Master's Thesis

Digital TV distribution in the DLNA/DVB-HN Home Network

Martin Emrich

Submitted on	:	December 21th 2010
Supervisor	:	Manuel Gorius, M.Sc.
1st Reviewer	:	Prof. Dr.-Ing. Thorsten Herfet
2nd Reviewer	:	Prof. Dr.-Ing. Philipp Slusallek

Saarland University
Faculty of Natural Sciences and Technology I
Department of Computer Science
Master's Program in Computer Science

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement under Oath

I confirm under oath that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken,..... .....

(Datum / Date) (Unterschrift / Signature)

Acknowledgment

The presented thesis was supported by Intel GmbH in the course of the “Digital Home Compliance Lab”.

Abstract
During the previous work, a DLNA/UPnP AV compatible Media Server capable of streaming live broadcast channels (TV and Radio) via the Real-time Transport Protocol (RTP) was developed. Now, we complete this with a matching Media Player, which receives and displays RTP streams received from this Media Server (and others). We phrase several properties such a scenario should have to provide a quality of experience similar to that of legacy Set Top Boxes, in terms of user interface and channel change delay, while providing compatibility with the UPnP AV and DLNA standards.

Contents

Conventions

CPlusPlusClass	A C++ class name. These are usually very descriptive, and thus used directly in the text.
library	A library or software package name.
Media Server, Media Player	Names with capital letters refer to the entities specified by the UPnP AV or DLNA standards as well as the concrete implementations.

1 Introduction

1.1 Motivation

In the past years, home entertainment and IP networks converged more an more. Users have their media files shared by their computer, and consume them on the TV screen using networked media players. Industry standards like DLNA[1] attempt to provide an ecosystem of home media servers, mobile phones and media player appliances compatible with each other.

One constraint is still present: All these systems focus on stored content. While live broadcast has become digital quite some time ago with DVB[2], it is still handled separately using receiver set-top-boxes, usually requiring separate cabling to each device.

Integrating live broadcast into the home entertainment networks could reduce cabling issues, allow wireless receiver boxes using Wi-Fi and provide new applications like watching TV recordings anywhere in the house. Finally, only one device (which could even be a software application on a video game console) could serve all home entertainment demands.

Such a change can only be successful if the end user has the same or superior quality of experience compared to legacy digital broadcast.

1.2 Goals

The goals of this work is to provide a Media Server/Media Player pair distributing live broadcast content in the home network. The Media Server contains tuner hardware for DVB broadcast, and provides the received services on the network. The Media Player discovers, enumerates and renders that live broadcast content. A basic implementation of the Media Server was developed during the previous bachelor thesis[13]. The system shall provide a quality of experience close to that of a regular DVB set-top-box: Reliable, error resilient media transport, user operation similar to a regular TV set (sequential switching through channels) and reasonably fast channel change times.

1.3 Previous Work

This work builds upon components created during the development of the `rtpserver` [13].

During this project we focus on live broadcast TV with these property assumptions:

- Video encoded with MPEG 2 in SD resolution (720x576 pixels), 25 frames per second, interlaced, maximum bitrate of 15 Mbit/s.

[1] http://www.dlna.org
[2] http://www.dvb.org

- Audio encoded with MPEG Layer II audio, typically 192-384 kbit/s, 48kHz sample rate.
- Multiple programs, each consisting of one video and one audio track, multiplexed into in one MPEG transport stream[23].

Some concepts may also apply to high definition (HD) video encoded with H.264 or VC-1, or alternative audio codecs such as Dolby Digital or DTS.

1.4 Quality of Experience

The term "Quality of Experience" used in here describes how a multimedia home network scenario composed of multiple devices compares to a "legacy" (non-networked) scenario with a single set-top-box in terms of

- **Rendering quality** of audio and video: Should be on par with DVB-S/C/T. As the digital content is processed unchanged, this is easily attained.
- **Reliability** against transmission distortions: DVB-IPTV[17] specifies a residual error rate of 10^{-6} and an end-to-end delay of 100ms.
- **Responsiveness** to user interaction, especially channel changes.
- **Features** like multi-user capability and content description (Electronic Program Guide (EPG)).

2 Home entertainment scenarios in the industry

Several entities in the entertainment industry and the standardization organizations have described how components in the home entertainment network could work together. The most relevant standards and infrastructures are UPnP, DLNA and DVB-HN.

2.1 Universal Plug'n'Play

Defined by the UPnP Forum[8], UPnP is a protocol stack designed for networked devices to interact with each other, without the need for user configuration. A UPnP *device* usually provides one or several *services* , while a *control point* is a client using these services.

Each *device* can contain other "embedded" *devices* as well as *services* . A *service* provides service actions that can be invoked by a control point to interact with the device. The top level *device* is called a *root device* .

All the following standards are based on Universal Plug'n'Play (UPnP).

2.1.1 Protocol stack overview

The UPnP protocol stack defines

- **Addressing:** Each UPnP device requires (at least) one IPv4 address. First, the device attempts to obtain an IP address via Dynamic Host Configuration Protocol (DHCP). If this fails, automatic private IP addressing[3] is used.
- **Discovery:** UPnP uses Simple Service Discovery Protocol (SSDP) for device discovery.

