Ein Beitrag zu MPEG-4 Broadcast

Ein MPEG-4 Transport im Delivery Multimedia Integration Framework (DMIF)
für das Broadcast Szenario unter Nutzung von IP-Multicast

Dissertation zur Erlangung des akademischen Grades
DOKTOR-INGENIEUR

des Fachbereichs
Elektrotechnik und Informationstechnik
der
FernUniversität in Hagen

Vorgelegt von
Dipl.-Ing. Michael Stepping
geboren in Hachenburg/Westerwald

Hagen 2004

Eingereicht am:	05.07.2004
Tag der mündlichen Prüfung:	01.10.2004
1. Berichterstatter:	Prof. Dr.-Ing. Firoz Kaderali
2. Berichterstatter:	Prof. Dr.-Ing. Bernd Krämer

Berichte aus der Kommunikationstechnik
herausgegeben von Prof. Firoz Kaderali

Michael Stepping

Ein Beitrag zu MPEG-4 Broadcast

Ein MPEG-4 Transport im Delivery Multimedia Integration Framework (DMIF) für das Broadcast Szenario unter Nutzung von IP-Multicast

Michael Stepping
Siegen

Printed in Germany

Michael Stepping, Gustav-von-Mevissen-Straße 28, 57072 Siegen

Telefon (02 71) 37 57 37 - 0 • Internet: www.stepping.de

eMail: Michael.Stepping@stepping.de

Danksagung

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrgebiet Kommunikationssysteme des Fachbereichs Elektrotechnik und Informationstechnik der FernUniversität in Hagen.

Mein besonderer Dank gilt Herrn Prof. Dr.-Ing. Firoz Kaderali. Durch seine Unterstützung bei der Erstellung der Arbeit, die Diskussionen und wertvollen Hinweise hat er diese Promotion überhaupt erst ermöglicht. Herrn Prof. Dr.-Ing. Bernd Krämer danke ich für das Interesse an dieser Arbeit und an der Übernahme des Korreferats.

Die vorliegende Arbeit basiert auf einem EU-Projekt, welches Herr Prof. Dr. Thomas Bonse zum FTK mitbrachte. Dadurch war früh der Grundstein für das Forschungsthema - Broadcast in MPEG-4 - gelegt. Ohne ihn wäre das Thema der Promotion so sicherlich nicht möglich gewesen. Daher mein besonderer Dank an ihn.

Für die interessanten Diskussionen, wertvollen Hinweise und Anregungen zu inhaltlichen Aspekten der Arbeit danke ich den Kollegen Dagmar Sommer, Thomas Paleschke und Christian Grosch. Herrn Peter Roer danke ich für die Ideen zum Thema Verkehrsformung. Herrn Prof. Dr. Werner Poguntke danke ich für das stets offene Ohr und seine Ratschläge bei der Erstellung dieser Arbeit.

Den Kollegen des Lehrgebiets danke ich für die zahlreichen Diskussionen und das intensive Korrekturlesen sowie ihrer Geduld mit mir, namentlich Gerd Steinkamp, Jörg Dora, Biljana Cubaleska, Alex Essoh und Thorsten Kisner. Frau Biljana Cubaleska habe ich auch meine heutige Begeisterung für MATLAB zu verdanken. Ohne diesen Impuls wäre es schwer geworden, die zahlreichen Auswertungen auch in grafischer Form zu erstellen.

Herrn Herwig Jahn, Herrn Mostafa Dashti und Frau Farkhondeh Mehdizadeh danke ich für die Erstellung ihrer Diplomarbeiten, womit sie wertvolle Vorarbeiten geleistet haben. Die Herren Rainer Dampmann, Stefan Neveling und Taher Nasched haben wesentlichen Anteil an der Implementierung des Prototypen durch das Umsetzen meiner Konzepte und Ideen. Ohne deren Hilfe und Ideen hätte der Prototyp nicht die heutige Stabilität erreicht. Den Herren Jens Vollmert und Marcus Ladwig danke ich für die Hilfe bei der Fertigstellung von Publikationen, der Erstellung von Grafiken sowie dem Korrekturlesen deutscher und englischer Texte. Die Herren Tekin Birduezen, Richard Sotke und Claus Behrens haben mich rund um das Thema MPEG-4 hilfreich unterstützt.

Frau Andrea Frank danke ich für die Erstellung von Hilfswerkzeugen und Herrn Firas Nasched für die Programmierung der Word-Makros, ohne die die Erstellung der Arbeit mit Word ein Desaster geworden wäre. Herrn Gerd Tübben danke ich für die umfassende Unterstützung mit technischem Equipment, sowie Herrn Matthias Schneider für die Versuche mit der Universität Siegen.

Die Arbeit am Lehrgebiet mit Herrn Kaderali, den Kollegen, Diplomanden, studentischen Hilfskräften und Praktikanten war immer sehr angenehm. Ich habe in den vergangenen Jahren stets sehr gerne mit ihnen zusammengearbeitet.

Für die Zusammenarbeit in den MPEG-Gremien, verschiedenen EU-Projekten und die Unterstützung mit MPEG-4-Playern und Autorenwerkzeugen darf ich den Herren Stefano Battista, Guido Franceschini, Jean-Claude Dufourd und Jürgen Deicke danken.

Schließlich gilt mein besonderer Dank meiner Frau, meinen Kindern, meiner elterlichen Familie und allen, die durch ihre Unterstützung unterschiedlichster Art die Arbeit mit vorangebracht haben und zu deren Gelingen beigetragen haben.

Schließlich und endlich enthält die Arbeit 13 Definitionen, 53 Gleichungen, 57 Tabellen und 282 Abbildungen.

Siegen, im Juli 2004

Kurzfassung

Die dieser Arbeit zu Grunde liegende Idee kann mit dem Schlagwort „Fernsehen über das Internet“ bezeichnet werden. Die stetig zunehmende Vernetzung der Haushalte ermöglicht es mittlerweile einer breiten Bevölkerungsschicht multimedial über das Internet zu kommunizieren.

Der Standard MPEG-4 (*Moving Pictures Experts Group*) liefert eine komprimierte, einheitliche Beschreibung (BIFS) für interaktive und objektbasierte Multimedia-Anwendungen (VRML) – die *MPEG-4 Applikation*. Diese multimedialen Datenströme werden als Multimedia-Objekte (AAC, AC-3, H.264, H.263, H.261, MPEG-2, ...) behandelt.

Der Transport dieser Datenströme geschieht mittels des Rahmenwerks *Delivery Multimedia Integration Framework* (*DMIF*), welches die eigentliche Quelle der Datenströme vor dem audio-visuellen Darstellungswerkzeug (Player) verbirgt. Damit erfolgt mit der Schnittstelle *DMIF Application Interface* (*DAI*) die Trennung in einen medien- und einen netzunabhängigen Teil.

Hier setzt die Arbeit an. *MPEG-4 Broadcast* ist der unidirektionale, gleichzeitige Versand von Datenströmen von einer Quelle an viele Empfänger (Push-Szenario). Hierfür wird in dieser Arbeit IP-Multicast verwendet, welches die Datenströme erst in den an der Verteilung beteiligten Routern vervielfältigt.

Die vorgestellte unidirektionale DMIF-Signalisierung für das MPEG-4 Broadcast-Szenario ist als *Generische Signalisierung* für schmalbandige Netze und als *IP-Multicast Signalisierung* (SAP/ SDP) für breitbandige Netze entwickelt worden. Transportkanalinformationen wie Adresse (Socket) und Dienstgüte sowie Kanalbeziehungen zwischen den logischen Kanälen (Elementar-Datenstrom) und den Transportkanälen werden übermittelt.

Für das bei dem Paradigma *Fernsehen* übliche spätere Einschalten eines Empfängers in eine laufende Übertragung, das sogenannte *Late Tuning-In*, überträgt die Signalisierung periodisch mit Karussell-Techniken die benötigten Informationen mehrfach. Auch das MPEG-4 typische Wurzelelement *InitialObjectDescriptor* (*IOD*) wird periodisch übertragen.

Für den Transport der Datenströme im Broadcast-DMIF wird das Echtzeit-Transportprotokoll *Real-Time Protocol* (*RTP*) über UDP/IP-Multicast verwendet und leistet die Inter- und Intra-Datenstrom-Synchronität. Aus den MPEG-4 Datenströmen werden die Zeitstempel aus dem spezifischen *Synchronisation Layer* entnommen. Die Informationen über den wahlfreien Zugriffspunkt (*Random Access Point*) müssen zusätzlich angepasst werden. Für das spätere Einschalten, *Late Tuning-In*, müssen des Weiteren statische Objekte (Stillbilder, ...) in einem Karussell periodisch übertragen werden. Beliebig große MPEG-4 PDUs werden in RTP-PDUs fragmentiert und reassembliert. Hierfür wurde das RTP-Protokoll erweitert. Das FlexMux-Werkzeug zum Multiplexen kleiner MPEG-4 PDUs wird nicht berücksichtigt.

UDP hat die Eigenschaft der Verkehrsverdrängung (*non-TCP-friendly traffic*), daher wird der Ausgangsverkehr der Quelle der Datenströme (Broadcast-Server) mittels der *Server Limited Bitrate* (SLB) limitiert. Diese Bitrate wird eingeführt und ist die obere Grenze der aktuellen Ausgangs-Spitzen-Bitrate beziehungsweise die maximale Server-Ausgangsdatenrate und wird nicht überschritten. Mittels Verkehrsglättung und -formung (Traffic Shaping, Rate Shaping) wird unter Verwendung verschiedener Scheduling-Algorithmen (FIFO, WRR, ...) der ankommende MPEG-4 Verkehr unter Ausnutzung statistischer Eigenschaften (Jitter, Delay) geglättet.

Für das Traffic Shaping wurde nicht der bekannte *Sliding Window*-Mechanismus, sondern die im Rahmen dieser Arbeit entwickelte *Virtuelle Blockierzeit* verwendet. Dieser Zusammenhang ermittelt aus der Größe der zu übertragenden PDU in Verbindung mit der *SLB* die virtuelle Übertragungszeit. In dieser Zeit darf keine weitere PDU den Broadcast-Server verlassen. Reale Einflüsse (Betriebssystem, Festplattenzugriffe, Netzwerkkarte) führen zu der Verfeinerung der *Realen Blo-*

ckierzeit. Empfängerseitig werden die Datenströme wieder entzerrt und in Echtzeit an den Player abgegeben.

Mittels der Verkehrstheorie wird gezeigt, dass die Puffergröße und die Serververweilzeit der PDUs abgeschätzt werden können und nur von der Verkehrsankunftsrate am Broadcast-Server und den statistischen Eigenschaften der Varianzen der Ankunfts- und Bedienrate abhängen. Ebenso wird gezeigt, dass die Größe des Client-Puffers in der Größenordnung des Server-Puffers liegen sollte.

Es wird ein modularer Demonstrator mit Software-Engineering-Methoden zur Verifikation der theoretischen Ansätze erstellt. Dieser besteht aus den Modulen *Broadcast-Server*, *Broadcast-DMIF-Server*, *Broadcast-DMIF-Client* und Player. Als Programm für die audio-visuelle Darstellung werden verschiedene MPEG-4 Player und als Datenströme werden selbst generierte MPEG-4 Applikationen verwendet. Hierbei werden bestehende Datenströme in MPEG-4 Audio- und Video-codierte Datenströme transcodiert. Die verkehrstheoretischen Ansätze und Abschätzungen, insbesondere das Traffic Shaping, werden mit dem Prototypen verifiziert.

Die entstandene Software ist mit den Methoden der objekt-orientierten Analyse, des Designs und der Programmierung (OOA, OOD, OOP) nach *Object Modelling Techniques* (*OMT*) erstellt worden. Die Ergebnisse wurden mit der *Unified Modelling Language* (*UML*) dokumentiert.

Der Demonstrator ist für Windows- und Linux-Betriebssysteme in portablem Source-Code in C++ erstellt. Dazu sind insbesondere die betriebssystemspezifischen Funktionen und Methoden in portablen Klassen gekapselt.

Abstract

The basic idea of this thesis can be described as “Television via the Internet”. The continuous growth of interconnected households enables a broad audience to use multimedia tools within the Internet.

The standard MPEG-4 (*Moving Pictures Experts Group*) provides a unique, compressed description (BIFS) for interactive and object-oriented Multimedia-Applications (VRML), the so called *MPEG-4 Application*. These multimedia streams are treated as Multimedia-Objects (AAC, AC-3, H.264, H.263, H.261, MPEG-2, ...).

These streams are transported using *Delivery Multimedia Integration Framework* (DMIF), which hides the original source of the streams from the audio-visual player. With this distinction a media-unaware part and a network-unaware part is separately served. The boundary is the *DMIF Application Interface* (*DAI*).

The discussions presented within this thesis are based on the described architecture. *MPEG-4 Broadcast* describes the unidirectional, simultaneous distribution of streams from one source to multiple receivers (Push-Scenario). This thesis is based on IP-Multicast mechanisms, which multiply data streams at the distribution-involved routers only.

The presented DMIF signalling within the *MPEG-4 Broadcast Scenario* is developed as a *generic signalling* for narrowband transmissions as well as an *IP-Multicast signalling* (SAP/SDP) for broadband networks. Transport channel information like addresses (sockets) and Quality of Service (QoS) information are transmitted as well as a mapping of logical channels (so called Elementary Streams) to the underlying transport channels.

To enable the well-known “Television” paradigm in the given environment, the *Late Tuning-In* of a terminal is supported by signalling information which announces the required information periodically in a carrousel. Additionally, the typical MPEG-4 root element *InitialObjectDescriptor* (*IOD*) is transmitted periodically as well.

The transport of the streams via the Broadcast-DMF is carried out by the *Real-Time Protocol* (*RTP*) via UDP/IP-Multicast. RTP provides the inter- and intra-stream-synchronicity. The time-stamps in an MPEG-4 stream are extracted from the specific *Synchronisation Layer.* Furthermore, the *Random Access Point* is modified. To enable the Late Tuning-In, static objects (still images) are carried out in a carrousel, too. MPEG-4 PDUs of any size are fragmented and re-assembled into RTP-PDUs. Therefore, the RTP-Protocol was extended accordingly. The FlexMux-Tool to multiplex small MPEG-4 PDUs is not considered in this thesis.

UDP has the characteristic to supersede traffic (*non-TCP-friendly traffic*), so the outgoing traffic of the source of the streams (broadcast server) is limited by the *Server Limited Bit rate* (SLB). This bit rate is defined and limits the upper border of the current outgoing bit rate. By introducing traffic shaping, rate shaping and different scheduling algorithms (like FIFO, WRR, ...), the arriving MPEG-4 traffic is smoothed using its own statistical features (jitter, delay). Traffic shaping is realised without using the well-known *Sliding Window*-Mechanism. Instead of this, a new *Virtual Blocking Time* is introduced. The PDU size to be transmitted in conjunction with the SLB leads to the virtual transmission time. During this time interval, no other PDU is allowed to leave the broadcast server. But other influences coming from the real environment of such a system (operating system, hard disc access, network interface card) lead to the improved *Real Blocking Time* algorithm. On behalf of the client, the streams are de-jittered and are delivered to the player in real-time.

By deriving relations from basic traffic theory, it is shown that the buffer sizes and the server waiting time for a distinct PDU is assessable and depends only on the arrival-rate at the broadcast

server, the statistical attributes of the variances of arrival rate and serving rate. Another result is that the size of the client buffer should be of the same size as the used server buffer.

The modular concept of the demonstrator is based on software-engineering principles and is required to verify the estimations determined theoretically. The demonstrator implementation consists of the Modules *Broadcast-Server*, *Broadcast-DMIF-Server*, *Broadcast-DMIF-Client* and *Player*. To playback audio-visual streams, several MPEG-4 players are used. The used streams appear as MPEG-4 applications and are hand-made results coming out of different processing steps. Existing streams are additionally transcoded into MPEG-4 audio- and video-coded streams. The deriving estimations and assessments from basic traffic theory are verified. Therefore, the main focus lies on the verification of the traffic shaping algorithms.

The developed software is realised by the use of *Object-oriented Analysis*, *Design* and *Programming* (*OOA*, *OOD*, *and OOP*) methods according to *Object Modelling Techniques* (*OMT*) and documented corresponding to *Unified Modelling Language* (*UML*) specifications.

The demonstrator is developed for Windows and Linux operating systems with portable source code in C++. Operating system specific function calls and methods are encapsulated in portable classes.

Inhaltsverzeichnis

1 Einführung und Übersicht

Die Anfänge dieser Arbeit wurden in dem EU-Esprit-Projekt *MPEG-4 PC* gelegt. Dieses EU-Projekt startete 1996, um den damals noch in der Entwicklung befindlichen Standard MPEG-4 auf einem *Personal Computer* (*PC*) zu implementieren. Ein Hauptaugenmerk bildete damals die Entwicklung von Hardware-beschleunigten MPEG-4 Grafikkarten, so wie dies schon zu Zeiten von MPEG-2 geschah. 1998 stellte sich jedoch heraus, dass kommende Rechner (damals Pentium II, 200 MHz) in kurzer Zeit schon leistungsfähig genug würden, so dass kein Bedarf an einer Hardware-Lösung bestand.

Noch im selben Jahr stimmte die EU-Kommission der Änderung des Projektziels dahingehend zu, dass das Broadcast-Szenario wissenschaftlich auf den praktischen Einsatz hin untersucht werden sollte. So wurde der Grundstein dieser Arbeit gelegt.

Eine Idee und somit der Grund für die Beteiligung der FernUniversität an diesem Projekt war das Ziel, über die bis dato bekannten Verfahren für multimediale Darstellungen hinaus Erfahrungen zu sammeln. Außerdem war das Broadcast-Szenario gerade deshalb interessant, weil es für die FernUniversität für die Verbreitung von multimedialen, elektronischen Lehrmaterialen, wie sie für die neu eingeführten Bachelor- und Master-Studiengänge schon vielfach in den letzten Jahren vorbereitet wurden, kostengünstig genutzt werden kann. Kostengünstig, weil ein einziger Datenstrom die FernUniversität verlässt und nur dieses Transfervolumen die Netzressourcen belastet und später zur Abrechnung gelangt.

Durch Verwendung der IP-Multicast Technologie, die über zehn Jahre nach der grundlegenden Arbeit von Deering [Dee91] in der Praxis erste langsame Konkretisierungen zeigt, wird dieser Datenstrom erst in den angeschlossenen Multicast-Netzen vervielfacht und gelangt dann zu den einzelnen Empfängern. Der Vorteil der Nutzung von Multicast-Techniken gegenüber Broadcast-Techniken ist, dass nur die empfangsbereiten Empfänger auch mit dem Datenstrom versorgt werden.

Die ***M**oving **P**ictures **E**xperts **G**roup (MPEG)* begann 1988 ihre Arbeiten zur Standardisierung von Audio- und Videokompressionstechniken und verabschiedete in der Folge eine Reihe von Normen. *MPEG-4* steht als Synonym für sehr kleine Videobitraten, was das ursprüngliche Ziel der Entwicklung (64 kbit/s - 384 kbit/s) von *MPEG-4 Video* war. Dieses ehrgeizige Ziel konnte erst mit der Weiterentwicklung des *MPEG-4 **A**dvanced **V**ideo **C**oded (AVC)* beziehungsweise ITU.T H.264 realisiert werden. Hier war die maßgebliche Antriebsfeder die Einführung der UMTS-Technologie (Mobilfunk dritter Generation), die Bewegtvideos mit einer nutzbaren Übertragungsrate von 384 kbit/s erstmals in akzeptabler Qualität ermöglichen.

Die Problemstellung dieser Arbeit ist, minimale Datenmengen[1] möglichst maximal zu verbreiten. Dazu folgen zwei Ansätze. Einerseits setzen Mechanismen an den Datenströmen selbst in Form von Kompressionsstandards an und andererseits in der Netzoptimierung wie Multicast. Ein Ansatzpunkt ergibt sich aus der Tatsache, dass Multimedia-Datenströme üblicherweise eine burstartige Verkehrscharakteristik aufweisen. Hierdurch kommt es normalerweise zu einer Überreservierung von Netzressourcen, wenn eine bestimmte Dienstgüte (*Quality of Service*) angestrebt wird. Der Lösungsansatz, der in dieser Arbeit verfolgt wird, ist die Glättung der Datenströme mit dem Ziel, die benötigte Übertragungsrate unter einem wohldefinierten, parametrisierbaren Limit zu halten. Im Broadcast-Szenario führt diese Lösung zu einer verzögerten Versendung der Datenpakete. Andererseits müssen diese in Echtzeit beim Endteilnehmer abspielbar sein. Die Fragen dieser Ar-

[1] beispielsweise komprimierte Videodatenströme

beit stellen sich bezüglich der *Signalisierung*, der *Verkehrsanpassung* und der verwendbaren *Pufferungsmechanismen*.

Die Arbeit gliedert sich wie folgt. Im Kapitel 2 werden die Grundlagen und Randbedingungen eines Broadcast-Szenarios eingeführt sowie die Datenverteiltechnik IP-Multicast und hier insbesondere den Ablauf beim Sender und Empfänger im LAN vorzustellen. Das Kapitel 3 gibt nach einem historischen Abriss der MPEG-Geschichte eine Einführung in die Audio- und Videokompressionstechniken und eine detaillierte Einführung in das MPEG-4 System wieder. Der Transport-Teil von *MPEG-4*, genannt *Delivery Multimedia Integration Framework*, kurz *DMIF* [14496-6], wird in Kapitel 4 eingeführt und unter dem besonderen Blickwinkel des Broadcast diskutiert. Daraus werden die beiden Arbeitsgebiete Signalisierung und Transport abgeleitet.

Im Bereich der Signalisierung wird in Kapitel 5 eine neues Signalisierungsprotokoll für das Broadcast-Szenario vorgestellt, welches mit generischen *DMIF* Protokollmeldungen als auch mit *Session Description Protocol* (*SDP*)-konformen Meldungen entwickelt wird.

Im Bereicht des Transports werden die Datenströme im Transport-System untersucht. Dazu wird im Kapitel 6 die Inter- und Intra-Datenstrom-Synchronisierung, sowie Konzepte für Puffermodelle und theoretische Betrachtungen von Warteschlangen-Systemen vorgestellt. Besonderes Augenmerk wird zum Abschluss auf die faire Nutzung der vorhandenen Netzressourcen gelegt.

Die Software des im Rahmen dieser Arbeit erstellten Prototyps wird in Kapitel 7 vorgestellt. Objektorientierte Analyse, Konzept und Design zeigen die Komplexität des Prototypen.

In Kapitel 8 werden die durchgeführten Versuche untersucht und diese mit den theoretisch erwarteten Werten verglichen.

Kapitel 9 endet mit einer Zusammenfassung und Bewertung sowie einem Ausblick auf die mit dieser grundlegenden Arbeit eröffneten Forschungsgebiete.

Die Beschriftungen aller Abbildungen in dieser Arbeit sind strikt in englischer Sprache beziehungsweise in englischen Fachworten gehalten. Ebenso sind im Text Fachbegriffe in englischer Sprache beibehalten worden und nur dort, wo die Bedeutung nicht sofort ersichtlich ist, kurz in deutscher Sprache erläutert. Dieses Vorgehen vereinfacht eine spätere englische Veröffentlichung für das internationale Fachpublikum.

2 Broadcast und Multicast

In diesem Kapitel werden Distributionsformen für Datenströme über Netze vorgestellt. Insbesondere wird die Verteilung von identischen Datenströmen an eine große Empfängerschar betrachtet.

2.1 Broadcast – Rundsendebetrieb

In diesem Kapitel wird der ***Broadcast***, deutsch *Rundsendebetrieb*, vorgestellt. Es wird absichtlich von Broadcast gesprochen, da damit nicht nur landläufig die drahtlosen Übertragungstechniken via terrestrischem Broadcast gemeint sind.

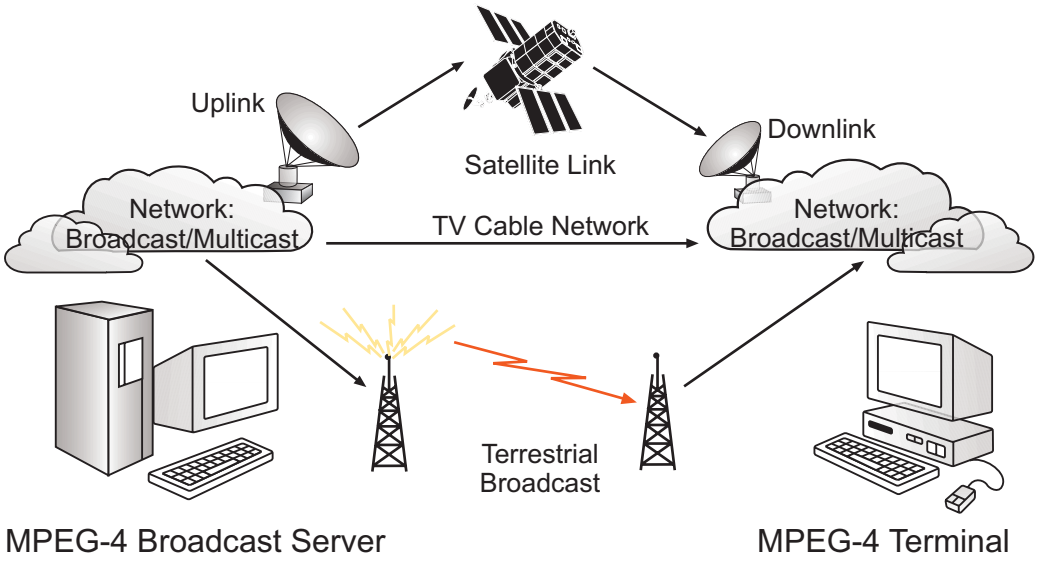


Abbildung 1: Mögliche Netzstruktur für ein MPEG-4 DMIF[1]-Broadcast-Szenario

In Broadcast Umgebungen wie Satellitenstrecken, Fernsehkabelnetzen oder terrestrischen Übertragungsverbindungen existiert im Allgemeinen nur ein einseitiger Informationsfluss vom Sender zum Empfänger. Dieses unidirektionale Szenario beinhaltet die weit verbreiteten Verteilungsstrategien für Informationsdienste (*Push-Dienste*).

In dieser Arbeit wird das Paradigma des konventionellen *Fernsehens* zu Grunde gelegt.

Dieses Szenario hat die sehr wichtige Randbedingung:

- Es besteht eine unidirektionale Verbindung vom Broadcast-Server zum Terminal und
- damit besteht kein Rückkanal vom Terminal (Anwender) zum Broadcast-Server.

Einige wesentliche Eigenschaften und Randbedingungen des *Fernsehens* sollen hier aufgeführt werden:

- kein Rückkanal (damit sind keine Anfragen des Terminals an den Broadcast-Server möglich),
- beliebige Auswahl ausgestrahlter Sendungen,
- nach Selektion eines Fernsehkanals sollen sofort die Inhalte präsentiert werden,
- Verwendung bestehender Internet-Infrastruktur und damit kostengünstiger Einsatz.

[1] Delivery Multimedia Integration Framework (DMIF) wird der für den MPEG-4 Transport zuständige Teil des MPEG-4 Standards genannt.

Mit dieser Aufzählung werden die größten technischen Schwierigkeiten im Umfeld von MPEG-4 beschrieben [BS00], [BS99a], [BS99c]. In den kommenden Kapiteln werden die Möglichkeiten und Einschränkungen von IP-Multicast und dem MPEG-4 Transport – DMIF – im Detail erarbeitet. Lösungen der Sachzwänge, die ein Broadcast-Szenario einführt, werden in Kapitel 5 vorgestellt.

Broadcast Szenario

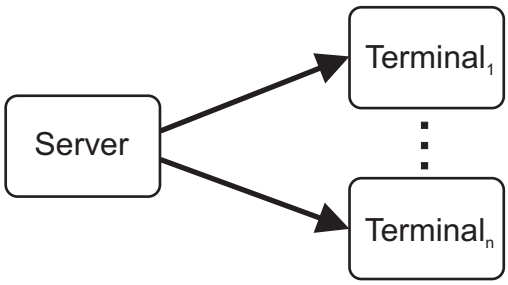


Abbildung 2: Gleichzeitiger Empfang an allen Terminals

Das Broadcast-Szenario zeichnet aus, dass *n* Empfänger ($Terminal_1$ bis $Terminal_n$) gleichzeitig dieselben Datenströme empfangen. Dieses Szenario wurde ursprünglich *Rundsendebetrieb* genannt, was in der deutschen Sprache allerdings mit der rein terrestrischen Ausstrahlung von Radio- oder Fernsehprogrammen über Antennen semantisch vorbelegt ist, auch wenn mittlerweile der Kabelfernseh- und der Satellitenempfang an Bedeutung zugenommen hat.

In dieser Arbeit werden die Terme *Player*, *Terminal* und *Client* synonym für den Empfänger der Datenströme verwandt, also das Empfangsgerät zur audiovisuellen Darstellung der Datenströme für den Betrachter.

Der Begriff *Server* bezeichnet den Sender der Datenströme.

Kanalwahl

Bei dem Begriff *Kanal* soll es sich um audio-visuelle Datenströme, wie in einem *Fernsehkanal*, handeln. Der Anwender kann zwischen mehreren *Programm-Kanälen* beliebig auswählen (*wahlfreier Zugriff*).

Spätes Einschalten (*Late Tuning-In*)

Eine wichtige Anforderung für ein Broadcast-Szenario ist die Möglichkeit, während einer laufenden Übertragung einzuschalten. Solch ein verspätetes Einschalten (*Late Tuning-In*) ist ohne weitere zusätzlich übertragene Informationen gewöhnlich nicht möglich.

Beim *Fernsehen*, was technisch *ein* und zusätzlich *kontinuierlicher* Datenstrom ist, kann zu beliebiger Zeit in den Datenstrom eingeschaltet werden und die Selbstsynchronisierungsmechanismen des Datenstromes sorgen für eine unverzügliche Darstellbarkeit der audio-visuellen Bestandteile des *Programm-Kanals*.

Wird jedoch in Betracht gezogen, dass bei MPEG-4 eine Szene auch statische Elemente beinhalten kann, die nur einmalig für eine Szene benötigt werden, so ergeben sich hieraus Probleme, wenn ein *Terminal* verspätet eingeschaltet wird. Die Übertragung des statischen Elements ist zeitlich gesehen schon geschehen und damit für das *Terminal* unwiderruflich verloren. Lösungen werden in Kapitel 5 *Broadcast* erarbeitet.

2.2 Kommunikationsformen für Gruppen

Internet-Anwendungen verwenden Punkt-zu-Punkt-Beziehungen (one-to-one, Abbildung 3), mehrfache Punkt-zu-Punkt-Beziehungen, Punkt-zu-Mehrpunkt-Beziehungen (one-to-many, Abbildung 4) oder Mehrpunkt-zu-Mehrpunkt-Kommunikationsbeziehungen (many-to-many, Abbildung 5). Eine oder mehrere Quellen senden an einen, mehrere oder alle Empfänger. Als Beispiele seien die Übertragung von Radiosendungen wie Webradio, Video- und Audiokonferenzen, replizierende Datenbanken, Webseiten und auch mittlerweile Dienstankündigungen mittels IP-Multicast zu nennen.

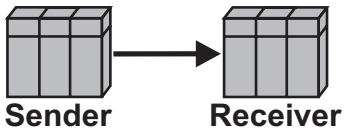


Abbildung 3: Punkt-zu-Punkt-Beziehung

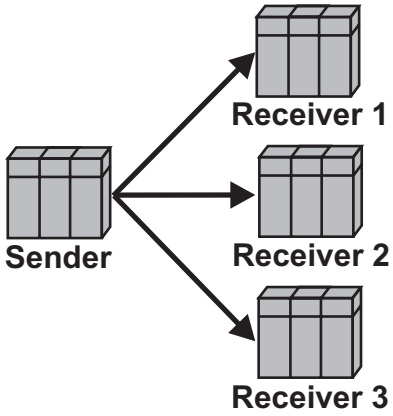


Abbildung 4: Punkt-zu-Mehrpunkt-Beziehung

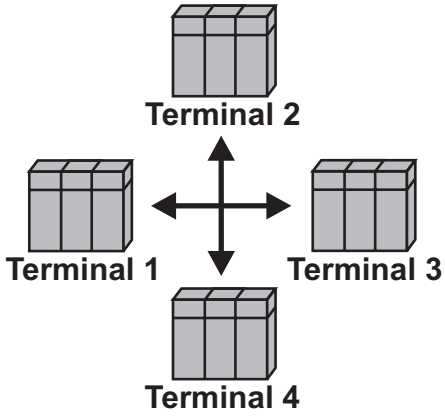


Abbildung 5: Mehrpunkt-zu-Mehrpunkt-Beziehung

Sollen also Datenströme gleichzeitig an viele Empfänger oder Gruppen von Empfängern versendet werden, muss ein geeignetes Übertragungsverfahren gewählt werden. In den folgenden Kapiteln werden die verschiedenen prinzipiellen Kommunikationsbeziehungen vorgestellt, um abschließend zu einer Empfehlung zu gelangen.

Es gibt die Möglichkeit, gleichzeitig Daten, Video und Audio zwischen mehreren Teilnehmern auszutauschen. Diese Form der Kommunikation wird als *Gruppenkommunikation* bezeichnet. Im Rahmen der Gruppenkommunikation können verschiedene Kommunikationsformen in Abhängigkeit der Anzahl der Sender und Empfänger unterschieden werden. Die Punkt-zu-Punkt-Kommunikationsbeziehung zwischen zwei Teilnehmern kann in diesem Sinne als Spezialfall der *Gruppenkommunikation* aufgefasst werden. Ebenso kann der *Broadcast* als ein Spezialfall der *Gruppenkommunikation* gesehen werden [Zit95] und [Zit96].

Unicast

Die Kommunikationsform *Unicast* (Abbildung 6) entspricht der traditionellen Punkt-zu-Punkt-Kommunikation. Es existiert genau ein Sender und ein Empfänger. Daraus leitet sich ab, dass der Austausch von Datenströmen unidirektional erfolgt.

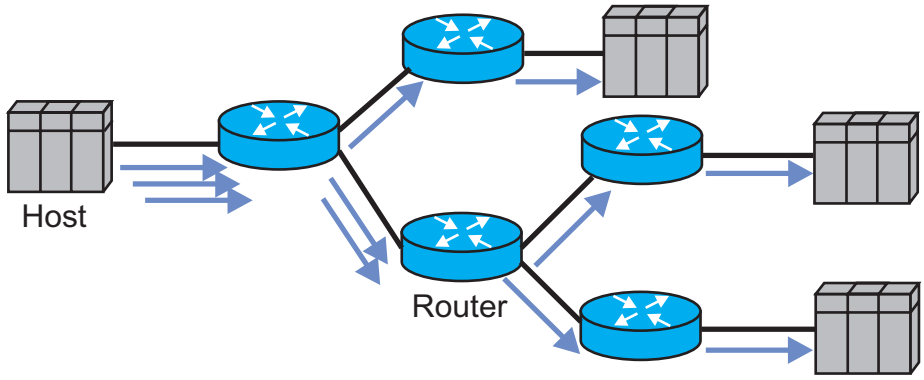


Abbildung 6: Unicast: n-fache Datenströme

Um eine Anzahl *n* von beliebigen Empfängern erreichen zu können, müssen *n*-fache undirektionale Punkt-zu-Punkt-Kommunikationsbeziehungen zwischen Server und Client (den *Hosts*) mittels Unicast etabliert werden. Der Sender (*Host*) muss dazu identische Datenströme *n*-fach zu den Empfängern (*Host*) replizieren und das Netz (*Router*) muss diese an die Empfänger (Teilnehmer) weiterleiten. Nachteilig ist die Belegung von Ressourcen im Server (Übertragungsbandbreite) und im Netz (Bandbreite der Router).

Broadcast

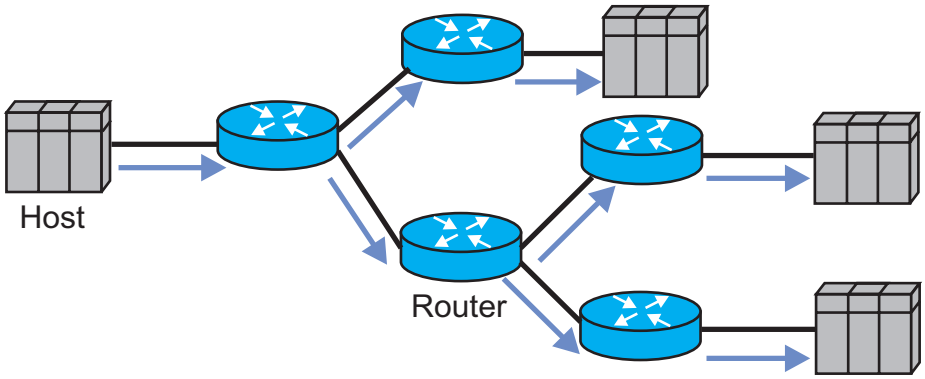


Abbildung 7: Broadcast: 1-facher Datenstrom

Die Kommunikationsform *Broadcast* (Abbildung 7) entspricht der traditionellen Punkt-zu-Mehrpunkt-Kommunikation in der Spezialform, dass alle Empfänger mit Datenströmen beliefert werden. Genau ein Sender (Host) sendet *einen* Datenstrom an *alle* Empfänger, im Gegensatz zur Unicast-Kommunikation. Daraus leitet sich ab, dass der Austausch von Datenströmen unidirektional erfolgt.

Die Gruppe der Empfänger E ist allerdings bei *Broadcast* nicht bestimmbar und ist die Menge aller empfangbaren Empfänger (im Gegensatz zu Multicast, wo nur „empfangswillige“ Empfänger Daten erhalten). Nachteilig bei diesem Verfahren ist, dass die Empfänger Nachrichten von diesem Sender erhalten, obwohl diese nicht angefordert waren.

Broadcast stellt in diesem Sinne eine Vereinfachung gegenüber Multicast dar, da keine Gruppen gebildet, adressiert und verwaltet werden müssen. Bekannte Szenarien für Broadcast-Kommunikation sind *Fernsehen* und *Rundfunk*.

Für das Verteilnetz bedeutet dies, dass Datenströme zu Empfängern repliziert werden, die nicht erwünscht sind. Somit entstehen auch im jeweiligen Netz vermeidbare Zusatzkosten.

Anycast

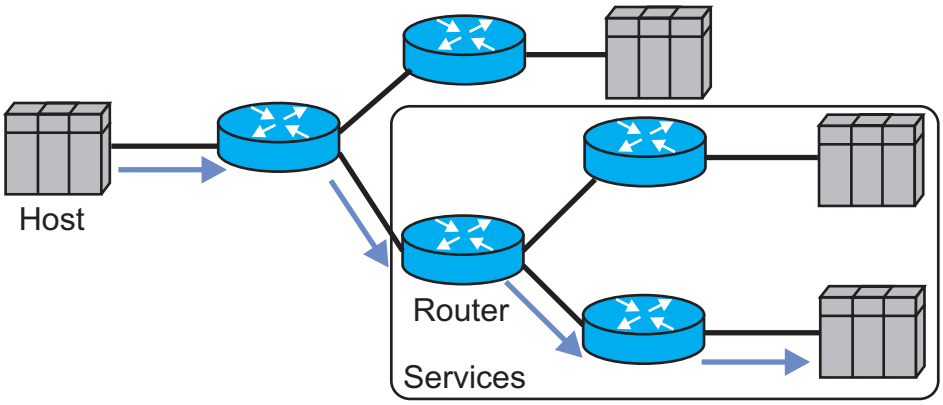


Abbildung 8: Anycast

Die Kommunikationsform *Anycast* (Abbildung 8) verwendet ebenfalls das Gruppenkonzept. Die Gruppe wird hier allerdings nicht zum eigentlichen Datenaustausch verwendet. Dieser findet per Unicast statt. Der Empfänger wird aber aus einer Gruppe heraus selektiert. Eine typische Anwendung von Anycast besteht in der Lokalisierung von Diensten in verteilten Systemen (*Services*).

Ein Anwender stellt eine Anfrage zu einem bestimmten Dienst an eine *Anycastgruppe*. Ein Auswahlmechanismus in dem *Anycast*-Kommunikationssystem selektiert dann einen einzelnen Rechner und leitet die Anfrage per Unicast an diesen Rechner weiter. Die verbleibenden Rechner der Gruppe sind in diesen Vorgang nicht mit einbezogen. Die Auswahl des Rechners aus der Gruppe der bereitstehenden Empfänger wird vom *Anycast*-Kommunikationssystem getroffen. Der Anwender hat hierauf keinen Einfluss.

Diese Kommunikationsform ist für den in dieser Arbeit verwendeten Broadcast nicht nützlich, da hier Dienstanfragen in einer Gruppe zu einem Dienstanbieter verteilt werden. Dies ist nützlich für Lastverteilung für Broadcast-Server, aber nicht für die Verteilung der Datenströme in ein Netz.

Multipeer

Eine *Multipeer-Kommunikation* liegt dann vor, wenn mehrere Sender an mehrere Empfänger Datenströme senden. Dies entspricht einer Mehrpunkt-zu-Mehrpunkt-Kommunikation (Abbildung 5). Multipeer stellt die vielfältigste Form der Gruppenkommunikation dar, da keinerlei Einschränkungen hinsichtlich der Anzahl von Sendern und Empfängern vorgenommen wird.

Die technische Realisierung lässt sich durch die Überlagerung verschiedener „einfacher“ Kommunikationsbeziehungen herstellen. Es können *n*-fache Unicast-Verbindungen von *m*-Sendern erstellt werden. Ferner können *m*-fache Multicast-Verteilbäume oder auch *m*-fache Broadcast-Verteilungen verwendet werden.

