben. Dies geschieht entweder immer wieder vom Anfang („loop“) oder im Wechsel von Vorwärts- und Rückwärtsanzeige („shuttle“).

Um Sequenzen nachverarbeiten zu können (MPEG-Codierung, Formatkonversion), müssen diese auf dem Videoserver gespeichert werden. Entsprechend besteht die Möglichkeit, Sequenzen vom Video-Server auf das VSS zu laden (*Abb. 25).*

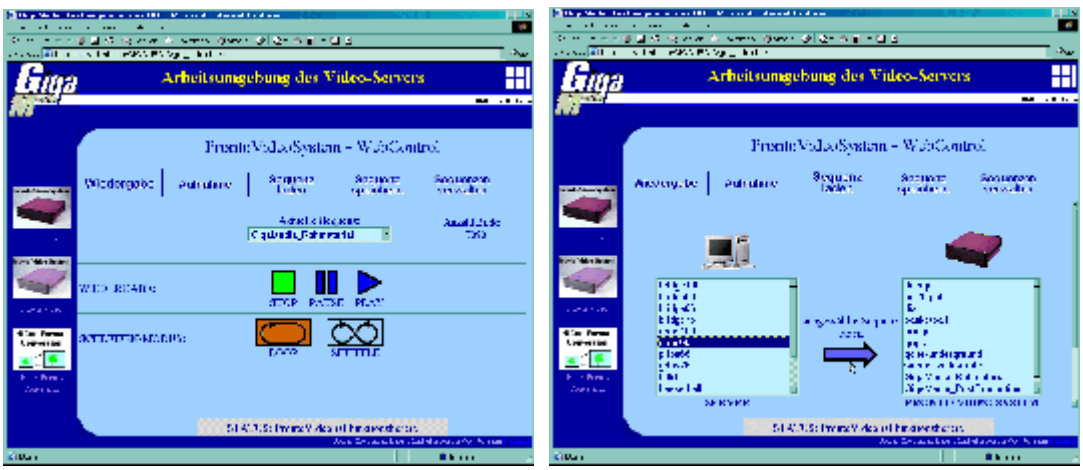

Abb. 25: Web-basierte Kontrolle des Video-Sequenz-Systems (VSS)

HiCon-Formatkonverter

Der in Kapitel 7 beschriebene Formatkonverter wurde als Postproduktionstool in die Web-Oberfläche integriert um eine Fernsteuerung zu ermöglichen.

Prinzipiell ist der Formatkonverter in der Lage, beliebige Ausgabeformate zu erzeugen. Allerdings sind unter Umständen aufwändige Einstellungen der Betriebsparameter nötig. Aus diesem Grund wurde ein reduzierter Funktionsumfang in das Web-Interface übernommen. Der Anwender hat die Möglichkeit, Sequenzen mit 60Hz, 75Hz oder 100Hz im Zeilensprung- oder im progressiven Format zu erzeugen. Die örtliche Bildauflösung kann frei gewählt werden. *Abb. 26* zeigt eine Darstellung des HiCon-Web-Interfaces.

Video-Switch

Weiterhin besteht die Möglichkeit, einen Video-Switch zu steuern, mit der verschiedene Quellen an unterschiedliche Senken geschaltet werden können. Nachdem am Ende der Postproduktionskette die produzierte Sequenz in das Zielformat konvertiert wurde (hier: XGA) wird eine spezielle Hardware benötigt, um diese auch in Echtzeit abspielen zu können. Hierfür wird ein RAM-basiertes Speichersystem eingesetzt. Aufgrund des höheren Datendurchsatzes kann das Ausgabeformat bis zu einer HDTV-Auflösung frei gewählt werden. Über das Web-Interface kann der Ausgang des VSS auf verschiedene hochauflösende Displays (Computer-Bildschirm, XGA-Projektor, Plasma-Display) geschaltet werden (*Abb. 26*).