  SSDP[20] is a simple, UDP-based protocol for discovering *devices* and *services* on the network. *Devices* entering the network announce their presence and that of their *services* , *devices* leaving the network also announce their intention to leave. Control points can actively probe the network for devices and services. The SSDP communication runs on the multicast channel 239.255.255.250:1900.
- **Device and Service description:** Each *device* and each of its *services* is described in an XML document, served by a built-in HTTP server.

  Each SSDP announcement message contain the corresponding description URL, while the service description URL(s) are in turn contained in the device description URL.

  The device description also contains the *device type*, whether it is a printer or maybe a multimedia server, and human-readable meta information about the device, e.g. brand, type, serial number and a friendly name of the device.

- **Control:** Each UPnP *device* provides a number of *services* , which in turn provide a number of *actions*. Service actions are invoked via SOAP, and provide the main form of interaction with an UPnP *device* .
- **Event notification:** UPnP also provides for asynchronous event notification by means of the GENA[4] architecture.

  As a HTTP based notification protocol, GENA provides a subscription-based notification via HTTP callback URLs: When a control point wants to be notified e.g. of changes in temperature from a networked thermometer, it subscribes to it. Now, the control point's notification URL is regularly posted with the current temperature value, without the control point having to poll.

2.1.2 Device Profiles

UPnP specifies dozens of *device profiles*, each describing a certain class of device with its required services and service actions. Among them are the ubiquitous Network Light Bulb (the simplest device possible), Internet Gateways[1] and, of course, home entertainment devices.

2.2 UPnP AV

The most basic home entertainment network scenario described here is the UPnP AV device architecture. UPnP AV defines (only) three device profiles: A *Media Server* that serves content, a *Media Renderer* that plays back content received from the server (not requiring any form of user interface), and finally a *Control Point*, which browses the content provided by the server and initiates playback on the renderer. These three form the basic "three box scenario" where the Control Point is a separate device. Most rendering devices on the market combine Media Renderer and Control Point, resulting in a device that both browses and renders the content. This is commonly called the "two box scenario" (See figure 2.1).

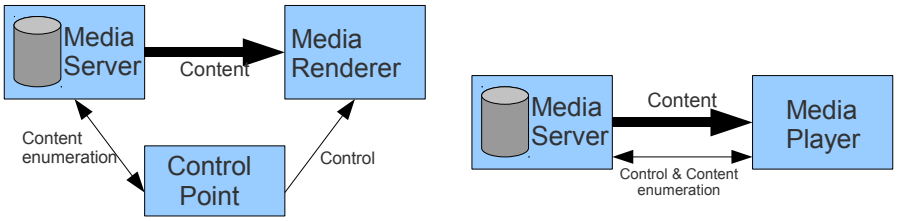

Figure 2.1: 2-Box and 3-Box home entertainment scenarios

[1] Home broadband routers which allow TCP/UDP port forwarding to arbitrary internal hosts via UPnP gave UPnP some bad press

2.2.1 UPnP Content directory

The most important *service* of a UPnP AV Media Server is the Content Directory service (CDS). It allows for browsing (and optionally searching) the available *Content Items* (representing e.g. a movie, a music track or a picture file) on the Media Server. The CDS is built hierarchically, the root *Content Container* may contain *Content Items* as well as other *Content Containers*.

Each *Content Item* has at least one *Content Resource* which basically describes a way to gain access to the item. A content Resource defines the content MIME type as well as the transport protocol used. A *Content Item* describing a broadcast TV station may contain resources with HTTP as well as with RTSP/RTP transport, or resources in different video resolutions.

Note that UPnP AV only specifies how the devices communicate, the technical content description (formats and codecs) are beyond its scope as well as a common transport protocol. Of course, e.g. a UPnP Media Renderer only accepting MPEG Layer III audio via HTTP won't work together with a Media Server providing A.52 audio via RTSP/RTP.

2.3 Digital Living Network Alliance

Dissatisfied with the interoperability problems between UPnP AV devices made by different vendors, companies in the IT and home entertainment sector formed the Digital Living Network Alliance[9]. The DLNA Networked Device Interoperability Guidelines[10] are based on UPnP AV, but define much more rigorous how the content is to be described, and how the devices have to behave.

DLNA also specifies certain content formats to be mandatory, to force a certain level of interoperability between devices made by different vendors. Furthermore, the DLNA specifies way more device classes than UPnP AV, to allow more devices to be compliant despite their limited capabilities (e.g. mobile phones).

Already today, there are thousands of different DLNA compliant devices available on the market[9].

2.3.1 Device Classes and Categories

DLNA not only distinguishes between devices, but also between Categories. While a Home Network Device (HND) is a regular device in the home network, a Mobile Handheld Device (MHD) may have different content format requirements (e.g. due to a smaller screen or lower processing power). A Home Infrastructure Device (HID) may be required to translate between these two categories. Within this thesis, only HNDs are of interest.

Device classes describe a certain system usage. The relevant device classes here are the Digital Media Server (DMS) (equivalent to an UPnP Media Server), the Digital Media Renderer (DMR) (UPnP Media Renderer), the Digital Media Player (DMP) (roughly[2] an UPnP Media Renderer with integrated Control Point) and the Digital Media Controller (DMC).

MPEG2 is the mandatory video codec, MPEG Layer 3 and LPCM are the mandatory audio formats. Optional formats include A.52 (Dolby Digital), DTS or MPEG4.

[2] A DLNA DMP does neither have to be controllable by external control points, nor discoverable by UPnP means.