Multicast

Die Kommunikationsform *Multicast* (Abbildung 9) entspricht der traditionellen Punkt-zu-Mehrpunkt-Kommunikation. Es existieren genau ein Sender und *n* Empfänger. Daraus leitet sich ab, dass der Austausch von Datenströmen unidirektional erfolgt.

Bisherige Realisationen eines Gruppendienstes sind in großen Netzen für einen praktischen Einsatz zu ineffizient bezüglich des Ressourcenverbrauchs im Netz. Entweder verwenden diese Lösungen das n-fache Versenden der Nachricht an die *n* Empfänger oder das Broadcasting der Datenströme an alle Empfänger im Netz, wobei ungewollte Nachrichten von Empfängern verworfen werden müssen.

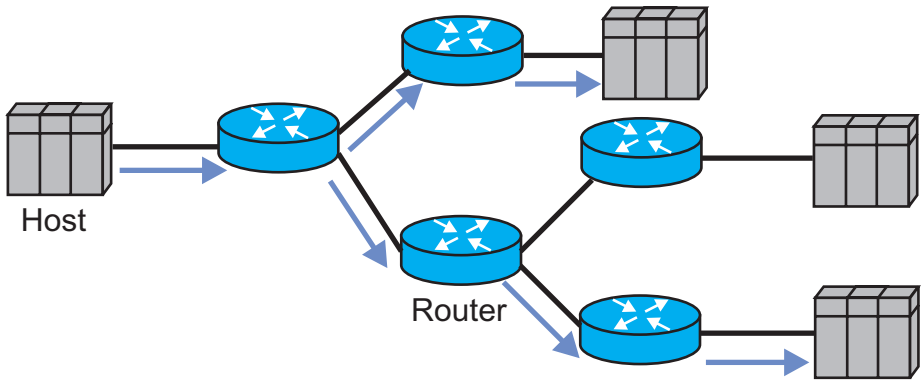


Abbildung 9: Multicast

Multicast-Empfänger, die Nachrichten des Senders erhalten möchten, treten dieser Gruppe bei. Multicast ermöglicht es, dass ausschließlich Gruppenmitglieder die Nachrichten des Senders erhalten. Für das Verteilnetz bedeutet dies, dass Datenströme nur zu Empfängern repliziert werden, die diese angefordert haben.

Effiziente Verteilung bedeutet in diesem Fall die minimale Nutzung der Ressourcen im Netz sowohl durch die Nutzdaten als auch durch die notwendige Signalisierung. Das heißt, dass das Kommunikationsprotokoll, das für die Verteilung der Daten zuständig ist, die Bandbreitennutzung, die Speichernutzung und den Rechenaufwand minimieren soll [Dei00].

Wichtigstes Ziel der Multicast-Protokolle ist die Minimierung des Ressourcenverbrauchs im Netz. Die einzelnen Teilziele beeinflussen sich allerdings reziprok zueinander:

- beispielsweise führt eine Minimierung der benötigten Bandbreite zu einem mit der Netzgröße exponentiell anwachsenden Rechenaufwand und
- zu einem großen Bandbreitenbedarf für Kontrollmeldungen (*Minimal Steiner Tree*-Problem).

Forschungsgebiete der IP-Multicast-Protokolle sind die Skalierbarkeit bezüglich der Netzgröße und der Charakteristik von Multicast-Gruppen, die Minimierung der Verzögerungszeiten zwischen Sender und Empfänger, Robustheit gegen Störungen (beispielsweise Topologieänderungen, Verhinderung von Schleifen), Einfachheit hinsichtlich des Netz-Managements und die Unabhängigkeit von Protokollen unterer Schichten.

2.3 IP-Multicast-Routingprotokolle

Für die Multicast-Verteilung müssen die Router Informationen untereinander austauschen. Dafür werden sogenannte IP-Multicast-Routingprotokolle benötigt. Nach einigen Vorbemerkungen zu IP-Multicast werden diese vorgestellt.

Der Sender adressiert seine IP-Datagramme an eine IP-Multicast-Adresse. Diese IP-Adresse für eine Multicast-Sitzung wird aus dem D-Klassen-Bereich der IP-Adressen gewählt, das heißt, von 224.0.0.0 bis 239.255.255.255. Im Gegensatz zu einer „normalen“ IP-Adresse, bezeichnet diese Adresse aber keinen bestimmten Rechner, sondern eine Gruppe von Rechnern, die IP-Multicastgruppe. Das IP-Datagramm wird mit Multicast-Routingprotokollen in die lokalen Netzte, wo Empfänger den Datenstrom abonniert haben, weitergeleitet. Indem ein Rechner auf diese IP-Adresse „hört“, tritt er der Gruppe bei, der Rechner wird ein Mitglied dieser IP-Multicast-Gruppe. Die Mit-

gliedschaft in einer Gruppe ist dynamisch und wird über sogenannte *Soft-States*[1] verwaltet. Beispielsweise läuft hier die Gruppenzugehörigkeit aus, wenn sich der Rechner nicht innerhalb einer parametrisierbaren Zeit zurückmeldet. Rechner können Gruppen zu jeder Zeit beitreten und sie verlassen, es gibt auch keine sonstigen Einschränkungen bezüglich des Standorts oder der Anzahl der Mitglieder in einer Gruppe.

Damit ergeben sich die drei Bereiche: Sender, Routing und Empfang.

Vorüberlegung

Eine Überlegung ist, dass sich der Empfänger beim Sender anmeldet. Damit kennt der Sender die Identität und die Anzahl der Empfänger, die den Datenverkehr abonnieren. Eine einfache Zugangskontrolle wäre möglich. Dies stößt aber bei dicht besetzten Multicast-Gruppen auf Skalierungsprobleme. Des Weiteren ist damit noch nicht das Transport- beziehungsweise das Verteilproblem der Datenströme gelöst.

Die andere Lösung ist, dass sich der Empfänger beim letzten Router (*Last Router*) anmeldet. Der letzte Router ist der Router, der das LAN mit den angeschlossenen Empfängern versorgt. Damit kann auch mehr als ein Sender an diese Gruppe, respektive die Empfänger, Datenströme versenden. Der Sender hat allerdings keine Information über die Gruppe der Empfänger (Anzahl, Identität, Zugehörigkeitsdauer, etc.). Eine gewünschte Zugangsbeschränkung ist nur per Verschlüsselung realisierbar [Gro01].

Aus Sicht der Router ergibt sich des Weiteren, dass diese für aktive Multicast-Gruppen deren ***O**utgoing **I**nter**f**aces (OIf)* speichern müssen, welche mit Datenströmen zu beliefern sind. Hier kann die Komplexität der Routingvorgänge erahnt werden.

In einem lokalen Netz ohne weitere Router (*Multihomed*) hat der Router Kenntnis darüber, welche Gruppen abonniert wurden, da die Hosts dies dem letzten Router mitgeteilt haben. In diesem Fall kann der Router also ungewünschte Multicast-Gruppen explizit abbestellen. Dieses Vorgehen wird als *Flood and Prune* verstanden. Des Weiteren kann der Router Multicast-Gruppen ohne Datenverkehr nach einer spezifizierten Zeit als unbenutzt markieren und den Datenstrom im in Senderichtung aufwärts liegenden Router abbestellen. Bei *Flood and Prune* werden Datenströme nur entgegengenommen, wenn sie auf dem ***I**ncoming **I**nter**f**ace (IIf)* des Routers angeliefert werden, über den auch die Unicast-Route zu dem Sender führt. Damit werden Schleifen vermieden.

Bei Multicast abonniert ein Empfänger die gewünschten IP-Multicast-Gruppen beim lokalen Router. Dieser fragt beim nächsten Router nach, ob dieser die Gruppen führt. Per *Prune* kündigt der Empfänger einzelne Gruppen beim letzten Router.

Distance Vector Multicast Routing Protocol (DVMRP)

Das erste Multicast-Routingprotokoll war das ***D**istance **V**ector **M**ulticast **R**outing **P**rotocol (DVMRP)* [RFC1075]. Die Referenzimplementation ist das *mrouted* auf dem das *Multicast-Backbone (MBone)* aufbaut. DVMRP basiert auf ***R**everse-**P**ath-**F**orwarding (RPF)*, das auch Grundlage der anderen Multicast-Routingprotokolle ist.

Reverse-Path-Forwarding akzeptiert zunächst nur Multicast-Pakete auf dem *Incoming Interface (IIf)*, auf dem die Unicast-Route zum Sender liegt. Existiert nun für eine Multicast-Gruppe keine Verteilerliste, dann trägt der Router alle benachbarten Router in eine neue Liste ein. An alle diese Router auf der Liste leitet das Protokoll Datenpakete weiter.

[1] Der Begriff *Soft-State* bezeichnet einen Zustand, der nach einer bestimmten Zeit ohne ein auftretendes externes Ereignis einen Zustandsübergang durchführt.

Schließlich löscht der Router periodisch die Listen für alle Multicast-Gruppen aufgrund der *Soft-States*. Wenn eine Liste leer ist, sendet der Router eine *Prune*-Nachricht für die jeweilige Multicast-Gruppe an den Router (*Upstream-Router*), von dem er die Multicast-Pakete erhält.

Will ein Empfänger nun eine Gruppe abonnieren, dann muss sein lokaler Router warten, bis für diese Gruppe das periodische Fluten stattfindet. Dieses wiederum findet nur für Gruppen mit Datenverkehr statt.

Eine DVMRP-Verfeinerung ist, dass der Router sich je ***O**utgoing-**I**nter**f**ace* (*Olf*) merkt, ob er von in Senderrichtung dahinter liegenden Routern erkannt wird. Hierzu muss DVMRP sich auch über Unicast-Routen austauschen, wofür ein eigenes Verfahren benutzt wird, welches auf Distanz-Vektoren aufbaut. Das stellt eine Verdoppelung des Verwaltungsaufwandes gegenüber dem Unicast-Routing dar.

Distanz-Vektoren sind Meldungen über den Austausch von Netz-Metriken. Die Metrik ist ein Maß für die Güte einer Route. Üblicherweise wird die Anzahl der Knoten (Router, Hops), also der auf dem Weg passierten Router, verwendet. Andere Formen der Netzkosten, wie zur Verfügung stehende Bandbreite, werden hierbei nicht beachtet. Das meistverwendete Distanz-Vektor-Verfahren beim Unicast-Routing ist das ***R**outing **I**nformation **P**rotocol (RIP)* [RFC2453].

Multicast Open Shortest Path First (MOSPF)

***O**pen **S**hortest **P**ath **F**irst (OSPF)* teilt nicht nur benachbarten Routern die eigenen Routen mit, sondern auch weiter entfernten Routern. Jeder Router baut eine Datenstruktur auf, in der alle Router und alle Routen gespeichert sind und sucht sich dann die optimale Route über *Dijkstra*s Algorithmus aus. ***M**ulticast **O**pen **S**hortest **P**ath **F**irst (MOSPF)* [RFC1584] benutzt OSPF für das Unicast-Routing.

Da alle Router und Routen pro Multicast-Gruppe in jedem Router gespeichert und zwischen Routern ausgetauscht werden, auch als *Link State Routing* bekannt, belegt MOSPF für die gleichen Daten viel mehr Netzbandbreite und Speicher im Router als DVMRP und außerdem verbraucht *Dijkstra*s Algorithmus viel Rechenzeit. MOSPF wird heute praktisch nicht mehr eingesetzt.

Protocol Independent Multicast-Dense Mode (PIM-DM)

DVMRP und *MOSPF* haben den Nachteil, dass sie von einem bestimmten Unicast-Routingprotokoll abhängen. Die ***I**nternet **E**ngineering **T**ask **F**orce (IETF)* hat zwei Multicast-Routingprotokolle definiert, die vom Unicast-Routing unabhängig ist.

Diese Protokolle heißen ***P**rotocol **I**ndependent **M**ulticast - **D**ense **M**ode* (PIM-DM) [PIM-DM] und ***P**rotocol **I**ndependent **M**ulticast - **S**parse **M**ode (PIM-SM)* [PIM-SM]. *PIM-DM* entspricht *DVMRP* ohne die Aussparung des in Senderichtung zurückliegenden Routers, erzeugt also mehr Datenverkehr. Es löst *DVMRP* in der Praxis trotzdem ab, da Cisco[1] in seinen Routern nur *PIM* unterstützt. Laut Cisco ist in allen Routern seit 1996 die Multicast-Unterstützung implementiert. Es liegt am ***I**nternet **S**ervice **P**rovider (ISP)*, ob dieser den Multicast-Verkehr auch zulässt.

Im *Dense Mode* wird angenommen, dass die Empfänger nahe beieinander liegen und die Router eine relativ hohe Zahl von Teilnehmern zu bedienen haben.

Core Based Tree (CBT)

Sparse-Multicast geht davon aus, dass nur wenige vereinzelte Empfänger pro Multicast-Gruppe existieren. Damit scheidet das Fluten (*Flooding*) aus. Anstelle dessen werden ***C**ore **B**ased **T**rees (CBT)* [RFC2201] und *PIM-SM* verwendet.

[1] Cisco stellt weltweit rund 90% aller IP-Router [Lei00a].

Bei spärlich verteilten Empfängern (*Sparse Mode*) muss jeder Router nur wenige Teilnehmer bedienen, dafür liegen die Empfänger einer Multicast-Gruppe weit verteilt.

Core Based Tree geht von einem zentralen Kern-Router (*Core*) aus, über den der gesamte Datenverkehr fließt. Ein lokaler Router meldet sich bei diesem netzweit bekannten Core-Router an. Die Anfrage kommt auf dem Weg zum Core-Router bei allen Routern vorbei, die auch dafür zuständig wären, den Multicast-Verkehr regulär weiterzuleiten. Der erste Router auf dem Weg, der die gewünschte Multicast-Gruppe bereits führt, leitet die Nachricht jetzt nicht mehr an den Core-Router weiter, sondern verteilt die Nachricht nun über das ***O**utgoing-**I**nter**f**ace (OIf)*, über den die Anfrage kam.

Bei *Core Based Trees* muss kein Router den kompletten Verteilungsbaum kennen. Damit wird kein wertvoller Speicherplatz in einem Router belegt - außer natürlich im Core-Router selbst, der alle Multicast-Gruppen kennen muss. Andererseits müssen alle Multicast-Datenströme durch den Core-Router verteilt werden, somit kann es also zu Bandbreiten-Engpässe kommen.

Protocol Independent Multicast-Sparse Mode (PIM-SM)

Im Unterschied zu *Core Based Trees* kennt ***P**rotocol **I**ndependent **M**ulticast - **S**parse **M**ode (PIM-SM)* [PIM-SM] keine Core-Router, sondern verwendet so genannte ***R**endezvous **P**oints (RP)*.

PIM-SM verwendet die gleichen Mechanismen wie *CBT*, also einen gemeinsamen Baum Verteilbaum für die Multicast-Gruppe.

Die Weiterentwicklung von *PIM-SM* und damit der Unterschied zu *CBT* ist, dass unterwegs liegende Router beziehungsweise der *Rendezvous Point* feststellen können, dass ein Umschalten vom *PIM-SM* in den *PIM-DM* nach einer parametrisierbaren Schwelle (*Threshold*) aufgrund gestiegenen Multicast-Verkehrs möglich ist.

Bei *PIM-SM* erfragen die Empfänger beim *Rendezvous Point (RP)*, welcher Sender eine gewünschte Gruppe führt; dann wird dorthin direkt geroutet.

Momentan werden bei *PIM-SM* die *Rendezvous Points* manuell bei den verschiedenen ***I**nternet **S**ervice **P**roviders (ISP)* konfiguriert. Das ***M**ulticast **S**ource **D**iscovery **P**rotocol* (*MSDP*) [MSDP] soll die Liste der *Rendezvous Points RP* abgleichen und ist noch in der Entwicklung bei der ***I**nternet **E**ngineering **T**ask **F**orce* (*IETF)*.

Border Gateway Protocol (BGP)

Als ***A**utonome **S**ysteme (AS)* werden die eigenständigen Teilnetze des Internet, beispielsweise das Netz eines ***I**nternet **S**ervice **P**roviders* (*ISP*), genannt. Innerhalb des Netzes eines ISP werden üblicherweise einheitliche Routing-Protokolle verwendet. Bei dem Zusammenschluss (*Interconnection*) von ISPs kommt das ***B**order **G**ateway **P**rotocol (BGP)* [RFC1771] zum Einsatz. Im IETF werden aktuelle Probleme beim Umsetzen der verwendeten Routing-Protokolle diskutiert. Erweiterungen für das Multicast-Routing sind in [RFC2283] aufgeführt.

Dynamische Addressen

Die Wahl einer IP-Adresse als IP-Multicast-Adresse ist im Moment ebenfalls noch in der Diskussion im IETF. Angedacht ist das ***M**ulticast **A**ddress **D**ynamic **C**lient **A**llocation **P**rotocol* (*MADCAP*) [MADCAP]. Es ist ein Protokoll, mit dem sich Sender eine Multicast-IP zuweisen lassen können. In der Praxis wird allerdings eine zufällige IP belegt und „gehofft“, dass diese noch nicht belegt ist.

Bandbreitennutzung

Bei Datenströmen über eine direkte Verbindung zum Empfänger passt sich der Datenstrom der verfügbaren Bandbreite an, wenn TCP verwendet wird. Bei multimedialen Datenströmen kommt aber üblicherweise das unkontrollierte Datagramm-Protokoll UDP (Kapitel 6) zum Einsatz.

MPEG schlägt ein Modell mit *Base-Layer* und *Enhancement-Layer* vor. Der Basis-Layer soll für ein brauchbares Bild sorgen, der optionale Enhancement-Layer soll die Bildqualität verbessern, auch wenn es nur in Fragmenten empfangen wird. An dieser Stelle setzen Forschungen zu *Quality of Service* und *Reliable Multicast* an. Es soll sichergestellt werden, dass der Empfänger zumindest das Basis-Layer komplett empfängt. Praktisch ist *Quality of Service* aber nicht netzweit verfügbar und bei den Autoren für MPEG-4 basiertes Material hat sich dieses geschichtete Qualitätsmodell nicht durchgesetzt.

Ausblick

Das IETF hat unter dem Arbeitstitel ***S**ource **S**pecific **M**ulticast (SSM)* [RFC3569] begonnen, die Skalierungsprobleme des *Rendezvous Points* aufzugreifen. Mittels des ***I**nternet **G**roup **M**anagement **P**rotocols*[1] *v3 (IGMPv3)* [IGMPv3] ist erstmals ein Sender direkt adressierbar.

Das Paradigma *Fernsehen*, ein Sender mit sehr vielen Empfängern, ist prinzipiell mit Multicast gelöst - bis auf die oben genannten Skalierungsprobleme bei den *Rendezvous Points*.

PIM beziehungsweise *SSM* entwickeln sich in diese Richtung. *Fernsehen* per *Multicast* scheint die erhoffte *Killer-Applikation* für Multicast zu sein, was aber noch einige Zeit benötigen wird.

Eine schon im praktischen Einsatz befindliche Anwendung sind Multiplayer-Spiele mit Multipeer-Kommunikationsbeziehungen. Allerdings sind hier nur Multiplayer-Spiele im LAN-Bereich bekannt, da für Multicast über das Internet die ISPs erst IP-Multicast für den breiten Endanwenderbereich freischalten müssten.

IP-Multicast hat somit auch über zehn Jahre nach seiner „Erfindung“ durch Steve Deering [DC85] [Dee91] immer noch viele ungelöste Fragestellungen und technische Probleme.

2.4 Einsatz für Broadcast-Szenarien

Um das Broadcast-Szenario in IP-Netzen zu realisieren, ist das Multicast-Verteilverfahren den anderen Verfahren vorzuziehen. Der Vorteil ist, dass der Sender nur einen Datenstrom in das Netz abgibt. Das Netz repliziert die Datenströme dort, wo es kostenminimal ist. Es werden nur die Empfänger mit Datenströmen versorgt, die diese auch angefordert haben.

Multicast folgt dem Push-Kommunikationsmodell. Neben der ursprünglichen Bedeutung, dass der Server die Datenströme auf den Client „schiebt“, kommt auch die Tatsache zum Tragen, dass der Client – Empfänger – auf die gewünschte Multicast-Übertragung eingestellt werden muss. Dies ist analog zu einer herkömmlichen Radio- oder Fernsehübertragung, bei der die Empfänger auf den entsprechenden Sender eingestellt werden müssen.

Im Falle von IP-Multicast, weist der Benutzer einfach seinen Rechner an, auf Übertragungen der gewünschten IP-Multicast-Gruppenadresse empfangsbereit zu warten. Der Sender, der den Datenstrom versendet, benötigt darüber hinaus keine Informationen, welche Empfänger sich für den Empfang entschieden haben; er muss lediglich die Datenströme an die entsprechende IP-Multicast-Gruppenadresse senden. Nur eine Kopie einer Multicast-Nachricht wird durch das Netz versendet und Kopien der Nachricht beziehungsweise der Datenströme werden nur erstellt, wenn sich die Signalwege an einem Router trennen. Daher birgt IP Multicast viele Performance-Verbesserungen und spart Bandbreite im Hinblick auf die Ende-zu-Ende-Kommunikation der Anwendungen.

Damit ist festzuhalten, dass IP-Multicast das ideale Verteilverfahren für das Broadcast-Szenario ist.

---

[1] Im nächsten Kapitel wird mehr über IGMP berichtet.

2.5 Gruppenmanagement im lokalen Netz

Sendende oder empfangende Rechner sind mit einem lokalen Netz verbunden und dieses ist an eine oder mehrere Router angeschlossen. Die Verteilverfahren, die die Router anwenden, wurden vorgestellt. Das Senden und das Empfangen von IP-Multicast-Datenströmen im LAN wird jetzt erläutert.

Senden

Zuerst wird der einfachere Fall des Versendens von IP-Multicast-Datagrammen betrachtet.

Möchte ein Sender einen Datenstrom an eine Multicast-Gruppe senden, so muss der Sender lediglich die IP-Datagramme, woraus die Datenströme bestehen, mit der IP-Multicast-Adresse der gewünschten Gruppe als Zieladresse versehen. Die Absenderadresse des Servers wird durch den IP-Protokollstapel automatisch ergänzt.

Die Empfangsadresse, respektive Zieladresse, muss die IP-Multicast-Gruppenadresse enthalten. Ein weiterer zu berücksichtigender Punkt ist die so genannte Lebenszeit eines IP-Datagramms für IP-Multicast. Es ist in der Praxis zu beachten, dass es nicht genügt, von der üblichen Lebenszeit *Time-To-Live (TTL)* eines Unicast-IP-Paketes auszugehen, da alle Implementierungen von Protokollstapeln für die Multicast-Kommunikationsbeziehungen eine eigene TTL, die Multicast-TTL, verwenden und diese standardmäßig auf den Wert 1 setzen.

Eine TTL von 1 bedeutet, dass das IP-Datagramm den nächsten Router nicht verlässt. Somit verbleiben diese Datagramme im LAN. Es muss also beim Sendevorgang zusätzlich die TTL der IP-Multicast-Datagramme auf einen gewünschten Wert gesetzt werden.

Bei der Multicast-TTL muss noch berücksichtigt werden, dass es im Multicast zwei Möglichkeiten der Reichweitensteuerung gibt. Eine Möglichkeit der Reichweitensteuerung erfolgt über die schon erwähnte Multicast-TTL. Mit einem geringen Wert kann die Reichweite eingeschränkt werden (TTL=16), mit einem großen Wert (TTL=128) ist eine weltweite Verteilung möglich. Das *Deutsche Forschungsnetz (DFN)* hatte dazu 1997 eine Übersicht erstellt, wie das DFN in einzelne Multicast-Regionen unterteilt ist und entsprechende TTL-Werte festgelegt sind.

Eine andere Möglichkeit der Reichweitensteuerung ist die administrative Reichweitensteuerung. Hier hat das DFN Multicast-Adressen in Adressbereiche für bestimmte Regionen festgelegt. Damit ist eine einfachere Reichweitensteuerung möglich.

Empfangen

Der Empfangsvorgang im lokalen Netz gestaltet sich hingegen sehr viel schwieriger als das Versenden. Es genügt nicht, dass ein Empfänger den Wunsch hat, einer Multicast-Gruppe beizutreten.

Auf Clientseite könnte der Empfangsvorgang ähnlich einfach sein, wie beim Sendevorgang. Der Empfänger bräuchte nur seine IP-Multicast-Adresse auf Empfang zu schalten und könnte sofort empfangen. Dafür müssten allerdings alle Datenströme permanent im LAN vorhanden sein, was dann in der Folge nicht mehr eine Multicast-Kommunikationsbeziehung sondern eine Broadcast-Kommunikationsbeziehung wäre. Daher muss es Mechanismen geben, woran der letzte Router erkennen kann, dass es für bestimmte Multicast-Gruppen Empfänger im LAN gibt, die Datenströme abonniert haben. Für nicht abonnierte Datenströme muss der letzte Router erkennen, dass diese Datenströme unerwünscht sind und dass er das LAN für diese Multicast-Gruppen aus dem Multicast-Verteilbaum abmelden muss.

Aus diesen Anforderungen heraus wurde das *Internet Group Management Protocol* (*IGMP*) entwickelt. Dieses regelt den Austausch zwischen dem letzten Router und den Empfängern, welche

Multicast-Gruppen abonniert werden. IGMP ist wie ICMP als Teil der Vermittlungsschicht zu betrachten.

Das IGMP Protokoll in der Version 1 [IGMPv1] kennt nur die Nachrichtentypen *Host Membership Query* und *Host Membership Report*. Ein Router kann mit IGMPv1 periodisch alle Hosts fragen, welche Gruppen sie empfangen möchten, und Hosts können als Antwort auf die Anfrage, oder spontan für eine neue Gruppe, einen Report absenden. Bei Version 1 kann ein Host eine Gruppe nicht gezielt verlassen. Er kann nur warten, bis der Router das nächste Mal eine Anfrage stellt und dann darauf nicht mehr antworten.

Version 2 [IGMPv2] fügt eine Nachricht hinzu, mit der Gruppen explizit verlassen werden können und spezifiziert, wie mehrere Multicast-Router in einem Ethernet koexistieren können. Version 3 [IGMPv3] erlaubt die Angabe von Senderlisten zur Sender-Auswahl. Ein Host kann beim Beitritt zu einer Gruppe erklären, dass er nur Daten von einem bestimmten Sender oder von einer Liste von Sendern empfangen möchte. Daneben kann er Sender explizit ausschließen. Derzeit unterstützen die meisten Betriebssysteme IGMPv2. IGMPv3 ist in einigen experimentellen Implementierungen praktisch umgesetzt, aber noch nicht weit verbreitet.

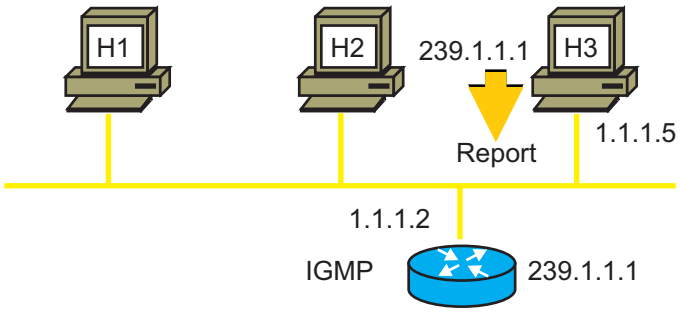


Abbildung 10: Beitritt zu einer Gruppe

Abbildung 10 stellt den Beitritt eines Empfängers zu einer IP-Multicast-Gruppe dar. Der Empfänger (*Host 3*) sendet eine *IGMP Report*-Meldung in das LAN mit der gewünschten, zu abonnierenden IP-Multicast-Gruppe (239.1.1.1). Der Router schaltet daraufhin den Datenstrom in das LAN durch und der Empfänger kann den IP-Multicast-Verkehr empfangen.

Falls dem Router die IP-Multicast-Gruppe nicht bekannt ist, fordert er diesen gemäß den beschriebenen Multicast-Routingprotokollen innerhalb des Netzes an. Dieser Vorgang ist nicht sichtbar, also transparent, für das lokale Netz.

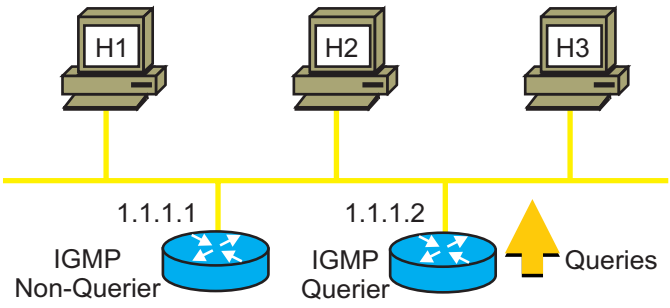


Abbildung 11: Bestimmung des Routers für die Gruppenverwaltung

Abbildung 11 stellt ein so genanntes *Multihomed* Netz dar. Hierunter wird verstanden, dass ein lokales Netz mit mehr als einem Router verbunden ist und von diesen mit Datenströmen versorgt wird. Jeder Router versendet *Query*-Meldungen. Stellt jedoch ein Router fest, dass *Query*-Meldungen im LAN vorhanden sind, ohne dass diese von ihm stammen, so ziehen die Router mit

niedrigeren IP-Adressen sich zurück. Damit ist der Router mit der höchsten IP-Adresse der so genannte letzte Router (*Last Router*) und wird damit der so genannte *IGMP-Querier*. Die anderen Router übernehmen, wenn der IGMP-Querier eine Anfrage nicht oder nicht rechtzeitig stellt, also ausgefallen zu sein scheint.

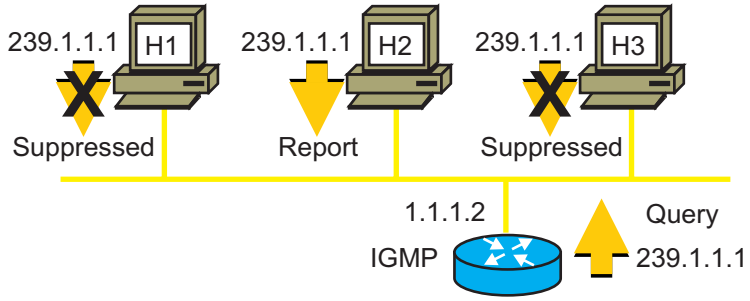


Abbildung 12: Aufrechterhaltung einer Gruppe

In der Version 1 des IGMP-Protokolls fragt der *IGMP-Querier* alle sechs Minuten im LAN nach, ob noch Bedarf an der speziellen Multicast-Gruppe besteht. Dies wird mittels der *Query*-Nachricht abgefragt (Abbildung 12). Einer der Empfänger (*Hosts*) des LAN beantwortet nach einer zufälligen Zeit diese Nachricht mit einer *Report*-Meldung. Empfängt der letzte Router keine *Report*-Meldung, so wird der Verkehr für diese Multicast-Gruppe im Multicastverteilbaum abgemeldet.

IGMP geht von einem *gesharten* Leitungsmedium aus und es wird daher angenommen, dass die *Report*-Meldung eines Hosts von den anderen im LAN angeschlossenen Hosts ebenfalls empfangen wird. Diese unterdrücken damit die weiteren *Report*-Meldungen an den letzten Router beziehungsweise den IGMP-Querier.

Forschungsgebiete sind hier die Verwendung von geswitchten LAN-Systemen, bei denen nur die angeschlossenen Hosts mit Datenströmen versorgt werden, die an der Kommunikation beteiligt sind. Üblicherweise werden bei Ethernet-Switches nur die beiden beteiligten Hosts direkt miteinander verschaltet, so dass benachbarte Hosts keinen Datenverkehr der Punkt-zu-Punkt-Kommunikation der beiden Hosts erhalten.

Hier hat die Firma CISCO eine Abwandlung von IGMP, das ***C**ISCO **G**roup **M**anagement **P**rotocol* (*CGMP*) entwickelt, welches so genanntes Layer-3 Sniffing betreibt. Das bedeutet, dass der Ethernet-Switch die IGMP-Meldungen bearbeitet und damit den Kommunikationsfluss mitverfolgt und auch mitverfolgen muss. Ursprünglich waren diese Überlegungen für IGMPv3 angedacht. Die Komplexität des Problems (Abbildung eines Shared Mediums durch dedizierte Verbindungen) und die damit verbundenen ungelösten Probleme sind Gegenstand weiterer Forschungen.

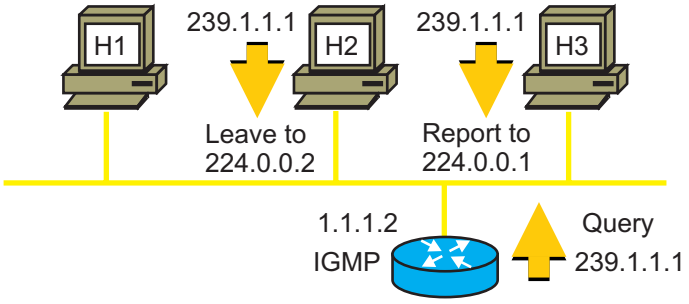


Abbildung 13: Aktives Verlassen einer Gruppe

In der Version 2 des IGMP Protokolls können Empfänger eine Gruppe aktiv verlassen, indem diese eine *Leave*-Meldung an die Multicast-Adresse *224.0.0.2* senden (Abbildung 13). Im Gegensatz da-

zu, konnte bei IGMP v1 die Mitgliedschaft nur durch ein Auslaufen des *Soft-States* nach sechs Minuten erfolgen. Falls sich der letzte Empfänger einer Gruppe in diesem LAN abmeldet, meldet der letzte Router dieses LAN aus dem Multicastverteilbaum ab. Der letzte Router fragt mit dem Query-Mechanismus im LAN nach, ob es noch aktive Empfänger gibt. Sollte sich kein Host mit einer Report-Meldung melden, so wird der Datenstrom im stromaufwärts liegenden Router und damit für dieses Netz abgemeldet (*Prune*). Diese Erweiterung des IGMP-Protokolls reduziert die Latenzzeit für das Abmelden einer IP-Multicast-Gruppe erheblich.

IGMP v3 erweitert die Möglichkeiten der Hosts dahingehend, dass diese den eingehenden IP-Multicast-Verkehr einer Gruppe nun zusätzlich auf bestimme Sender hin ausfiltern können.

2.6 Zusammenfassung

In diesem Kapitel wurden die verschiedenen Arten von Kommunikationsbeziehungen diskutiert und festgestellt, dass bei der Umsetzung des Paradigmas *Fernsehen* die Kommunikationsbeziehung *IP-Multicast* am besten geeignet ist.

Die Techniken für IP-Multicast wurden dann vorgestellt und die Problematiken im lokalen Netz für einen Sender und einen Empfänger ausführlich dargestellt.

3 MPEG-4

In diesem Kapitel wird zunächst die Standardisierung und die verschiedenen Normen zu multimedialen Elementen vorgestellt. Eine kurze Einführung in die Kompression von Stand- und Bewegtbildern folgt. Anschließend wird die MPEG-4 Terminalarchitektur und zur Bereitstellung von multimedialen Datenströmen das *Delivery Multimedia Integration Framework* eingeführt. Darauf aufbauend wird der Begriff der *MPEG-4 Applikation* definiert. Um die in den MPEG-4 Datenströmen enthaltenen Zeitstempel zu ermitteln, werden die MPEG-4 spezifischen Deskriptoren benötigt, mit denen dieses Kapitel abschließt.

3.1 MPEG Standards

Die ***M**oving **P**ictures **E**xperts **G**roup (MPEG)* wurde Ende der 80er Jahre zur Festlegung eines digitalen Standards für Bewegtbilddarstellung ins Leben gerufen. Zu dieser Zeit standen bereits verschiedene Verfahren zur Verfügung. Zu den bis dato wichtigsten Vertretern gehörten ***M**otion-JPEG (M-JPEG)* und die Empfehlung (*Recommendation*) *H.261* des ***C**omité **C**onsultatif **I**nternational **T**élégraphique et **T**éléphonique*[1] (*CCITT)*, die im weiteren Verlauf noch detailliert erläutert werden.

MPEG ist die Arbeitsgruppe *ISO/IEC JTC 1/SC 29/WG11* mit dem Titel *Coding of moving pictures and audio* der ***I**nternational **O**rganisation for **S**tandardisation (ISO*). Ziel dieser Arbeitsgruppe MPEG ist die Entwicklung internationaler Standards für Komprimierung, Dekomprimierung, Verarbeitung und verschlüsselte Darstellung von beweglichen Abbildungen, Tönen (Audio) und deren Kombination, um eine breite Vielzahl von Anwendungsmöglichkeiten zu erfüllen.

MPEG hielt die erste Tagung in Ottawa, Kanada, 1988 ab. Seitdem wird sich zirka viermal im Jahr weltweit getroffen, um an den Normen zu arbeiten. Vorsitzender der MPEG-Arbeitsgruppe ist Leonardo Chiariglione, der an allen bisherigen 66 Sitzungen (Oktober 2003) teilgenommen hat.

Der Prozess der Standardisierung der MPEG-Normen wird nun im Detail beschrieben.

Das erste Dokument, das für die Audio- und Videocodierung bei diesem Prozess entsteht, wird ***V**erification **M**odel* (*VM*) bezeichnet. In MPEG-1 und MPEG-2 wurde dieses *Simulation- and Testmodell* genannt. Das *Verification Modell* beschreibt in einer Art Programmiersprache beziehungsweise Blockschaltbildern, den Betrieb der Encoder und der Decoder. Das VM wird für Simulationen benutzt, um die Leistung der Coder-Entwürfe zu optimieren. Wenn MPEG ausreichendes Vertrauen in die Stabilität des Entwicklungsstandes hat, wird ein sogenannter ***W**orking **D**raft (WD)* erstellt. Dieser ist bereits in Form eines internationalen Standards erstellt, wird aber von MPEG für mögliche Überarbeitungen intern gehalten. Entsprechend dem aufgestellten Zeitplan und wenn der *Working Draft* stabil genug wurde, wird zum ***C**ommittee **D**raft (CD)* übergeleitet. Dieser wird dann zur Abstimmung an die nationalen Normungsgremien (in Deutschland das DIN), die so genannten ***N**ational **B**odies (NB)*, mit der Bitte um Abstimmung (*Balloting*) gesandt. Liegt die Anzahl der positiven Stimmen über dem Quorum, wird aus dem *Committee Draft (CD)* ein ***F**inal **C**ommittee **D**raft (FCD)*. Allerdings müssen die Bemerkungen der nationalen Normungsgremien (*NB*) in Betracht gezogen werden. Nach einer gründlichen Überarbeitung werden die Dokumente wieder an die *NB*s für eine zweite Abstimmung gesandt. Liegt die Anzahl der positiven Stimmen wiederum über dem Quorum wird der *Final Committee Draft* zum ***F**inal **D**raft **I**nternational **S**tandard (FDIS)*. ISO führt danach nur noch eine Ja/Nein-Abstimmung mit den *National Bodies* durch. Hier sind keine technischen Änderungen mehr erlaubt. Das Dokument wird nach erfolgreicher Abstimmung endlich zum ***I**nternational **S**tandard (IS)*. Jede durchlaufene Stufe wird veröffentlicht, so dass die jeweiligen Do-

[1] heute ***I**nternational **T**elecommunications **U**nion-**T**elecommunication* (*ITU-T*)

Seiten 18-284 sind in der vollständigen Ausgabe enthalten.

Die vollständige Ausgabe kann beim Autor erworben werden, sowohl in elektronischer Fassung (PDF) als auch in gebundener Ausgabe.

Michael Stepping

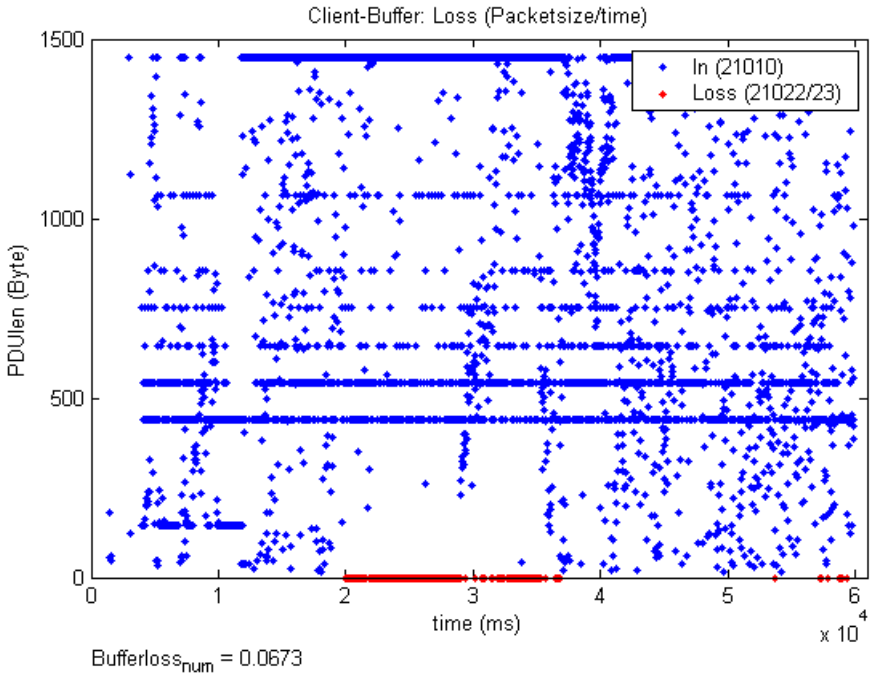


Abbildung 241: Verluste Server-DA* - DAI, TS, RS FCFS, 670 kbit/s

Abbildung 241 stellt die Paketverluste am DAI als absolute Anzahl bei der *Server Limited Bitrate* B_{SLB} = 600 kbit/s dar. Dies ist deckungsgleich mit den durch das serverseitige Rate Shaping verursachten Verlusten (Abbildung 223). Insofern sind keine augenfälligen zusätzlichen Verluste aufgetreten. Bei der *Server Limited Bitrate* B_{SLB} = 1200 kbit/s treten keine Verluste auf, da serverseitig das Rate Shaping nicht einsetzen muss (Kapitel 8.4.4).