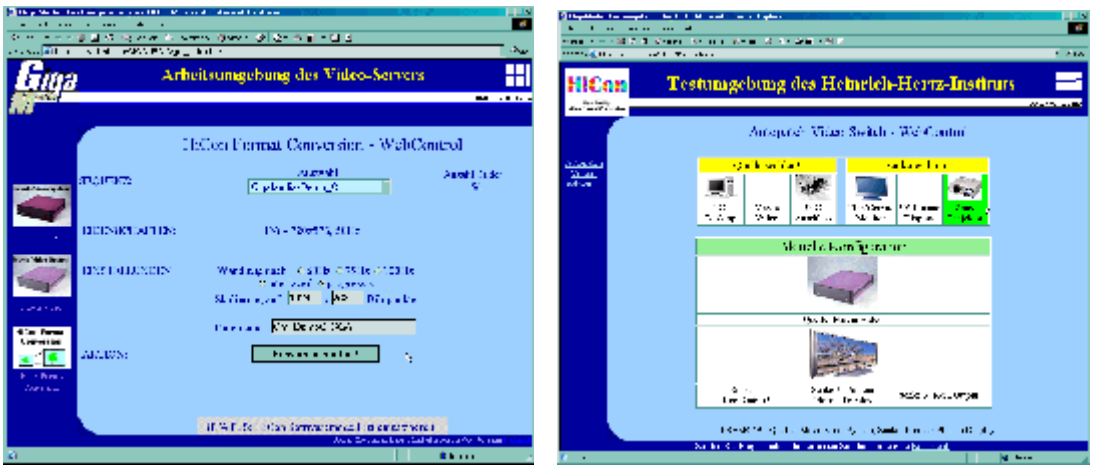

Abb. 26: Web-basierte Kontrolle: Formatkonversion und Sterueung des Video-Switches

8.3 Zeitlicher Ablauf der verteilten Produktion

Der zeitliche Ablauf der Produktion sowie die dabei involvierte Netzbenutzung sind in *Abb. 27* dargestellt.

Die einzelnen Szenen werden im Produktionshaus gefilmt und auf einer MAZ aufgezeichnet. Nach dem Abfilmen aller Szenen werden diese über die SDI-ATM-Strecke des GTB zum Post-Produktionshaus übertragen. Diese Übertragung wird von einer Videokonferenz zwischen den Technikern auf beiden Seiten begleitet. Die Ansteuerung der beiden Speichergeräte kann wahlweise vom Sende- oder Empfangsort erfolgen. Im Postproduktionshaus kann dann die Nachverarbeitung (Schneiden, Einfügen von Effekten, Grafik, Schriften usw.) stattfinden. Anschließend wird der fertiggestellte Film über SDI-ATM zum Produktionshaus zurückübertragen.

Das fertiggestellte Material wird nun MPEG-4 codiert und während einer Dreier-Konferenz zwischen Produktionshaus, Post-Produktionshaus und Kunden diskutiert.

Dabei kann insbesondere von den MPEG-4 Player-Eigenschaften wie schnelles Vor- und Zurückspulen sowie Standbild Gebrauch gemacht werden, um Passagen in der Produktion zwischen den drei Partnern im Detail zu diskutieren.