2.3.2 Transports

The mandatory transport for DLNA devices is *http-get*, where the content is acquired by the client using a HTTP[19] *GET* request. Optional is the *rtsp-rtp-udp* transport, where the content transfer is negotiated via RTSP[19] and accomplished via the RTP[28] protocol. The RTP transport is explicitly recommended[10] for live content.

2.3.3 Media formats

To ensure that a DMP offers only the content items on remote servers in can actually render, DLNA introduces many different *content profiles*[11] which precisely specify the media format used by each content item, including container format, audio and video codecs, resolution etc.

As an example, the `MPEG_TS_SD_EU` profile, which is among the mandatory profiles for the European market, has the following requirements[3]:

- Container: MPEG transport stream with or without additional 4-byte timestamp field per packet
- MIME type: *video/vnd.dlna.mpeg-tts* if timestamp field is present, *video/mpeg* otherwise
- 1 single program per TS, including PAT and PMT (See section 3.3.1)
- Stream complies with European DVB specifications
- MPEG-2 video stream, main profile @ main level, 4:2:0 chroma subsampling, maximum bitrate less or equal to 15Mbit/s
- MPEG Layer I or Layer II audio with certain sample- and bit rates
- 25 frames/s interlaced (50 fields/s) video, in certain resolutions (all having the full (576 lines) or half (288 lines) PAL vertical resolution) and aspect ratios

2.3.4 DLNA Compliance tests

Besides the Interoperability Guidelines[10], DLNA provides its members with a Compliance Test Tool (CTT) to verify a device's compliance with the DLNA guidelines. The CTT has a set of automatic or manual tests for each of the *mandatory* requirement. Probably for economic reasons, the CTT currently provides no tests for the optional components in the DLNA guidelines, which includes the entire stack of RTP-transport-related rules.

Passing all tests of the CTT is of course not sufficient, each device must be certified by the DLNA before it may bear the "DLNA certified" label.

DLNA also regularly organizes "Plugfests" for its members, where members can bring their devices to test the interoperability between different vendors.

[3] The list is only exemplary and not complete

2.4 DVB-HN

European Telecommunications Standards Institute (ETSI), The issuer of the Digital Video Broadcasting (DVB) standards, is also working on the convergence between home entertainment networks and digital broadcast.

Their standard *DVB-HN*[33][16][18] intends to augment the existing DLNA infrastructure with guidelines to support broadcast TV and radio received both via separate media and via IP networks (DVB-IPTV).

2.4.1 Logical Functions

To describe the new device classes, DVB-HN introduces three *Logical Functions*:

- **HN-DP** (Discovery Point): Advertises content via the UPnP AV ContentDirectory *service* . Translates service information such as DVB-IPTV Service Discovery And Selection (SD&S, Section 7.2), Broadband Content Guide (BCG) and DVB-S/T/C Service Information (SI, , Section 7.1) data into UPnP ContentDirectory.
- **HN-SP** (Streaming Point): Streams the actual content via (unicast or multicast) RTP into the home network. Translates (if necessary) and forwards content received from the IPI-1 network or DVB-S/T/C to the home network.
- **HN-RP** (Rendering Point): Receives and renders (and/or stores) content received via RTP from the home network.

2.4.2 Device Classes

DVB-HN adds these new device classes:

- The DVB-HN compatible DLNA DMS becomes a **DVB-MS**, it implements HN-DP and HN-SP
- The DVB-HN compatible DLNA DMR becomes a **DVB-MR**, it implements HN-RP.
- TheUnidirectional Gateway Device (DVB-UGD) is a DVB-MS (DVB-MS) that also has a tuner to provide programs received via DVB-S/C/T.
- A Bidirectional Gateway Device (DVB-BGD) consists of a DVB-MS and a Delivery Network Gateway (DNG), which provides access to a DVB-IPTV[15] delivery network[4].

For DVB-IPTV content that is available via Real Time Protocol (RTP) multicast, DVB-HN[16] specifies the additional *dvb-igmp* transport. This is used to map available DVB-IPTV channels to the Content Directory. Compatible clients can join a specified multicast channel via Internet Group Management Protocol (IGMP) and receive the channel broadcast using RTP multicast directly from the service provider.

[4]Discovery and transport of content delivered via the DVB-IPTV IPI-1 interface is beyond the scope of this thesis

2.5 DVB-IPTV

As more and more homes got broadband internet access with bandwidths sufficient for digital TV distribution, ETSI started working on the DVB-IPTV standard[17] (formerly known as DVB IP infrastructure (DVB-IPI)). DVB-IPTV is not based on UPnP, but focuses on the delivery of broadcast content as well as Content-on-Demand (CoD) from the service provider to the home network end devices (Home Network End Device (HNED)s).

DVB-IPTV describes discovery and selection of the available channels (See section 7.2 and the transport of media streams via multicast RTP (for live broadcast media) or unicast RTP (for CoD). Also covered are Digital Rights Management (DRM) issues.

DVB-IPTV itself does not cover use cases *within* the home network, e.g. transfer of user-generated content from one HNED to another.

3 Basic transport protocols and formats

RTP and HTTP are the most used protocols for directly[1] transmitting multimedia content in the home network.