8.4.7.4 Traffic Shaping mit Rate Shaping WRR

Es wird serverseitig das Traffic Shaping in Verbindung mit Rate Shaping getestet. Als Rate Shaping-Strategie wird WRR angewendet.

Der Verlauf des Jitters zwischen Server-DAI* und DAI bei Verwendung von Traffic Shaping mit B_{SLB} = 670 kbit/s und Rate Shaping mit der WRR-Strategie ist der Gleiche wie bei der FCFS-Strategie für das Rate Shaping (Abbildung 240). Der Anstieg der Verzögerungszeit ist bedingt durch die serverseitig verspätete Versendung der PDUs (Abbildung 226).

Bei einer B_{SLB} = 1200 kbit/s ist der Verlauf des Jitters gleich der Referenzmessung (Abbildung 233 und Abbildung 234).

Die Verweilzeiten der PDUs im Clientpuffer entspricht den Verläufen der Abbildung 238 und Abbildung 239.

Die Auswertung der Messung ergibt, dass es keine augenfällige Änderung des Verhaltens gegenüber dem Traffic Shaping mit Rate Shaping mit FCFS-Strategie gibt. Ausgangsbitrate, Pufferverweilzeiten, Pufferfüllstände und Verluste sind in beiden Fällen gleich.

8.5 Zusammenfassung

Eine Variation der Bedienstrategien der Warteschlangen hat entgegen den Erwartungen keinen messbaren Einfluss auf den Ausgang des Server-Systems an der Netz-Schnittstelle.

Um die Übertragungsrate beeinflussen zu können, wird Traffic Shaping benötigt. Ist die Ankunftsrate des Systems höher als die Abgangsrate kommt es zu einer kontinuierlichen Vergrößerung der Warteschlange im Server. Da keine PDUs verworfen werden, werden die PDUs älter als nach der Echtzeitbedingung erlaubt ist und die audiovisuelle Darstellung der MPEG-4 Applikation wird verlangsamt.

Der Einsatz eines Rate Shaping-Algorithmus ist notwendig, um die Überlast-Situation abbauen zu können. Es werden dann Pakete aus den Warteschlangen verworfen. Die Messergebnisse lassen die Schlussfolgerung zu, dass der gewählte Algorithmus für die Rate Shaping-Strategie keinen messbaren Einfluss hat. Wird jedoch die *Server Limited Bitrate* B_{SLB} genügend hoch angesetzt, so müssen keine Pakete verworfen werden und es ist der Einsatz des Rate Shaping nicht notwendig.

Aus den Messwerten geht ebenfalls hervor, dass die Player bei Unterschreitung einer bestimmten Bitrate beziehungsweise zu vielen gehäuften Verlusten keine vernünftige audiovisuelle Wiedergabe leisten können. Aus den Messungen ist die *Server Limited Bitrate* für VBR codierte Datenströme um 50 % höher anzusetzen als der Mittelwert der MEPG-4 Applikation im Langzeitmittelwert ist.

Eine weitere Beobachtung ist, dass CBR-codierte Datenströme gebündelt in einer MPEG-4 Applikation keine großen Schwankungen der Datenrate von sich aus haben. Diese Applikationen haben von sich aus keine ausgeprägten Bursts in der Datenübertragung. Somit ist die Verwendung der *Server Limited Bitrate* B_{SLB} bei diesen MPEG-4 Applikationen in der Größe des Langzeitmittelwertes möglich.

9 Zusammenfassung

Die vorliegende Arbeit befasst sich mit dem Thema *MPEG-4 Broadcast.* Es wurden zuerst die verschiedenen Kommunikationsbeziehungen diskutiert mit dem Ergebnis, dass IP-Multicast die geeignete Verteiltechnik für die Realisierung der Anforderung Broadcast ist. Die unidirektionalen Verbindungen von einem Server zu mehreren Terminals bei einem reinen Broadcast-Szenario wurden mit den sehr langen Laufzeiten bei satellitengestützten und terrestrischen Übertragungstechniken begründet.

Der Rahmen zur Signalisierung und zum Transport von Datenströmen innerhalb MPEG-4 ist das *Delivery Multimedia Integration Framework (DMIF).* Die Besonderheit eines MPEG-4 Systems ist, dass dieses als Client auf einem Terminal aktiv Datenströme anfordert (Pull-Szenario). Das Broadcast-Szenario hingegen liefert die Datenströme vom Server zum Client (Push-Szenario). Aus der Signalisierung für bidirektionale Client-Server-Kommunikation (*Remote-Interactive-Szenario*) wurde ein Signalisierungsprotokoll für das Broadcast-Szenario entwickelt. Hierbei wurden für die sogenannte *Generische Signalisierung* MPEG-4 DMIF-konforme Protokollmeldungen und für die *IP-Multicast-Signalisierung* entsprechende Protokollmeldungen erarbeitet. Die Abbildungsvorschriften zwischen *Lokaler Syntax* und *Transfersyntax* der jeweiligen Signalisierungs-Meldungen wurden festgelegt. Ein Vergleich beider Verfahren ergibt, dass die *Generische Signalisierung* um den Faktor 3 geringere Bitraten benötigt als bei der *IP-Multicast-Signalisierung*. Die *IP-Multicast-Signalisierung* hat im Gegenzug den Vorteil, dass sie einfach lesbaren ASCII-codierten Text verwendet. Die entworfenen Signalisierungen wurden an Hand eines Demonstrators verifiziert.

Für das bei dem Paradigma *Fernsehen* übliche spätere Einschalten in eine laufende Übertragung, das sogenannte *Late Tuning-In*, wird bei MPEG-4 ein Karussell benötigt. Dieses muss alle statischen Elemente einer MPEG-4 Applikation periodisch wiederholen, damit ein später eingeschaltetes Terminal in die Lage versetzt wird, Datenströme zu öffnen und audiovisuell darzustellen. Es wurde festgestellt, dass mit der Karussell-Technik Redundanz in das System eingeführt wird und durch das periodisch wiederholte Aussenden der Daten schon eine Grundübertragungsrate im Netz belegt wird. Daraus folgt die Anforderung, dass der Autor einer MPEG-4 Applikation zu verwendende statische Elemente unter Berücksichtigung der erzeugenden Grundlast auswählen sollte.

Der Transport der multimedialen Datenströme mittels Broadcast-DMIF ist der zweite Schwerpunkt dieser Arbeit. Der Broadcast-Server entnimmt den MPEG-4 Datenströmen die Zeitstempel der zu versendenden Dateneinheiten und übergibt die einzelnen PDUs dem Broadcast-DMIF in Echtzeit. Um die Zeitstempel auslesen zu können, werden detaillierte Informationen über die Abhängigkeiten der einzelnen Elemente einer MPEG-4 Applikation benötigt. Es muss der initiale Objekt-Deskriptor *InitialOD,* dann der Objekt-Deskriptoren-Datenstrom *OD* mit der Szenenbeschreibung *BIFS* verarbeitet werden, damit letztendlich die Konfiguration der Paket-Header der MEPG-4 Datenströme bekannt ist. Erst mit diesen Informationen lassen sich die Zeitstempel der PDUs der multimedialen Datenströme verarbeiten und entsprechend in Echtzeit verarbeiten. Damit entsteht ein Verkehrsprofil an der Schnittstelle *DMIF-Application Interface (DAI)* des Transportsystems Broadcast-DMIF. Ziel dieser Arbeit ist es, im Client an dessen Schnittstelle *DAI* zum Player das identische Verkehrsprofil wiederherzustellen. Mit dem Demonstrator wurde gezeigt, dass dies mit dem entwickelten Broadcast-DMIF erreicht wird.

Bei der Diskussion der Netz-Protokolle wurde festgestellt, dass die Verwendung des RTP/UDP-IP-Multicast-Protokollstapels sowohl für die Signalisierung als auch für den Datenstromtransport op-

timal ist. Das RTP-Protokoll gewährleistet durch die transportierten Zeitstempel die Inter- und Intra-Datenstrom-Synchronität innerhalb und zwischen den übertragenen Datenströmen.

Die Funktionsfähigkeit des Broadcast-Szenarios über IP-Multicast wurde zuerst im LAN gezeigt und später LAN-übergreifend sowohl mit einem Nachbarlehrgebiet als auch mit einer anderen Universität nachgewiesen. Es wurde nicht überprüft, ob das Überschreiten der Grenze zwischen *Autonomen Systemen* (*AS*) für IP-Multicast funktioniert, dass diese Tests nicht spezifisch für diese Arbeit sind. Dies sind grundsätzliche Dienste eines *Internet Service Providers (ISP).*

Auf Grund der verkehrsverdrängenden Eigenschaft des ungesicherten und nicht fluß-gesteuertem Protokolls UDP wurden Mechanismen zur Regelung der benötigten Übertragungsrate einer MPEG-4 Applikation eingeführt. Es wurde eine obere Schranke für die verwendbare Übertragungsrate, die *Server Limited Bitrate* B_{SLB}, verwendet. Somit ist das Ziel, die Ankunftsraten der Datenströme innerhalb der *Server Limited Bitrate* einzuregeln. Dazu wurden die Begriffe *Traffic Shaping* für die Verkehrsformung unter Beibehaltung der mittleren Bitrate und *Rate Shaping* für die Ratenanpassung bei Änderung der mittleren Bitrate geprägt. Mit Mitteln der Verkehrs- und Bedientheorie wurde eine Abschätzung der benötigten Puffergrößen im Broadcast-DMIF-Server und Broadcast-DMIF-Client getroffen. Diese Abschätzungen wurden an Hand der durchgeführten Messungen mit dem Demonstrator bestätigt. Weiterhin wurden verschiedene Bedienstrategien für die Bedieneinheit der Warteschlangen und für das Traffic- und Rate Shaping untersucht, was zu dem Ergebnis führte, dass die *Weighted Round Robin*-Strategie ein fairer Algorithmus bei der Behandlung unterschiedlicher Datenströme ist. Dies konnte bei den durchgeführten Messungen mit dem Demonstrator nicht bestätigt werden. Hier waren vielmehr die verschiedenen untersuchten Algorithmen gleichwertig. Untersucht wurden die Strategien: *First-Come First-Serve*, *Round Robin* und *Weighted Round Robin* mit dynamischer und statischer Gewichtung. Das Ergebnis ist, dass die Ausgangsbitrate nicht von einer gewählten Bedienstrategie abhängt.

Die Ausgangsbitrate ist nur von dem gewählten Algorithmus für das Traffic Shaping abhängig. Hier wurde der Mechanismus der „virtuellen Übertragungszeit“ entwickelt, um eine Bitratenkontrolle unabhängig von der tatsächlichen Übertragungsrate des tatsächlichen Netzes zu erlangen. Eine andere Strategie, konstante Sendepause nach jeder versendeten PDU zu verwenden, führt nicht zur Einhaltung einer Server Limited Bitrate B_{SLB}.

Ein Ansatz für die Dimensionierung der durchsetzbaren Bitrate ist, den Mittelwert der Übertragungsrate einer MPEG-4 Applikation zu verwenden. Die Messungen führten aber zu dem Ergebnis, dass eine *Server Limited Bitrate* mindestens 50 % höher als der Mittelwert gewählt werden muss. Andernfalls sind die Datenströme derart verzögert oder mit Verlusten behaftet, dass in der Praxis die Player keine audiovisuelle Darstellung mehr vornehmen konnten.

Es bestätigte sich, dass Datenströme, die schon mit konstanter Bitrate (*CBR*) codiert wurden, gut für die Verkehrsformung geeignet sind.

Es wurde gezeigt, dass eine Ratenanpassung zwingend erforderlich ist, wenn die Ankunftsrate der Datenströme größer ist als die *Server Limited Bitrate*. Weiterhin wurde gezeigt, dass dann nur mit Verwendung des Rate Shapings die Echtzeitforderung am Client eingehalten werden kann. Bei der Betrachtung der verschiedenen Bedienstrategien für das Rate Shaping konnte keine optimale Strategie gefunden werden. Bei den Messungen mit dem Demonstrator wurden keine augenfälligen Unterschiede zwischen den Algorithmen *First-Come First-Serve, Round Robin und Weighted Round Robin* festgestellt. Mittels dieser Algorithmen werden PDUs ausgewählt, die verworfen werden müssen.

Zur Vermeidung von zusätzlichen Netzbelastungen durch die IP-Fragmentierung zu großer Datenpakete werden diese im Broadcast-DMIF auf die Größe einer parametrisierbaren *Maximum Transmission Unit Size (MTU)* gesplittet. Hier konnte auf Grund der durchgeführten Messungen festgestellt werden, dass große PDUs weniger Belastung des empfangenden Rechners bedeuten. Viele, kleine PDUs beanspruchen hingegen die Hardware des Rechners derart (Anzahl der Interrupts), dass die Player keine audiovisuelle Darstellung mehr darstellen konnten. Diese Erkenntnis steht im Widerspruch zu [Lin92], der bei Meldungen des ISDN-D-Kanal-Protokolls herausfand, dass kleinstmögliche Meldungen optimal sind.

Der Demonstrator wurde mittels der *Object Modelling Technique* entworfen und in der Programmiersprache C++ implementiert. Verschiedene Entwurfsmethoden nach [GHJV95] wurden verwendet.

Im praktischen Betrieb mussten äußere Einflüsse auf den Demonstrator erkannt und beseitigt werden. Dazu gehören die Prozessumschaltungen eines Betriebssystems sowie die verschiedenen Prioritätsklassen von Prozessen. Zum Einen musste der Thread zum Versenden der PDUs auf die Priorität *Real-Time* angehoben werden um Aktionen im Millisekundenbereich zu ermöglichen und zum Anderen musste Prozessorzeit abgeben werden, damit nach einer bestimmten Anzahl übergebener PDUs (*MaximumBurstSize*) andere Prozesse der Maschine das tatsächliche Versenden der PDU durchführen können.

Zum Schluss soll noch erwähnt werden, dass der Broadcast-DMIF so allgemein gehalten worden ist, dass der Transport nicht auf MPEG-4 spezifische Datenströme beschränkt ist. Wobei dieses Entwicklungsziel bisher nicht überprüft werden konnte.

Zukünftige Forschungsziele können verbesserte Algorithmen für die Bedienstrategie der Warteschlangen, des Traffic- und des Rate Shapings sein. Weiterhin können die Software des Broadcast-DMIF und der Player optimiert werden.

Anhang

A Abkürzungsverzeichnis und Glossar

A.1 Abkürzungen, Glossar

Bezeichner	Gebiet	Beschreibung
AAC	MPEG-4	Advanced Audio Coding
ABNF	IETF	Augmented Backus-Naur Form, Grammatik
ALF		Application Level Framing
API	Developer	Application Programming Interface
Amd	MPEG	Amendment, Ergänzung, neue Version des Standards
Ankunftsrate		Mittlere Anzahl von Ankünften je Zeiteinheit
ASCII		American Standard Code for Information Interchange
ATM	ATM	Asynchronous Transfer Mode
AU, Access Unit	MPEG-4	Access Unit, beinhaltet eine generische, formatabhängige Einheit, eines Elementardatenstroms, beispielsweise ein JPEG-Frame, ein I-Frame, etc.
AVC	MPEG-4	Advanced Video Coding, auch als H.264 bezeichnet
AVO		Audio Video Objekt
AVT WG	IETF	Audio-Video Transport Working Group, Arbeitsgruppe des IETF
Befehl		Aufruf der Schnittstellenfunktion für Dienstelemente eines Protokolls
BIFS	MPEG-4	Binary Format for Scenes, Szenenbeschreibung
CAT	MPEG-4	Channel Association Tag
CBR		Constant Bitrate
CCITT		Comité Consultatif International Télégraphique et Téléphonique, heute ITU
CD	MPEG	Committee Draft
CE	MPEG	Core Experiment
CELP	MPEG	Code Excited Linear Prediction
CIF	MPEG	Common Intermediate Format
CR LF		Carriage Return, Wagenrücklauf, zurück an Zeilenanfang und Line Feed, Zeilenvorschub, neue Zeile

CSELT/ TILAB		Centro Studi et Laboratori Telecommunicazioni, Forschungszentrum der Telecom Italia, heute Telecom Italia Lab, TILAB
CSRC	RTP	Contributing Source Identifier
CTS	MPEG	Construction Timestamp
DAI	MPEG-4	DMIF Application Interface
DCT		Diskrete Kosinus Transformation
DMIF	MPEG-4	Delivery Multimedia Integration Framework
DNI	MPEG-4	DMIF Network Interface
DNS		Domain Name System
DPI	MPEG-4	DMIF Plug-In Interface
DS	MPEG-4	DMIF Signalling
DSM	MPEG	Digital Storage Media
DSM-CC	MPEG	Digital Storage Media - Command and Control
DTS	MPEG	Delivery Timestamp, Decoding Timestamp
DVMRP		Distance Vector Multicast Routing Protocol
EBR	Netze	Equivalent Bitrate. Bitrate einer hypothetischen Verbindung mit konstanter Bitrate, die unter den gleichen Randbedingungen die gleiche Übertragungskapazität benötigt wie die betrachtete Verbindung mit variabler Bitrate.
ENST		École Nationale Superieur des Télécommunications
ES	MPEG-4	Elementary Stream, Elementardatenstrom
ESI	MPEG-4	Elementary Stream Interface
ESID	MPEG-4	Elementary Stream ID, eindeutiger Bezeichner für einen Elementardatenstrom
FBA	MPEG-4	Facial and Body Animation
FCD	MPEG	Final Committee Draft
FDIS	MPEG	Final Draft International Standard
FTP		File Transfer Protocol
GMC	MPEG	Global Motion Compensation, Bewegungsschätzung
GOP	MPEG	Group of Pictures, in MPEG-4 aber VOP
HDTV		High Definition Television

HTML		Hypertext Markup Language
HTTP		Hypertext Transfer Protocol
IANA	IETF	Internet Assigned Numbers Authority
ICMP		Internet Control Message Protocol
IETF		Internet Engineering Task Force, Standardisierungsgremium des Internet
IGMP		Internet Group Management Protocol
InterNIC	IETF	Internet Network Information Center
IOD	MPEG-4	Initial Object Descriptor
IP	IETF	Internet Protocol
IPMP	MPEG-4	Intellectual Property Management and Protection
IPR		Intellectual Property Rights
IS	MPEG	International Standard
ISDN		Integrated Services Digital Network
ISO		International Organisation for Standardisation
ITU		International Telecommunications Union
JPEG		Joint Pictures Expert Group
LAN		Local Area Network
MBone	IETF	Multicast Backbone on the Internet
MCU		Multipoint Control Unit
MDCT		Modifizierte Diskrete Kosinus Transformation
Meldung		Aufruf der Schnittstellenfunktion für Dienstelemente eines Protokolls
MFC	Developer	Microsoft Foundation Class Library
MIME		Multipurpose Internet Mail Extension
MOSPF		Multicast Open Shortest Path First
MPEG		Moving Pictures Experts Group
MTU		Maximum Transmission Unit Size, maximale Größe eines Pakets
NB	MPEG	National Body
NTP		Network Time Protocol
OD	MPEG-4	Object Descriptor, Objekt-Deskriptor

OMG		Object Management Group
OSI		Open System Interconnection
PAR		Positive Acknowledgement with Retransmission
PCM		Pulse Code Modulation
PDU		Protocol Data Unit
PSNR		Peak Signal to Noise Ratio
PT	RTP	Payload Type
PTS	MPEG	Presentation Timestamp
QCIF	MPEG	Quarter Common Intermediate Format
QoS		Quality of Service
RAP	MPEG	Random Access Point, wahlfreier Zugriff, Synchronisationspunkt
Rendering	MPEG	Der Prozess zur Erstellung von Pixeln für die Anzeige beziehungsweise Töne für das Hören
RPC		Remote Procedure Call
RR	RTP	Receiver Report
RTCP	RTP	RTP Control Protocol
RTP	RTP	Real-Time Transport Protocol
RTSP	RTP	Real-Time Streaming Protocol
SAP		Session Announcement Protocol
SAP		Service Access Point
SBR	Netze	Sustainable Bitrate. Obere Grenze der mittleren Bitrate. Aufrechterhaltbare oder durchsetzbare Bitrate
SDES		Source Description
SDF		Shortest Deadline First
SDP		Session Description Protocol
SDR		Session Directory
SIP		Session Initiation Protocol
SL		Synchronisation Layer
SMIL		Synchronized Multi-media Integration Language

SNHC	MPEG	Synthetic and natural hybrid coding, Codierung von synthetischen Objekten (Gesichts- und Körperanimation) und Codierung von natürlichen Objekten, beispielsweise aus einem Film, Stillbild
SNR		Signal to Noise Ratio
SR	RTP	Sender Report
SSRC	RTP	Synchronisation Source Identifier
SVG		Scalable Vector Graphics
TAT	MPEG-4	TransMux Channel Association Tag
TCP		Transmission Control Protocol
TransMux	MPEG-4	Generische Abstraktion für beliebige Transportprotokolle
TTL		Time-To-Live
TTS	MPEG-4	Text-to-speech
UDP		User Datagram Protocol
UM	Developer	Unified Method
UML	Developer	Unified Modeling Language
UMTS		Universal Mobile Telecommunication System, Mobilfunk 3. Generation
URI		Universal Resource Identifier
URL		Uniform Resource Locator
URN		Universal Resource Name
VBR		Variable Bitrate
VM	MPEG	Verification Model
VOH	MPEG	Video Object Header
VOL	MPEG	Video Object Layer
VOP	MPEG	Video Object Plane, das gleiche wie in MPEG-2 ein „Frame“, allerdings möglicherweise mit einer beliebigen Kontur
VRML	MPEG-4	Virtual Reality Modelling Language
W3C		World Wide Web Consortium
WD	MPEG	Working Draft
WWW		World Wide Web
XDR		External Data Representation
XMT	MPEG-4	Extensible MPEG-4 textual format

A.2 Mathematische Bezeichnungen

W(y)	Wahrscheinlichkeit, mit der die Wartezeit unter dem Wert y liegt.
σ_a^2	Varianz Ankunftsabstand
σ_b^2	Varianz Bediendauer
$\overline{T_A}$	Mittlerer Ankunftsabstand
$\overline{T_B}$	Mittlere Bediendauer
$\overline{\rho}$	Mittlere Auslastung der Bedieneinheit
λ_{SBR}	Durchsetzbare Übertragungsrate, Sustainable Bitrate
Q	Warteschlangenlänge
$\overline{Q}$	mittlere Warteschlangenlänge
T_N	Verweilzeit im Netz
T_C	Wartezeit in der Client-Warteschlange
T_S	Wartezeit in der Server-Warteschlange
$\overline{T_c}$	mittlere Client-Wartezeit
$T_W, \overline{T_W}$	Wartezeit, mittlere Wartezeit

B Unified Modeling Language

Der Standard der Notation für die objektorientierte Modellierung ist die ***U**nified **M**odeling **L**anguage (UML)*.

UML ist in erster Linie eine Modellierungssprache und keine Modellierungsmethode[1] und ist eine Notation zur Spezifikation, Konstruktion, Visualisierung und Dokumentation von Modellen für Softwaresysteme. Entsprechend geeignet ist diese für die Umsetzung in Programmiersprachen, also der Code-Generierung.

UML definiert einen Satz von Modellelementen mit zugeordneten graphischen Symbolen, aus denen Modelle konstruiert werden können, die

- einen Überblick über das Gesamtsystem,
- das extern erkennbare Verhalten des Systems,
- die logische Struktur des Aufbaus,
- allgemeine und spezielle Ablaufstrukturen sowie Modulstrukturen und Konfigurationen

repräsentieren.

Ausgehend von der historischen Entwicklung der UML wird die objektorientierte Modellierung als wichtige Anwendung der Objekttechnologie herausgestellt.

Die Modellierungsmethoden von *Grady Booch* und *James Rumbaugh* haben sich Anfang der 90er Jahre zu den Modellierungsmethoden schlechthin entwickelt [RBP+91] [Boo94].

Die Modellierungsmethode von *Rumbaugh* war dabei an die strukturierten Modellierungsmethoden angelehnt. 1995 begannen *Booch* und *Rumbaugh* ihre Modellierungsmethoden zunächst in Form einer gemeinsamen Notation zur ***U**nified **M**ethod (UM)* zusammenzuführen. Später integrierte *Ivar Jacobson* die von ihm beschriebenen Anwendungsfälle (*Use Cases*). Aus dieser Zusammenarbeit ist die gemeinsame Beschreibungssprache *UML* für die objektorientierte Analyse und das objektorientierte Design entstanden. Die Modellierungsmethoden von *Booch*, *Rumbaugh* und *Jacobson* und deren Zusammenführung zur UML bilden einen Quasi-Standard.

UML wurde 1997 in der Version 1.1 bei der ***O**bject **M**anagement **G**roup (OMG)* standardisiert und wird permanent weiterentwickelt.

B.1 Konzepte der objektorientierten Systementwicklung

Die grundlegenden Konzepte der objektorientierten Systementwicklung sind *Abstraktion* und *Kapselung*. Es resultieren operative Bausteine in Gestalt von *Objekten*, die jeweils einen wohldefinierten und begrenzten Aufgabenbereich abdecken. Die Eigenschaften und das Verhalten gleichartiger Objekte werden in abstrakter Form durch eine *Klasse* spezifiziert.

Abstraktion, als ein Prozess betrachtet, bezeichnet die Vorgehensweise, die wesentlichen Details zu ermitteln und die unwesentlichen Details zu ignorieren [Ber93].

Die *Kapselung* sagt aus, dass zusammengehörende Attribute und Operationen, in einer Einheit, der *Klasse,* gekapselt sind. Abbildung 242 veranschaulicht die Kapselung der Daten eines Objektes.

[1] im Sinne von Handlungsanweisung

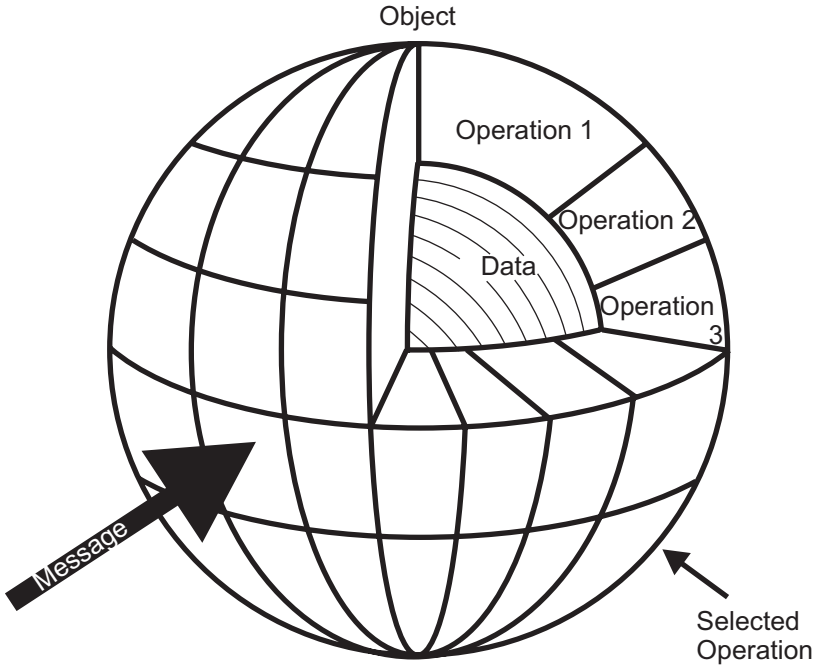


Abbildung 242: Kapselung (*Information Hiding*)

In objektorientierten Systemen gibt es im Wesentlichen folgende Begriffe und Komponenten [Bur92]:

• Klasse	• Objekt	• Attribut
• Operation (Methode)	• Polymorphismus	• Prozess
• Ereignis	• Assoziation/Rolle	• Aggregation
• Vererbung		

Tabelle 52: Begriffe objektorientierter Systeme

Diese Begriffe objektorientierter Systeme lassen sich aus der Sicht der objektorientierten Modellierung in die zwei folgenden Kategorien einordnen.

B.1.1 Statische Aspekte

Bei den statischen Aspekten sind die Beschreibungselemente mit deklarativem Charakter zu betrachten. Sie beschreiben die Systemstruktur, die Klassen und die zwischen ihnen existierenden Beziehungen zusammen mit den Restriktionen, die erfüllt sein müssen. Das sind im Wesentlichen die *Klasse*, die *Rolle* bzw. *Assoziation* und das *Modul*. Die Klasse bringt die in ihr enthaltenen Attribute und Methoden mit ein. Klassen werden oft als abstrakte Datentypen bezeichnet, die um die Vererbung erweitert wurden. Die Rolle erlaubt Restriktionen bezüglich beteiligter Klassen und bezüglich ihrer Kardinalität. Inverse Rollen können zu Assoziationen zusammengefasst werden.

B.1.2 Dynamische Aspekte

Bei den dynamischen Aspekten ist die Verhaltensweise des Systems zu betrachten. Charakteristisch für dynamische Aspekte sind Werte. Hierzu gehören alle Systemkomponenten mit Instanzcharakter. Die wesentlichen Begriffe sind *Objekt*, *Prozess* und *Ereignis*. Das Objekt ist die Instanz einer Klasse und tritt im Umfeld dieser Instanzierung mit einem Objektzustand auf. Der Prozess ist die Instanz einer Methode. Ereignisse sind zu differenzieren in reine Zustandsänderungen, also

implizit auftretende Ereignisse und explizit ausgelöste Ereignisse, die Botschaften (*Messages*) genannt werden.

B.2 Objektorientierte Modellierung mit der UML

Die Modellierungssprache UML liefert einen Standard für die Notation eines objektorientierten Modells. Die objektorientierten Modellierungsmethoden und die Modellierungssprache stehen in einem engen Zusammenhang. Daher werden im Folgenden die wichtigsten Modellierungsmethoden zusammen mit den Modellelementen der UML erklärt [BJR97a], [BJR97b], [BJR97a], [Oes98]:

- Anwendungsfalldiagramme (*use case*)
- Basiselemente zur Darstellung von Klassendiagrammen
- Beziehungselemente zur Darstellung von Klassendiagrammen
- Klassendiagramme (*class diagram*)
- Verhaltensdiagramme

B.2.1 Anwendungsfalldiagramme

Das Anwendungsfallmodell (*use case model*) beschreibt die externen Akteure (*actors*) und die Szenarien (*use cases*) des Systems [JCJ+92].

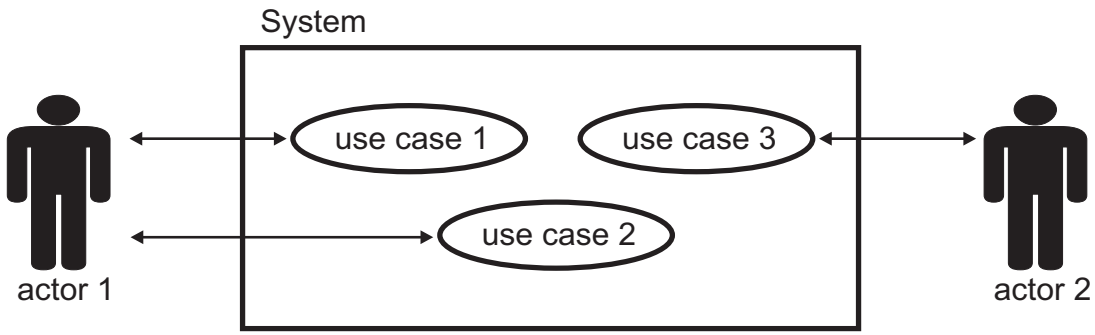


Abbildung 243: Akteure (*actors*) und Szenarien (*use cases*)

Die Akteure repräsentieren die zukünftigen Anwender. Sie definieren die Rollen, in denen die Anwender auftreten können. Bei den Akteuren kann es sich um eine Person, ein anderes System oder auch eine Maschine handeln. Ein Szenario (*use case*) ist eine Sequenz von Transaktionen, die ein Akteur im Dialog mit dem System ausführt. Eine Transaktion kann aus mehreren ausführbaren Aktionen bestehen. Sie endet, wenn das Szenario auf eine neue Eingabe wartet. Das Szenario wird in frei formulierbarem Text beschrieben. Es spezifiziert die Interaktionen, die zwischen einem Bediener und dem System stattfinden. Die Menge aller Szenarien beschreibt demnach alle Möglichkeiten, wie das System benutzt werden kann.

Mittels der Akteure lassen sich die Szenarien identifizieren. Für jeden Akteur wird entschieden, welche Szenarien er ausführt.

Aus den Szenarien leitet Jacobson Objekte beziehungsweise Klassen ab. Hierbei ist für die Modellierung entscheidend, von welchen Klassen Daten benötigt werden, beziehungsweise, welche Klassen Teilaufgaben durchführen sollen.

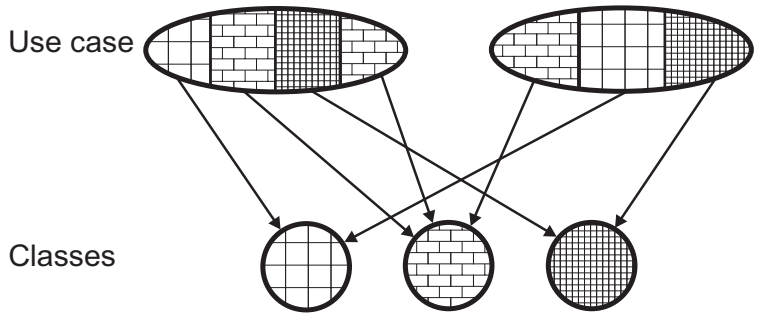


Abbildung 244: Extraktion von Klassen

Aus dem Anwendungsfallmodell leitet Jacobson ebenfalls die *Operationen* ab [JCJ+92]. Für jedes Szenario erstellt er ein Sequenzdiagramm (*Sequence diagram*), das zeigt, wie die beteiligten Objekte das Szenario durch geeignete Kommunikation realisieren.

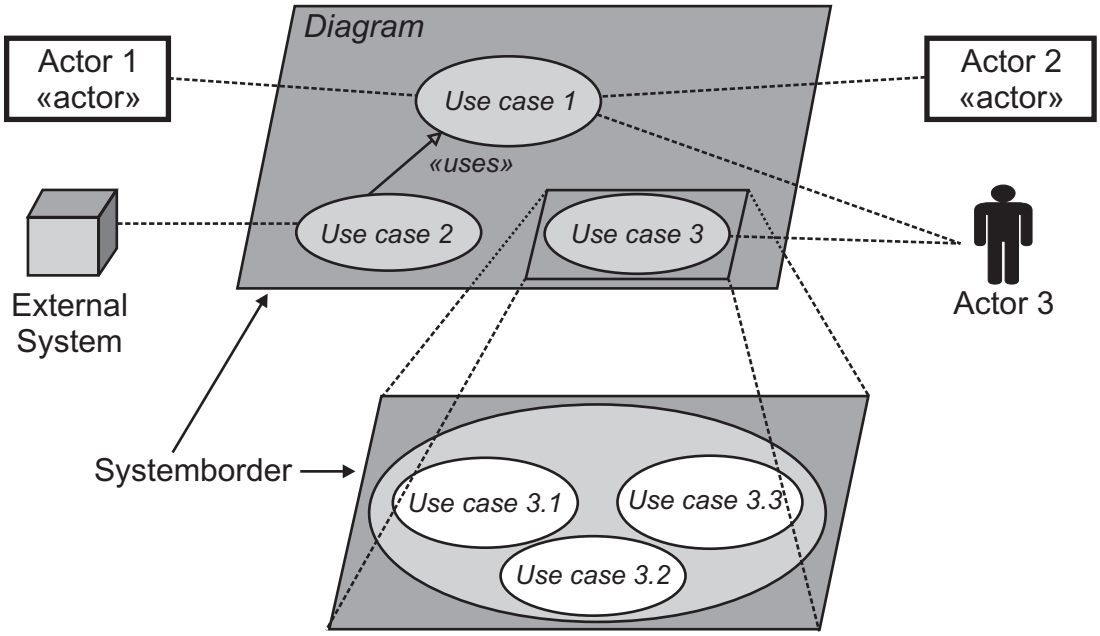


Abbildung 245: Hierarchische Gliederung von Anwendungsfällen (*use case*)

Ein *Anwendungsfalldiagramm* enthält eine Menge von *Anwendungsfällen*, die durch einzelne Ellipsen dargestellt werden und eine Menge von Akteuren und Ereignissen, die daran beteiligt sind. Die Anwendungsfälle sind durch Linien mit den beteiligten Klassen verbunden. Ein Rahmen um die Anwendungsfälle symbolisiert die Systemgrenzen. Anwendungsfalldiagramme können hierarchisch verschachtelt werden, das heißt, ein Anwendungsfall in einem Anwendungsfalldiagramm kann durch ein weiteres Diagramm in detailliertere Anwendungsfälle untergliedert werden.

B.2.2 Basiselemente zur Darstellung von Klassendiagrammen

Die einzelnen Basiselemente der UML zur Darstellung von Klassendiagramm werden hier erläutert.

B.2.2.1 Objekt

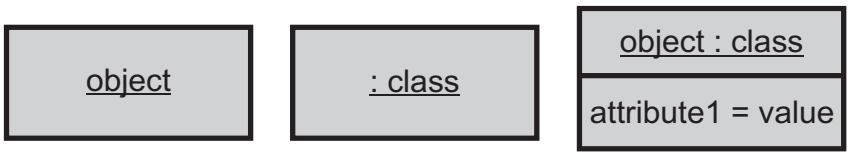


Abbildung 246: Notation: Objekt und Klasse

In der objektorientierten Softwareentwicklung besitzt ein *Objekt* bestimmte Eigenschaften und reagiert mit einem definierten Verhalten auf seine Umgebung. Außerdem besitzt jedes Objekt eine *Identität*, die es von allen anderen Objekten unterscheidet. Die Eigenschaften eines Objekts werden durch dessen Attributwerte ausgedrückt, sein Verhalten durch eine Menge von Operationen (*Methoden*).

Objekte werden durch Rechtecke dargestellt (Abbildung 246), die entweder nur ihren Namen (linke Abbildung), zusätzlich auch den Namen ihrer Klasse (mittlere Abbildung) oder auch die Werte bestimmter oder aller Attribute tragen. Werden die Attributwerte angegeben, wird das Rechteck in zwei durch eine horizontale Linie getrennte Rechtecke aufgeteilt (rechte Abbildung). Zur Unterscheidung von der Klassen-Notation wird der Name des Objekts unterstrichen. Der Objektname beginnt gewöhnlich mit einem Kleinbuchstaben.

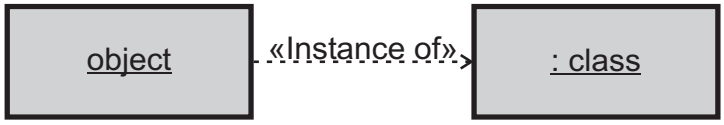


Abbildung 247: Instanzierung in der UML

Instanzierungsbeziehungen (Abbildung 247) sind Klassen-Objekt - Beziehungen und werden durch einen gestrichelten Pfeil dargestellt. Das Objekt zeigt auf seine Klasse.

B.2.2.2 Klasse

Eine *Klasse* (*class*) beschreibt eine Kollektion von Objekten mit gleichen Eigenschaften (Attribute), gemeinsamer Funktionalität (Operationen), gemeinsamen Beziehungen zu anderen Objekten und gemeinsamer Semantik [RJB95]. Ferner legt die Klasse fest, wie neue Objekte dieser Klasse erzeugt werden. Eine Klasse definiert also die Eigenschaften und das Verhalten ihrer Objekte.

B.2.2.3 Attribut

Die *Attribute* beschreiben die Daten beziehungsweise die Eigenschaften einer Klasse. Alle Objekte einer Klasse besitzen dieselben Attribute, jedoch unterschiedliche Attributwerte (*values*).

B.2.2.4 Operationen

Operationen beschreiben die Funktionalität eines Objekts. Eine Operation kommuniziert mit anderen Objekten über Ein- und Ausgabeparameter. Alle Objekte einer Klasse verwenden dieselben Operationen. Diese Objekt-Operationen werden jeweils auf ein einzelnes Objekt angewandt. Jede Operation kann auf alle Attribute eines Objekts dieser Klasse zugreifen.

Die Menge aller Operationen wird als das Verhalten der Klasse bezeichnet.

B.2.2.5 Darstellung

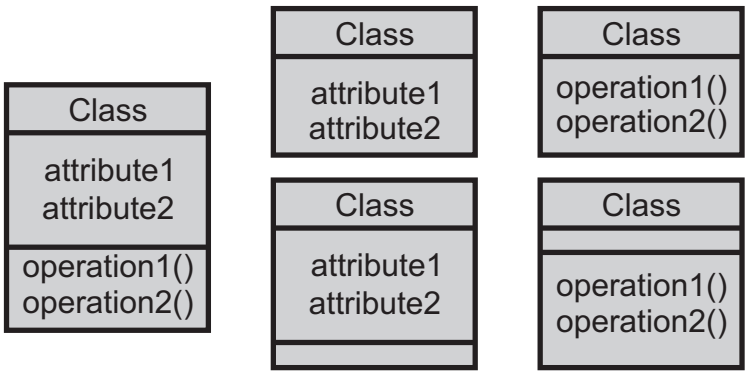


Abbildung 248: Notierung einer Klasse (*class*)

Klassen werden durch Rechtecke dargestellt, die entweder nur den Namen der Klasse oder zusätzlich auch Attribute und Operationen tragen. Dabei werden diese drei Abschnitte - *Klassenname*, *Attribute*, *Operationen* - jeweils durch eine Linie getrennt. Klassennamen beginnen mit einem Großbuchstaben und sind Substantive im Singular.