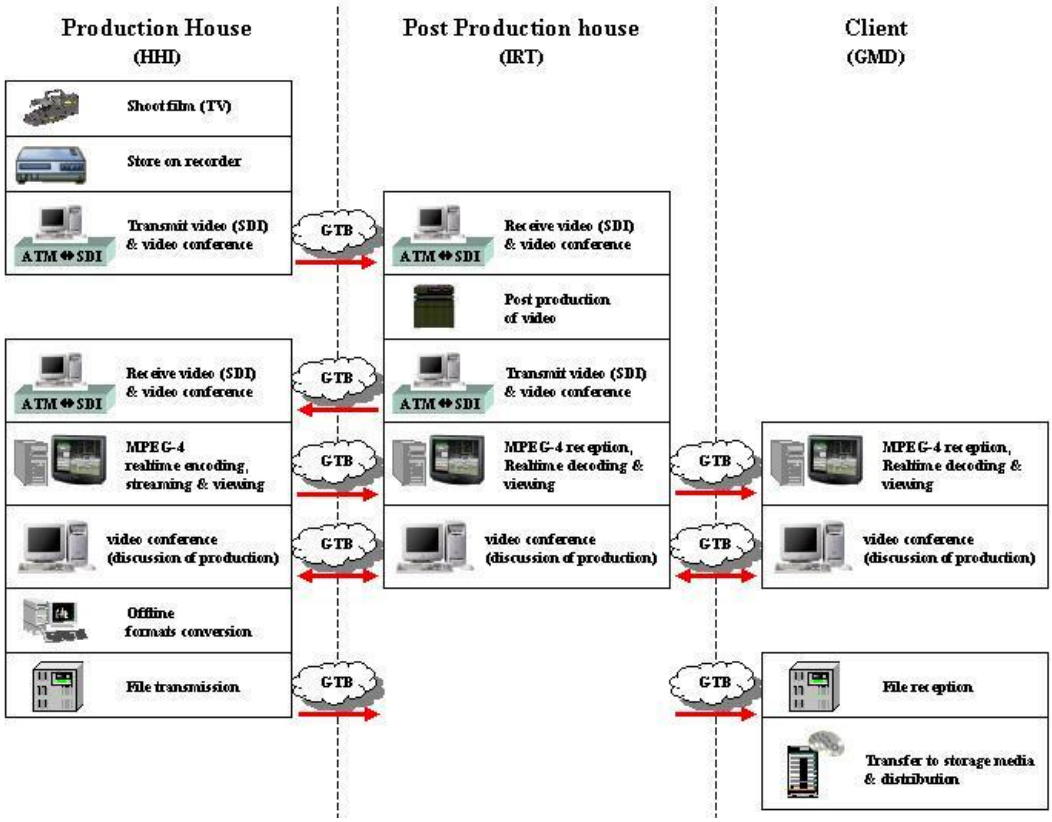

Abb. 27: Zeitlicher Ablauf einer Werbefilm-Produktion

Im Rahmen der Diskussion werden dann letzte Änderungen für das Endprodukt besprochen. Nach einem weiteren Nachbearbeitungsschritt durch das Post-Produktionshaus wird nun der fertige Werbefilm wieder zum Produktionshaus geschickt. Mittels MPEG-4 Streaming erfolgt eine Übertragung zum Kunden, der den Werbefilm in dieser Form akzeptiert. Es erfolgt nun eine off-line Formatkonversion auf das Ausgabeformat (1024×768 Bildpunkte, progressiv 1:1, 60Hz) der Plasma-Displays. Das konvertierte Material wird anschließend in hoher Qualität mit 20 Mbit/s MPEG-2 codiert. Hierzu wird ein MPEG-2-Encoder eingesetzt, der ebenfalls off-line arbeitet, da sich der Einsatz von Hardware-Encodern für solche Spezialformate nicht lohnt. Die komprimierten Daten werden in ein Datenfile geschrieben und über eine 2 Mbit/s Verbindung zum Kunden übertragen. Von dort erfolgt eine Überspielung zu den Geräten, die in den Supermärkten installiert sind.

Bei der Übertragung des komprimierten Videos werden MPEG-2 Transportströme mit Hilfe des Real-time Transport Protocol (RTP) über IP übertragen. Die Steuerung erfolgt dabei über das Real-time Streaming Protocol (RTSP) (siehe auch Kapitel 6).

8.4 Erfahrungen aus dem Echtzeitbetrieb

Bei den Untersuchungen im Rahmen des Demonstratortests konnte in der Regel eine völlige Fehlerfreiheit der Übertragung festgestellt werden.

Das Aufsetzen der ATM-Verbindung zwischen beteiligten Partnern erfolgte reibungslos ohne größere Probleme. Die Qualität des übertragenen Videosignals war für das untersuchte Applikationsszenario absolut geeignet.

Die Steuerung der Übertragung über das Web-Interface funktionierte weitestgehend fehlerfrei. In den Tests konnte die Benutzerfreundlichkeit der Oberfläche untersucht und anhand der Ergebnisse optimiert werden.

Die parallele Videokonferenz, über die die beteiligten Techniker an den drei Standorten kommunizieren konnten, war ein hilfreiches Instrument, mit dessen Hilfe eine Reihe von Problemen besprochen und gelöst werden konnte.

Insgesamt kann gesagt werden, dass sich das im GigaMedia-Projekt entwickelte Konzept der verteilten Produktion in den begrenzten Tests, die die beteiligten Partner durchführen konnten, bewährt hat. Allerdings müsste dieses Konzept im Rahmen einer realen Produktion durch professionelle Anwender noch einmal überprüft und ggf. optimiert werden.

9 Öffentlichkeitsarbeit und Veröffentlichungen

Das Projekt GigaMedia wird im Web unter der Adresse http://www.fokus.gmd.de/glip/gigamedia präsentiert. Hier finden sich unter der Rubrik ‚Info' die Folien der Abschlussvorträge, die eine gute Übersicht über das Projekt bieten, ähnlich diesem Dokument.