3.1 HTTP

From an application point-of-view, Hyper Text Transport Protocol (HTTP)[19] is the simplest protocol to use, and thus it is very widely used. HTTP is a TCP-based protocol, well known from the World Wide Web. TCP guarantees reliable, in-order transfer of data, but does not guarantee timely arrival. As long as the available bandwidth is large enough and the latency requirements low, HTTP nevertheless works well as a media transport protocol. This is why it is also the first choice in many home entertainment network standards.

3.2 RTP

In contrast, RTP[28] is an UDP based protocols designed for real-time operation. RTP provides unreliable, in-order transfer of data. This means that, in contrast to HTTP, a lost packet does not stall the transfer of data, but the receiver must be prepared for lost data packets. As RTP per se is unidirectional, it allows for multicast scenarios where one stream serves multiple receivers. Especially the multicast capability makes it interesting for live media broadcast applications.

One RTP session (identified by an UDP address/port pair) can carry different (e.g. audio or video) streams, distinguishable by a payload ID and a 32-bit Synchronization Source (SSRC), identifying a single content source (e.g. a camera. This feature is mainly used in video conferencing applications like H.323; In the live media broadcast applications described here, one single MPEG transport stream (see below) is used to multiplex different elementary streams, resulting in only one stream from the RTP point of view.

With Real Time Control Protocol (RTCP), RTP provides a companion protocol designed for the exchange of statistical information about the stream between sender and receiver. RTCP conveys information about network jitter and the amount of lost packets, allowing the sender to adapt to changing network conditions. To keep the bandwidth ratio of RTCP vs. RTP low, RTCP defines strict report intervals depending on the number of participants in the session.

Especially unicast RTP sessions are usually negotiated using another protocol called Real Time Streaming Protocol (RTSP)[29], which is in its syntax closely related to HTTP. It is used to describe, set up, control and tear down RTP sessions. The session description in all standards described herein is encoded as Session Description Protocol (SDP)[22] documents.

[1] in contrast to file sharing protocols like Samba/CIFS, NFS, etc.

Most multicast live media broadcast services over RTP are always running; clients can join these existing multicast RTP streams via IGMP. This just informs the next hop router that the requested multicast stream is required in the client's network segment.

3.3 MPEG Transport Stream

The MPEG Transport Stream (TS)[23] is the type of content container we focus on here.

A MPEG-TS is split into fixed-size packets of 188 bytes. It typically consists of multiple elementary streams, each identified by a 13-bit Program Identifier (PID) in the packet header. The small packet size of only 188 bytes makes MPEG-TS a good choice for channels prone to errors, as single lost packets only affect a small amount of transported data.

A *program* within the transport stream is a set of individual elementary streams, synchronized by a Program Clock Reference (PCR) carried on one its PIDs, forming essentially a virtual channel within the transport stream. The PCR is a 27MHz reference clock used to synchronize encoder and decoder clocks.

Due to the high number of available PIDs, the transport stream is capable of carrying many independent programs, each consisting of several elementary streams. The transport stream also carries Program Specific Information (PSI) data, describing the stream.

3.3.1 MPEG Program Specific Information

The PSI data is necessary to identify each PID carried in the stream, and to associate the elementary (audio and video) streams with each others. It consists of various tables. The most important are:

- The *Program Association Table* (PAT) on PID 0x0 lists the programs in the stream and their Program Map Tables' PIDs.
- The *Conditional Access Table* (CAT) on PID 0x1 (which is only present when the content is encrypted) describes how to decrypt the streams.
- Each program has a *Program Map Table* (PMT), associating video, audio and other elementary streams to one program. It also contains for example language information for each audio stream.
- The *Network Information Table* (NIT) on PID 0x10 describes information about the network the stream is being transmitted on.

Other table types are introduced with DVB-SI, see section 7.1 below.

4 Error Correction

As the broadcast content is transmitted via the RTP protocol, which uses the UDP protocol, packet loss can occur, especially on congested networks or wireless links. While the MPEG transport stream as well as the MPEG codecs are designed to cope with a certain level of packet loss, the quality of experience can be improved by including error correction schemes that repair the damage inflicted by lost packets. We will take a look at error correction basics, existing protocols and look how error correction can be applied in the multimedia home network.

4.1 Error correction schemes

There are several ways to correct transmission errors over an unreliable network:

4.1.1 Forward Error Correction

Forward Error Correction (FEC) *proactively* adds redundancy to the stream: It produces n coded packets from $k \leq n$ data packets by computing $n-k$ redundancy packets.

The receiver can now reconstruct the data packets from any combination of at least k received packets. The obvious disadvantage is the additional bandwidth requirement. On the other hand, the correction is possible without further interaction with the sender.

Excessive use of FEC, especially with a large data block size k increases the latency, as the data packets have to be accumulated to compute the redundancy.

4.1.2 Retransmission of lost packets

The alternative is to use Automatic Repeat reQuest (ARQ). Whenever the receiver detects that a packet is lost (e.g. by the means of a sequence number gap), it *reactively* sends a retransmission request (negative acknowledgment, NACK) to the sender. The sender now can transmit the lost packet again. Alternatively, the clients regularly send positive acknowledgments (ACKs) for received data, if the sender misses ACKs, it then sends the retransmissions.

On links with a rather high round trip time (like a broadband internet connection), this scheme requires a higher delay, as the retransmission requires another network round trip. Thus the receiver has to add a delay buffer to have the time for the retransmission before the packet is needed for rendering. The advantage is the lower bandwidth requirement, especially on networks with a rather low error rate.