Attribute werden mindestens mit ihrem Namen aufgeführt und können zusätzlich Angaben zu ihrem Typ, das heißt, ihrer Klasse, einem Initialwert, Eigenschaftswerten und Zusicherungen enthalten.

Operationen werden ebenfalls mindestens mit ihrem Namen, und zusätzlich durch mögliche Parameter, deren Klasse und Initialwerte sowie eventuellen Eigenschaftswerten und Zusicherungen notiert.

B.2.3 Beziehungselemente zur Darstellung von Klassendiagrammen

Die einzelnen Elemente der UML zur Darstellung von statischen Beziehungen werden hier erläutert.

B.2.3.1 Vererbung

Vererbung bedeutet, dass eine Klasse über die Eigenschaften und das Verhalten einer anderen Klasse verfügen kann.

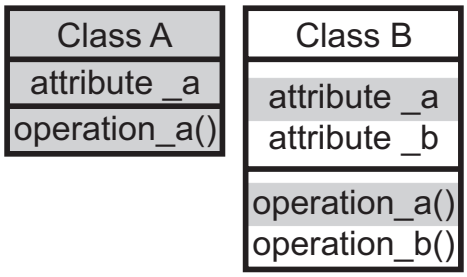


Abbildung 249: Die Klasse B erbt Eigenschaften und Verhalten der Klasse A

In der Abbildung 249 besitzt die Klasse B außer ihren eigenen Attributen und Operationen auch die Attribute und Operationen der Klasse A. Sie kann die geerbten Eigenschaften und das geerbte Verhalten redefinieren. Die Klasse, die Eigenschaften weitergibt, wird *Oberklasse* oder *Basisklasse* genannt, während die erbende Klasse *Unterklasse* oder *Subklasse* genannt wird. Das hierbei angewendete Prinzip wird gewöhnlich *Generalisierung* (*generalisation*) beziehungsweise *Spezialisierung* genannt [RJB95].

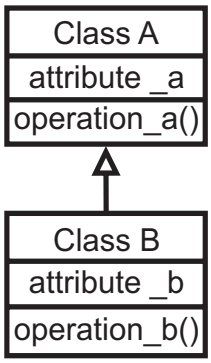


Abbildung 250: Vererbung in UML-Notation

Dieses Prinzip begründet die Wiederverwendungsmöglichkeiten in der Objektorientierung. Die Klasse A ist eine Generalisierung der Klasse B und die Klasse B ist eine Spezialisierung der Klasse A. Die Vererbungsbeziehung wird mit einem großen, nicht ausgefüllten Pfeil dargestellt, wobei der Pfeil von der Unterklasse zur Oberklasse zeigt.

B.2.3.2 Assoziation

Eine *Assoziation* ist die allgemeinste Beziehung zwischen zwei Klassen in einem objektorientierten System. Eine Assoziation beschreibt eine Verbindung zwischen Klassen. Assoziationen sind notwendig, damit Objekte miteinander kommunizieren können. Die konkrete Beziehung zwischen zwei Objekten dieser Klassen wird *Objektverbindung* (*link*) genannt [RJB95]. Objektverbindungen sind also die Instanzen einer Assoziation.

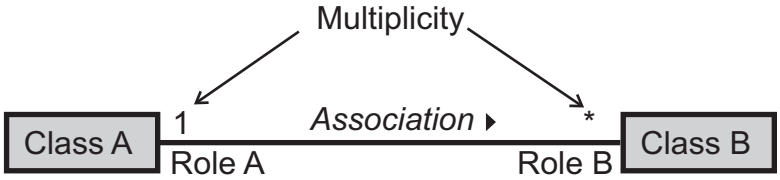


Abbildung 251: Assoziation in der UML

Die *Kardinalität (multiplicity*) einer Assoziation gibt an, mit wie vielen Objekten der gegenüberliegenden Klasse ein Objekt assoziiert sein kann.

Jede Assoziation kann mit einem Namen versehen werden, der die Beziehung durch ein Verb näher beschreiben sollte.

Beziehungen werden durch eine Linie zwischen den beteiligten Klassen dargestellt. An den jeweiligen Enden kann die Kardinalität der Beziehung angegeben werden. Jede Beziehung wird mit einem kursiv geschriebenen Namen versehen. Um die Klassennamen und die Bezeichnung der Beziehungen in der richtigen Richtung lesen zu können, kann neben dem Beziehungsnamen ein kleines ausgefülltes Dreieck gezeichnet werden, dessen Spitze in die Leserichtung zeigt.

An jedem Ende einer Beziehung können zusätzlich Rollennamen (*role*) angegeben werden. Ein Rollenname beschreibt, wie das Objekt durch das in der Assoziation gegenüberliegende Objekt gesehen wird.

B.2.3.3 Aggregation

Die *Aggregation* zeigt eine Zugehörigkeit mehrerer *Teile* (*parts*) zu einem *Ganzen* (*assembly*) an. Das Ganze ist dabei ein Aggregat, dessen Behandlung auch die Teile betrifft. Die Aggregatklasse enthält beispielsweise Operationen, die keine unmittelbare Veränderung im Aggregat selbst bewirken, sondern die Nachricht an seine Teile weiterleiten. Im Gegensatz zur Assoziation führen die

beteiligten Klassen also keine gleichberechtigte Beziehung, sondern eine Klasse (Aggregat) bekommt eine besondere Rolle und übernimmt stellvertretend die Verantwortung und Führung.

Je nach Sicht wird die Aggregation als *enthält*-Beziehung (*has*-relationship) oder als *ist-Teil-von*-Beziehung (*is-part-of*-relationship) interpretiert.

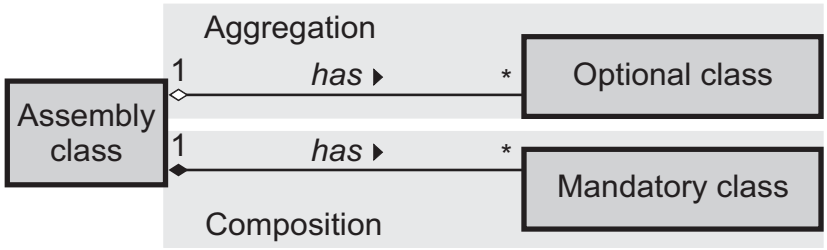


Abbildung 252: Komposition und Aggregation in der UML

Eine Aggregation wird wie eine Assoziation als Linie zwischen zwei Klassen dargestellt und zusätzlich mit einer kleinen Raute versehen (Abbildung 252). Die Raute steht auf der Seite des Aggregats. Im Übrigen gelten alle Notationskonventionen der Assoziation. Der Wert der Kardinalität auf der Seite des Aggregats ist „1“, wenn dort eine explizite Angabe fehlt. Ein Teil kann gleichzeitig zu mehreren Aggregationen gehören.

B.2.3.4 Komposition

Eine *Komposition* ist eine spezielle Variante der Aggregation. Die Komposition beschreibt Beziehungen, in denen die Teile vom Aggregat existenzabhängig sind. Mit der Löschung des Aggregates werden auch alle abhängigen Teile gelöscht. Wird hingegen ein Einzelteil gelöscht, bleibt das Aggregat erhalten. Existenzabhängigkeiten, wie sie durch die Komposition beschrieben werden, ermöglichen nicht nur die realitätsnahe Modellierung, sondern sind auch ein Mittel zur Vereinfachung von Konsistenzbedingungen.

Die Komposition wird wie die Aggregation als Linie zwischen zwei Klassen gezeichnet und mit einer kleinen Raute auf der Seite des Ganzen versehen. Im Gegensatz zur Aggregation wird die Raute jedoch ausgefüllt. Die Abbildung 252 zeigt die Notation von Komposition und Aggregation in der UML.

B.2.4 Klassendiagramme

Das statische Modell repräsentiert eine logische Abstraktion des eigentlichen Anwendungssystems, die auf statische Aspekte beschränkt ist. Das *Klassendiagramm* zeigt die logische Architektur eines Anwendungssystems und eine Menge statischer Modellelemente, vor allem Klassen und ihre Beziehungen. Hierbei kann es sich, wie oben beschrieben, sowohl um Vererbungsbeziehungen, als auch um Nutzungsbeziehungen handeln.

Inhalt dieser Diagrammform sind die statischen Strukturen der Klassen inklusive Vererbung und Assoziationen, gegebenenfalls Klasseninhalte, also Operationen und Attribute.

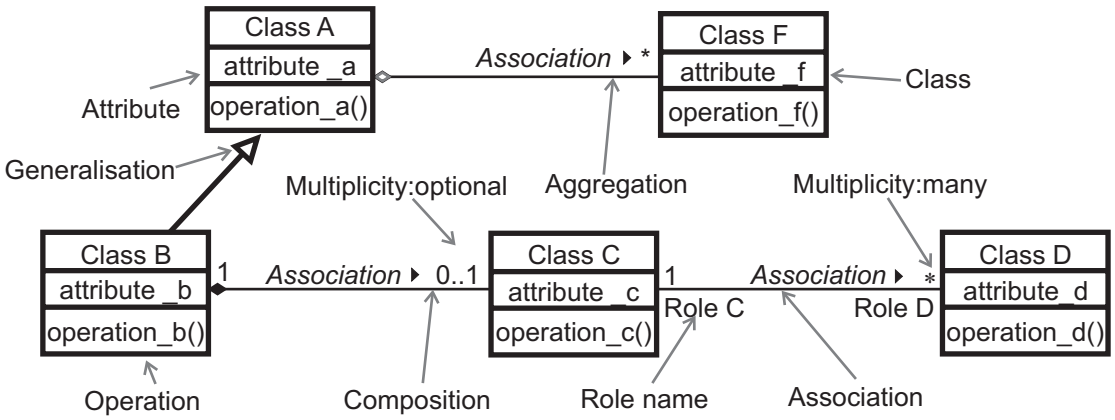


Abbildung 253: UML-Klassendiagramm – statisch

B.2.5 Verhaltensdiagramme

Das dynamische Modell ist die Abstraktion des Anwendungssystems unter Berücksichtigung dynamischer Aspekte. Bei der Erstellung des Modells wird das innere dynamische Verhalten eines bestimmten Objekts analysiert, das mit anderen Objekten interagiert. Dabei wird untersucht, wie das Objekt agiert und auf Ereignisse reagiert. Bei seiner Erstellung wird vom statischen Modell ausgegangen.

B.2.5.1 Zustandsdiagramme

Zustandsdiagramme (*State diagrams*) sind eine verbreitete Methode, um das Verhalten eines Systems zu beschreiben. Sie beschreiben alle möglichen Zustände, die ein bestimmtes Objekt annehmen kann. In den meisten objektorientierten Methoden werden Zustandsdiagramme für einzelne Klassen entworfen, um das Verhalten während der Lebenszeit eines einzelnen Objektes aufzuzeigen. Alle Objekte einer Klasse besitzen dasselbe Zustandsdiagramm, jedoch kann jedes Objekt einen individuellen Zustand einnehmen.

Ein Zustandsdiagramm besteht aus *Zuständen* und *Zustandsübergängen*. Jeder Zustand

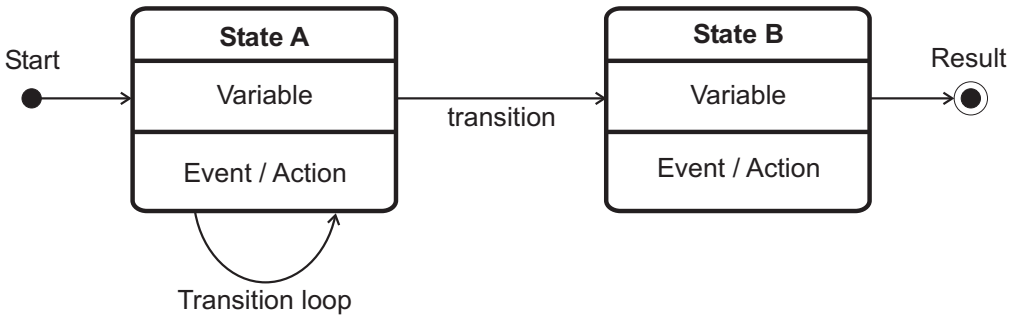


Abbildung 254: UML-Zustandsdiagramm

repräsentiert eine bestimmte Situation, in der sich das Objekt befindet. In diesen Zustand gelangt das Objekt durch ein entsprechendes *Ereignis* (*event*). Der Übergang in den nächsten Zustand wird auch als *Transition* bezeichnet.

Ein Objekt kann im Allgemeinen mehrere Zustände einnehmen. Zu einem Zeitpunkt befindet es sich jedoch in genau einem Zustand. Tritt in einem Zustand ein Ereignis ein, so erfolgt ein Zustandsübergang (*transition*). Es gibt genau einen Anfangszustand nach der Erzeugung des Objektes.

Ein Zustand im Zustandsdiagramm wird als Rechteck mit abgerundeten Ecken dargestellt. Sie können einen Namen beinhalten und optional in bis zu drei Bereiche geteilt werden. Im oberen Bereich steht der Name des Zustandes. In einem weiteren Bereich können existierende Zustandsvariablen (Attribute der Klasse) aufgelistet werden. Der dritte Bereich innerhalb des Zustandssymbols enthält eine Liste möglicher interner Ereignisse, Bedingungen und aus ihnen resultierende Operationen. Die Aktionsbeschreibung kann ein Operationsname sein oder frei formuliert werden, ferner kann sie Zustandsvariablen, Attribute der Klasse oder Parameter der eingehenden Transition enthalten.

Ein Startzustand wird als kleiner ausgefüllter Kreis gezeichnet, die Endzustände jeweils durch einen nicht ausgefüllten Kreis, in dem ein kleinerer, ausgefüllter Kreis liegt.

Die Pfeile werden mit Transitionsbeschreibungen versehen.

B.2.5.2 Aktivitätsdiagramm

Während in einem Zustandsdiagramm die Reaktionen eines Objekts auf diverse Ereignisse dargestellt werden, macht ein *Aktivitätsdiagramm* die Einzelheiten beim internen Ablauf der Operationen des Objekts sichtbar. Die Beschreibung nebenläufiger *Aktivitäten* wird durch Aktivitätsdiagramme unterstützt. Ein Aktivitätsdiagramm ist eine spezielle Form des Zustandsdiagramms, das Aktivitäten enthält. Der Zustand heißt hier Aktivität, und orientiert sich an der Operation (Methode), die ausgeführt wird. Eine Aktivität ist ein einzelner Schritt in einem Verarbeitungsablauf. Dementsprechend ist jede Aktivität selbst wieder sinnvoll als eigenes Aktivitätsdiagramm darstellbar. Eine Aktivität kann mehrere ausgehende Transitionen haben, wenn diese durch Bedingungen unterschieden werden können.

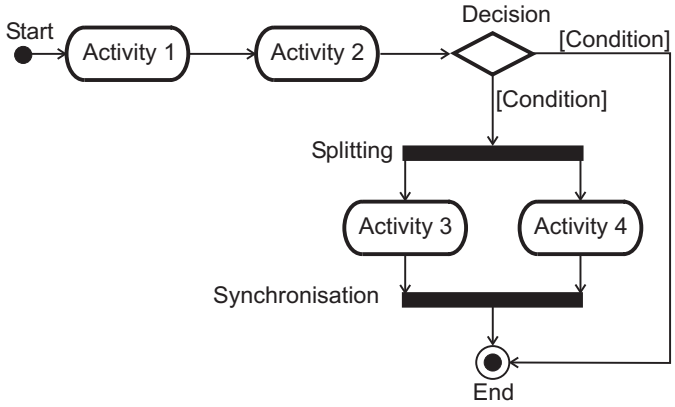


Abbildung 255: UML-Aktivitätsdiagramm

Dargestellt wird eine Aktivität durch eine Figur mit gerader Ober- und Unterseite und konvex geformten Seiten. Die Figur enthält eine Aktionsbeschreibung, welches ein Name oder eine frei formulierte Beschreibung sein kann.

Eingehende Transitionen lösen eine Aktivität aus. Sofern zu einer Aktivität mehrere eingehende Transitionen existieren, kann jede dieser Transitionen unabhängig von den anderen die Aktivität auslösen.

Die ausgehenden Transitionen werden durch Pfeile dargestellt. Sie können mit Bedingungen versehen werden, die in eckigen Klammern stehen. Bei den Bedingungen sollte es sich um boolesche Ausdrücke handeln. Außerdem können Transitionen synchronisiert und geteilt werden.

B.2.5.3 Sequenzdiagramm

Das *Sequenzdiagramm* (*Sequence diagram*) ist eine Methode, um die Kommunikation zwischen Objekten mittels *Botschaften* (*messages*) zu modellieren. Eine Sequenz ist eine Reihe von Botschaften, die eine ausgewählte Menge von Objekten in einer zeitlich begrenzten Situation austauscht, wobei der zeitliche Ablauf betont wird. Eine Botschaft ist eine Aufforderung eines *Senders* an einen *Empfänger*, eine Dienstleistung zu erbringen. Der Empfänger interpretiert diese Botschaft und führt eine Operation aus. Eine Botschaft besteht aus dem Namen der Operation und den Argumenten (Eingabeparametern), die diese Operation benötigt. Der Sender der Botschaft weiß nicht, wie die entsprechende Operation ausgeführt wird. Sender und Empfänger sind meistens Objekte.

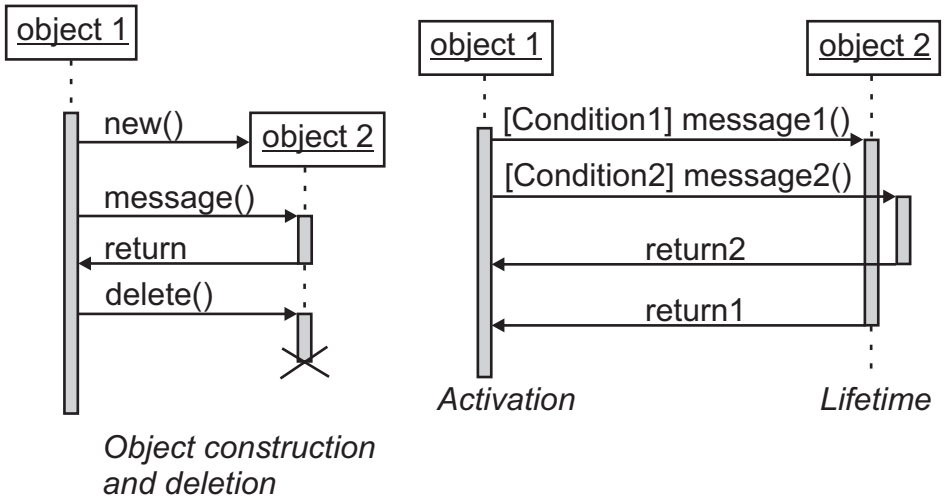


Abbildung 256: UML-Sequenzdiagramm

Dieser Diagrammtyp wurde lange Zeit im Bereich der Telekommunikation benutzt [JCJ+92]. Er ist in der Analysephase zur Spezifikation des dynamischen Verhaltens einsetzbar [Boo94]. In einem Sequenzdiagramm wird ein Objekt durch ein Objektsymbol über einer gestrichelten, vertikalen Linie dargestellt. Diese vertikale Linie wird die *Lebenslinie* (*lifetime*) des Objekts genannt. Die Lebenslinie repräsentiert die Existenz des Objekts während der Interaktion. Die Botschaften werden als waagerechte Pfeile zwischen den Objekt-Lebenslinien gezeichnet. Die Überlagerung der Lebenslinien durch breite, nicht ausgefüllte (oder graue) senkrechte Balken, symbolisiert *Aktivierungen*, die explizit auftreten, wenn eine Methode aktiv ist, weil sie entweder gerade ausgeführt wird oder auf das Zurückkehren einer Unterprozedur wartet. Wie in den Aktivitätsdiagrammen können die Botschaften mit Bedingungen versehen werden, die in eckigen Klammern stehen.

C Erstellung von Inhalten

C.1 Das Autorensystem

Die französische Hochschule ***É**cole **N**ationale **S**upérieur des **T**elecommunications* (*ENST*), Paris, hat innerhalb der MPEG-4 Standardisierungsaktivitäten ein einfaches Autorenwerkzeug ***M**PEG-4 **D**ev**S**tudio (MDS)* zur Verifikation der Konformität von Bitströmen erstellt. Dieses Werkzeug ist eine betriebssystemunabhängige Implementation in Java und erfordert fundierte Kenntnisse der Elemente der Szenenbeschreibung mittels ***B**inary **I**nformation **F**or **S**cenes (BIFS)*. Als Vorteil für die Verwendung von BIFS ist die Tatsache anzusehen, dass BIFS auf den Elementen und Prinzipien von VRML [VRML97], einer weiteren Arbeitsgruppe des ISO (ISO/IEC JTC1 SC24), basiert.

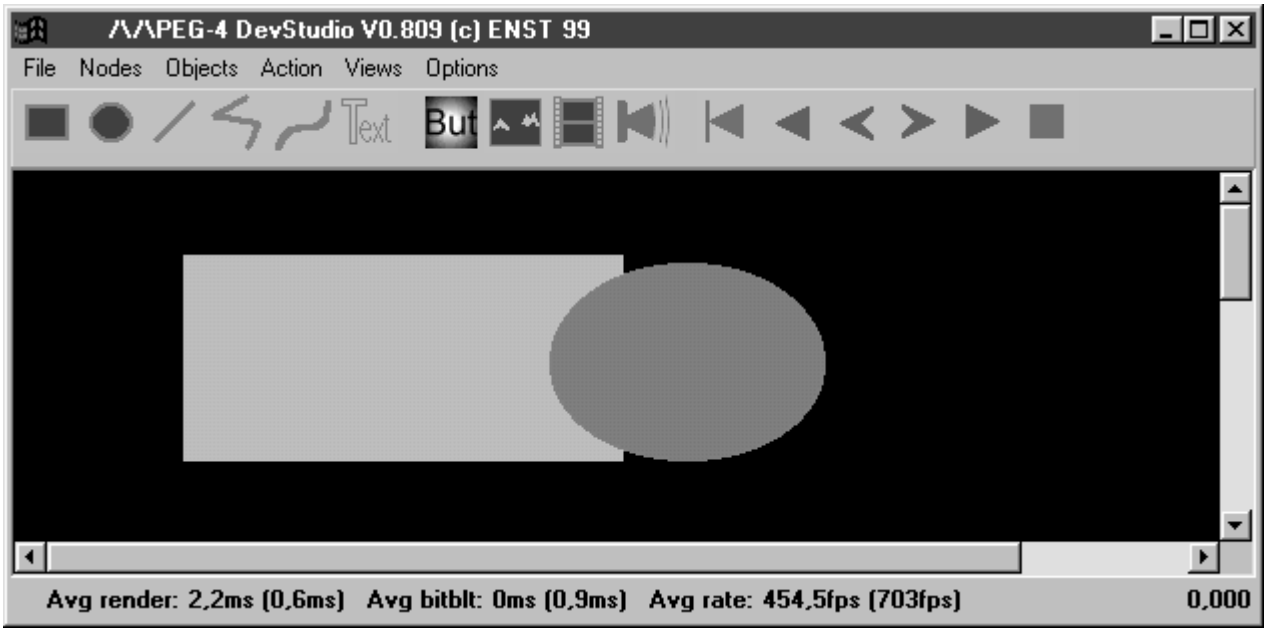


Abbildung 257: MPEG-4 Autorenwerkzeug MDS (Quelle ENST)

Die Abbildung zeigt eine sehr einfache Szene in dem Autorenwerkzeug MDS. Anstelle der beiden Objekte (Ellipse, Rechteck) können prinzipiell auch beliebige andere multimediale Objekte verwendet werden (Audio, Video, etc.). Aktuell kann das MDS beim Erstellen allerdings nur JPEGs darstellen, so dass für andere Objekte (beispielsweise MPEG-4 Video) ein Platzhalter gesetzt werden muss. Dieser Platzhalter muss allerdings ebenso wie jedes andere Objekt entsprechende Attribute zur Darstellung besitzen.

Interaktivität wird durch Sensoren erreicht. Es gibt beispielsweise den *TouchSensor*, welcher Nachrichten (*Events*) beim Überfahren, beim Anklicken, beim Verlassen usw. generiert. Weiterhin gibt es den wichtigen *TimeSensor*, welcher nach bestimmten Zeiten Nachrichten (*Events*) generiert. Diese Nachrichten können mittels eines Bedingungsknotens (*Conditional*) in Verbindung mit einem Weiterleitungsknoten (*Route*) an weitere Knotenelemente gesendet werden. Damit ist es relativ einfach Verzweigungen in Lehrinhalten zu ermöglichen, seien es Multiple-Choice-Tests oder das bestimmte Springen innerhalb einer MPEG-4 Applikation.

C.2 Die Szenenbeschreibung

Die Autorenwerkzeuge haben die Szenenbeschreibung als so genannten *BIFSText* zu exportieren. Abbildung 258 zeigt einen kleinen Auszug aus dem *BIFSText*, welcher aus der Szene in Abbildung 259 generiert wurde.

Die Verwandtschaft von *BIFSText* zu VRML ist deutlich zu erkennen. Der Unterschied ist darin zu sehen, dass VRML seine Beschreibungen in unkomprimiertem Klartext abspeichert, hingegen BIFS die Szenenbeschreibung nach festgelegten Algorithmen unter Zuhilfenahme von Werkzeugen *BIFSEncoder* binär codiert ablegt. Dieses führt zu einer Datenkompression der Szenenbeschreibung. Ergänzend soll noch darauf hingewiesen werden, dass auch bestehende VRML Anwendungen in MPEG-4 Applikationen überführt werden können, was eine Reduzierung der Größe der ursprünglichen VRML Szenenbeschreibung zur Folge hat.

```
Group {
   children [
     Transform2D {
       Translation -352.0 288.0
       Children [
          DEF I0 Group { # Root
            Children [
               DEF I5 Transform2D {
                 Translation 193.0 -92.0
                 Children [
                    DEF I4 Shape {
                      geometry
                      DEF I3 Rectangle {
                        size 219.0 102.0
                      }
                      appearance
                      DEF I1 Appearance {
...
```

Abbildung 258: Szenenbeschreibung BIFS als BIFSText

C.3 Generierung einer MPEG-4-Applikation als .mp4-Datei

Die Szenenbeschreibung und die Objektdeskriptorenbeschreibung werden als Eingangsdateien für den *Multiplexer* verwendet. Dieser Multiplexer versieht jeden Elementardatenstrom mit den nötigen *Inter-Objekt-Synchronisationsinformationen*, wie z. B. den *Lieferzeitstempel DTS*.

Die Abbildung zeigt schematisch diese Erstellungsprozedur einer MPEG-4 Applikation. Es ist ebenfalls dargestellt, dass der *Elementardatenstrom (Elementary Stream) ES* vor dem Multiplexen mittels *Muxer MUX* durch den medienspezifischen Encoder bereits codiert worden ist.

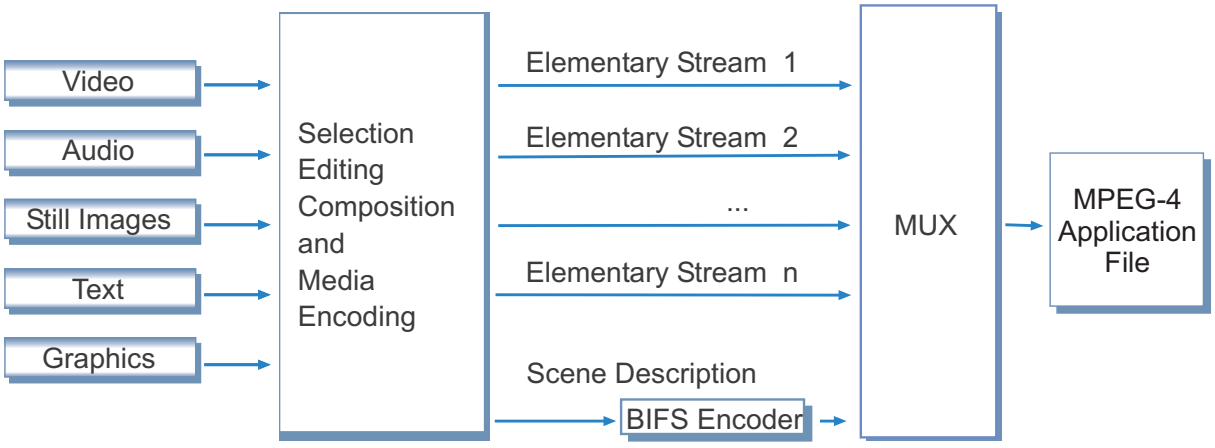


Abbildung 259: Generierung von .mp4-Dateien (MPEG-4 Applikation)

Das MPEG-4 Dateiformat besteht zum einen aus flexibel gemultiplexten Objekten (*Interleaving*, *FlexMux*), als auch den Hinweisspuren (*Hint Tracks*). Diese enthalten Informationen zu Synchronisationspunkten (*Random Access Points*), so dass ein wahlfreier Zugriff in eine MPEG-4 Applikati-

on an beliebiger Stelle ermöglicht wird. Dies wird beispielsweise in der Fernlehre insbesondere dann benötigt, wenn ein Studierender eine Übung bis zu einem bestimmten Punkt bearbeitet hat, aber die Übung zu einem späteren Zeitpunkt an dieser Stelle fortführen möchte.

C.4 Der Autoring-Prozess

Derzeit bedarf es für die Erstellung von MPEG-4 Inhalten der folgenden Schritte:

Erstellung einer MPEG-4 Applikation

Unter dem Begriff *MPEG-4 Applikation* wird die komplette und ablauffähige MPEG-4 Szene mit allen Dokumenten und Teildokumenten verstanden. In der Praxis ist meistens eine Applikation in *einer* Datei (MP4 File) abgelegt.

Somit muss zu Beginn des Erstellungsprozesses zu einer solchen MPEG-4 Applikation entweder eine bereits vorliegende Applikation zwecks Modifikation geöffnet werden oder es muss eine neue Applikation generiert werden. Eine neue MPEG-4 Applikation ist im MDS-Tool zunächst leer; nur das hierarchische Wurzelknotenelement (*Root Node*) ist enthalten.

Einführung von Objekten

Um die Applikation mit Leben zu füllen, werden üblicherweise mehrere verschiedene Szenen entworfen. Diese Szenen werden als Teilbäume in der hierarchischen Szenenbeschreibung *BIFSText* abgelegt. Die benötigten nativen multimedialen Objekte müssen getrennt zur Verfügung stehen. Bewegte Texte und Animationen können direkt in BIFS beschrieben werden. Soll ein Video verwendet werden, stehen üblicherweise Codecs, wie H.263, MPEG-1, MPEG-2 oder MPEG-4 Video, zur Verfügung.

Editieren

Als nächstes erfolgt die zeitliche und räumliche Komposition der Szenen. Dazu müssen die Attribute der Objekte, Knoten (*Node*), Wege (*Route*), Ereignisabhängigkeiten (*Events*) und das Verhalten (*Behaviour*) gesetzt werden. Interaktivitäten müssen sowohl als Element (*TouchSensor*) als auch als Attribut gesetzt werden.

Encoder/Decoder Parameter

Damit der Player beim Abspielen der MPEG-4 Applikation Informationen zu den Elementardatenströmen erhält und implizit den objektspezifischen Decoder instanzieren kann, müssen Szenenbeschreibung und Objektdeskriptoren die Attribute zu den verwendeten Elementardatenströmen enthalten. Diese Attribute müssen vor dem Multiplexen gesetzt worden sein.

Produktion des MPEG-4 Files (MP4 Datei)

Die in *BIFSText* abgelegte Szenenbeschreibung wird in die binäre Form mittels des *BIFSEncoders* überführt. Der Stream Multiplexer fügt jedem Elementardatenstrom Synchronisationsinformationen hinzu. Diese erfolgen zum einen objektspezifisch (Framerate, u. a.) und zum anderen gemäß den Vorgaben des Autors, zum Beispiel bei einem Szenenwechsel. Abschließend erfolgt der Aufruf des Konverters *MP4Tool* zur Erzeugung des endgültigen MPEG-4 Dateiformats.

Es sind nun neben den nativen multimedialen Objekten noch die Dateien für BIFSText, Objektdeskriptoren, binäres BIFS und die MPEG-4 Applikation im MPEG-4 Dateiformat entstanden.

MPEG-4 Player

Die MPEG-4 Applikation ist fertig gestellt und kann auf einem MPEG-4 Player dargestellt werden. Die in dieser Arbeit verwendeten MPEG-4 Player sind

- die Referenzsoftware *IM-1-2D Player* [MPEG4-5]
- der *Osmo4* der *Ècole Nationale Supérieure des Télécommuncations (ENST)* und
- der *SplayD2* von *bSoft Ltd*., Maccerata, Italien.

C.5 Verwendete Programme

Programm	URL	Verwendungszweck
mds	n/a	Autorenwerkzeug
Mpeg4ip	http://mpeg4ip.sf.net	mp4 muxing
bSoft Tools	http://www.bsoft.info	mp4 muxing
mp4edit.exe	http://www.comelec.enst.fr/~dufourd/mpeg-4/tools.html	BIFS Nachbearbeitung
faac	http://faac.sourceforge.net/	aac encoding
BeSweet/GUI	http://dspguru.doom9.net/	mp3 encoding
Lame	http://mitiok.cjb.net/	mp3 encoding
XviD	http://www.xvid.org/ (http://roeder.goe.net/~koepi/xvid.shtml)	mp4 Video Codec
DivX	http://www.divx.com/	mp4 Video Codec
VirtualDubMOD	http://virtualdubmod.sourceforge.net/	Video transcoding
DVD Decrypter	http://www.doom9.net/	DVD ripping
QuickTime6Pro	http://www.apple.com/	MOV -> AVI

Tabelle 53: Überblick über die verwendeten Programme

C.5.1 Video/Audio Quellen zum Codieren vorbereiten

Die einzelnen Datenströme werden zuerst aus dem quellspezifischen Format in reine Rohdaten (Raw-Video, Wave-Audio) decodiert. Danach erfolgt die Weiterverarbeitung zu MPEG-4 Video und MPEG-4 Audio.

Die einzelnen Schritte werden nun detailliert erläutert.

Beispiel: Quicktime Movie: Matrix Reloaded Trailer (trailer_final_1000_dl.mov)

1. Die *mov* Datei mit *Quicktime Pro 6* öffnen.
2. Das Video exportieren, *AVI* als Format auswählen. Unter Optionen den Sound deselektieren und bei den Video Einstellungen keine Kompression einstellen.
3. Dieselbe Vorgehensweise erfolgt für den Sound. Hier ist WAV als Exportformat auszuwählen. Dann sind die Einstellungen unter Optionen hinsichtlich Stereo zu wählen.

Audio und Video Datenstrom liegen nun im Rohformat zur Weiterverarbeitung vor (Audio: Wave, Video: Raw-Video RGB).

C.5.1.1 Videomaterial codieren

Die Rohdaten (Video) sollen nun in MPEG-4 Videodatenströme überführt werden.

Dazu wird das Programm *VirtualDupMOD* gestartet.

Videodatenstrom öffnen

- Kompression einstellen und *XviD/DivX* oder alternativen *MPEG-4 Videocodec* auswählen und konfigurieren.
  Beispiel Videocodec XviD:
  - In der XviD Konfiguration den Encoder Mode: „2 Pass – 1 Pass“ auswählen und unter „Advanced options...“ anpassen. [http://roeder.goe.net/~koepi/xvid.shtml]. Aus Kompatibilitätsgründen wird auf *B-Frames* (-1), *Enable Lumi masking*, *Enable Greyscale*, *Enable interlacing*, *Use croma motion*, *Quarterpel* und *Global Motion Compensation* verzichtet.
  - Unter *Motion search precision* wird *6 Ultra High* ausgewählt. Der verwendete Quantization Type ist *H.263* und der *VHQ mode* bleibt ausgeschaltet. Der Job wird in die Jobcontrolliste mit „F7“ eingestellt und „Don’t run this job now; add it to job control so I can run it in batch mode“ wird eingeschaltet.
  - Gleiche Einstellungen für den zweiten Durchlauf (2 Pass –2 Pass Int.) wählen und ebenfalls in die Jobcontrolliste einfügen.

Über „F4“ wird *Job control* aufgerufen und die anstehenden Jobs abgearbeitet.

Ergebnis: MPEG-4 Video im AVI Container

C.6 Audiomaterial nach AAC/MP3 encoden

Wahlweise kann der Ton als *MPEG-1 Layer 3 (MP3)* oder als *MPEG-4 Audio (AAC)* codiert werden. Entscheidungskriterium ist hier die Verfügbarkeit der Decoder im Player.

C.6.1 Advanced Audio Codec (AAC)

C.6.1.1 Erstellen des AAC Audio Datenstroms für die mpeg4ip Werkzeuge

Zum Erstellen des *AAC* Datenstroms wird *faac.exe* (Quelle http://faac.sourceforge.net/) verwendet. Nachdem in das Verzeichnis mit der ausführbaren Datei *faac* und den umzurechnenden Audio-Datenströmen gewechselt worden ist, wird folgendes eingegeben:

faac.exe -m 4 -o LC -q 100 audio.wav audio.aac

C.6.1.2 Erstellen des AAC Audiodatenstroms mit den Werkzeugen von bSoft

Zum Erstellen des AAC Datenstroms und der „*.info“-Datei wird *Am4EncInfo.exe* (beruht auf *faac*) verwendet. In der Info-Datei stehen der Zeitstempel *Delivery Timestamp* (*DTS*) und die zugehörige Anzahl Bytes für die Größe des Frames. Das Argument *-r* bewirkt, dass ein Raw AAC Audio Datenstrom erzeugt wird.

Am4EncInfo.exe -m 4 -o LC -r audio.wav audio.am4

Es sind nun die Dateien *audio.am4* mit Raw AAC Audio und *audio.am4.info* mit dem Tupel <DTS; Bytes> entstanden.

C.6.2 Audio MPEG-1 Layer 3 (MP3)

Wenn die MPEG-4 Applikation MP3-codierte anstelle AAC-codierte Audiodatenströme verwenden soll, kann der Datenstrom mit *BeSweetGUI* und *Lame* erstellt werden.

Dazu sind in *BeSweetGUI* die Pfade für *BeSweet/Lame*, *Input* und *Output* anzupassen sowie das entsprechende *MP3 Profile* auszuwählen. Der Codiervorgang wird mit dem Menüpunkt *wave to mp3* gestartet.

Danach muss der Name der mp3-Datei in *audio.am2* umbenannt werden und es wird mit dem bSoft-Werkzeug *Am2DecInfo.exe* die dazugehörige „*.info“ Datei erzeugt.

Am2DecInfo.exe audio.am2

Es sind nun die Dateien *audio.am2* mit MP3 Audio und *audio.am2.info* mit dem Tupel <DTS; Bytes> entstanden.

C.6.3 Modifikation der Deskriptoren

Die Datei *Am2Vm4Desc.txt* enthält die Objektdeskriptoren. Tabelle 54 stellt die benötigten Parameter gegenüber.

MP3	AAC

Media{
MediaIdentifier 300
UrlFlag 0
Reserved 0
Stream [
Stream{
StreamIdentifier 301
	Media{
	MediaIdentifier 300
	UrlFlag 0
	Reserved 0
	Stream [
	Stream {
	StreamIdentifier 301

Decoder Decoder {
MediaType 107 # 0x6B -- mp3
StreamType 5 # Audio
BufferSize 1050
DecoderInfo DecoderInfoAudio
{
SamplesPerFrame 1152
SamplesPerSecond 22050
#Frequenz
BytesPerSample 2
Channels 2
}
}
	Decoder Decoder {
	MediaType 64 # Mp4 Audio 0x40
	StreamType 5 # Am4 Stream
	BufferSize 400
	DecoderInfo DecoderInfoAudioMpeg
	{
	AudioObjectType 2
	SamplingFrequencyIndex 4
	ChannelConfig 2
	}
	}

SyncLayer SyncLayer {
FlagUseAccessUnitStart TRUE
FlagUseAccessUnitEnd TRUE
FlagUseTimeStamps TRUE
TimeStampResolution 1000
TimeStampLength 16
}
	SyncLayer SyncLayer {
	FlagUseAccessUnitStart TRUE
	FlagUseAccessUnitEnd TRUE
	FlagUseTimeStamps TRUE
	TimeStampResolution 1000
	TimeStampLength 28
	}

Mux Mux {
FileData "audio.am2"
FileInfo "audio.am2.info"
StreamFormat "Am2"
}
	Mux Mux {
	FileData "audio.am4"
	FileInfo "audio.am4.info"
	StreamFormat "Am4"
	}

}] }	}] }

Tabelle 54: MP3, AAC Objektdeskriptoren

Für die Sample-Frequenz der AAC-Audio-Datenströme stehen die Frequenzen nach Tabelle 55 zur Verfügung.

Index	Frequenz (Hz)	Index	Frequenz (Hz)
0	96000	5	32000
1	88200	6	24000
2	64000	7	22050
3	48000	8	16000
4	44100	9	12000

Tabelle 55: AAC Frequenz-Indizes

C.7 Erstellen der MPEG-4 Applikation aus den einzelnen Datenströmen

An dieser Stelle wird das Erstellen der eigentlichen MPEG-4 Dateien erklärt.