Außerdem wurde das Projekt GigaMedia bei den folgenden Veranstaltungen präsentiert und in den entsprechenden Proceedings bzw. Zeitschrift publiziert:

- 34. DFN-Betriebstagung, 6./7. Februar 2001, Berlin, http://www.dfn.de/dfn/dfn-bt/
- TERENA Networking Conference 2001, 14-17.Mai 2001 , Antalya, http://www.terena.nl/conf/tnc2001/proceedings/session6a.html
- 9. Dortmunder Fernsehseminar, 24.-26. Sept. 2001, Dortmund, http://www-kt.e-technik.uni-dortmund.de/Veranstaltungen/fernseh01/
- "Projekt GigaMedia: Gigabit-Media-Dienste für Kooperative Postproduktion von Film und Video", DFN Journal, Heft 54, November 2000

10 Referenzen

[Apple01] http://www.apple.com/quicktime/

[ATM98] M. Orzessek, P. Sommer: „ATM & MPEG-2: intergrating digital video into broadband networks", Prentice-Hall 1998

[Diffserv] http://www.ietf.org/html.charters/diffserv-charter.html

[FeKP98] Wu-chi Feng, Brijesh Krishnaswami, Arvind Pradhudev: "Proactive Buffer Management for the Streamed Delivery of Stored Video", ACM Multimedia, 1998

[FeRe97] Wu-chi Feng, Jennifer Rexford: "A Comparison of Bandwidth Smoothing Techniques for the Transmission of Prerecorded Compressed Video", IEEE INFOCOM, April 1997

[flex] http://www.ma.adfa.oz.au/Local/Info/flex/flex_toc.html

[GMED_D12] Sebastian Zander et al.: „Konzepte für Videoübertragung, Prefetch und Previewing", DFN Gigabit-Media-Dienste für kooperative Postproduktion von Film und Video (GigaMedia), November 2000

[ISO93] ISO/IEC JTC 1/SC29/WG 11 Editorial Group Document 658-660: „MPEG-2 Standard", ISO/IEC 1993

[LinDiff01] http://diffserv.sourceforge.net/

[MaFlo97] J. Mahdavi, S. Floyd: "TCP-Friendly Unicast Rate-Based Flow Control", Technical note sent to the end2end-interest mailing list, Januar 1997

[NetFilt01] http://netfilter.samba.org/

[PFTK98] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose: "Modeling TCP throughput: a simple model and its empirical validation", ACM SIGCOMM, September 1998

[RaTh98] R. Ramanujan, K. Thurber: "An Active Network Based Design of a QoS Adaptive Video Multicast Service", SCI'98, 1998

[ReTo99] Jennifer Rexford, Don Towsley: "Smoothing Variable-Bit-Rate Video in an Internetwork", IEEE/ACM, 1999

[RFC1889] H. Schulzrinne, S.Casner, R. Frederick, V. Jacobson: „RTP: A Transport Protocol for Real-Time Applications“, RFC1889, Januar 1996

[RFC1890] H. Schulzrinne: „RTP Profile for Audio and Video Conferences with Minimal Control“, RFC1890, Januar 1996

[RFC2068] R. Fielding, J. Gettys, J. Mogul, H. Nielsen, T. Berners-Lee: „Hypertext transfer protocol – HTTP/1.1“, RFC2068, Januar 1997

[RFC2250] D. Hoffman, G. Fernando, V. Goyal, M. Civanlar: „RTP Payload Format for MPEG1/MPEG2 Video“, RFC2250, Januar 1998

[RFC2326] H. Schulzrinne, A. Rao, R. Lanphier: „Real Time Streaming Protocol (RTSP)”, RFC2326, April 1998

[RFC2343] M. Civanlar, G. Cash, B. Haskell: „RTP Payload Format for Bundled MPEG“, RFC2343, Mai 1998

[RNKA97] R. Ramanujan, J. Newhouse, M. Kaddoura, A. Ahamad, E. Chartier, K. Thurber: "Adaptive Streaming of MPEG Video over IP networks", IEEE LCN'97, November 1997

[RSFT97] J. Rexford, S. Den, J. Dey, W. Feng, J. Kurose, J. Stankovic, D. Towsley: "Online Smoothing of live variable-bit-rate video" in Proc. Network and OS Support for Digital Audio and Video, pp. 249-257, May 1997

[rtptools] http://www.cs.columbia.edu/~hgs/rtp/

[Strad00] Stradis, Inc., http://www.stradis.com

[SZKT96] J. Salehi, Z. Zhang, J. Kurose, D. Towsley: "Supporting Stored Video: Reducing Rate Variability and End-to-End Resource Requirements through Optimal Smoothing", ACM SIGMETRICS, 1996

[UDPT] http://www.cs.columbia.edu/~lennox/udptunnel/

[VFJF99] B. Vandalore, W. Feng, R. Jain, S. Fahmy: "A Survey of Application Layer Techniques for Adaptive Streaming of Multimedia", Journal of Real Time Systems, April 1999

[Wuerz98] http://www-info3.informatik.uni-wuerzburg.de/

[yacc] http://hera.mni.fh-giessen.de/~hg52/lv/compiler/skript/html/node57.html