Hybrid error correction

As both schemes have their advantages and drawbacks depending on network conditions and content data rate, it can be a good idea to combine them to provide both proactive and reactive error correction. When the level of redundancy is reduced, network bandwidth is saved, at the higher risk of having unrecoverable errors. In this case an added ARQ mechanism can now fix the remaining errors.

4.1.3 AHEC

Adaptive Hybrid Error Correction (AHEC)[31], developed at the Telecommunications Lab of Saarland University, is a HEC scheme intending to provide predictable reliability under predictable delay.

AHEC combines FEC and ARQ to minimize bandwidth usage for redundancy when it is not needed. For each block of data, redundancy is being computed by the sender depending on parametrization, but not yet transmitted. All sent packets within a reasonable timeframe are kept in a retransmission queue. Only when the receiver indicates that the block cannot be decoded due to packet loss by sending negative feedback (NACK), the sender will send more redundancy packets. This may repeat until a predefined time limit (the maximum latency, hence "predictable delay") is reached. Of course, the sender must keep all packets for this timeframe in a resend queue in case a receiver sends a re-request for them.

Note that only in a corner case, where $k = n = 1$ (Block size of 1, thus no parity packets), the retransmissions always contain copies of the payload data. This mode is called AHEC-PR[32] (AHEC Packet Repetition). Otherwise, the NACKs always result in redundancy packets.

In a multicast environment, usually several receivers fail to receive the same packets, so multiple receivers may benefit from a single retransmission. To reduce the number of multicast NACK reports in this case, the NACK are sent by the receivers with a short, random delay. The feedback sent first by the receiver with the shortest random feedback delay suppresses the NACK by all other receivers.

4.2 Implementations

4.2.1 RTP/AVPF

The Extended RTP Profile for RTCP-Based Feedback (RTP/AVPF)[25] specifies how sender and receiver can exchange feedback about lost data in an RTP session.

RTP/AVPF provides both ACK (indicating received data) and NACK (indicating lost data) based feedback via RTCP FB (feedback) messages. To avoid excessive bandwidth consumption by feedback RTCP packets, RTP/AVPF retains the bandwidth limitations of RTCP[28], except that the minimum interval of 5 seconds is abolished for feedback messages. Additionally an `early feedback` packet is allowed, which may be sent immediately *once* per RTCP report interval. This provides better scalability at the price of less timely feedback, which reduces the correction performance.

The feedback can be generic transport-based (e.g. a lost RTP packet) as well as media-specific (e.g. lost video frame) or application-specific.

Note that there is no specification on how to react to the feedback; FEC is possible just as ARQ or codec-specific actions.

4.2.2 RTP Retransmission Payload Format

RTP Retransmission Payload Format[26] (RFC4588), or RTP/RPF for short, provides different mechanisms for retransmitting lost packets. All of them multiplex with the regular RTP transmission in a way backwards-compatible to endpoints not supporting this retransmission scheme.

To ensure this backwards-compatibility, the retransmission packets are not simply duplicates of the original packets (which would distort RTCP statistics), but they are multiplexed with the original stream either using SSRC-multiplexing or using session-multiplexing. The former uses the same RTP session for the retransmission data, but assigns a different SSRC to the retransmission data. The latter uses two separate RTP sessions, which means using different UDP ports. In either case, the original sequence number of the lost packet is included in the retransmission packet. The association between original and retransmission stream is done via the SDP document.

4.2.3 RTP/FEC

Using the example of a simple XOR operation to generate the parity packets, RTP/FEC[1] specifies the RTP packet structure for redundancy packets. It supports Uneven Level Protection (ULP), where more "important" payload packets (e.g. reference frames in video streams) get a higher level of redundancy than less important parts. Instead of XOR, other FEC algorithms such as Reed-Solomon are possible.

One important property of RTP/FEC is that the payload packets are sent just as if there was no FEC, while the FEC overhead is entirely contained in the redundancy packets. This makes it compatible with multicast scenarios where non-FEC-aware receivers are present.

RTP/FEC requires session multiplexing, i.e. the payload stream and the FEC stream always share the same SSRC, but are sent on different address/port combinations. This allows receivers to decide individually whether they want to receive FEC or not.

4.2.4 PRRT

Predictable Reliable Realtime Transport (PRRT)[21], developed at the Telecommunications Lab of Saarland University, provides a Linux kernel-based implementation of AHEC. It is designed as a drop-in replacement for UDP, allowing the developer to quickly convert UDP-based applications to use advanced error correction.

Instead of extending existing UDP-based protocols like RTP, the socket library implements a new IP protocol based on UDP. It extends the standard UDP header by 12 bytes to convey packet type, sequence numbers, timestamps and statistical information.

Protocol parameters are set by the application using a special `proc(5)`[36] file (although future implementations might accomplish this using a more usual `ioctl(2)`[36] interface).

The advantage of designing a new protocol for this purpose is the small overhead compared to more bloated protocols like RTP. On the other hand, there is no backwards compatibility to RTP clients. To remedy this, there is already a proposal[21] to implement AHEC on top of standard RTP extensions like RTP/AVPF and RTP/FEC.

4.2.5 AHEC via RTP/AVPF and RTP/FEC

Here, the goal is to provide AHEC on top of RTP while retaining backwards compatibility with endpoints not capable of AHEC.