Mit den *mpeg4ip* Tools lässt sich eine MPEG-4 Datei erzeugen.

Es wird das Programm *mp4creator60.exe* aus den *mpeg4ip* Werkzeugen (http://mpeg4ip.sf.net) benötigt. Wenn Video-Datenstrom und Audio-Datenstrom in dasselbe Verzeichnis abgelegt worden sind, wird das gewünschte mp4 File mit folgenden Befehlen erzeugt.

mp4creator60 -c=video.avi -rate=24 Matrix-Reloaded.mp4

mp4creator60 -c=audio.aac -I -rate=24 -optimize Matrix-Reloaded.mp4

Es folgt eine Tabelle mit Parametern für das *mp4creator* Werkzeug:

-create=<input-file>	Create track from <input-file>input files can be of type: .aac .mp3 .divx .mp4v .m4v .cmp .xvid
-extract=<track-id>	Extract a track
-delete=<track-id>	Delete a track
-hint[=<track-id>]	Create hint track
-interleave	Use interleaved audio payload format
-list	List tracks in mp4 file
-mpeg4-video-profile=<level>	Mpeg4 video profile override
-mtu=<size>	MTU for hint track
-optimize	Optimize mp4 file layout
-payload=<payload>	Rtp payload type (use 3119 or mpa-robust for mp3 RFC 3119 support
-rate=<fps>	Video frame rate, e.g. 30 or 29.97
-timescale=<ticks>	Time scale (ticks per second)

-use64bits	Use for large files
-verbose[=[1-5]]	Enable debug messages
-version	Display version information

Tabelle 56: Parameter *mp4creator*

Beim Erstellen der mp4 Dateien erzeugt *mp4creator60.exe* automatisch die entsprechenden BIFS/OD Einträge in der mp4 Datei. Hier ein Beispiel:

```
OrderedGroup {
   children [
      Sound2D {
         spatialize false
         source AudioSource {
            url [ "10" ]
         }
      }
      Shape {
         appearance Appearance {
            texture MovieTexture {
               url [ "20" ]
            }
         }
         geometry Bitmap {
            scale 1.0 1.0
         }
      }
   ]
}
InitialObjectDescriptor {
   objectDescriptorID 1
   ODProfileLevelIndication 255
   sceneProfileLevelIndication 255
   audioProfileLevelIndication 15
   visualProfileLevelIndication 1
   graphicsProfileLevelIndication 255
   includeInlineProfileLevelFlag false
   esdescr [
      ES_Descriptor {
         es_id 2
         streamPriority 0
         decConfigDescr DecoderConfigDescriptor {
         objectTypeIndication 2
         streamType 3
         upStream false
         bufferSizeDB 24
         maxBitrate 192
         avgBitrate 1
         decSpecificInfo BIFSv2Config {
            use3DMeshCoding false
            usePredictiveMFField false
            isCommandStream true
            pixelMetrics true
            pixelWidth 1000           # mit MP4edit.exe eingefügt
            pixelHeight 540           # mit MP4edit.exe eingefügt
            nodeIDBits 0
            routeIDBits 0
            protoIDBits 0
         }
```

```
      }
      slConfigDescr SLConfigDescriptor {
      }
    }
    ES_Descriptor {
      es_id 1
      streamPriority 0
      decConfigDescr DecoderConfigDescriptor {
        objectTypeIndication 1
        streamType 1
        upStream false
        bufferSizeDB 140
        maxBitrate 264
        avgBitrate 1
      }
      slConfigDescr SLConfigDescriptor {
        }
      }
    ]
}
AT 0 {
   UPDATE OD [
     ObjectDescriptor {
       objectDescriptorID 10
       esdescr [
         ES_Descriptor {
           es_id 3
           streamPriority 0
           decConfigDescr DecoderConfigDescriptor {
             objectTypeIndication 64
             streamType 5
             upStream false
             bufferSizeDB 472
             maxBitrate 149552
             avgBitrate 129287
             decSpecificInfo DecoderSpecificInfoString {
               info "ABA@"
             }
           }
           slConfigDescr SLConfigDescriptor {
           }
         }
       ]
     }
   ] # end UPDATE OD
   UPDATE OD [
     ObjectDescriptor {
       objectDescriptorID 20
       esdescr [
         ES_Descriptor {
           es_id 4
           streamPriority 0
           decConfigDescr DecoderConfigDescriptor {
             objectTypeIndication 32
             streamType 4
             upStream false
             bufferSizeDB 109008
             maxBitrate 8940376
             avgBitrate 5143833
             decSpecificInfo DecoderSpecificInfoString {
               info
"@@@@@A;@@C@@@@@A;E@K@@@@@A@@@@@@@AB@@@@F8G?O?O@J:J8O:BBA<JCA@@@@@A;BEHGFFIDDC@C
@C@CIGO@@"
```

```
                }
              }
              slConfigDescr SLConfigDescriptor {
              }
            }
          ]
        }
    ] # end UPDATE OD
}
```

Mit Hilfe von *MP4edit.exe* können nach Bedarf Änderungen in BIFS und OD vorgenommen werden. Beispielsweise können die Zeilen `pixelWidth 1000` und `pixelHeight 540`, wie oben dargestellt, nachträglich eingebaut werden. Aus diesen Werten kann der Player entnehmen, mit welchen Abmessungen die Szene einer MPEG-4 Applikation dargestellt werden soll.

C.8 bSoft Tools

Das Erstellen einer MPEG-4 Applikation unter Verwendung der *bSoft* Tools ist etwas aufwendiger, erlaubt aber auch einen größeren Einfluss auf die Gestaltung der Szenenbeschreibung und es gibt neben Video- und Audiodatenströmen auch die Möglichkeit Bilder mit in die mp4 Datei zu multiplexen und die Szene interaktiv zu gestalten. In gewissen Grenzen geht das auch mit einer von den *mpeg4ip Tools* erstellten mp4 Datei, welche dann mit *Mp4edit* nachbearbeitet werden kann.

C.8.1 Übersicht der Werkzeuge

Am4EncInfo.exe: Erzeugt einen Raw AAC Datenstrom und die zugehörige Info-Datei. Letztere wird beim multiplexen in die mp4 Datei benötigt.

Am2DecInfo.exe: Erzeugt eine Info-Datei aus einer mp3 Audio-Datei. Die Info-Datei wird beim Multiplexen der mp3 Datei in die mp4 Datei benötigt.

Sm4Enc.exe: Erstellt aus drei Dateien die Informationen zum Aufbau der Szene (Am4Vm4Node.txt), Beschreibung der einzelnen Datenströme (Am4Vm4Desc.txt) und die in der Anfangsbeschreibung (Am4Vm4Init.txt) enthalten BIFS und OD.

Mp4Mux.exe: Multiplext nach Angaben des *InitialOD (Am4Vm4Init.txt)* und der Szenenbeschreibung (*Am4Vm4Desc.txt*) aus den vier Datenströmen: *audio.am4 (raw-aac)*, *video.vm4 (raw-mpeg4-Video)*, *OD (Am4Vm4.init)* und *BIFS (Am4Vm4.node)* die mp4 Datei (MPEG-4 Applikation).

avi2raw60.exe: Wird zum extrahieren des *raw Video* Datenstroms aus dem *avi File* benötigt.

Vx0DecInfo.exe: Erzeugt eine Info-Datei von einem *raw Video* Datenstrom, die beim multiplexen in die mp4 Datei benötigt wird.

C.8.2 Vorgehensweise

C.8.2.1 Video

RAW Video Datenströme aus dem AVI exportieren.

avi2raw60.exe video.avi ***video.vm4***

Info-Datei zum Video-Datenstrom erstellen.

Vx0DecInfo.exe 0 24 0 video.vm4

(Vx0DecInfo.exe <Offset> <Rate> < Pictures> <File Name>)

Offset: milliseconds added to DTS/CTS

Rate: if Rate == 0 then use the timestamps

Pictures: if Pictures == 0 then decode the whole sequence

Folgende Dateien werden erzeugt:

video.vm4.info (Info-Datei zum Videodatenstrom)

video.desc.info (Informationen zur Initialisierung – Entspricht der ersten Zeile aus *video.vm4.info*)

C.8.2.2 Audio

Info-Datei zum Audiodatenstrom erstellen:

BIFS und OD erstellen aus den Dateien *Am4Vm4Init.txt, Am4Vm4Node.txt*, *Am4Vm4Desc.txt:*

Sm4Enc.exe Am4Vm4 Am4Vm4Init.txt Am4Vm4Node.txt

Am4Vm4Desc.txt > EncAm4Vm4.trace

C.8.3 InitialOD (Am4Vm4Init.txt)

Aufbau der Dateien für eine mp4 Datei mit AAC Audio- und MPEG-4 Videodatenstrom:

```
#########################################################
# Desc

MediaInitial
{
  MediaIdentifier  100
  MediaProfile     1
  SceneProfile     1
  AudioProfile     2
  VideoProfile     1
  GraphicsProfile  1
  Stream
  [
    Stream
    {
      StreamIdentifier   101
      Decoder Decoder
      {
        MediaType   1    # Mpeg4 System
        StreamType  3    # Node Stream
        BufferSize  16000
        DecoderInfo DecoderInfoSystem
        {
          NodeIdBits  16
          RouteIdBits 16
          CommandFlag 1
          PixelMetric 1
          SizeFlag    1
          SizeHor     1000
          SizeVer     540
        }
      }
      SyncLayer SyncLayer
      {
        FlagUseAccessUnitStart  TRUE
        FlagUseAccessUnitEnd    TRUE
        FlagUseTimeStamps       TRUE
        TimeStampResolution     1000
        TimeStampLength         16
      }
      Mux Mux   {
        FileData "Am4Vm4.node"
        FileInfo "Am4Vm4.node.info"
```

```
          StreamFormat "Node"
        }
      }
      Stream {
        StreamIdentifier   102
        Decoder Decoder {
          MediaType   1    # Mpeg4 System
          StreamType  1    # Desc Stream
          BufferSize  16000
        }
        SyncLayer SyncLayer {
          FlagUseAccessUnitStart  TRUE
          FlagUseAccessUnitEnd    TRUE
          FlagUseTimeStamps       TRUE
          TimeStampResolution     1000
          TimeStampLength         16
        }
        Mux Mux {
          FileData "Am1Vm4.desc"
          FileInfo "Am1Vm4.desc.info"
          StreamFormat "Desc"
        }
      }
   ]
}
```

C.8.4 Szenenbeschreibung (Am4Vm4Node.txt)

```
 ########################################
# Node

ReplaceScene
{
   TimeComposition  500
Scene Group
{
   children [
     Transform2D {
       # scale1.0  1.0
       children [
         #### VIDEO + AUDIO ########################
         DEF Sha001 Shape {
           appearance Appearance {
             texture MovieTexture {
               url [ 200 ]
             }
           }
           geometry Bitmap {
           }
         }
         Sound2D {
           source AudioSource {
             NumChan     2
             startTime   0
             stopTime    -1
             url [ 300 ]
           }
         }
       ]
     }
   ]
}
}
```

C.8.5 Objektdeskriptoren (Am4Vm4Desc.txt)

```
##########################################
# Desc

UpdateMedia {
   TimeComposition  0
   Media [

     Media {   ################################################
        MediaIdentifier  200
        UrlFlag0
        Reserved  0
        Stream
        [
           Stream
           {
              StreamIdentifier 201
              Decoder Decoder
              {
                 MediaType 32 # 0x20 -- Mpeg4
                 StreamType 4 # Video -- Mpeg4
                 BufferSize 40000
                 DecoderInfo DecoderInfoVideoMpeg
                 {
                    FileData  "video.vm4"
                    FileInfo  "video.desc.info"
                 }
              }
              SyncLayer SyncLayer
              {
                 FlagUseAccessUnitStart  TRUE
                 FlagUseAccessUnitEnd TRUE
                 FlagUseTimeStamps  TRUE
                 TimeStampResolution  1000
                 TimeStampLength  16
              }
              Mux Mux
              {
                 FileData  "video.vm4"
                 FileInfo  "video.vm4.info"
                 StreamFormat  "Vm4"
              }
           }
        ]
     }
     Media  ################################################
     {
        MediaIdentifier  300
        UrlFlag0
        Reserved  0
        Stream
        [
           Stream
           {
              StreamIdentifier 301
              Decoder Decoder
              {
                 MediaType 64  # Mpeg4 Audio 0x40
                 StreamType 5  # Audio Am4 Stream
                 BufferSize 400
                 DecoderInfo DecoderInfoAudioMpeg
                 {
```

```
                AudioObjectType  2
                SamplingFrequencyIndex  4
                ChannelConfig 2
              }
            }
            SyncLayer SyncLayer
            {
              FlagUseAccessUnitStart  TRUE
              FlagUseAccessUnitEnd TRUE
              FlagUseTimeStamps  TRUE
              TimeStampResolution  1000
              TimeStampLength  28
            }
            Mux Mux
            {
              FileData "audio.am4"
              FileInfo "audio.am4.info"
              StreamFormat "Am4"
            }
          }
        ]
      }
   ]
}
```

C.8.6 Multiplexen und Erstellen der mp4 Datei

Mp4Mux.exe multiplext nach Angaben von *Am4Vm4Init.txt* und *Am4Vm4Desc.txt* die mp4 Datei aus den einzelnen Datenströmen zusammen.

Mp4Mux.exe Matrix.mp4 Am4Vm4Init.txt Am4Vm4Desc.txt

Die vorangegangenen Schritte lassen sich in einer Stapeldatei automatisieren.

D Realisierung eines Schalters in einer MPEG-4 Applikation

MPEG-4 stellt Elemente zur Verfügung, die eine Interaktivität des Benutzers erlauben. Dieses Kapitel führt in die Elemente und den Erstellungsprozess für interaktive Szenen einer MPEG-4 Applikation ein.

Mit einem Schalter können in einer Szene einer MPEG-4 Applikation verschiedene Zustände herbeigeführt werden. Diese beiden Abbildungen illustrieren beispielhaft das Aussehen eines Schalters, wie er im Folgenden dann implementiert werden soll.

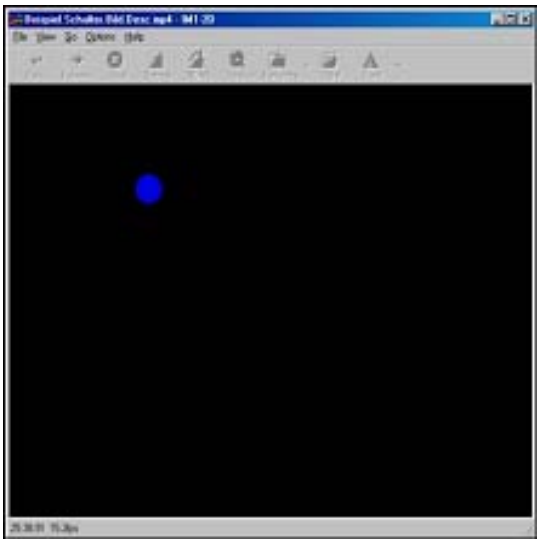


Abbildung 260: Schalter Zustand „blau“

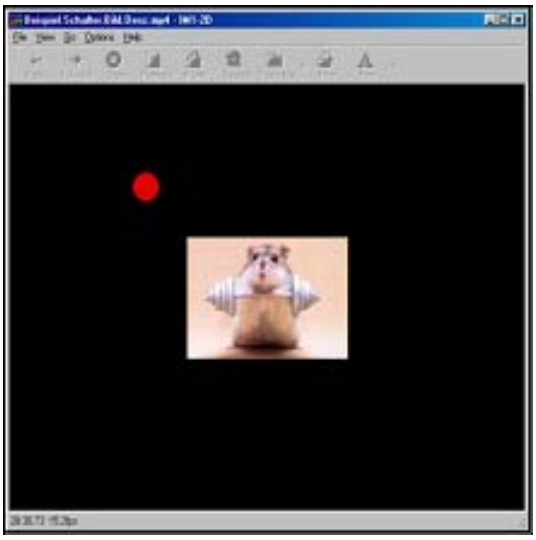


Abbildung 261: Schalter Zustand „rot“

Abbildung 260 zeigt die Szene mit dem Schalter im Zustand *blau*. Es ist kein weiteres Objekt sichtbar. Durch Drücken des Schalters wird ein Zustandsübergang initiiert, der den Übergang zu dem Zustand, wie er in Abbildung 261 dargestellt ist, veranlasst. Dieser Zustand stellt den Schalter im Zustand *rot* dar, welcher ein zusätzliches Objekt, ein Stillbild im JPEG-Format, beinhaltet. Der Vorgang ist reversibel angelegt. Durch Drücken des Schalters *rot* soll der Zustand *blau* wieder eingenommen werden.

D.1 Grundlegende Elemente in MPEG-4

Zur Realisierung eines Schalters in MPEG-4 werden die Elemente *TouchSensor Nodes*, *Route, Conditional Nodes* und *Switch Nodes* benötigt.

Die nun folgende Aufzählung erfolgt unter dem zeitlichen Aspekt der beteiligten Instanzen.

TouchSensor Node

In Verbindung mit einem Objekt (beispielsweise der blaue Kreis) erlangt dieses Objekt durch Verknüpfung mit dem *TouchSensor Node* in eine berührungssensitive Fläche beziehungsweise allgemein ein berührungssensitives Objekt. Wird dieses Objekt „berührt“, so generiert der *TouchSensor Node* Ereignisse. Diese Ereignisse sind unter anderem *isOver* und *isActive*. Weitere Ereignisse sind detailliert in [VRML97] und [MPEG4-1] beschrieben.

Route

Der *Route Node* ist am Ende des BIFSTexts zu platzieren. In Verbindung mit einem *(Touch)Sensor* bestimmt der *Route Node* das interaktive Verhalten einer Szene.

Conditional Node

Der *Conditional Node* sorgt bei Aufruf dafür, dass die parametrisierten Befehle ausgeführt werden.

Switch Node

Die *Switch Nodes* stellen Zustände einer Szene einer MPEG-4 Applikation zur Verfügung.

D.2 Definition der Elemente

Es wird das Aussehen, die Anordnung und die Elemente definiert. Der zugehörige BIFSText ist im Kapitel *D.4 BIFSText* abgedruckt.

Layout

Für das Layout werden folgende Elemente benötigt:

blauer Kreis, roter Kreis, TouchSensor, Stillbild JPEG

Statische Zustände

Mit Hilfe des *Switch S1* (*Zeile 03*) werden die beiden Schalterzustände realisiert.

Die erste Gruppe *Group S1C1* (*Switch1Choice1*)(*Zeile 07,* siehe BIFSText in D.4 BIFSText) des *Switch* wird als Anfangsauswahl mit *whichChoice 0* (*Zeile 04*) definiert.

Gruppierung der Elemente zu den Zuständen

Blauer Zustand

Zuerst ist nur ein blauer Kreis (Abbildung 260) mit dem Radius 16 (*Zeile 13-17*) zu sehen. Dessen weiteres Aussehen unter *appearance* (*Zeile 18-26*) festgelegt wird. Das Ganze wird mit dem *Children node (Zeile 12)* zusammengefasst. Diese Gruppe wird mit *DEF Ta1 TouchSensor{}* (*Zeile 09*) als *TouchSensor Ta1* definiert. Dieser so konstruierte blaue Schalter wird als *Gruppe S1C1* (*Zeile 7*) definiert. Durch *whichChoice 0 (Zeile 4)* wird beim ersten Aufruf der Szene die erste Auswahl des *Switches S1* angezeigt.

Roter Zustand

Der rote Kreis (Abbildung 261) erscheint, wenn auf den blauen Kreis gedrückt wurde. Dieser hat ebenfalls einen Radius von 16 (*Zeile 39-43*) und ist exakt an der gleichen Position wie der blaue Kreis. Es wird das Erscheinungsbild in *appearance* (*Zeile 44-51*) eingestellt. Die Position des *Shape* (der Kreis) wird durch *translation* (*Zeile 37*) bestimmt, welches ein Bestandteil der Obergruppe *Transform2D* (*Zeile 36*) ist. Das alles wird wiederum mit *children* zusammengefasst. Mit *DEF Ta2 TouchSensor {} (Zeile 35)* wird diese Gruppe als *TouchSensor Ta2* definiert. Der auf diese Art definierte rote *TouchSensor Ta2* ist Bestandteil der zweiten Auswahl des *Switches S1*. Diese Auswahl *choice* wird zur leichteren Übersicht als *S1C2* definiert.

Initialisierung

Die Initialisierung erfolgt durch:

```
Group {
   children [
      to do
   ]
}
```

Des Weiteren wird der Zustand *blau* als Anfangszustand festgelegt.

D.3 Funktionsweise

Der Anwender drückt (klickt) auf den blauen Schalter. Das Bild wird gewechselt, wie in Abbildung 261 dargestellt. Der Anwender drückt auf den roten Schalter, das Bild wechselt zum Aussehen der Abbildung 260.

TouchSensor

Wird der *TouchSensor Ta1* durch drücken aktiviert, wird das Ereignis *active* generiert.

Route

Nun führt die Verzweigung *ROUTE Ta1.isActive TO Con1.activate* (*Zeile 125*) und die Verzweigung *ROUTE Ta1.isActive TO Con3.activate* (*Zeile 133*) dazu, dass die Verzweigungspunkte *Conditional Con1* (*Zeile 98*) und *Con3* (*Zeile 108*) ausgeführt werden.

Conditional

Con1 sorgt dafür, dass der Befehl aus dessen Puffer *REPLACE S1.whichChoice BY 1 (Zeile 99)* ausgeführt wird, so das anstelle der ersten Auswahl jetzt Auswahl zwei *Conditional S1C2* angezeigt wird (Abbildung 261).

Con3 wiederum sorgt dafür, dass der Befehl aus seinem Puffer *REPLACE S2.whichChoice BY 1 (Zeile 99)* ausgeführt wird, so dass anstelle der Auswahl *S2C1* jetzt Auswahl zwei *S2C2* (*Zeile 69-90*) angezeigt wird (Abbildung 260).

In dem markierten *###Bereich###* zwischen Zeile 72 und 92 kann jedes gewünschte zu schaltende Objekt eingefügt werden. Wie hier in dem Beispiel (*Zeile 79-88*) ein Bild unter Verwendung von *geometry Bitmap -> texture ImageTexture*. Die *url 3* (Zeile 83) verweist auf das Objekt mit der ID 3. Die benötigten Informationen um das Objekt einzubinden werden unter *objectDescriptorID 3* gefunden.

Bei erneutem Drücken der Schaltfläche (für den Anwender stellt sich der rote Kreis und der blaue Kreis ja als ein Schalter mit zwei Farben dar) mittels des *TouchSensors* wiederholt sich der Vorgang, nur das die *Switch Nodes* in diesem Fall wieder ihren Anfangszustand einnehmen. Der Anfangszustand wird durch *whichChoice* für den jeweiligen *Switch* festgelegt. Die Anzahl der Gruppen (*Group*) innerhalb des *Switches* entspricht der Anzahl der unterschiedlichen Zustände des *Switches*. Zählbeginn ist bei 0. Null ist somit die erste Auswahl des *Switches*. Zur besseren Übersicht werden die einzelnen Zustände per *DEF* einem Namen zugewiesen. Was aber nicht zwingend erforderlich, ist da mit dem Aufruf von *Conditional* jeweils nur der Wert von *whichChoice* verändert wird.

D.4 BIFSText

Datei: Beispiel.Schalter.Bild.Node.txt

```
01:Group {  # Beginn einer Szene, unabhängig von den besprochenen Gruppen
02:  children [
03:     DEF S1 Switch {  # Switch S1 definieren -SCHALTER-
04:       whichChoice 0 # Anfangsauswahl festlegen hier 0 (erste)
05:                     # -> Auswahl S1C1
06:       choice [ # Gruppe blaue Schaltfläche
07:         DEF S1C1 Group { # Erste Auswahl (0) als S1C1 definieren
08:           children [
09:             DEF Ta1 TouchSensor {}     # Definiert Touchsensor Ta1
10:             Transform2D {
11:               translation 25.0 -25.0  # Positionsangabe
12:               children [
13:                 Shape {
```

```
14:                     geometry
15:                     Circle { radius 16.0 }
18:                     appearance Appearance {
20:                       material Material2D {
21:                         emissiveColor 0.77 0.39 0.0
23:                         transparency 0.0
24:                         filled TRUE
25:                       }
26:                     }
27:                   }
27                  ]
27:               }
28:             ]
29:           }
30:       ]
31:     }
32:     # Gruppe rote Schaltfläche
33:     DEF S1C2 Group { # Zweite Auswahl (1) unter S1C2 definieren
34:       children [
35:         DEF Ta2 TouchSensor {}       # Definiert Touchsensor Ta2
36:         Transform2D {
37:           translation 25.0 -25.0     # Positionsangabe
38:           children [
39:             Shape {
40:               geometry
41:               Circle { radius 16.0 }
44:               appearance Appearance {
46:                 material Material2D {
47:                   emissiveColor 0.99 0.0 0.02
48:                   transparency 0.0
49:                   filled TRUE
50:                 }
51:               }
52:             }
53:           ]
54:         }
55:       ]
56:     }
58:     # Switch S1 Ende - Schalter -
59:
60:     DEF S2 Switch {  # Switch S2 definien
61:       whichChoice 0 # Anfangsauswahl festlegen (0) erste
62:                     # -> Auswahl Switch2Choice1
63:       choice [
64:         DEF S2C1 Group {
65:           children [] } # Leere Auswahl S2C1
66:                # Erst nach dem aktivieren des Touchsensors Ta1 soll
67:                # etwas angezeigt werden
68:
69:     DEF S2C2 Group { # Zweite Auswahl unter S2C2 definieren
70:       children [
71:
72:################ Hier das zu schaltende Objekt einfügen #############
73:
74:         DEF Bild Transform2D {  # Bild welches bei aktivierten
75:                                 # Schalter angezeigt werden soll
77:           translation 184.0 -136.0
78:           children [
79:             Shape {
80:               geometry Bitmap {}
81:               appearance Appearance {
82:                 texture ImageTexture {
83:                   url 3  # siehe weiter unten Datei:
```

```
84:                              # Beispiel.Schalter.Bild.Desc.txt
85:                              # objectDescriptorID 3
86:                     }
87:                   }
88:                 }
89:               ]
90:             }
92:#####################################################################
93:         ]
94:       }
95:   ]
96:}
97:#--------------------------------
98:   DEF Con1 Conditional {
99:     buffer { REPLACE S1.whichChoice BY 1 }   # Auswahl 2 (1) im Switch
100:                                             # S1 aktivieren ->
101:                                             # Schalterzustand ändern
102: }
103: DEF Con2 Conditional {
104:    buffer { REPLACE S1.whichChoice BY 0 }   # Auswahl 1 (0) im Switch
105:                                             # S1 aktivieren ->
106:                                             # Schalterzustand ändern
107: }
108: DEF Con3 Conditional {
109:    buffer { REPLACE S2.whichChoice BY 1 }    'Auswahl 2 (1) im Switch
110:                                             # S2 aktivieren ->
111:                                             # Zeige Bild „Beispiel.jpg"
112: }
113: DEF Con4 Conditional {
114:    buffer { REPLACE S2.whichChoice BY 0 }   # Auswahl 1 (0) im Switch
115:                                             # S2 aktivieren ->
116:                                             # Bild ausblenden
117: }
120: ]
121:}
```

D.4.1 Route

```
125:ROUTE Ta1.isActive TO Con1.activate # Warten, dass der Touchsensor Ta1
126:                                    # aktiviert wurde und Conditional
127:                                    # Con1 ausführen
128:
129:ROUTE Ta2.isActive TO Con2.activate # Warten, dass der Touchsensor Ta2
130:                                    # aktiviert wurde und Conditional
131:                                    # Con2 ausführen
132:
133:ROUTE Ta1.isActive TO Con3.activate # Warten, dass der Touchsensor Ta1
134:                                    # aktiviert wurde und Conditional
135:                                    # Con3 ausführen
136:
137:ROUTE Ta2.isActive TO Con4.activate # Warten, dass der Touchsensor Ta2
                                        # aktiviert wurde und Conditional
                                        # Con4 ausführen
```

D.4.2 Update OD

```
UPDATE OD [
  {
    objectDescriptorID 3
    esdescr [
      ES_Descriptor {
        es_id  3
        muxInfo muxInfo {
          fileName "Beispiel.jpg"
```

```
          }
          decConfigDescr DecoderConfigDescriptor {
            bufferSizeDB  500000
          }
          slConfigDescr SlConfigDescriptor {
          }
        }
      ]
    }
]
```

D.5 Beispiel: InitialOD und OD

Datei: Beispiel.Schalter.Bild.Desc.txt

```
InitialObjectDescriptor
{
  objectDescriptorID 1
  ODProfileLevelIndication       1
  sceneProfileLevelIndication    1
  audioProfileLevelIndication    1
  visualProfileLevelIndication   1
  graphicsProfileLevelIndication 1
  esDescr [
    {
      ES_ID 1
      muxInfo {
        fileName Beispiel.Schalter.Bild.Node.od
        streamFormat BIFS
      }
      decConfigDescr {
        streamType 1 ## OD
        bufferSizeDB 500
      }
      slConfigDescr {
        useAccessUnitStartFlag TRUE
        useAccessUnitEndFlag TRUE
        timeStampResolution 1000
        timeStampLength 14
      }
    }
    {
      ES_ID 2
      muxInfo {
        fileName Beispiel.Schalter.Bild.Node.bif
        streamFormat BIFS
      }
      decConfigDescr {
        streamType 3 ## BIFS
        bufferSizeDB 2000
      }
      slConfigDescr {
        useAccessUnitStartFlag TRUE
        useAccessUnitEndFlag TRUE
        timeStampResolution 1000
        timeStampLength 14
      }
    }
  ]
}
#Object Descriptor
{
  objectDescriptorID 3
```

```
  esdescr [
    ES_Descriptor {
      es_id  3
      muxInfo muxInfo {
        fileName "Beispiel.jpg"
        streamFormat  JPEG
        maxAhead 10
      }
      decConfigDescr DecoderConfigDescriptor {
        objectTypeIndication 0x6C
        streamType           4
        upStream             FALSE
        bufferSizeDB         500000
        maxBitrate           0
        avgBitrate           0
      }
      slConfigDescr SlConfigDescriptor {
        predefined  0
        useAccessUnitStartFlag      TRUE
        useAccessUnitEndFlag        TRUE
        useRandomAccessPointFlag    TRUE
        hasRandomAccessUnitsOnlyFlag FALSE
        usePaddingFlag              FALSE
        useTimeStampsFlag           TRUE
        useIdleFlag                 FALSE
        durationFlag                FALSE
        timeStampResolution         1000
        OCRResolution               1000
        timeStampLength             16
        OCRLength                   0
        AU_Length                   0
        instantBitrateLength        0
        degradationPriorityLength   0
        AU_seqNumLength             8
        packetSeqNumLength          8
        startDecodingTimeStamp      0
        startCompositionTimeStamp   0
        AU_seqNumLength             0
        packetSeqNumLength          0
      }
    }
  ]
}
```

D.6 MPEG-4 Knotenelemente (Nodes)

Hier erfolgt eine Kurzübersicht der zur Realisierung eines Schalters in einer MPEG-4 Applikation notwendigen *Node*s, des *ROUTE* Befehls und des Grundgerüsts der Szenenbeschreibung.

Szenengrundgerüst

```
  Group{
    Children [
      ...
    ]
  }
```

Einfache Form (Kreis)

```
  Shape {
    geometry
      Circle { radius 16.0 }  # Kreis mit einem Radius von 16
      material Material2D {   # Material Eigenschaften
        emissiveColor 0.77 0.39 0.0
```

```
        transparency 0.0
        filled TRUE
      }
}
```

Kennung einer *Node* zu ordnen

Mit *DEF* kann einer *Node* eine eindeutige Kennung zugeordnet werden.

```
DEF Kennung1 Group{ ...}
```

Realisierung verschiedener Zustände (Switch)

```
Switch {
  whichChoice 0         #Fall auswählen
  choice [
    DEF case1 Group { children [...] }
    DEF case2 Group { children [...] }
    …
  ]
}
```

Interaktives Element (TouchSensor)

Ein *TouchSensor* wird mit folgender *Node* erstellt.

```
TouchSensor {}
```

Das Knotenelement liefert bei einem Druck auf die Schaltfläche (*Button*) das Ereignis *active*, bei Überfahren der Schaltfläche wird das Ereignis *over* generiert.

Code in einer bedingten Ausführung (Conditional)

Trifft für ein Knotenelement *Conditional* das Ereignis *active* ein, so wird der bestimmte Code ausgeführt.

```
Conditional {
  buffer {Code der bei Aktivierung ausgeführt werden soll}
}
```

Bestimmen der Bedingung (ROUTE)

```
ROUTE Ta1.isActive TO Con1.activate
```

Wenn das Knotenelement *Ta1* das Ereignis *active* meldet, wird das Knotenelement *Con1* mit dem Ereignis *active* signalisiert.

E Datenmaterial

E.1 „KillerBean 2“

Ausgangsmaterial „KillerBean2.avi“

Datei-Name	KillerBean2.avi	
Datei-Format	AVI	
	Video	**Audio**
Format	DivX 4 (Mpeg4 Video)	mp3
Mittlere Bitrate	539,9 kBit/s 65,8 kByte/s	95k
CBR/VBR	VBR	CBR
Länge	7:21	7:21
Datei-Größe	28,6 MB	5,2 MB
Bildschirmgröße	640x360	
Bildwiederholungsrate	29,97	
Besonderes	-	-

MP4-Datei „KillerBean2.mp4“

\Test\007Richard\001a_KillerBean_Sample_ AAC(2003)

Datei-Name	KillerBean2.mp4	
Datei-Format	mp4 File	
	Video	**Audio**
Format	DivX (Mpeg4 Video)	aac
Mittlere Bitrate	539,9 kBit/s, 65,8 kByte/s	128 kBit/s
CBR/VBR	VBR	VBR
Länge	7:21	7:21
Datei-Größe	28,6 MB	6,39 MB
Bildschirmgröße	640x360	
Bildwiederholungsrate	29,97	
Besonderes	-	-

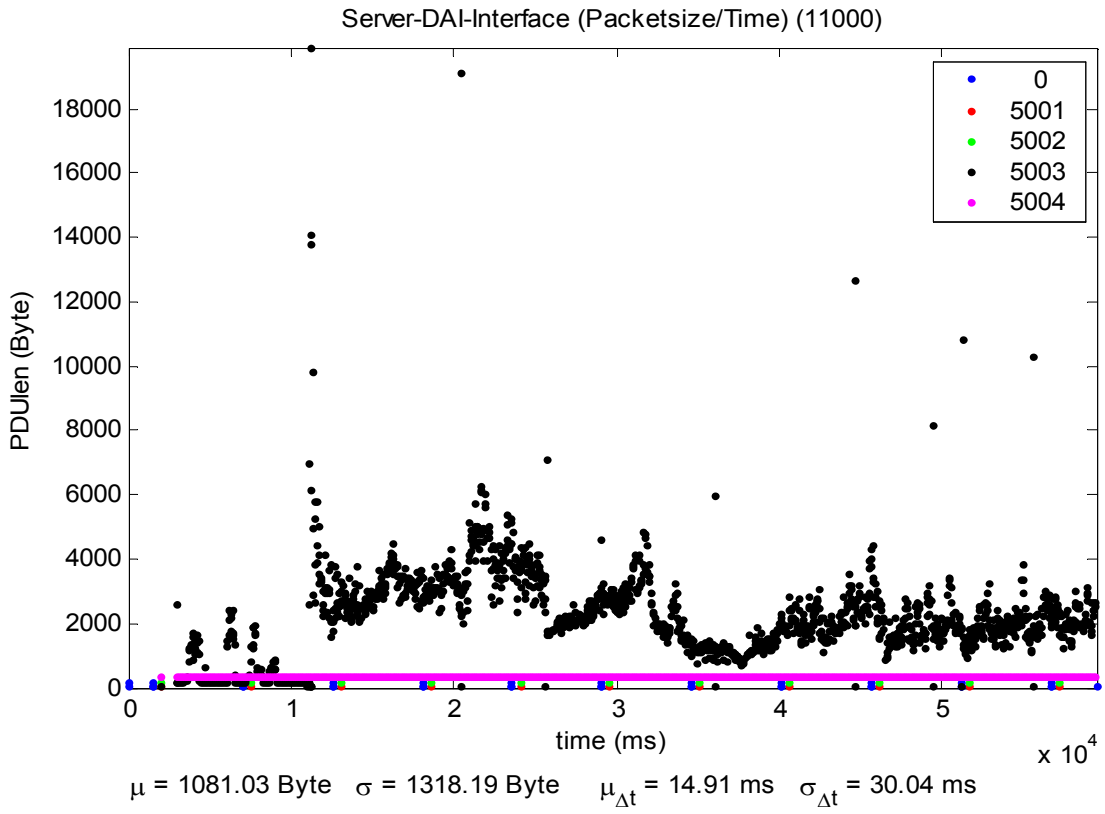


Abbildung 262: Datenströme am Server-DAI*: KillerBean2 AAC

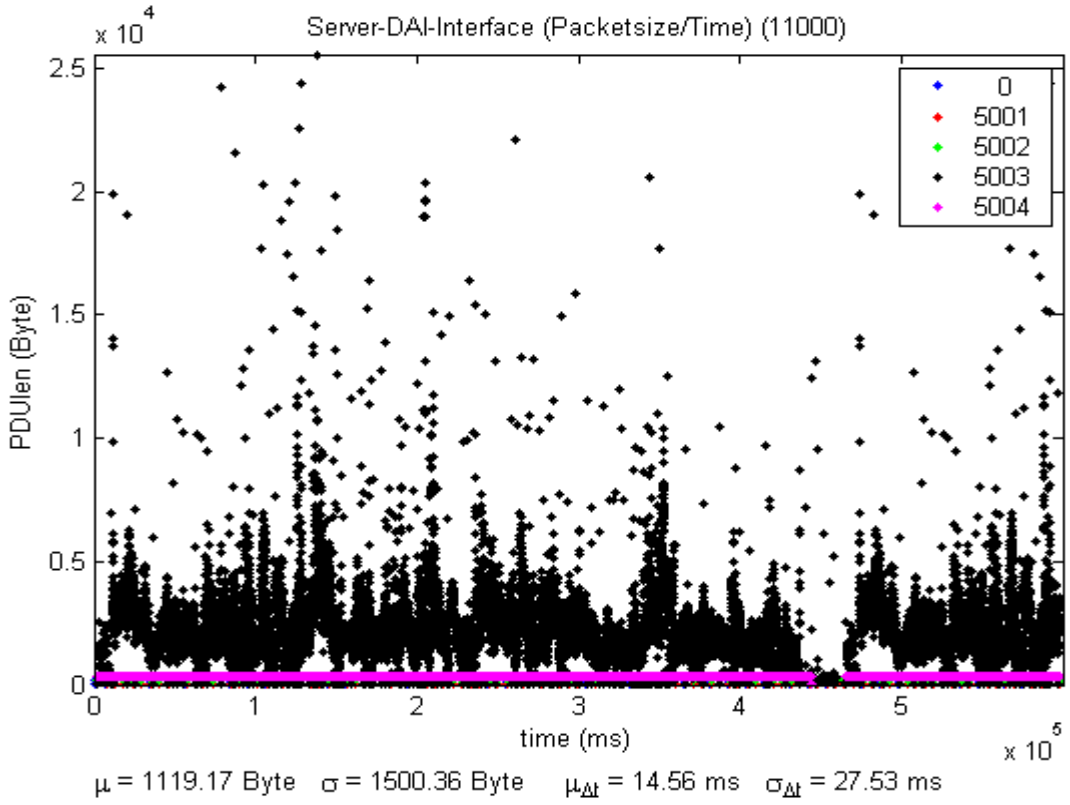


Abbildung 263: Datenströme (10 Min.) am Server-DAI*: KillerBean2 AAC

MP4-Datei „KillerBean2_mp3_.mp4“

\Test\007Richard\001b_KillerBean_Sample_mp3

Datei-Name	KillerBean2_mp3_.mp4	
Datei-Format	mp4 File	
	Video	**Audio**
Format	DivX (Mpeg4 Video)	mp3
Mittlere Bitrate	539,9 kBit/s, 65,8 kByte/s	149 kBit/s
CBR/VBR	VBR	VBR
Länge	7:21	7:21
Datei-Größe	28,6 MB	7,85 MB
Bildschirmgröße	640x360	
Bildwiederholungsrate	29,97	
Besonderes	-	-