One of the main drawbacks of RTP/AVPF is the strict feedback timing schedule: RTP/AVPF deliberately limits the feedback frequency, based on the calculation in RTP[28]. Depending on the receiver group size, only a certain number of RTCP feedback messages (including RTP/AVPF NACKs) are allowed in a rather long time period, intended to limit the total bandwidth consumption of RTCP in relation to the related RTP stream. AHEC proposes[30] a modification to this schedule to allow multiple feedback rounds (as long as the application-specific maximum delay is not exceeded). Also the requirement of "one feedback per report interval" is removed, allowing immediate feedback at any time.

The RTP and the protocol extensions made by RTP/AVPF and RTP/FEC already provide the basic building blocks required for AHEC:

- RTP packets contains sequence numbers and timestamps
- A RTCP feedback packet type to convey the receivers' NACKs
- Statistical information is contained in the RTCP sender and receiver reports
- A separate, independent channel for FEC data (RTP/FEC makes no assumption about the error coding scheme used)
- Coding parameters can be sent in the RTP/FEC packets, too.

As the RTP session component itself is not modified with respect to a simple RTP session, a non-AHEC-capable client can still join such a session; It would ignore the extended attributes in the sessions' SDP, won't join the FEC session and will never send a NACK feedback message. Only the NACK feedback message would have to be clarified, to distinguish between a NACK for RTP/RPF (maybe from an endpoint only supporting RTP/AVPF and RTP/RPF) and a NACK for more AHEC redundancy packets.

4.2.6 DVB RET

The DVB Project provides its own RTP retransmission solution (ARQ) based on RTP/AVPF in the form of a RET (retransmission) server within DVB-IPTV/IPI-1[17]. Building upon RTP/AVPF and RTP/RPF, considerations for multicast scenarios are added:

In a multicast live media broadcast (LMB) scenario on the delivery network, the packet loss can occur higher up in the delivery network architecture, affecting many HNEDs at once. To avoid a storm of NACK packets, the RET server in this case sends a multicast feed-forward (FF) message to all its clients to indicate that the repair is already in progress.

4.3 Error Correction in the home network

The question remaining is clearly how an effective error correction scheme can be integrated in the existing DLNA and DVB-HN standards.

DLNA compliant devices that intend to support RTP error correction are required to support ARQ using RTP/AVPF and the RTP/RPF.

DVB-HN devices supporting any form of error correction are actually required to use RTP/RPF; no other retransmission schemes are allowed[18].

Neither DLNA nor DVB-HN do make any provisions for FEC.

4.3.1 Home network conditions

In the home network the number of nodes (and thus the number of stream receivers) is usually very small[1].

The bandwidth available is usually very high. A current network device usually has Fast Ethernet with 100Mbit/s, and wireless devices support at least 802.11g with 54Mbit/s. Compared to an SD video stream with a maximum of 15Mbit/s, there is enough bandwidth available[2].

While a switched wired network can be considered almost error-free, a wireless link is quite prone to intermittent packet loss, especially if an endpoint itself is moving or people are walking in the signal's path.

Latency wise, a home network is almost optimal, with round trip times well below 5ms.

4.3.2 Extending DLNA/DVB-HN with AHEC

Unfortunately, DLNA/DVB-HN and AHEC already have a conflict in the packet format:

Both DLNA/DVB-HN require RTP/RPF if any form of error correction is to be supported. In AHEC on the other hand, the reaction to receiver feedback is always more FEC data, thus requiring RTP/FEC transport if implemented RFC-compliant.

Integrating AHEC in DLNA/DVB-HN would thus require one of the following:

- Either remove the strict rule of "If error correction, then RTP/AVPF with RTP/RPF" for the *rtsp-rtp-udp* transport
- Or introducing a new transport with AHEC support, maybe called *rtsp-rtp-udp-ahec*, which includes specifications on how to implement AHEC. A serving endpoint (DMS, DVB-MS) could offer it alongside a resource for regular *rtsp-rtp-udp* transport.

As AHEC can be implemented backwards compatible (see section 4.2.5) to plain RTP, the first alternative seems to be more feasible. A full AHEC-protected backwards compatible RTP session could look like this:

All components are session-multiplexed, i.e. a separate UDP port pair for content, RTP/RPF and RTP/FEC.

[1] Already a number of 5 concurrent receivers can safely be considered a border case.

[2] Even taking the much lower net bandwidth of Wi-Fi into account

Responding to the RTSP `DESCRIBE` request, the serving endpoint provides a SDP document describing the stream, with all fields required by DLNA and DVB-HN. In addition, it contains RTP/FEC-required fields, RTP/RPF-required fields and additional attributes with AHEC coding parameters. All of UPnP, DLNA and DVB-HN hold the guideline of "parse and interpret what you know, or parse and ignore what you do not know" very high, thus any not-AHEC-aware DLNA/DVB-HN client device must ignore the additional SDP information.