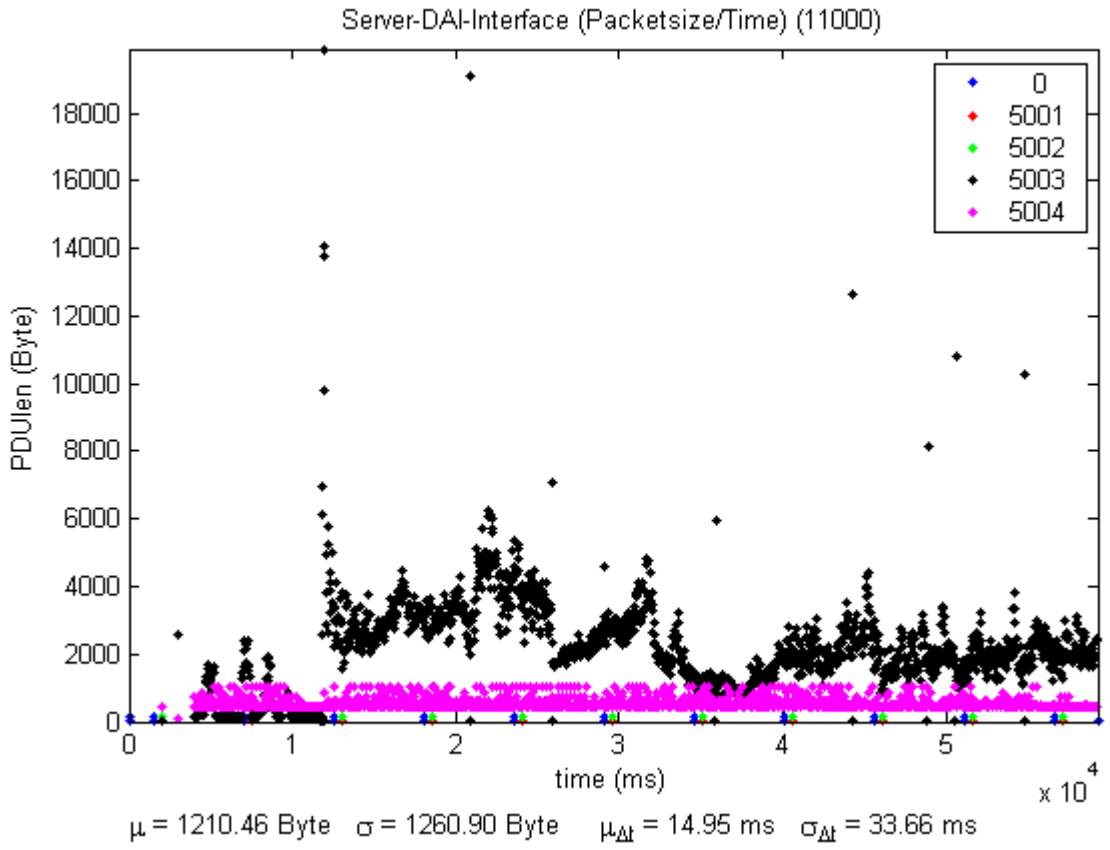


Abbildung 264: Datenströme am Server-DAI*: KillerBean2 MP3

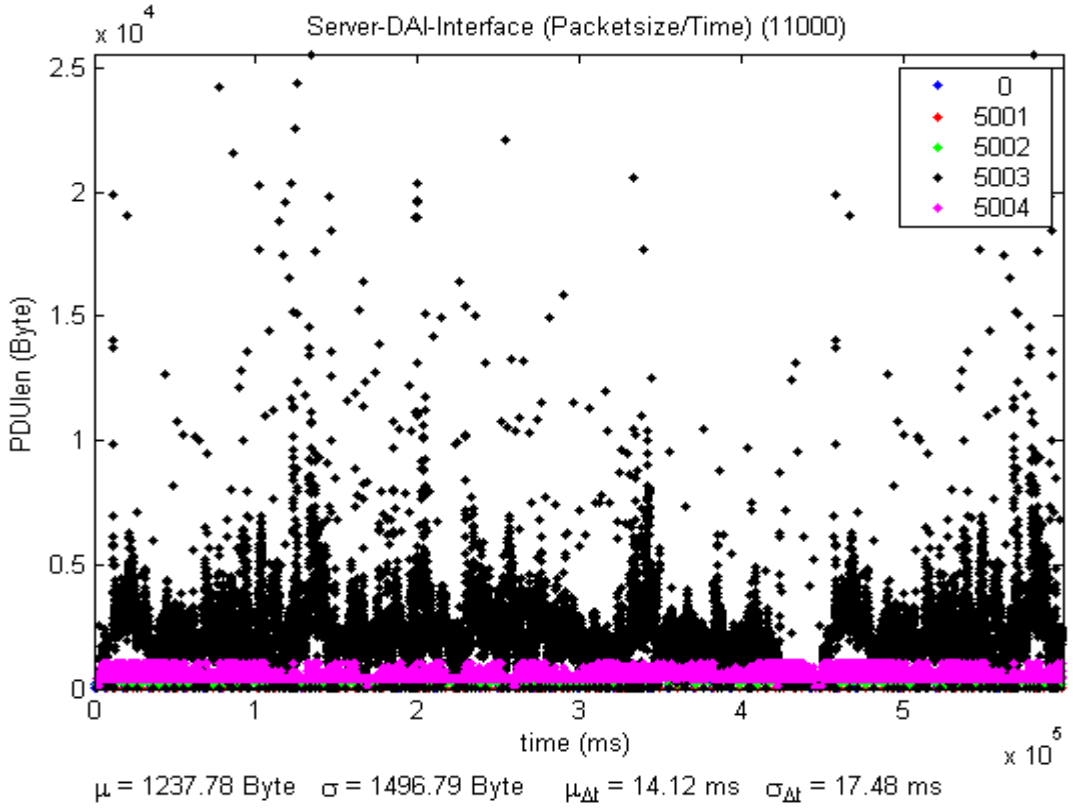


Abbildung 265: Datenströme (10 Min.) am Server-DAI*: KillerBean2 MP3

E.2 „Roxette-Fingertips93-CBR600“

Ausgangsmaterial „DVD“

\Test\007Richard\002_Roxette-Fingertips93\source

Datei-Name	vts_02.idx, vts_02_0.ifo, vts_02_1.vob, vts_02_3.vob	
Datei-Format	vob (DVD–RIP einzelnes Kapitel)	
	Video	**Audio**
Format	Mpeg 2 Video	ac3 (2ch) (Dolby Surround)
Mittlere Bitrate	6000 kBit/s (750 kByte/s)	192 kbit
CBR/VBR	VBR	CBR
Länge	3:42	3:42
Datei-Größe	135 MB (mit Audio)	-
Bildschirmgröße	720x576	
Bildwiederholungsrate	25	
Besonderes	Interlaced	-

MP4-Datei „Roxette_CBR600.mp4“

\Test\007Richard\002_Roxette-Fingertips93\XviD_CBR600_496x368_aac-LC

Datei-Name	Roxette_CBR600.mp4	
Datei-Format	mp4 File	
	Video	**Audio**
Format	XviD (Mpeg4 Video)	aac_LC
Mittlere Bitrate	597,8 kBit/s (73 kByte/s)	128 kbit/s
CBR/VBR	CBR	VBR
Länge	3:42	3:42
Datei-Größe	15,9 MB	3,46 MB
Bildschirmgröße	496x368	
Bildwiederholungsrate	25	
Besonderes	-	-

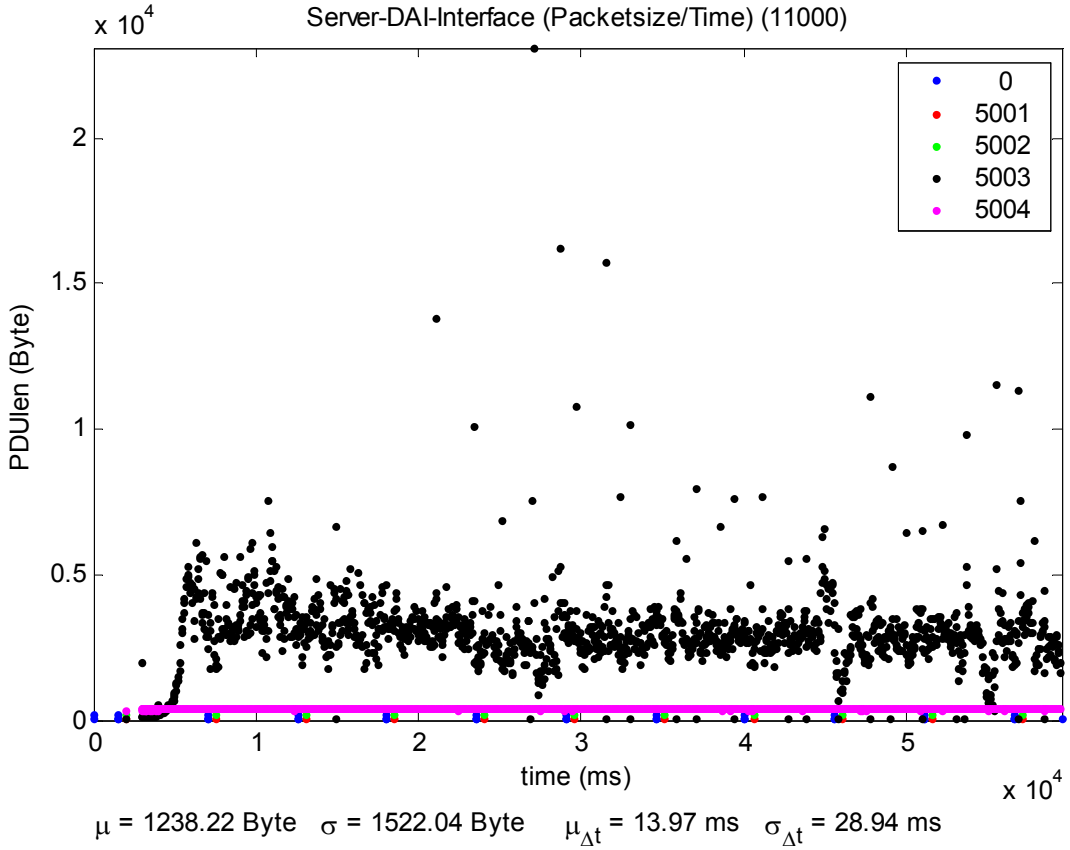


Abbildung 266: Datenströme am Server-DAI*: Roxette CBR 600

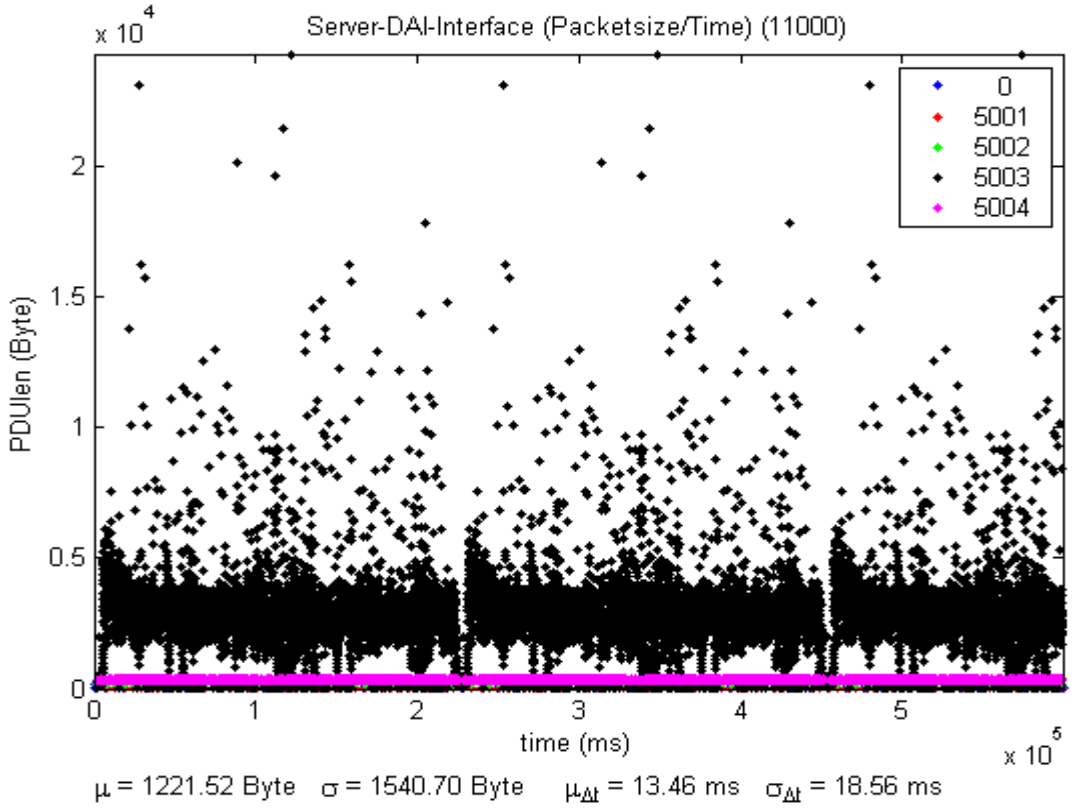


Abbildung 267: Datenströme (10 Min.) am Server-DAI*: Roxette CBR 600

E.3 „Roxette-Fingertips93-VBR600“

Ausgangsmaterial „DVD“

\Test\007Richard\002_Roxette-Fingertips93\source

Datei-Name	vts_02.idx, vts_02_0.ifo, vts_02_1.vob, vts_02_3.vob	
Datei-Format	vob (DVD–RIP einzelnes Kapitel)	
	Video	**Audio**
Format	Mpeg 2 Video	ac3 (2ch) (Dolby Surround)
Mittlere Bitrate	6000 kBit/s (750 kByte/s)	192 kbit
CBR/VBR	VBR	CBR
Länge	3:42	3:42
Datei-Größe	135 MB (mit Audio)	n/a
Bildschirmgröße	720x576	
Bildwiederholungsrate	25	
Besonderes	B-Frames, etc. Interlaced	-

MP4-Datei „Roxette_VBR600.mp4“

\Test\007Richard\002_Roxette-Fingertips93\XviD_VBR600__496x368_aac-LC

Datei-Name	Roxette_VBR600.mp4	
Datei-Format	mp4 File	
	Video	**Audio**
Format	XviD (Mpeg4 Video)	aac_LC
Mittlere Bitrate	612,5 kBit/s (74,8 kByte/s)	128k
CBR/VBR	VBR	VBR
Länge	3:42	3:42
Datei-Größe	16,9 MB	3,46 MB
Bildschirmgröße	496x368	
Bildwiederholungsrate	25	
Besonderes	-	-

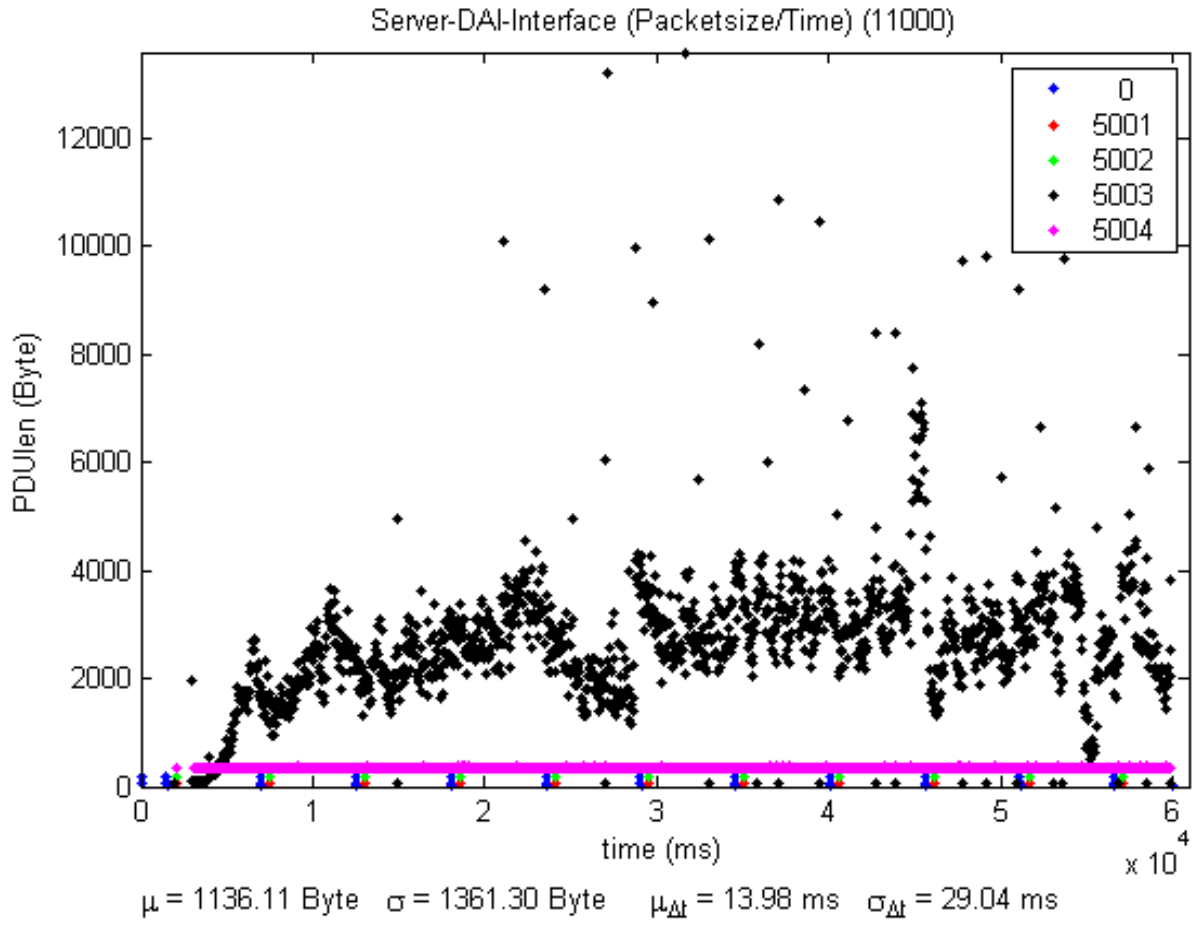


Abbildung 268: Datenströme am Server-DAI*: Roxette VBR 600

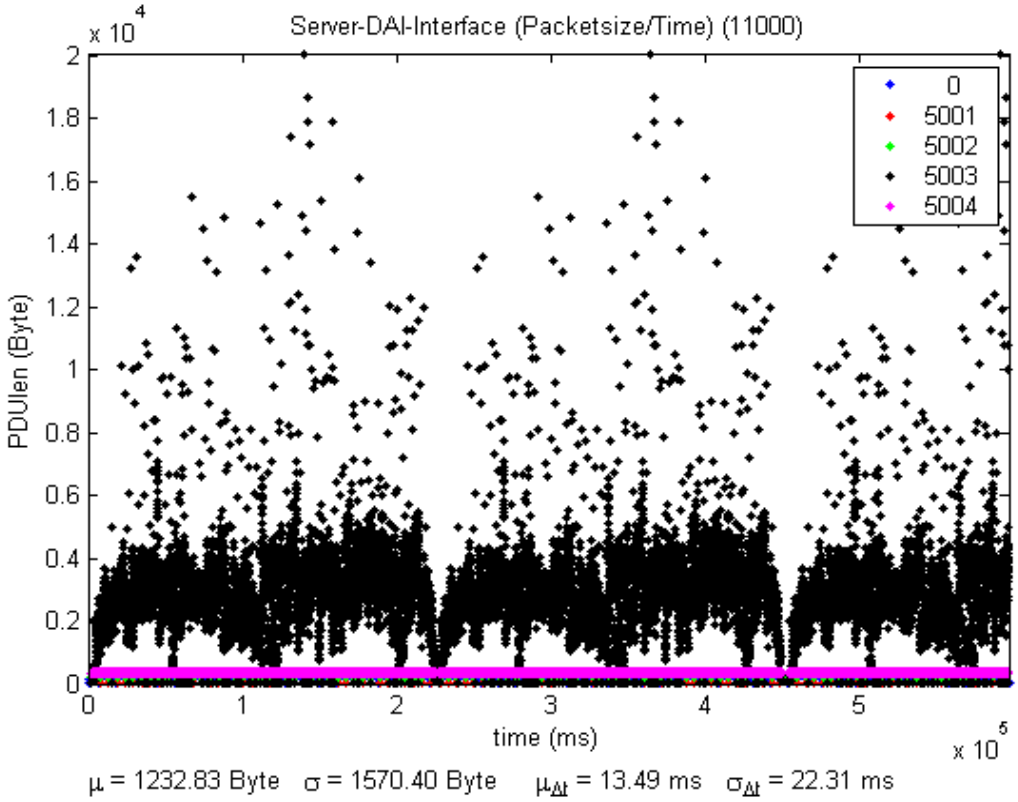


Abbildung 269: Datenströme (10 Min.) am Server-DAI*: Roxette VBR 600

E.4 „Roxette-Fingertips93-VBR1200“

Ausgangsmaterial „DVD“

\Test\007Richard\002_Roxette-Fingertips93\source

Datei-Name	vts_02.idx, vts_02_0.ifo, vts_02_1.vob, vts_02_3.vob	
Datei-Format	vob (DVD–RIP einzelnes Kapitel)	
	Video	**Audio**
Format	Mpeg 2 Video	ac3 (2ch) (Dolby Soround)
Mittlere Bitrate	6000 kBit/s (750 kByte/s)	192 kbit
CBR/VBR	VBR	CBR
Länge	3:42	3:42
Datei-Größe	135 MB (mit Audio)	-
Bildschirmgröße	720x576	
Bildwiederholungsrate	25	
Besonderes	Interlaced	-

MP4-Datei „Roxette_VBR1200.mp4“

\Test\007Richard\002_Roxette-Fingertips93\XviD_VBR1200_640x480_aac-LC

Datei-Name	Roxette_VBR1200.mp4	
Datei-Format	mp4 File	
	Video	**Audio**
Format	XviD (Mpeg4 Video)	aac_LC
Mittlere Bitrate	1209,6 kBit/s (147,7 kByte/s)	128k
CBR/VBR	VBR	VBR
Länge	3:42	3:42
Datei-Größe	32,2 MB	3,46 MB
Bildschirmgröße	640x480	
Bildwiederholungsrate	25	
Besonderes	-	-

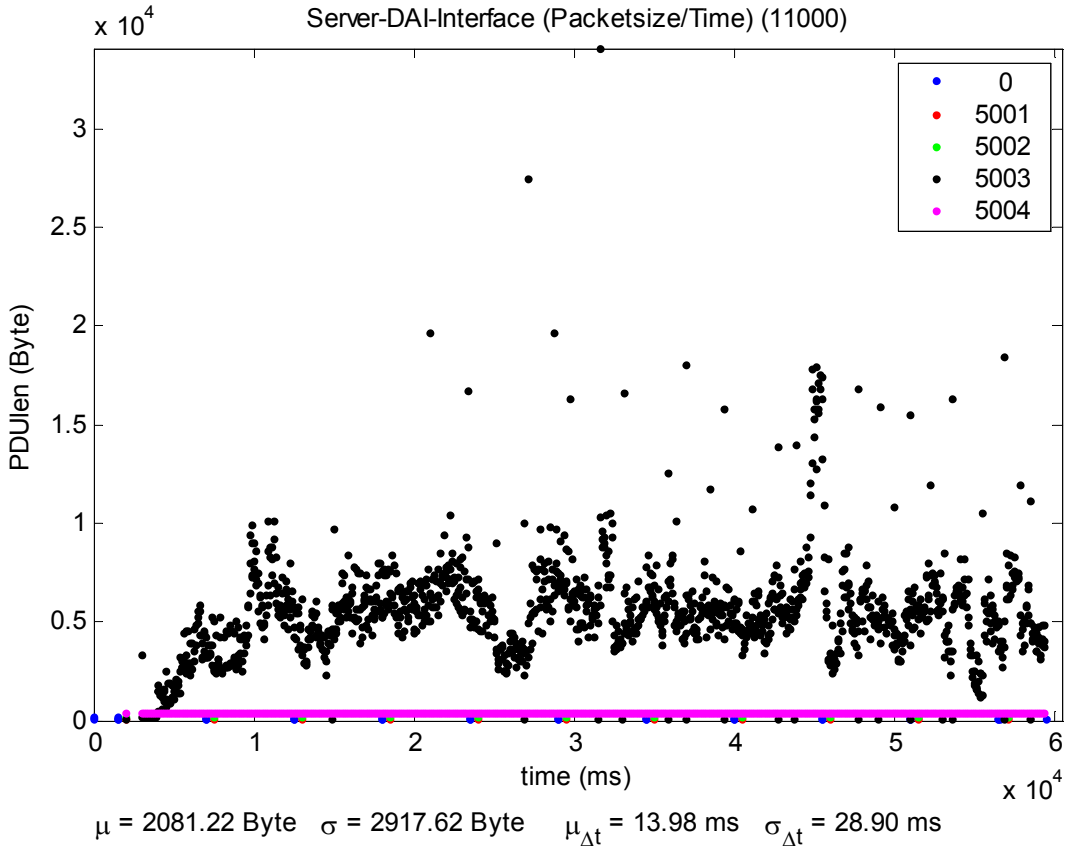


Abbildung 270: Datenströme am Server-DAI*: Roxette VBR 1200

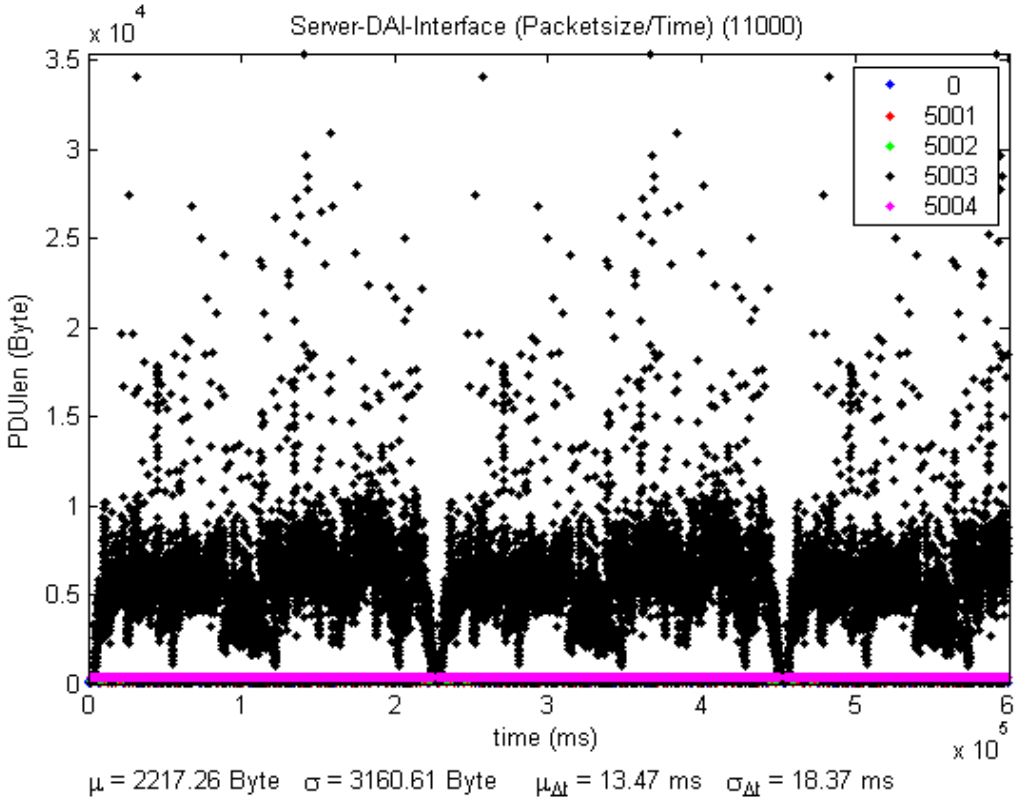


Abbildung 271: Datenströme (10 Min.) am Server-DAI*: Roxette VBR 1200

E.5 „Roxette-Queen of Rain“

Ausgangsmaterial „DVD“

\Test\007Richard\003_Roxette-Queen of Rain\Source

Datei-Name	vts_02.idx, vts_02_0.ifo, vts_02_1.vob, vts_02_3.vob	
Datei-Format	vob (DVD–RIP einzelnes Kapitel)	
	Video	**Audio**
Format	Mpeg 2 Video	ac3 (2ch) (Dolby Soround)
Mittlere Bitrate	9000 kBit/s (1125 kByte/s)	192 kbit
CBR/VBR	VBR	CBR
Länge	5:02	5:02
Datei-Größe	149 MB (mit Audio)	-
Bildschirmgröße	720x576	
Bildwiederholungsrate	25	
Besonderes	Interlaced	-

MP4-Datei „Roxette_Qeen-of-Rain.mp4“

\Test\007Richard\003_Roxette-Queen of Rain\

Datei-Name	Roxette_Qeen-of-Rain[DivX.aac-LC].mp4	
Datei-Format	mp4 File	
	Video	**Audio**
Format	DivX (Mpeg4 Video)	aac_LC
Mittlere Bitrate	1001.7 kByte/s	128 kbit
CBR/VBR	VBR	VBR
Länge	5:02	5:02
Datei-Größe	29,9 MB	3,46 MB
Bildschirmgröße	640x480	
Bildwiederholungsrate	25	
Besonderes	-	-

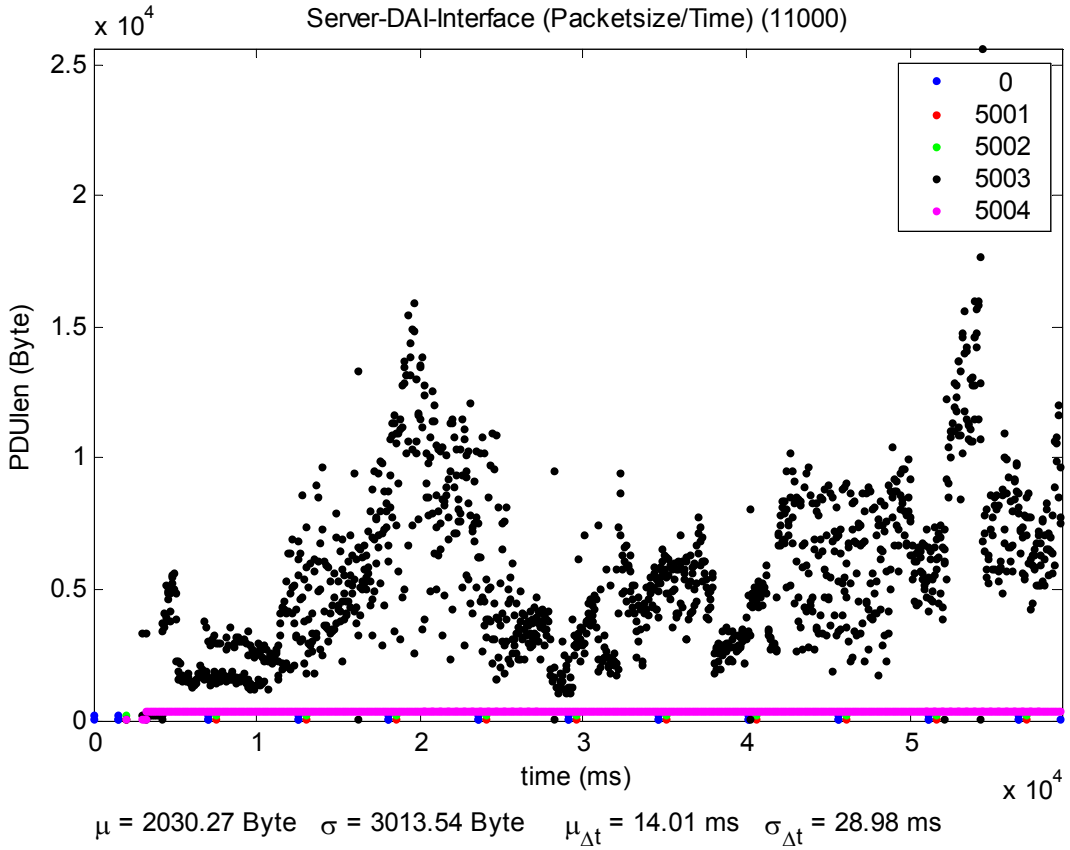


Abbildung 272: Datenströme am Server-DAI*: Roxette Queen of Rain

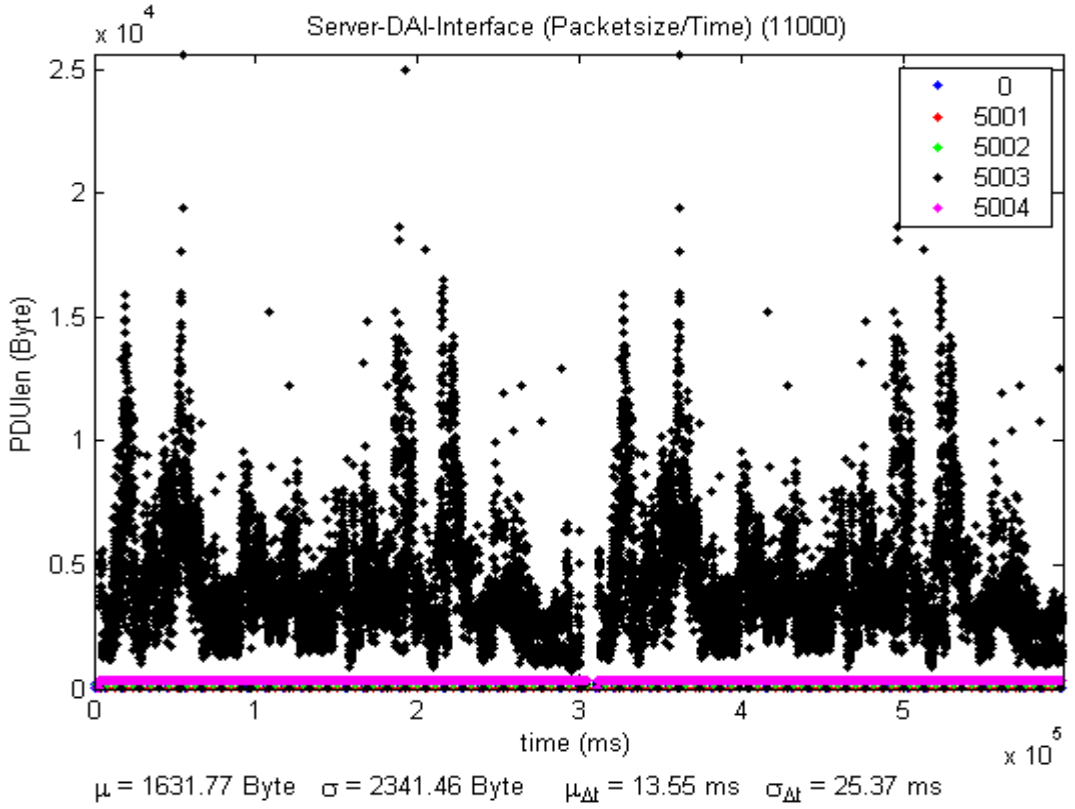


Abbildung 273: Datenströme (10 Min.) am Server-DAI*: Roxette Queen of Rain

E.6 „Matrix_XP“

Ausgangsmaterial „MATRIX_de_ultra.mpeg“

Datei-Name	MATRIX_de_ultra.mpeg	
Datei-Format	mpeg 1	
	Video	**Audio**
Format	Mpeg 1 Video	mpeg 1 Audio
Mittlere Bitrate	1000 kBit/s (125 kByte/s)	192 kbit
CBR/VBR	CBR	CBR
Länge	3:52	3:52
Datei-Größe	31,6 MB (mit Audio)	3,46 MB
Bildschirmgröße	320x288	
Bildwiederholungsrate	25	
Besonderes	-	-

MP4-Datei „Matrix-XP.mp4“

\Test\007Richard\004_matrix_xp

Datei-Name	Matrix-XP[XviD-B-Frames_mp3].mp4
Datei-Format	mp4 File

	Video	Audio
Format	XviD (Mpeg4 Video)	mp3
Mittlere Bitrate	109.1 kByte/s	128k
CBR/VBR	VBR	VBR
Länge	3:52	3:52
Datei-Größe	24,8 MB	3,45 MB
Bildschirmgröße	396x280	
Bildwiederholungsrate	25	
Besonderes	B-Frames	-

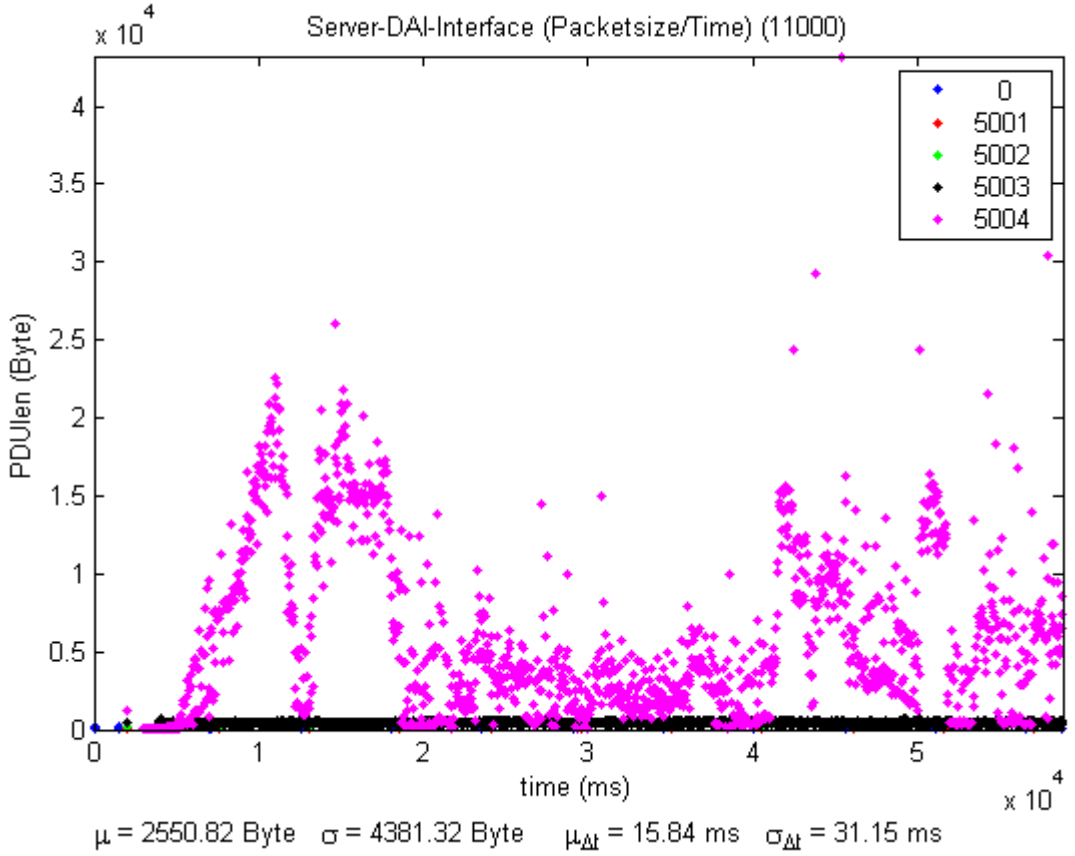
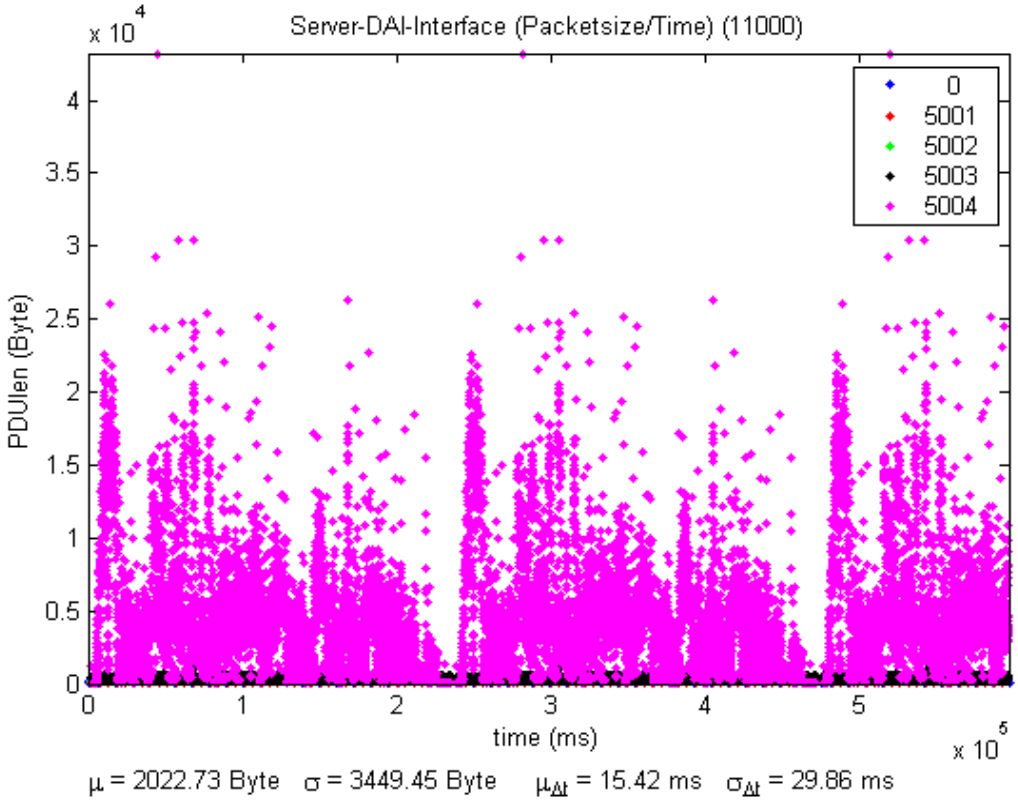


Abbildung 274: Datenströme am Server-DAI*: Matrix XP



E.7 „Tomb Raider 2 Trailer“

Ausgangsmaterial „tomb_raider.mov“

Datei-Name	tomb_raider_2-tlr_m480.mov	
Datei-Format	mov	
	Video	**Audio**
Format	Sorenson3	Q Design Music 2
Mittlere Bitrate	152,6 KByte/s	13,6 kByte/s
CBR/VBR	VBR	n/a
Länge	2:26	2:26
Datei-Größe	21,8 MB	1,9 MB
Bildschirmgröße	480x260	
Bildwiederholungsrate	24	
Besonderes	-	-

MP4-Datei „tomb_raider_2.mp4“

\Test\007Richard\005_tomb_raider_2

Datei-Name	tomb_raider_2-tlr_m480-[XviD-Bframes-GMC_aac-LC].mp4	
Datei-Format	mp4 File	
	Video	**Audio**
Format	XviD (Mpeg4 Video)	aac-LC

Mittlere Bitrate	128.9 KByte/s	128 kbit
CBR/VBR	VBR	VBR
Länge	2:26	2:26
Datei-Größe	19,9 MB	2,29 MB
Bildschirmgröße	480x260	
Bildwiederholungsrate	24	
Besonderes	B-Frames GMC	-

E.8 „Terminator 3 Trailer“

Ausgangsmaterial „t3-domestic.mov“

Datei-Name	t3-domestic_trailer_m480.mov	
Datei-Format	mov	
	Video	**Audio**
Format	Sorenson3	Q Design Music 2
Mittlere Bitrate	137,5 kByte/s	15,5 KByte/s
CBR/VBR	-	-
Länge	2:29	2:29
Datei-Größe	20,0 MB	2,2 MB
Bildschirmgröße	480x208	
Bildwiederholungsrate	24	
Besonderes	-	-

MP4-Datei „t3-domestic_trailer.mp4“

\Test\007Richard\006_t3-domestic_trailer

Datei-Name	t3-domestic_trailer_m480_XviD-B-Frams.mp4	
Datei-Format	mp4 File	
	Video	**Audio**
Format	XviD (Mpeg4 Video)	aac-LC
Mittlere Bitrate	75.5 KByte/s	128k
CBR/VBR	VBR	VBR
Länge	2:29	2:29
Datei-Größe	11,2 MB	2,30 MB
Bildschirmgröße	480x260	
Bildwiederholungsrate	24	
Besonderes	B-Frames	-

E.9 „Hikaru no GO (ep1) –german Subtitle“

Ausgangsmaterial „Hikaru_no_Go.avi”

Datei-Name	[GFTP]_Hikaru_no_Go_01_(german).avi	
Datei-Format	avi	
	Video	**Audio**
Format	XviD	mp3
Mittlere Bitrate	908,8 kBit/s (110.9 kByte)	14000 – 176400 112 kBit
CBR/VBR	VBR	VBR
Länge	23:28	23:28
Datei-Größe	153 MB	18,8 MB
Bildschirmgröße	640x480	
Bildwiederholungsrate	23.98	
Besonderes	-	-

MP4-Datei „HikaruNoGO.ep1german.mp4“

Datei-Name	HikaruNoGO.ep1german.SUB[mpeg4ip_0.9.7.2][XviD_aac-LC].mp4	
Datei-Format	mp4 File	
	Video	**Audio**
Format	XviD	aac-LC
Mittlere Bitrate	908,8 kBit/s (110.9 kByte)	128k
CBR/VBR	VBR	VBR
Länge	23:28	23:28
Datei-Größe	153 MB	21,1 MB
Bildschirmgröße	640x480	
Bildwiederholungsrate	23.98	
Besonderes	-	-

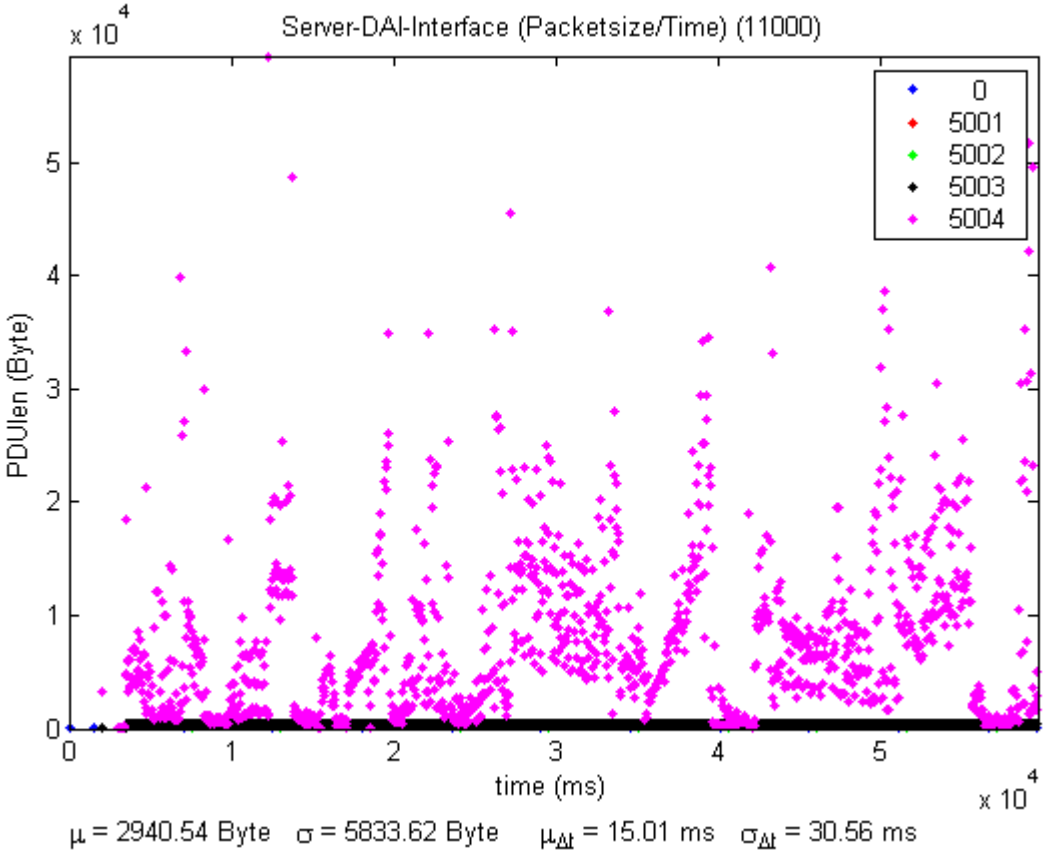


Abbildung 275: Datenströme am Server-DAI*: Hikaru No Go, German

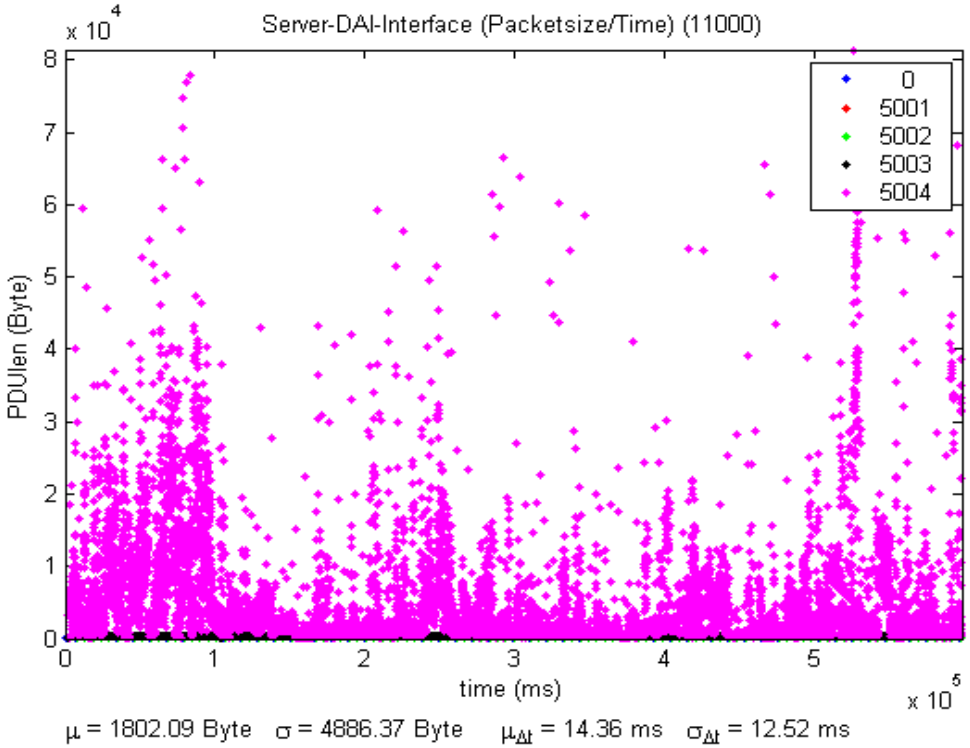


Abbildung 276: Datenströme (10 Min.) am Server-DAI*: Hikaru No Go, German

E.10 „Hikaru no GO (ep1) – italy Subtitle“

Ausgangsmaterial “Hikaru no Go - 001 (ita).avi”

Datei-Name	[HnG-ITP] Hikaru no Go - 001 (ita)[XviD][44ED39B2].avi	
Datei-Format	avi	
	Video	**Audio**
Format	XviD	mp3
Mittlere Bitrate	1116,3 kBit/s (136.3 kByte)	16000 – 1920000 128 kBit
CBR/VBR	VBR	VBR
Länge	22:54	22:54
Datei-Größe	183 MB	21,1 MB
Bildschirmgröße	640x480	
Bildwiederholungsrate	23.98	
Besonderes	-	-

MP4-Datei „Hikaru no Go - 001 (SUB_ita).mp4“

Datei-Name	Hikaru no Go – 001 (SUB_ita)mpeg4ip_0.9.8[XiviD_mp3].mp4	
Datei-Format	mp4 File	
	Video	**Audio**
Format	XviD	mp3
Mittlere Bitrate	1116,3 kBit/s (136.3 kByte)	16000 – 1920000 128 kBit
CBR/VBR	VBR	VBR
Länge	22:54	22:54
Datei-Größe	183 MB	21,1 MB
Bildschirmgröße	640x480	
Bildwiederholungsrate	23.98	
Besonderes	-	-

E.11 „Matrix Reloaded Trailer“

Ausgangsmaterial „trailer_final_1000_dl.mov”

Datei-Name	trailer_final_1000_dl.mov	
Datei-Format	mov	
	Video	**Audio**
Format	Sorenson3	aac
Mittlere Bitrate	617.6 kByte/s	39 kByte /s
CBR/VBR	VBR	VBR
Länge	2:31	2:31
Datei-Größe	91,5 MB	5,7 MB

Bildschirmgröße	1000x540	
Bildwiederholungsrate	24	
Besonderes	-	-

MP4-Datei „Matrix_Reloaded_Trailer.mp4“

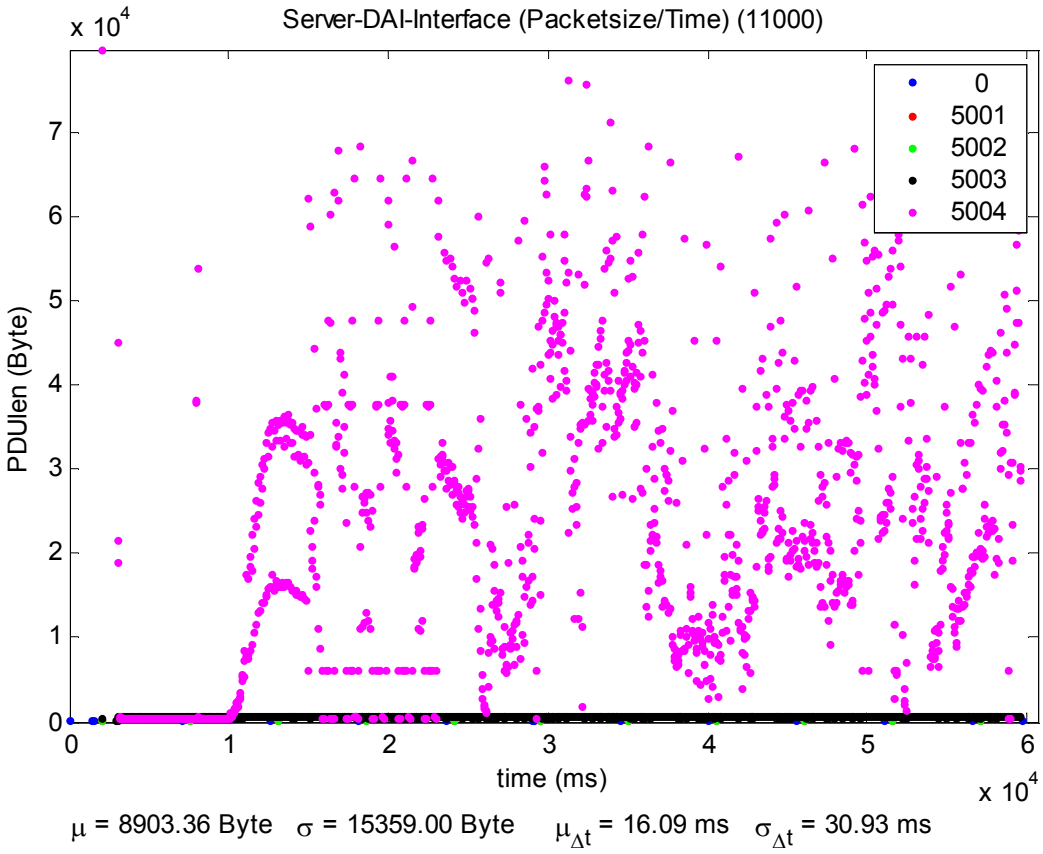


Abbildung 277: Datenströme am Server-DAI*: Matrix Reloaded Trailer

Datei-Name	Matrix_Reloaded_Trailer-[mpeg4ip0.9.8][XviD-B-Frames_aac-LC].mp4	
Datei-Format	mp4 File	
	Video	**Audio**
Format	XviD	aac
Mittlere Bitrate	5517,8 kBit/s (632.1 kByte/s)	16000 – 17640000 128 kBit
CBR/VBR	VBR	VBR
Länge	2:31	2:31
Datei-Größe	93.2 MB	2,4 MB
Bildschirmgröße	1000x540	
Bildwiederholungsrate	24	
Besonderes	B-Frames	-

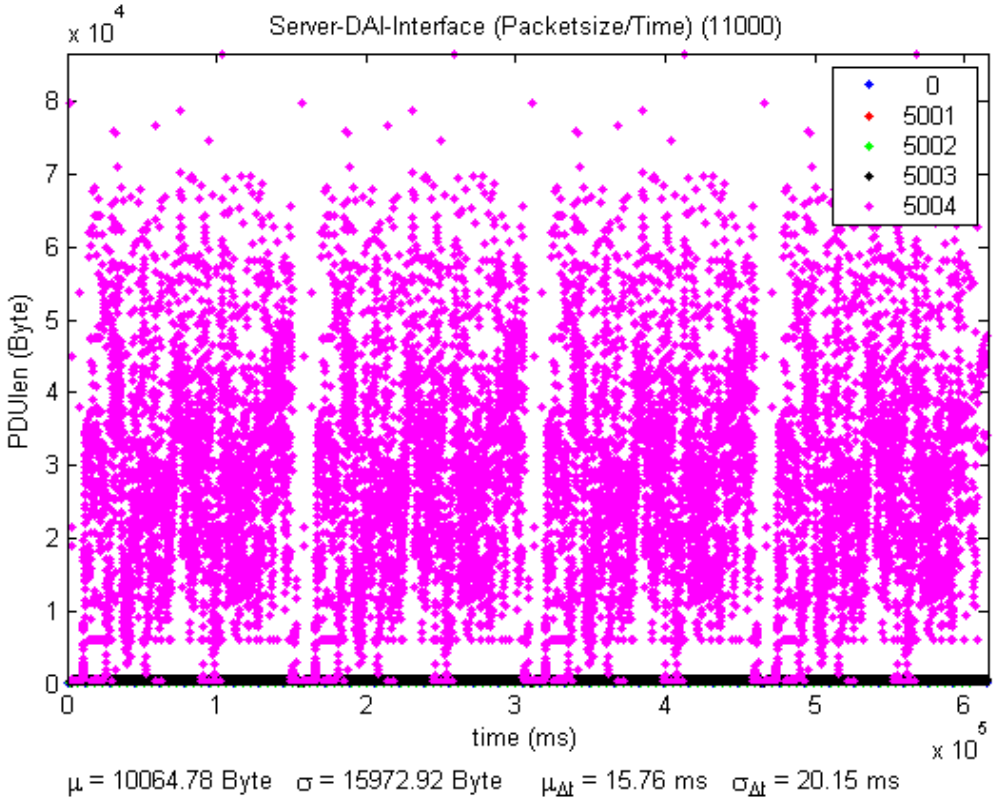


Abbildung 278: Datenströme (10 Min.) am Server-DAI*: Matrix Reloaded Trailer

E.12„Watch-Out“ (http://watch.out.free.fr/)

Ausgangsmaterial „HIGH_Watch-Out”

Datei-Name	HIGH_Watch-Out_!_Divx_pro5.avi	
Datei-Format	avi	
	Video	**Audio**
Format	DivX 5	mp3
Mittlere Bitrate	550.8 kBit/s (67.2 kByte/s)	7000 – 88200 56 kBit
CBR/VBR	VBR	VBR
Länge	8:16	8:16
Datei-Größe	32.8 MB	3,29 MB
Bildschirmgröße	720x576	
Bildwiederholungsrate	24	
Besonderes	-	-
Bild-Format	jpg	
Bild-Größe	3.606 Bytes	

MP4-Datei „watchout.mp4“

Datei-Name	watchout.mp4	
Datei-Format	mp4 File	
	Video	**Audio**
Format	DivX 5	mp3
Mittlere Bitrate	550.8 kBit/s (67.2 kByte/s)	7000 – 88200 56 kBit
CBR/VBR	VBR	VBR
Länge	8:16	8:16
Datei-Größe	32.8 MB	3,29 MB
Bildschirmgröße	720x576	
Bildwiederholungsrate	24	
Besonderes	-	-
Bild-Format	jpg	
Bild-Größe	3.606 Bytes	

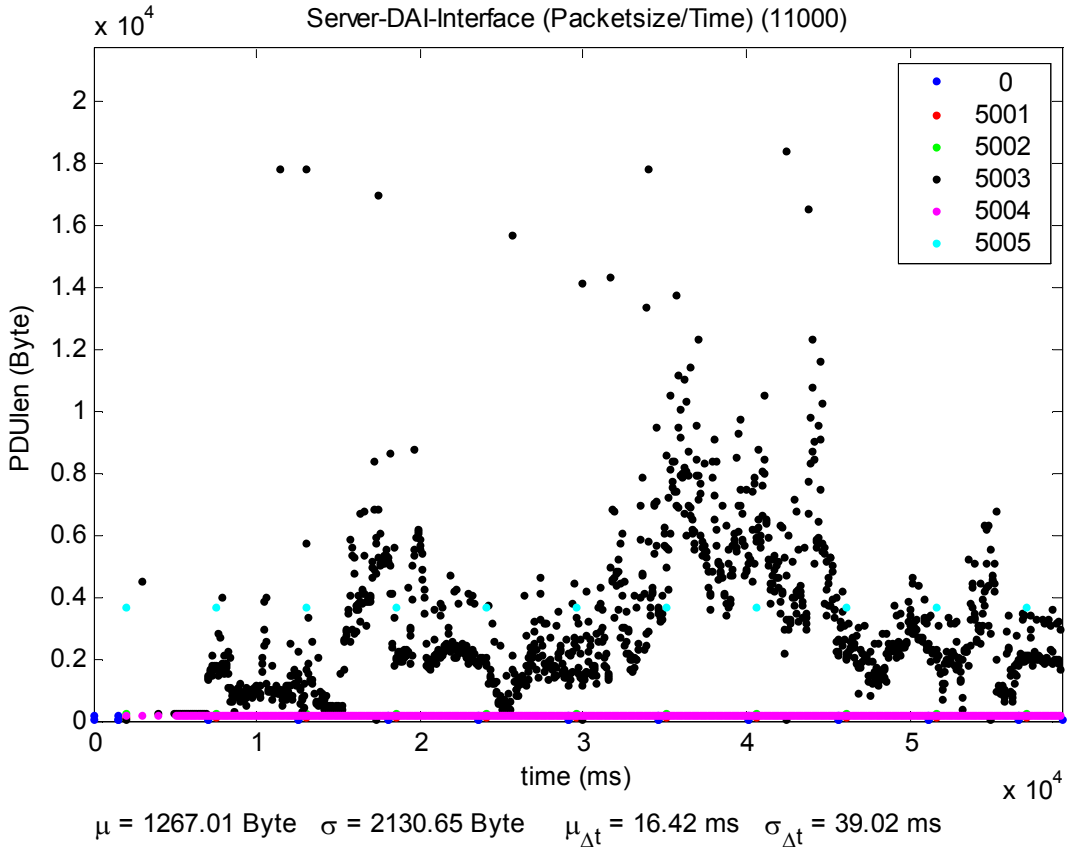


Abbildung 279: Datenströme am Server-DAI*: Watchout

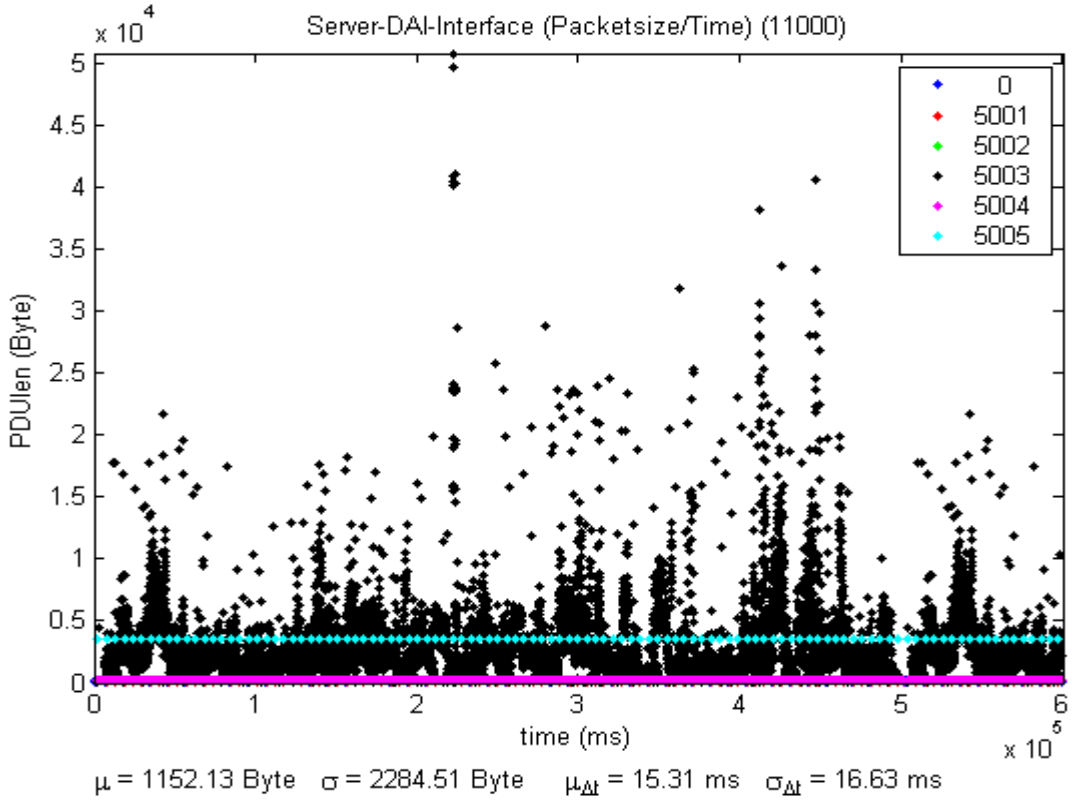


Abbildung 280: Datenströme (10 Min.) am Server-DAI*: Watchout

F Übersicht der MPEG Standards

Stand (Dezember 2003)

Norm	Teil	Titel
11172	**MPEG-1**	**Coding of moving pictures and associated audio at up to about 1.5 Mbit/s**
	Part 1	Systems
	Part 2	Video
	Part 3	Audio
	Part 4	Conformance testing
	Part 5	Software simulation
13818	**MPEG-2**	**Generic coding of moving pictures and associated audio**
	Part 1	Systems
	Part 2	Video
	Part 3	Audio
	Part 4	Conformance testing
	Part 5	Software simulation
	Part 6	System extensions - DSM-CC
	Part 7	Audio extension - NBC mode
	Part 8	VOID - (withdrawn)
	Part 9	System extension RTI
	Part 10	Conformance extension - DSM-CC
	Part 11	IPMP on MPEG-2 Systems
14496	**MPEG-4**	**Coding of audio-visual objects**
	Part 1	Systems
	Part 2	Visual
	Part 3	Audio
	Part 4	Conformance testing
	Part 5	Reference Software
	Part 6	Delivery Multimedia Integration Framework
	Part 7	Optimised software for MPEG-4 tools

	Part 8	4 on IP framework
	Part 9	Reference Hardware Description
	Part 10	Advanced Video Coding
	Part 11	Scene Description and Application Engine
	Part 12	ISO Base Media File Format
	Part 13	IPMP Extensions
	Part 14	MP4 File Format
	Part 15	AVC File Format
	Part 16	Animation Framework eXtension (AFX)
	Part 17	Streaming Text Format
	Part 18	Font compression and streaming
15938	**MPEG-7**	**Multimedia Content Description Interface**
	Part 1	Systems
	Part 2	Description Definition Language
	Part 3	Visual
	Part 4	Audio
	Part 5	Multimedia Description Schemes
	Part 6	Reference Software
	Part 7	Conformance
	Part 8	Extraction and Use of MPEG-7 Descriptions
	Part 9	Profiles
	Part 10	Schema definition
21000	**MPEG-21**	**Multimedia Framework**
	Part 1	Vision, Technologies and Strategy
	Part 2	Digital Item Declaration
	Part 3	Digital Item Identification and Description
	Part 4	IPMP
	Part 5	Rights Expression Language
	Part 6	Rights Data Dictionary
	Part 7	Digital Item Adaptation

	Part 8	Reference Software
	Part 9	File Format
	Part 10	Digital Item Processing
	Part 11	Evaluation Tools for Persistent Association
	Part 12	Test Bed for MPEG-21 Resource Delivery
	Part 13	Scalable video coding
	Part 14	Conformance

Tabelle 57: MPEG Standards und Teile

G Automatisierter Test

Für das Testen des Broadcast-Szenarios unter der Verwendung des Broadcast-Servers müssen Durchläufe mit verschiedenen Einstellungen vorgenommen werden. So müssen beispielsweise Tests mit verschiedener Bandbreite oder unterschiedlichen Rate-Shaping beziehungsweise Traffic-Shaping Strategien durchgeführt werden. Diese Einstellungen befinden sich in der *BroadcastServer.ini*-Datei. Damit die Parameter nicht manuell adaptiert werden müssen, wurden einige Programme entwickelt, die die Tests automatisieren.

Dazu werden zwei Rechner über einen Ethernet-100 MBit/s-Switch verbunden. Ein Rechner wird als Broadcast-Server und der andere als Terminal eingerichtet. Auf dem Server läuft das Programm *BroadcastCtrl.exe*, das den Broadcast-Server mit den gewünschten Einstellungen startet, die Log-Dateien in einen vorgegebenen Ordner kopiert und diese auswertet. Auf dem Client läuft *OsmoseTester.exe*, dieses Programm wartet auf Nachrichten von BroadcastCtrl und führt bestimmte Operationen wie das Starten oder Beenden des Players *Osmose* aus.

Nachdem der Test vollständig durchlaufen ist und alle Log-Dateien generiert wurden, kann mit Hilfe des Programms *LogFileConvert.exe* die Generierung der Matlab Diagramme durchgeführt werden.

Das Programm *CheckTestFolder.exe* dient nur der Ergänzung und ermöglicht ein schnelles Überprüfen, welche Testläufe vollständig erstellt wurden.

G.1 BroadcastCtrl

Dieses Programm steuert den Broadcast-Server und erzeugt einen kompletten Satz von Messdaten, analysiert diese und konvertiert sie ins Matlab Format. Mit *StartTest* beginnt ein Durchlauf, wobei die Parameter der *BroadcastServer.ini* im Ausgabefenster (*Broadcast Server Settings*) angezeigt werden.

Folgende Eingabeparameter werden benötigt:

- Path To Broadcast Server:
  Pfad zum Ordner in dem sich die Datei *BroadcastServer_Auto.exe* befindet. Dieser Server beginnt im Gegensatz zum normalen Broadcast Server nach dem Start automatisch mit dem Broadcasten der MPEG-4 Applikation. Außerdem muss in diesem Ordner ein Unterordner Matlab vorhanden sein, in dem sich das Konvertertool *CSVToMatlab.exe* befindet, sowie ein Unterordner *vid* in dem sich die MP4-Dateien befinden.
- Path To Log Files:
  Pfad zum Ordner in dem die Ausgabe-Log-Dateien (*srv_rtp.log*, *cli_rtp.log*, *srv_out.csv*) gespeichert werden. In diesem Ordner wird je nach ausgewähltem Test ein Unterordner generiert, in dem die einzelnen Tests (Test001,Test002…) gespeichert werden. Wichtig ist, dass beim Client, dem *OsmoseTester* der gleiche Pfad angegeben wird, damit Client und Server die Log-Dateien in den gleichen Ordner kopieren. Dazu sollte idealerweise auf dem Client ein Netzlaufwerk verbunden werden (siehe *OsmoseTester*).

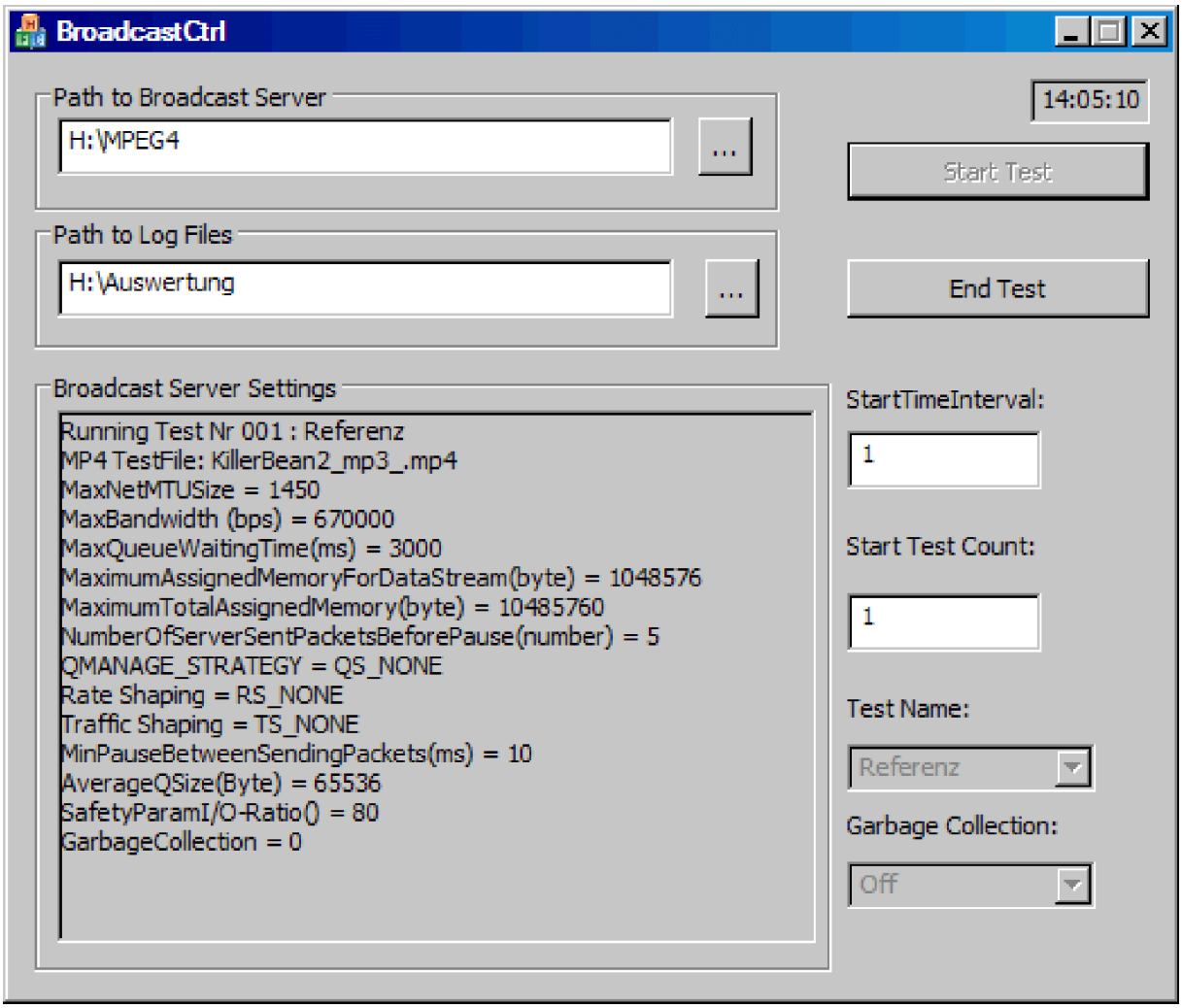


Abbildung 281: Anwendung „BroadcastCtrl“

- Start Time Intervall:
  Abspieldauer der MPEG-4 Applikation in Minuten.
- Start Test Count:
  Gibt den Testlauf an bei dem gestartet werden soll, falls ein vorheriger Durchlauf abgebrochen wurde.
- Test Name:
  Auswahl der zu testenden Kategorie. Folgende Einstellungen sind möglich
  - Referenz Referenz mit allen MPEG-4 Applikationen
  - Scheduling Scheduling mit *Killerbean* MPEG-4 Applikationen
  - Diverse Diverse Einstellungen mit *Killerbean* MPEG-4 Applikationen
  - AllTests führt die drei oben genannten Tests hintereinander aus.
- Garbage Collection:
  Hier kann die Garbage Collection am Client ein oder ausgeschaltet werden, bei On/Off werden beide Fälle durchlaufen

G.2 OsmoseTester

Steuert den Client *Osmose* und kopiert die entstehenden Log-Dateien (*cli_rtp.log*) zum Server. *Osmose* wird automatisch gestartet sobald der Broadcast Server durch *BroadcastCtrl* gestartet wird.

Folgende Eingabeparameter werden benötigt:

- Path To Osmose:
  Pfad zum Ordner in dem sich *Osmose.exe* (der MPEG-4 Player) befindet.
- Path To Log Files:

Pfad zum Ordner in den die Log-Dateien kopiert werden sollen. Dieser Ordner muss derselbe sein, den auch der Broadcast-Server verwendet.

Beispiel für Ordner:

Auf dem Server (IP Adresse des Servers: „132.176.12.1“):

Path To Log Files: „e:\Auswertung“

Laufwerk E freigegeben als „Laufwerk_E“

Am Client:

Netzlaufwerk „Q:“ verbinden mit „132.176.12.1\\Laufwerk_E“

Path To Log Files (OsmoseTester) : „Q:\Auswertung“

G.3 CSV2Matlab

Das Werkzeug *CSV2Matlab* wurde in Kapitel 8 vorgestellt. Aufgrund der Log-Dateien des Broadcast-Server-DMIFs *srv_rtp.log* und des Broadcast-Client-DMIFs *cli_rtp.log* wird die Datei *srv_out.csv* gebildet, die mit den entwickelten *Matlab*-Skripten weiterverarbeitet wird.

G.4 LogFileConverter

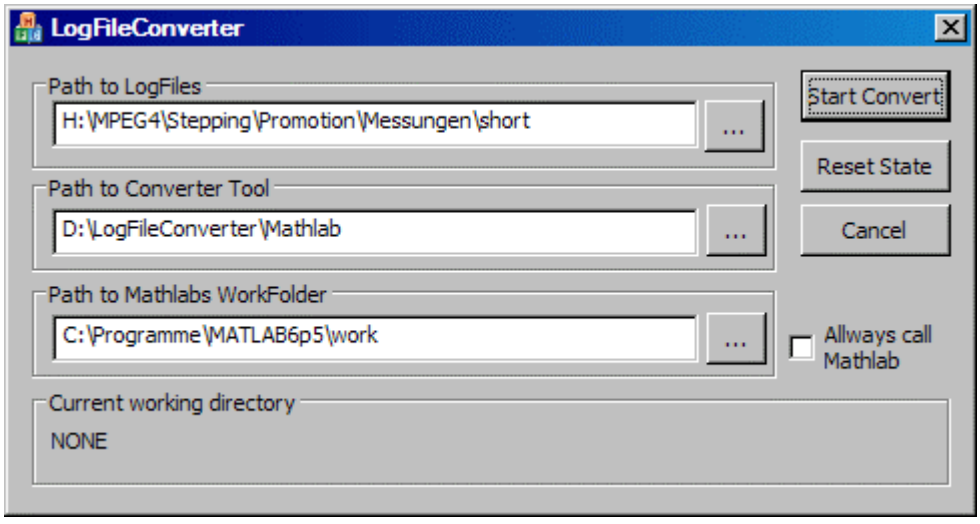


Abbildung 282: Anwendung „LogFileConverter“

Das Programm *LogFileConverter* generiert aus den in den Testläufen erzeugten Log-Dateien mittels Aufruf von *Matlab* Diagramme. Dabei erkennt das Programm selbständig, ob die *srv_out.csv* Datei vorhanden ist. Falls das nicht der Fall ist, wird das Konverter Tool *CSV2MatLab.exe* aufgerufen. Mit *StartConvert* wird gestartet, dabei wird bei jedem Test vermerkt, ob dieser bereits in Bearbeitung ist, wodurch ein paralleles Bearbeiten durch mehrere Rechner möglich ist. Mit *ResetState* wird der Status (*Ordner in Bearbeitung*/*fertig bearbeitet*) zurückgesetzt, so dass bei einem erneu-

ten *StartConvert* wieder alle Ordner prozessiert werden können. Beispielsweise kann, wenn das Konverter-Tool oder Matlab nicht korrekt zu Ende rechnet, mit *ResetState* der Status dann zurückgesetzt und erneut *StartConvert* aufgerufen werden. Zu dieser Zeit darf kein paralleles Arbeiten stattfinden, da der Verriegelungsmechanismus außer Kraft gesetzt ist. Das Programm erkennt, ob bereits Bilder generiert wurden. Ist das zutreffend, wird der entsprechende Testlauf übersprungen.

Wird *Allways call Matlab* aktiviert, dann wird *Matlab* immer aufgerufen, beispielsweise, wenn ein neues *Matlab*-Skript erstellt wurde.

Folgende Eingabeparameter werden benötigt:

- Path To LogFiles:
  Pfad zum Ordner, in dem sich die Log-Dateien befinden
- Path To Converter Tool:
  Pfad zum Ordner, in dem sich das Konverter Tool (CSV2MatLab.exe) befindet
- Path To Matlabs Work Folder:
  Pfad zum Ordner, in dem sich die Matlab-Skripte befinden. Standardmäßig ist dies „[PATHTOMATLAB]\work“. Dieses muss unbedingt eingehalten werden, da vom Arbeitsordner ausgehend die *Matlab.exe* gesucht wird.