Depending on the client's capabilities, it may then use different error correction components:

- **RTP-only**: Receives only the content stream. Any FEC data on the RTP/FEC session is ignored.
- **RTP with RTP/AVPF and RTP/RPF**: Understands SDP attributes for RTP/RPF, therefore receives content stream and RTP/RPF session. Sends only regular RTP/AVPF NACK feedback messages, and thus receives only regular retransmissions.
- **RTP with RTP/FEC**: Understands SDP attributes for RTP/FEC. If it supports the FEC scheme proposed, it receives content stream and RTP/FEC session. Potentially makes use of any FEC data received on the RTP/FEC session.
- **RTP with both RTP/RPF and RTP/FEC**: Combination of the previous two cases.
- **RTP with full AHEC awareness**: Understands all EC-related SDP attributes, therefore listens on all three RTP sessions. May send regular RTP/AVPF NACK feedback (to receive retransmissions) and also special AHEC NACK feedback to request additional FEC data.

4.3.3 Conclusion

In typical home network conditions, a simple ARQ can repair most damage done to the stream. Due to the low latency and the limited number of nodes, the scheme used may request retransmissions as often as required, by not adhering to the feedback timing constraints of RTP/AVPF[25], but adjusted by the AHEC-PR analysis.

To increase effectiveness of the error correction, AHEC could be introduced into DLNA/DVB-HN compatible home network devices, if the specifications receive minimal modifications.

5 Reducing channel change delay

One major issue regarding the quality of experience of broadcast-enabled home entertainment network system is the time required for channel changes.

As many people watching TV quickly switch through the available channels searching for an interesting program ("zapping"), the time between the user pressing the "channel up"/"channel down" button and the next channel starting to be rendered defines how fast the user finds an interesting program. Thus it is reasonable to reduce this delay as far as possible.

5.1 Delay analysis

First, we have to determine the minimum delay possible, i.e. how fast a channel change can happen if all delay sources are minimized. Thus we have to take a look at the channel change process and the delay each step imposes on the process. We assume a delay source to be instantaneous (thus already minimized) if its impact on the delay is below the average users attention span. This includes the network round trip time on the home network.

As a starting point, the process of changing a channel on the current DMS/DMP scenario requires these steps with the given delays (server and client running on the same host, using RTSP/RTP media transport to stream PAL SD MPEG2 encoded broadcast TV).

1. The user presses the channel up/channel down button. The input processing on a modern system is assumed to be instantaneous.

2. The client has to determine the next channel to switch to. This issues a ContentDirectory Search() call to the server. Including network overhead, in practice this takes well below 50ms.

3. The client requests the new channel from the server.

4. The server switches the tuner card to the new channel. Depending on medium and hardware, this takes up to 850ms (including RTSP handshake)[1]: $t_{tuner} = 0.85s$

5. The server starts streaming the new channel to the client. We chose to neglect the network RTT here.

6. The client starts receiving the stream, and first fills a receiver buffer with the stream. The buffer size which influences the buffering delay t_{Buffer} is variable, its minimum sizes depends on various factors like the decoder used or the stream properties.

[1] Measurements show tuner delays of ca. 800-850ms with an USB DVB-T tuner, and ca. 450-500ms with a PCI DVB-S tuner

7. When the client starts receiving the stream, it starts at a random position with no regard for the stream structure. But the video decoder has to wait for the beginning of a Group Of Pictures (GOP). A GOP is a set of video frames, starting with a I-Frame, and followed by a set of P- and B-Frames referencing the I-Frame. Because of this, video decoding can only start at the next GOP. In PAL DVB SD content (broadcast at $f = 25$ frames per second), a GOP is usually 15 frames long. In the worst case, streaming starts just after the first byte of the current GOP, resulting in a maximum decoder start delay $t_{decoder} = 15 \cdot \frac{1s}{25} = 600\text{ms}$.

   Similar constraints apply for the audio stream, but the audio payload unit (frame) sizes are much shorter than a video GOP (One MPEG Layer 2 audio frame is 1152 samples long, at a sampling frequency of 48kHz this results in a frame duration of only 24ms).

8. In a DVB MPEG transport stream, the audio stream(s) are delayed w.r.t. the video streams. Thus, after decoding the first video frame, the renderer has to wait for the corresponding audio frame to be played at the same time before playback commences (See figure 5.1. This delay is ca. $t_{AVoffset} = 0.7s$.

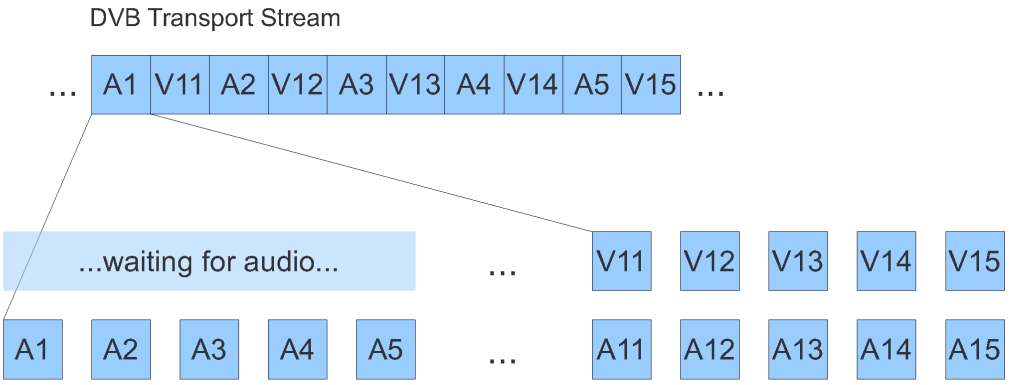

Figure 5.1: Audio/Video delay in the multiplexed DVB Transport Stream

A decoder not aware of the A/V offset needs a rather big buffer to ensure playback without frame drops. This buffer includes the before mentioned A/V offset as well as the unused data before the next video GOP begins, thus $t_{buffer} \geq t_{AVoffset} + t_{decoder}$.