G.5 CheckTestFolder

Dies ist ein einfaches Testprogramm, das den angegeben Pfad (*Path to Log Files*) durchsucht und eine Log-Datei (*d:**Output.txt*) generiert, worin folgende Informationen enthalten sind:

- Path: Pfad zum Test Ordner relativ zum angegeben Startverzeichnis
- CL: Client Log File (cli_rtp.log) vorhanden (0/1)
- SL: Server Log File (srv_rtp.log) vorhanden (0/1)
- CM: Konvertiertes File (srv_out.csv) vorhanden (0/1)
- PF: Ordner bereits Bearbeitet/ zurzeit in Bearbeitung
- PF: Ordner Bilder (Picture) vorhanden (0/1)
- NrPics: Anzahl der Diagramme im Bilder (Picture) Ordner

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[AS98] F.-U. Andersen und J. Schmidt: Weltweit Video – MBone: Multimedia im Internet, Verlag Heinz Heise, c't, Ausgabe 21, Seite 262 ff, 1998

[BB92] B. Breutmann, R. Burkhardt: Objektorientierte Systeme, Grundlagen – Werkzeuge, Einsatz, Hanser, München, Wien, 1992

[Ben97] H. Benoit: Digital Television MPEG-1, MPEG-2 and principles of the DVB system, AJ Wiley & Sons, 1996, 1997

[Ber93] E. Berard: Essays on Object-Oriented Software Engineering, Volume I, Prentice Hall, Englewood Cliffs, 1993

[BJR97a] G. Booch, I. Jacobson und J. Rumbaugh: Unified Modeling Language, Notation Guide, Version 1.5, 2003, zu beziehen über http://www.omg.org

[BJR97b] G. Booch, I. Jacobson und J. Rumbaugh: Unified Modeling Language, UML Semantics, Version 1.5, 2003, zu beziehen über http://www.omg.org

[BJR97c] G. Booch, I. Jacobson und J. Rumbaugh: Unified Modeling Language, UML Summary, Version 1.5, 2003, zu beziehen über http://www.omg.org

[Boo94] G. Booch: Object Oriented Design with Aplications, Second Edition, The Benjamin/-Cummings Publishing Company, Redwood City, 1994

[Bra97] R. Braden: Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification, RFC 2205, Internet Engineering Task Force, September 1997

[BS00] T. Bonse und M. Stepping: MPEG-4 - Neue Nutzungspotentiale in der Fernlehre, FERNSEH- und KINO-TECHNIK, 54. Jahrgang Nr. 1-2/2000, Seiten 43-51, Hüthig-Verlag, Heidelberg

[BS98] T. Bonse, M. Stepping, F. Casalino, M. Quaglia, G. Franceschini: MPEG-4 Systems, concepts and implementation, ECMAST Konferenz, 26.05.-28.05.1998, Berlin, Seiten 504 - 517, Springer-Verlag, Heidelberg

[BS99a] T. Bonse und M. Stepping: Broadcast Multimediaübertragung auf der Basis von MPEG-4, 8. Dortmunder Multimediaseminar der FKTG, 27.-29.09.1999, Dortmund

[BS99b] T. Bonse und M. Stepping: MPEG-4 PC - Authoring and Playing of MPEG-4 Content, EMMSEC Konferenz, Proceedings on Business and Work in the Information Society: New Technologies and Applications, 21.-23.6.1999, Stockholm, Schweden, Cheshire Henbury Verlag, Großbritannien

[BS99c] T. Bonse, M. Stepping, P. Gerken, S. Schultz, G. Knabe, F. Casalino, G. Di Cagno, M. Quaglia, J.-C. Dufourd, S. Boughoufalah, F. Bouilhaguet, U. Mayer, J. Deicke, M. Glesner: MPEG-4 PC - Authoring and Playing of MPEG-4 Content for Local and Broadcast Applications, ECMAST Konferenz, Mai 1999, Madrid, Spanien, Seiten 108 - 119, Springer-Verlag, Heidelberg

[CRVG00] D.Curet, C.Roux, M.Veillard, E. Le Gall: A FlexMux tool with lower overhead, ISO/IEC JTC1/SC29/WG11 Beitrag M6612, 2000

[CIR98] D.Curet, C.Islas, K.Renout: MPEG4 SL and FlexMux management, fixed length SL-PDUs, ISO/IEC JTC1/SC29/WG11 Beitrag M3901, 1998

[CGR+03] D.Curet, E.Gouleau, S.Relier, C.Roux, P.Clement, G.Cherry: RTP Payload Format for MPEG-4 FlexMultiplexed Streams, Internet draft: draft-curet-avt-rtp-mpeg4-flexmux-04.txt, Internet Engineering Task Force, work in progress, 2003

[Das00] M. Dashti: Entwicklung eines Mappings von MPEG-4 DMIF-Signalisierungsmeldungen auf IP-Multicast und Erstellung eines Datentransportmechanismus mittels IP-Multicast, Diplomarbeit Universität Dortmund in Zusammenarbeit mit FernUniversität in Hagen, Universität Dortmund, 2000

[DBB99] J.-C. Dufourd, S. Boughoufalah und F. Bouilhaguet: Release of ENST tools for BIFS/MP4 editing and encoding, ISO/IEC JTC1/SC29/WG11, Beitrag M4690, 1999

[DC85] S. E. Deering, and D. R. Cheriton: Host Groups: A Multicast Extension for Datagram Internetworks, in Proceedings of the Ninth Data Communications Symposium, ACM/IEEE, Seiten 172 - 179, September 1985

[Dee91] S. Deering: Multicast Routing in a Datagram Internetwork, PhD thesis, Stanford University, 1991

[Dei00] J. Deicke: Schedulingstrategien für das Multiplexen audiovisueller Datenströme unter Berücksichtigung von Dienstgüte, Dissertation, TH Darmstadt, Shaker-Verlag, 2000

[DIN44300] DIN 44300-1: Informationsverarbeitung - Begriffe - Teil 1: Allgemeine Begriffe, Beuth-Verlag, Berlin, 1995

[DKR99] I. Defée, J. Kangasoja, M. Rustarion: COMIQS System for Commercial Presentations on the Internet, ECMAST Konferenz, 26.-28.5.1999, Madrid, Spanien, Seiten 262 - 280, Springer-Verlag, Heidelberg

[EEIG97] European Economic Interest Grouping (EEIG): European Information Technology Observatory 1997 (EITO), EITO, 1997

[Ele97] A. Eleftheriadis: Flavor: A Language for Media Representation, Proceedings, ACM Multimedia ’97 Conference, Seattle, Washington, Seiten 1 - 9, November 1997

[Ele99] A. Eleftheriadis: Traffic Shaping, E-Mail Konversation über den IP-Multicast Reflektor der Audio-Video-Transport-Group AVT des IETF, 28.01.1999, Archiv unter http://www.ietf.org/mail-archive/web/avt/current/

[Ens98] J. Enssle: Modellierung und Leistungsuntersuchung eines verteilten Video-On-Demand-Systems für MPEG-codierte Videodatenströme mit variabler Bitrate, 68. Bericht über verkehrstheoretische Arbeiten, Universität Stuttgart, 1998

[EU96] EU Esprit-Projekt 23191: MPEG-4 PC – MPEG-4 System Implementation and Tools for PC, Nov. 96 – Aug. 99

[EU97] EU Esprit Project 23191: Deliverable D5.1 – Software Architecture, project report, 1997

[EU98a] EU Esprit Project 23191: Deliverable D7.1 – Specification of Scenario, project report, 1998

[EU98b] EU Esprit Project 23191: IDeliverable D5.1a – Intermediate Deliverable on the Status of WP5 Developments, project report, 1998

[EU98c] EU Esprit Project 23191: Technical Annex, 1998

[EU98d] EU Esprit Project 23191: Deliverable D5.1a – Network architecture for broadcast scenario, project report, 1998

[EU98e] EU ACTS Project COMIQS: Deliverable – Choice of network architecture, public project report, 1998

[EU99] M. Stepping: EU Esprit-Projekt 23191 Deliverable 5.6a: Streams management tools for broadcast scenario, FernUni Hagen, 1999

[GBL+98] J. D. Gibson, T. Berger, T. Lookabaugh, D. Lindbergh, R. L. Baker: Digital Compression for Multimedia – Principles & Standards, Morgan Kaufmann Publishers, San Francisco, 1998

[GHJV95] E. Gamma, R. Helm, R. Johnson, J. Vlissides: Design Patterns, Elements of Reusable Object-Oriented Software, Addison Wesley Longman, Massachusetts, 1995

[GP98] E. Gelenbe und G. Pujole: Introduction to Queueing Networks, John Wiley & Sons, Chichester, England, 1987, 1998

[Gra97] R. Grabowski: The Web Publisher's Illustrated Quick Reference - Covers HTML 3.2 & VRML 2.0, Springer-Verlag Berlin, 1996

[Gro01] C. Grosch: Über Sicherheit und Anonymität in Multicastumgebungen, Dissertation, FernUniversität Hagen, Fachbereich Elektrotechnik, Berichte aus der Kommunikationstechnik, Band 9, Shaker-Verlag, Aachen, 2001

[H261] ITU-T: Line Transmission of Non-Telephone Signals – Video Codec For Audiovisual Services at p * 64 kbit/s, ITU-T Rec. H.261, 1993

[H263] ITU-T: Video coding for low bitrate communication, ITU-T Recommendation H.263, Vers. 1, 1995, Vers. 2 (H.263+), 1998, Vers. 3 (H.263++), 2000

[H323] ITU-T: Visual telephone systems and equipment for local area networks which provide a non guaranteed quality of service, ITU-T Rec. H.323, 1996

[Her98] C. Herpel: Elementary Stream Management in MPEG-4, IEEE Trans. on Circuits and Systems for Video Technology, Seiten 315 - 324, 1998.

[Hod98] P. Hoddie: Comments on MP4 Intermedia Format VM, ISO/IEC JTC1/SC29/WG11, M3858: July 1998

[HS+98] P. Hoddie und D. Singer et. al.: The QuickTime File Format as the Basis for MPEG-4 Intermedia Format, ISO/IEC JTC1/SC29/WG11, M2980, 1998

[HW96] J. Hartman und J. Wernecke: The VRML 2.0 Handbook, Addison Wesley, 1996

[IGMPv1] S. Deering: Host Extensions for IP Multicasting, RFC 1112, Internet Engineering Task Force, 1989, (Stichwort: IGMP)

[IGMPv2] W. Fenner: Internet Group Management Protocol, Version 2, RFC 2236, Updates RFC 1112, Internet Engineering Task Force, 1997

[IGMPv3] B. Cain, S. Deering, I. Kouvelas, B. Fenner und A. Thyagarajan: Internet Group Management Protocol, Version 3, RFC 3376, Internet Engineering Task Force, Oktober 2002

[ISO93] ISO/IEC 2382-26: Information technology - Vocabulary - Part 26: Open systems interconnection, ISO/IEC, 1993

[ISO94] ISO/IEC 7498-1: Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model, ISO/IEC, 1994

[ISO97] ISO/IEC 7498-3: Information technology - Open Systems Interconnection - Basic Reference Model: Naming and addressing, ISO/IEC, 1997

[ITG98] ITG: ITG 5.2-03 - Nachrichtenverkehrstheorie - Begriffe, ITG Fachgruppe 5.2.1, VDE-Verlag, Berlin, Offenbach, 1998

[JCJ+92] I. Jacobson, M. Christerson, P. Jonsson, G. Övergaard: Object-Oriented Software Engineering – A Use Case Driven Approach, Addison Wesley, Wokingham, 1992

[JPEG] ISO/IEC 10918-1:1994 und ITU-T T.81: Information technology - Digital compression and coding of continuous-tone still images, Part 1-4, ISO/IEC JTC1/SC29/WG1 und ITU-T, 1994

[JPG2000] ITU-T and ISO/IEC: Information technology - JPEG 2000 image coding system, Part 1-11, ISO/IEC 15444, ITU-T Recommendation, T.800, 2002

[Kad91] F. Kaderali: Digitale Kommunikationstechnik I, Vieweg-Verlag, Braunschweig, 1991

[Kad95] F. Kaderali: Digitale Kommunikationstechnik II, Vieweg-Verlag, Braunschweig, 1995

[Kad03] F. Kaderali: Vorlesungsskript Kommunikationsprotokolle, Kurs-Nr. 02552, FernUniversität Hagen, 1995, 2003

[KBS+00] F. Kaderali, T. Bonse, M. Stepping und G. Steinkamp: MPEG-4 - der neue Austauschstandard für Autorensysteme in der Fernlehre der Virtuellen Universität?, Vernetztes Lernen mit digitalen Medien, Poster, Konferenz D-CSCL, 23.-24.03.2000, Darmstadt

[KF99] M. W. Koyabe, G. Fairhurst, Wide Area Multicast Internet Access via Satellite, presented at the 5th European Conference on Satellite Communications (ECSC), Toulouse, France, 03.-05.11.1999.

[KF01] M. W. Koyabe, G. Fairhurst, RELIABLE MULTICAST VIA SATELLITE: A Comparison Survey and Taxonomy, International Journal of Satellite Communications (IJSC), Volume 19, No 1, Seiten 3 - 28, Januar/Februar 2001.

[Kir97] C. Kirsch: Programm-Aktualisierung per Netz, Verlag Heinz Heise, iX, Ausgabe 8, Seite 96 ff, 1997

[Kle76a] L. Kleinrock: Queueing Systems, Volume 1, Theory, John Wiley & Sons, 1976

[Kle76b] L. Kleinrock: Queueing Systems, Volume 2, Computer Applications, John Wiley & Sons, 1976

[KlGa96] L. Kleinrock and R. Gail: Queueing Systems, Problems and Solutions, John Wiley & Sons, 1996

[Koe97] R. Koenen: Overview of the MPEG-4 Standard, ISO/IEC JTC1/SC29/WG11, N1730, Stockholm, 1997

[Koe01] R. Koenen: Overview of the MPEG-4 Standard, ISO/IEC JTC1/SC29/WG11, N2725, 2001

[KP94] F. Kaderali, W. Poguntke: Graphen, Algorithmen und Netze: Grundlagen und Anwendungen in der Nachrichtentechnik, Vieweg-Verlag, 1995

[KPC97] R. Koenen, F. Pereira, and L. Chiariglione: MPEG-4: Context and Objectives, Signal Processing: Image Communication, Special Issue on MPEG-4, Vol. 9, Nr. 4, Seiten 295 - 304, Elsevier Science, 1997

[Lei00] F. von Leitner: Die Kunst des Weglassens - Grundlagen der Audio-Kompression, Verlag Heinz Heise, c't, Ausgabe 3, Seiten 130 ff, 2000

[Lei00a] F. von Leitner: Multicast: Sendeverfahren für Audio und Video im Netz, Verlag Heinz Heise, c't, Ausgabe 12, Seiten 212 ff, 2000

[Lin92] G. Lin: Simulative Leistungsuntersuchung des D-Kanal-Protokolls an den ISDN-Benutzer-Netz-Schnittstellen, Dissertation, FernUniversität Hagen, Fachbereich Elektrotechnik, 1992

[Lor00] M. Lorenz: Traffic Control mit Linux, Verlag Heinz Heise, iX, Ausgabe April, Seite 134 ff, 2000

[LS00] S. Latrou and I. Stavrakakis: A Dynamic Regulation and Scheduling Scheme for Real-Time Traffic Management, IEEE/ACM Transactions on Networking, Vol 8, No 1, Seiten 60 - 70, Februar 2000

[MADCAP] S. Hannah, B. Patel and M. Shah: Multicast Address Dynamic Client Allocation Protocol (MADCAP), RFC 2730, Internet Engineering Task Force, 1999

[May98] G. Mayer-Schwarzenberger: MPEG, JPEG & Co.: Wege der Datenreduktion, Heidelberg, Hüthig-Verlag, 1998

[Meh00] F. Mehdizadeh: Entwicklung eines Mappings von MPEG-4 DMIF-Signalisierungsmeldungen auf das Real-Time Transport Protocol RTP und Erstellung eines Datentransportmechanismus, Diplomarbeit Universität Dortmund in Zusammenarbeit mit FernUniversität in Hagen, Universität Dortmund, 2000

[Mey98] D. Meyer: Administratively Scoped IP Multicast, RFC 2365, Internet Engineering Task Force, 1998

[MJPG] ITU-T and ISO/IEC: Information technology - JPEG 2000 image coding system - Part 3: Motion JPEG 2000, Part 3, ISO/IEC 15444, ITU-T Recommendation, T.800, 2001

[MPEG1] ISO/IEC 11172:1993: Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s, Part 1-5, 1993

[MPEG2] ITU-T and ISO/IEC JTC1: Generic coding of moving pictures and associated audio information - Part 1-10, ITU-T Rec. H.262 - ISO/IEC 13818, 1994, 2000

[MPEG2-1] ITU-T and ISO/IEC JTC1: Generic coding of moving pictures and associated audio information - Part 1: Systems, ITU-T Rec. H.262 - ISO/IEC 13818-1, 1994, 2000

[MPEG2-2] ITU-T and ISO/IEC JTC1: Generic coding of moving pictures and associated audio information - Part 2: Video, ITU-T Rec. H.262 - ISO/IEC 13818-2, 1994

[MPEG2-6] ITU-T and ISO/IEC JTC1: Generic coding of moving pictures and associated audio information - Part 6: Digital Storage Media – Command and Control, ITU-T Rec. H.262 - ISO/IEC 13818-2, 1994

[MPEG4] ISO/IEC IS 14496: Information Technology – Generic Coding of Audio - Visual Objects, Part 1 - 6, ISO/IEC JTC1/SC29/WG11, 1999, 2000

[MPEG4-1] ISO/IEC IS 14496-1: Information technology – Generic Coding of Audio-Visual Objects – Part 1: Systems, International Standard, ISO/IEC JTC1/SC29/WG11, N2601, Version 1, 1998

[MPEG4-1v3] ISO/IEC IS 14496-1: Information technology – Generic Coding of Audio-Visual Objects – Part 1: Systems, International Standard, ISO/IEC JTC1/SC29/WG11, N5277, Version 3, 2002

[MPEG4-2] ISO/IEC IS 14496-2: Information technology – Generic coding of moving pictures and associated audio information – Part 2: Visual, International Standard, ISO/IEC JTC1/-SC29/WG11, N2602, Version 1, 1998

[MPEG4-2v2] ISO/IEC IS 14496-2: Information technology – Generic coding of moving pictures and associated audio information – Part 2: Visual, International Standard, ISO/IEC JTC1/SC29/WG11, N4350, Version 2, 2001

[MPEG4-3] ISO/IEC IS 14496-3: Information technology – Generic coding of moving pictures and associated audio information – Part 3: Audio, International Standard, ISO/IEC JTC1/-SC29/WG11, N2603, 1998

[MPEG4-5] ISO/IEC IS 14496-5: Information technology – Generic coding of moving pictures and associated audio information – Part 5: Reference Software, International Standard, ISO/IEC JTC1/SC29/WG11, 1998 – 2003, kontinuierliche Entwicklung

[MPEG4-6v1] ISO/IEC IS 14496-6: Information technology – Generic coding of moving pictures and associated audio information – Part 6: Delivery Multimedia Integration Framework, International Standard, ISO/IEC JTC1/SC29/WG11, N2606, Version 1, 1998

[MPEG4-6] ISO/IEC IS 14496-6: Information technology – Generic coding of moving pictures and associated audio information – Part 6: Delivery Multimedia Integration Framework, International Standard, ISO/IEC JTC1/SC29/WG11, N3713, Version 2, 2000

[MPEG4-8] ISO/IEC IS 14496-8: Information technology – Generic coding of moving pictures and associated audio information – Part 8: MPEG-4 on IP Framework, International Standard, ISO/IEC JTC1/SC29/WG11, N4172, 2002

[MSDN] Microsoft: Windows NT 4.0 DDK, Microsoft Developer Network Library Edition, Microsoft 1998, 2004

[MSDP] B. Fenner, D. Meyer: Multicast Source Discovery Routing Protocol, RFC 3618, Internet Engineering Task Force, 2003, (Stichwort MSDP)

[NAAM03] A. Nafaa, T. Ahmed, Y. H. Aoul, A. Mehaoua: RTP4MUX: A Novel MPEG-4 RTP Payload For Multicast Video Communications Over Wireless IP, Packet Video 2003 Konferenz, 28.-29.04.2003, Nantes, Frankreich, http://www.polytech.univ-nantes.fr/pv2003/-papers/pv/html/abstract/a1014.htm

[Oes98] B. Oestereich: Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified Modeling Language, R. Oldenbourg, München, Wien, 1998

[Paw98a] D. Pawson: Analysis of MP4 Sample Table for Random Access, ISO/IEC JTC1/SC29/-WG11, M3612, 1998

[Paw98b] D. Pawson: Proposed Glossary for M4F VM, ISO/IEC JTC1/SC29/WG11, M3851, 1998

[Paw98c] D. Pawson: Amended MP4 text with regard to FlexMux, ISO/IEC JTC1/SC29/WG11, M3852, 1998

[PE98] A. Puri and A. Eleftheriadis: MPEG-4: An Object-Based Multimedia Coding Standard Supporting Mobile Application, ACM Mobile Networks and Applications Journal, Volume 3, Issue 1, Seiten 5 - 32, 1998

[PE02] F. Pereira (Hrsg.), T. Ebrahimi (Hrsg.): The MPEG-4 Book, Prentice-Hall, New Jersey, 2002

[Per96] F. Pereira: MPEG-4: a New Challenge for the Representation of Audio-Visual Information, Picture Coding Symposium 96, Seiten 447 - 451, Melbourne, Australia, 1996

[PIM-DM] A. Adams, J. Nicholas, W. Siadak: Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised), Internet Draft draft-ietf-pim-dm-new-v2-05.txt, work in progress, Internet Engineering Task Force, Juni 2004

[PIM-SM] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, L. Wei: Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification, RFC 2117, 1997, RFC 2362, 1998

[PK96] F. Pereira, and R. Koenen: Very Low Bitrate Audio-Visual Applications, Signal Processing: Image Communication, Vol. 9, Nr. 1, Seiten 55 - 77, November 1996

[Pos80] J. Postel: User Datagram Protocol, RFC 768, Internet Engineering Task Force, 1980, (Stichwort UDP)

[Pos81]] J. Postel: Internet Protocol, RFC 791, Internet Engineering Task Force, 1981, (Stichwort IP, IPv4)

[Pos81] J. Postel: Transmission Control Protocol, RFC 793, Internet Engineering Task Force, 1981, (Stichwort TCP)

[Rob94] T. G. Robertazzi: Computer Networks and Systems, Springer-Verlag, New York, 1990, 1994

[Rei97] U. Reimers: Digitale Fernsehtechnik – Datenkompression und Übertragung für DVB – Digital Video Broadcasting, Springer-Verlag, Berlin, 1995, 1997

[RBP+91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy und W. Lorensen: Object-Oriented Modeling und Design, Prentice Hall, Englewood Cliffs, 1991

[RFC1075] D. Waitzmann, C. Partridge, S. Deering: Distance Vector Multicast Routing Protocol, RFC 1075, Internet Engineering Task Force, 1988, (Stichwort: DVMRP)

[RFC1129] D. Mills: Internet Time Synchronization: the Network Time Protocol, RFC 11129, Internet Engineering Task Force, 1989

[RFC1305] D. Mills: Network Time Protocol (Version 3) Specification and Implementation, RFC 1305, Internet Engineering Task Force, 1992, (Stichwort: NTP)

[RFC1584] J. Moy: Multicast Extension to OSPF, RFC 1584, Internet Engineering Task Force, 1994, (Stichwort: MOSPF)

[RFC1771] Y. Rekhter, T. Li: A Border Gateway Protocol 4 (BGP-4), RFC 1771, Internet Engineering Task Force, 1995, (Stichwort BGP)

[RFC1819] L. Delgrossi, L. Berger: Internet Stream Protocol Version 2 (ST2) Protocol Specification Version ST2+, RFC 1819, Internet Engineering Task Force, 1995, (Stichwort ST2)

[RFC2047] K. Moore: Multipurpose Internet Mail Extensions (MIME), Part Three: Representation of Non-ASCII Text in Internet Message Headers, RFC 2047, Internet Engineering Task Force, 1996

[RFC2201] A. Ballardie: Core Based Trees (CBT) Multicast Routing Architecture, RFC 2201, Internet Engineering Task Force, 1997

[RFC2234] D. Crocker, P. Overell: Augmented BNF for Syntax Specifications: ABNF, RFC 2234, Internet Engineering Task Force, 1997, (Stichwort ABNF)

[RFC2250] D. Hoffman, G. Fernando, V. Goyal, M. Civanlar: RTP Payload Format for MPEG1/-MPEG2 Video, RFC 2250, Internet Engineering Task Force,1998

[RFC2283] T. Bates, R. Chandra, D. Katz, Y. Rekhter: Multiprotocol Extensions for BGP-4, RFC 2283, Internet Engineering Task Force, 1998, (Stichwort MBGP)

[RFC2309] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, L. Zhang: Recommendations on Queue Management and Congestion Avoidance in the Internet, RFC 2309, Internet Engineering Task Force, 1998

[RFC2326] H. Schulzrinne, A. Rao, R. Lanphier: Real Time Streaming Protocol (RTSP), RFC 2326, Internet Engineering Task Force, 1998, (Stichwort RTSP)

[RFC2453] G. Malking: RIP Version 2, RFC 1388, RFC 1723, RFC 2453, Internet Engineering Task Force, 1998, (Stichwort RIP)

[RFC2365] D. Meyer: Administratively Scoped IP Multicast, RFC 2365, Internet Engineering Task Force, 1998

[RFC3569] S. Bhattacharyya: An Overview of Source-Specific Multicast (SSM), RFC 3569, Internet Engineering Task Force, 2003, (Stichwort SSM)

[Ric85] L. Richter: Betriebssysteme, B.G. Teubner, Stuttgart, 1977, 1985

[Ric99] J. Richter: Programming Applications for Microsoft Windows, Microsoft Press, Redmond, 4. Auflage, 1999

[RJB95] J. Rumbaugh, I. Jacobson und G. Booch: The Unified Modelling Language for Object Oriented Development, Rational Software Corporation, Santa Clara, 1995, (Stichwort: UML)

[Roe99] P. Roer: Verkehrssteuerung in ATM-Netzen – Untersuchung des Worst Case-Verkehrs GCRA-konformer Zellströme, Dissertation, FernUniversität Hagen, Fachbereich Elektrotechnik, Berichte aus der Kommunikationstechnik, Band 2, Shaker-Verlag, 1999

[RTP96] H. Schulzrinne, S. Casner, R. Frederick und V. Jacobsen: RTP: A Transport Protocol for Real-Time Applications, RFC 1889, Internet Engineering Task Force, 1996, (Stichwort: RTP, RTCP)

[RTP] H. Schulzrinne, S. Casner, R. Frederick und V. Jacobsen: RTP: A Transport Protocol for Real-Time Applications, RFC 3550, Internet Engineering Task Force, 2003, ersetzt RFC 1889 [RTP96].

[RTPa96] H. Schulzrinne: RTP Profile for Audio and Video Conferences with Minimal Control, RFC 1890, Internet Engineering Task Force, 1996

[RTPa] H. Schulzrinne und S. Casner: RTP Profile for Audio and Video Conferences with Minimal Control, RFC 3551, Internet Engineering Task Force, 2003, ersetzt RFC 1890 [RTPa96]

[RTP03] J. van der Meer, D. Mackie, V. Swaminathan, D. Singer, P. Gentric: RTP Payload Format for Transport of MPEG-4 Elementary Streams, RFC 3640, Internet Engineering Task Force, November 2003

[SAP] M. Handley, C. Perkins, E. Whelan: Session Announcement Protocol, RFC 2974, Internet Engineering Task Force, Version 2, 2000. Version 1 wurde nicht RFC. (Stichwort SAP)

[SB02] M. Stepping und C. Bechter: MPEG-4 In Mobile Environments, Proceedings on Globalization, Innovation and Human Resource Development for Competitive Advantage, Vol. 1, pp. 477-487, Bangkok, 2002

[Sch95] Klaus Schroiff: Multimedia-Datenformate, zu beziehen über http://i31www.ira.uka.de/-semin94/06_MPEG/main_html.html

[Sch96] H. Schulzrinne: RTP Profile for Audio and Video Conferences with Minimal Control, RFC 1890, Internet Engineering Task Force, 1996

[SDP] M. Handley und V. Jacobson: SDP: Session Description Protocol, RFC 2327, Internet Engineering Task Force, 1998

[SDP03] M. Handley, V. Jacobson, C. Perkins: SDP: Session Description Protocol, Internet draft: draft-ietf-mmusic-sdp-new-15.txt, Internet Engineering Task Force, work in progress, 2003

[Sin98] D. Singer: Contact points between systems and elementary streams, ISO/IEC JTC1/-SC29/WG11, M3552, 1998

[SIP] M. Handley, H. Schulzrinne, E. Schooler und J. Rosenberg: SIP: Session Initiation Protocol, RFC 2543, Internet Engineering Task Force, 1999

[Sis00] D. Sisalem: TCP-Friendly Congestion Control for Multimedia Communication in the Internet, Dissertation, Technische Universität Berlin, Springer-Verlag, 2000

[Som01] D. Sommer: Zur Prädiktion variabler Videobitströme für deren optimierte Übertragung in paketvermittelnden Netzen, Dissertation, FernUniversität Hagen, Fachbereich Elektrotechnik, Berichte aus der Kommunikationstechnik, Band 8, Shaker-Verlag, 2001

[SS98] D. Singer und M. Speer: Proposed Revised Intermedia Format (M4F) VM text, ISO/IEC JTC1/SC29/WG11, M3319, 1998

[SSW03] K. Sühring, H. Schwarz, T. Wiegand: Effizienter kodieren - Details zum kommenden Videostandard H.264/AVC, Verlag Heinz Heise, c't, Ausgabe 6, Seiten 266 ff., 2003

[Sta90] W. Stallings: Handbook of Computer Communication Standards - Local Area Network Standards, Howard W. Sams and Company, 1990

[Ste00] M. Stepping: Betriebssysteme, F. Kaderali (Hrsg.), Band Unternehmensnetze, Seiten 8.3.1-1 – 8.3.1-13, DWD-Verlag, Köln 2000

[Ste00a] M. Stepping: Broadcast-DMIF (DNI-Messages and Broadcast Server), ISO/IEC JTC1/-SC29/WG11, M5797, März 2000

[Ste00b] M. Stepping: SL-PDU fragmenting over RTP, ISO/IEC JTC1/SC29/ WG11, Beitrag M5798, März 2000

[Ste02] M. Stepping: E-Learning Content Creation with MPEG-4, Advances in Infrastructure for e-Business, e-Education, e-Science and e-Medicine on the Internet in L'Aquila, Italien, 2002

[Ste93] M. Stepping: Bereitstellung der SINEC-H1 Funktionalität (OSI-Transport-Protokoll) auf der Basis einer handelsüblichen Ethernet-Netzwerkkarte unter dem Betriebssystem Microsoft Windows NT, Diplomarbeit, Universität Siegen, 1993

[Ste+98] M. Stepping, et al.: DMIF Application Interface: Syntax Definition, ISO/IEC JTC1/ SC29/WG11 Beitrag M4182, 1998

[Ste98] M. Stepping: DAI Syntax definition: Source files, C++, Java, ISO/IEC JTC1/SC29/ WG11 Beitrag M4191, 1998

[Ste99a] M. Stepping: DAI Syntax definition - several additions V1/V2, ISO/IEC JTC1/SC29/-WG11 Beitrag M4507, 1999

[Ste99b] M. Stepping: DMIF Group and Broadcast Signalling: comments and DNI messages, ISO/IEC JTC1/SC29/WG11 Beitrag M4508, 1999

[Ste99c] M. Stepping: Monitoring DMIF, ISO/IEC JTC1/SC29/WG11, Beitrag M4509, 1999

[Ste99d] M. Stepping: Rechnergestütztes Telefonieren – Computer Telefonie, F. Kaderali (Hrsg.), Band Unternehmensnetze, Seiten 9.1-1 – 9.1-17, DWD-Verlag, Köln, 1999

[Ste99e] M. Stepping: Netzwerkkarten, F. Kaderali (Hrsg.), Band Unternehmensnetze, Seiten 8.2.2-1 – 8.2.2-6, DWD-Verlag, Köln, 1999

[Ste99f] M. Stepping: Prinzipieller Aufbau von Betriebssystemen, F. Kaderali (Hrsg.), Band Unternehmensnetze, Seiten 8.1.2-1 – 8.1.2-15, DWD-Verlag, Köln 1999

[Stm99] R. Steinmetz: Multimediatechnologie, Springer-Verlag, 2. Auflage, 1999

[Str92] B. Stroustrup: Die C++ Programmiersprache, Addison Wesley Deutschland, 2. Auflage, 4. Überarbeitung, 1992, 1994

[Str94] B. Stroustrup: Design und Entwicklung von C++, Addison Wesley Deutschland, 1994

[SW49] C. Shannon, W. Weaver: The Mathematical Theory of Communication, Urbana, Ill., 1949

[Tan90] A. Tanenbaum: Computer-Netzwerke, Pearson-Studium, München, 3. Auflage, 2000

[Tan90a] A. Tanenbaum: Betriebssysteme – Entwurf und Realisierung, Teil 1, Prentice-Hall, München, Wien, London, 1990

[Tan95] A. Tanenbaum: Distributed Operating Systems, Prentice-Hall, New Jersey, 1995

[VRML97] ISO/IEC 14772-1:1997: Information technology - Computer graphics and image processing - The Virtual Reality Modelling Language (VRML) - Part 1: Functional specification and UTF-8 encoding, ISO/IEC, 1997

[WZ99] R. Wittmann, M. Zitterbart: Multicast: Protokolle und Anwendungen, dpunkt-Verlag, Heidelberg, 1999

[X.208] IS 8824 Information Processing Systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1) (ebenfalls als ITU-T Rec. X.208 veröffentlicht)

[X.209] IS 8825 Information Processing Systems - Open Systems Interconnection - Specification of Basic Encoding Rules for ASN.1 (ebenfalls als ITU-T Rec. X.209 veröffentlicht)

[Zer99] K. Zerbe: Bauplan für Objekte - Eine Einführung in objektorientierte Verfahren mit der Unified Modelling Language, Verlag Heinz Heise, c't, Ausgabe 21, Seiten 338 ff,1999

[Zit95] T. Braun, M. Zitterbart: Hochleistungskommunikation, Band1: Technologie und Netze, Oldenburg-Verlag, München, Wien, 1995

[Zit96] T. Braun, M. Zitterbart: Hochleistungskommunikation II: Transportdienste und -protokolle, Oldenburg-Verlag, 1996

Eigene Veröffentlichungen

[BS98] T. Bonse, M. Stepping, F. Casalino, M. Quaglia, G. Franceschini: MPEG-4 Systems, concepts and implementation, ECMAST Konferenz, 26.05.-28.05.1998, Berlin, Seiten 504 - 517, Springer-Verlag, Heidelberg

[BS99a] T. Bonse und M. Stepping: Broadcast Multimediaübertragung auf der Basis von MPEG-4, 8. Dortmunder Multimediaseminar der FKTG, 27.-29.09.1999, Dortmund

[BS99b] T. Bonse und M. Stepping: MPEG-4 PC - Authoring and Playing of MPEG-4 Content, EMMSEC Konferenz, Proceedings on Business and Work in the Information Society: New Technologies and Applications, 21.-23.06.1999, Stockholm, Schweden, Cheshire Henbury Verlag, Großbritannien

[BS99c] T. Bonse, M. Stepping, P. Gerken, S. Schultz, G. Knabe, F. Casalino, G. Di Cagno, M. Quaglia, J.-C. Dufourd, S. Boughoufalah, F. Bouilhaguet, U. Mayer, J. Deicke, M. Glesner: MPEG-4 PC - Authoring and Playing of MPEG-4 Content for Local and Broadcast Applications, ECMAST Konferenz, Mai 1999, Madrid, Spanien, Seiten 108 - 119, Springer-Verlag, Heidelberg

[EU99] M. Stepping: EU Esprit-Projekt 23191 Deliverable 5.6a: Streams management tools for broadcast scenario, FernUniversität in Hagen, 1999

[Ste99d] M. Stepping: Rechnergestütztes Telefonieren – Computer Telefonie, F. Kaderali (Hrsg.), Band Unternehmensnetze, Seiten 9.1-1 – 9.1-17, DWD-Verlag, Köln, 1999

[Ste99e] M. Stepping: Netzwerkkarten, F. Kaderali (Hrsg.), Band Unternehmensnetze, Seiten 8.2.2-1 – 8.2.2-6, DWD-Verlag, Köln, 1999

[Ste99f] M. Stepping: Prinzipieller Aufbau von Betriebssystemen, F. Kaderali (Hrsg.), Band Unternehmensnetze, Seiten 8.1.2-1 – 8.1.2-15, DWD-Verlag, Köln 1999

[Ste00] M. Stepping: Betriebssysteme, F. Kaderali (Hrsg.), Band Unternehmensnetze, Seiten 8.3.1-1 – 8.3.1-13, DWD-Verlag, Köln 2000

[BS00] T. Bonse und M. Stepping: MPEG-4 - Neue Nutzungspotentiale in der Fernlehre, FERNSEH- und KINO-TECHNIK, 54. Jahrgang Nr. 1-2/2000, Seiten 43-51, Hüthig-Verlag, Heidelberg

[KBS+00] F. Kaderali, T. Bonse, M. Stepping und G. Steinkamp: MPEG-4 - der neue Austauschstandard für Autorensysteme in der Fernlehre der Virtuellen Universität?, Vernetztes Lernen mit digitalen Medien, Poster, Konferenz D-CSCL, 23.-24.03.2000, Darmstadt

[Ste02] M. Stepping: E-Learning Content Creation with MPEG-4, Advances in Infrastructure for e-Business, e-Education, e-Science and e-Medicine on the Internet in L'Aquila, Italien, 2002

[SB02] M. Stepping und C. Bechter: MPEG-4 In Mobile Environments, Proceedings on Globalization, Innovation and Human Resource Development for Competitive Advantage, Vol. 1, pp. 477-487, Bangkok, 2002

Beiträge zur MPEG-Standardisierung

[Ste+98] M. Stepping, et al.: DMIF Application Interface: Syntax Definition, ISO/IEC JTC1/-SC29/WG11 Beitrag M4182, 1998

[Ste98] M. Stepping: DAI Syntax definition: Source files, C++, Java, ISO/IEC JTC1/SC29/-WG11 Beitrag M4191, 1998

[Ste99a] M. Stepping: DAI Syntax definition - several additions V1/V2, ISO/IEC JTC1/SC29/-WG11 Beitrag M4507, 1999

[Ste99b] M. Stepping: DMIF Group and Broadcast Signalling: comments and DNI messages, ISO/IEC JTC1/SC29/WG11 Beitrag M4508, 1999

[Ste99c] M. Stepping: Monitoring DMIF, ISO/IEC JTC1/SC29/WG11, Beitrag M4509, 1999

[EU99] M. Stepping: EU Esprit-Projekt 23191 Deliverable 5.6a: Streams management tools for broadcast scenario, FernUni Hagen, 1999

[Ste00a] M. Stepping: Broadcast-DMIF (DNI-Messages and Broadcast Server), ISO/IEC JTC1/-SC29/WG11, M5797, März 2000

[Ste00b] M. Stepping: SL-PDU fragmenting over RTP, ISO/IEC JTC1/SC29/ WG11, Beitrag M5798, März 2000

Betreute Diplomarbeiten

[Meh00] F. Mehdizadeh: Entwicklung eines Mappings von MPEG-4 DMIF-Signalisierungsmeldungen auf das Real-Time Transport Protocol RTP und Erstellung eines Datentransportmechanismus, Diplomarbeit Universität Dortmund in Zusammenarbeit mit FernUniversität in Hagen, Universität Dortmund, 2000

[Das00] M. Dashti: Entwicklung eines Mappings von MPEG-4 DMIF-Signalisierungsmeldungen auf IP-Multicast und Erstellung eines Datentransportmechanismus mittels IP-Multicast, Diplomarbeit Universität Dortmund in Zusammenarbeit mit FernUniversität in Hagen, Universität Dortmund, 2000

[Jah00] H. Jahn: Vergleich von Warteschlangen- und Puffermodellen unter dem spezifischen Aspekt verschiedenartiger MPEG-4 Multimedia-Datenströme und Entwicklung eines konkreten Puffermodells innerhalb des Broadcast MPEG-4 Delivery DMIF über IP-Multicast unter der Randbedingung einer konstanten Gesamtausgangsbandbreite (Rate Shaping), Diplomarbeit, FernUniversität Hagen, 2000

[Kar03] W. Karl: Erweiterung des Java-OfflineNavigators um authentische und vertrauliche Kommunikation, Diplomarbeit, FernUniversität Hagen, 2003

Index

Lebenslauf

Name:	Michael Stepping
Geburtsdatum:	1. Mai 1969
Geburtsort:	Hachenburg / Westerwald
Wohnort:	Gustav-von-Mevissen-Straße 28 57072 Siegen
Familienstand:	verheiratet, 2 Kinder
Schulbesuch:	1975 - 1979 Grundschule Wilnsdorf 1979 - 1985 Realschule Wilnsdorf 1985 - 1988 Gymnasium Am Giersberg, Siegen
Studium:	1988 - 1993 Universität -GH- Siegen, Fachbereich Elektrotechnik Fachrichtung Technische Datenverarbeitung
Stipendium:	1991 - 1993 Siemens AG, Köln
Diplomarbeit:	In Zusammenarbeit mit Siemens AG, Köln: „Bereitstellung der SINEC-H1 Funktionalität (OSI-Transport-Protokoll) auf der Basis einer handelsüblichen Ethernet-Netzwerkkarte unter dem Betriebssystem Microsoft Windows NT“.
	06.08.1993 Diplom-Ingenieur der Elektrotechnik
Werdegang:	1993 - 1997 Firma Quintec Gesellschaft für Datentechnik mbH, Overath, Systementwickler
	1997 - 2003 FernUniversität in Hagen, Kommunikationssysteme wissenschaftlicher Mitarbeiter
Dissertation:	01.10.2004 Ein Beitrag zu MPEG-4 Broadcast

Berichte aus der Kommunikationstechnik

Herausgeber: Firoz Kaderali

Lehrgebiet Kommunikationssysteme, FernUniversität in Hagen

Bisher sind in dieser Reihe folgende Bände erschienen:

	Helge Winterstein	Simulation zur Leistungsanalyse und Optimierung des Token-Bus-Protokolls
	Guochun Lin	Simulative Leistungsuntersuchung des D-Kanal-Protokolls an den ISDN Benutzer-Netz-Schnittstellen
	Johannes Wissing	Dynamisches Bandbreitenmanagement am ATM-Netzzugang für burstartige, verzögerungstolerante Datenströme
	Christoph Bach	Leistungsbewertung und Optimierung von zellbasierten Medienzugriffsverfahren für lokale Gbit/s-Netze
	Andreas Grebe	Reservierungsverfahren in regionalen Gbit/s-Netzen – Leistungsbewertung und Optimierung
	Burkhard Heyber	Zur Güte von Zufallsprozessen in der kryptologischen Praxis
	Thomas Hermann	Vergleichende Bewertung von Verfahren zur Benutzerauthentifikation
Band 1	Andreas Rieke	Die Anwendung von Verschlüsselungsverfahren in ATM-Systemen
Band 2	Peter Roer	Verkehrssteuerung in ATM-Netzen
Band 3	Kyamakya Kyandoghere	Issues in Telecommunication Networks Resiliency Analysis and Design Concepts
Band 4	Markus Schneider	Über Methoden der Generierung binärer Pseudozufallsfolgen zur Stromverschlüsselung
Band 5	F. Kaderali (Hrsg.)	Anonymität im Internet
Band 6	Dirk Westhoff	Ein dezentrales Agentensystem unter Berücksichtigung von mehrseitiger Sicherheit
Band 7	Gerd Bauer	Optimierung von statischen Routingverfahren in speziellen Graphenklassen
Band 8	Dagmar Sommer	Zur Prädiktion variabler Videobitströme für deren optimierte Übertragung in paketvermittelnden Netzen
Band 9	Christian Grosch	Über Sicherheit und Anonymität in Multicastumgebungen
Band 10	Bernhard Löhlein	Entwurf und Analyse kryptographisch sicherer Keystreamgeneratoren zur Stromverschlüsselung
Band 11	Thomas Demuth	Ein Beitrag zur Anonymität in Kommunikationsnetzen
	Michael Stepping	Ein Beitrag zu MPEG-4 Broadcast