5.2 Reducing the delay

The two biggest contributors to the channel change delay are the tuner hardware t_{tuner} and the buffering required at the receiver t_{buffer}. For faster channel changes, both can and should be reduced by clever tuner hardware management.

5.3 Predictive Tuning

The tuner delay t_{tuner} can be reduced to zero if the media server has the tuner already tuned to the new channel when the request for the new channel arrives. For this prediction, we make use of the typical user behavior: When "zapping" through the channels, it is very likely that the user changes the channels sequentially in the order of the channel list.

Because of that, when changing from channel *Ch1* to *Ch2*, the Media Player can assume that a change to channel *Ch3* is imminent, and can notify the Media Server. If the Media Server has free tuner resources, it can tune a free tuner to *Ch3* before the user has switched to that channel.

5.4 Fast Channel Change

With the RTP multicast transport, filling the receiver buffers takes the same amount of time as the buffer content takes to be rendered, as the data is sent and received at the same speed as it is intended to be rendered.

In cooperation with Predictive Tuning, the buffer delay t_{buffer} can be significantly reduced by filling the buffer at increased speed at the beginning of the streaming session. As the server has the channel already tuned, it can cache a reasonably sized backlog of the stream in a ring buffer. When the client issues the RTSP PLAY request, the server's ring buffer contents can be quickly transferred into the client buffer, before the client switches to the RTP stream.

This idea has been used recently in several proposals:

5.4.1 Unicast-Based Rapid Acquisition of Multicast RTP Sessions (RAMS)

This IETF Draft[38] proposes bursting the backlog buffer to the client via RTP unicast, built upon the RTP/AVPF profile[25](See section 4.2.1) controlled by RTCP[28].

The time between the receiver joining a multicast session and actually having the required information ("*Reference Information*") is therein called *Acquisition Delay*. Note that by "required information", the draft means various things: Not only a certain buffer size required by the decoder, but also periodically sent encryption keys in DRM-protected streams or stream descriptions such as PSI data.

The RAMS-compatible multicast stream source or an intermediate retransmission server stores the Reference Information, ready to be transmitted as soon as a new client joins the multicast session. This role can run on the same server that is running the feedback target role for error correction methods.

The following is the simplified process of receiving the Reference Information: Before joining the multicast RTP session, the client and the retransmission server negotiate the additional RTP unicast port used for RAMS. e.g. via a SDP document.

Next, the client sends a RAMS request (`RAMS-R`) via RTCP. This is acknowledged by the server with a `RAMS-I` response. Now the RTP unicast burst is sent until the client terminates the burst. When the client has received the burst transmission, it terminates it by sending a `RAMS-T` message. The client now joins the multicast RTP session, and repairs the possible gap

between the RTP unicast burst data and the multicast data using the RTP/AVPF retransmission mechanisms.

5.4.2 DVB Fast Channel Change

The DVB Project's *Server-Based Fast Channel Change for DVB-IPTV Systems*[34] uses RAMS. The document clarifies the use of RAMS in the context of DVB-IPTV transmissions, including the combination of RAMS and the LMB RET (Live Media Broadcast Retransmission) error correction.

5.4.3 Fast Channel Change in the home network

The before mentioned schemes all assume that the Reference Information is always available, as the Retransmission Server is always joined to all possible multicast channels. On a DVB-UGD with tuner hardware, this is not always the case, as the data is only available after the tuner is tuned to the requested channel. But in combination with Predictive Tuning, a certain amount of data is already available to reduce the channel change delay.

Neither DLNA nor DVB-HN propose any form of fast channel change. Here is a FCC proposal for within the home network:

A simple method would be to send the burst data as part of the RTSP *PLAY* response. When the client indicates support for RTSP-based fast-channel-change (FCC) in the request, the server sends the requested amount of buffer data in the form of RTP packets as the body of the RTSP response. Using the TCP in this way has advantages and disadvantages compared to the UDP-based RTP approach of RAMS:

Sending the burst data via RTP implies the possibility of packet loss, which has to be repaired later using a rerequest scheme. A TCP-based transmission guarantees the orderly transmission of all burst data. On the other hand, the TCP protocol's slow start algorithm denies the full network throughput right from the start.

Also the use of the RTSP message of course limits this approach to scenarios where RTSP is actually used. Nevertheless it would be a simple approach within the home network, where RTSP is used for all internal RTP sessions.

RTSP headers

To initiate RTSP-based fast channel change, some new RTSP headers need to be introduced. See figure 5.2 for an overview.

X-DHCL-FCC-Range : In the RTSP `PLAY` request, the client indicates that it supports RTSP-based FCC and describes how much data it requests. The syntax is like this (see RFC2616[19] for grammar):

```
           FCC-Request ::= "x-dhcl-fcc-range" ":" LWS dhcl-fcc-value
         dhcl-fcc-value ::= dhcl-fcc-value-bytes | dhcl-fcc-value-timespan
   dhcl-fcc-value-bytes ::= 1*DIGIT "bytes"
dhcl-fcc-value-timespan ::= 1*DIGIT "ms"
```

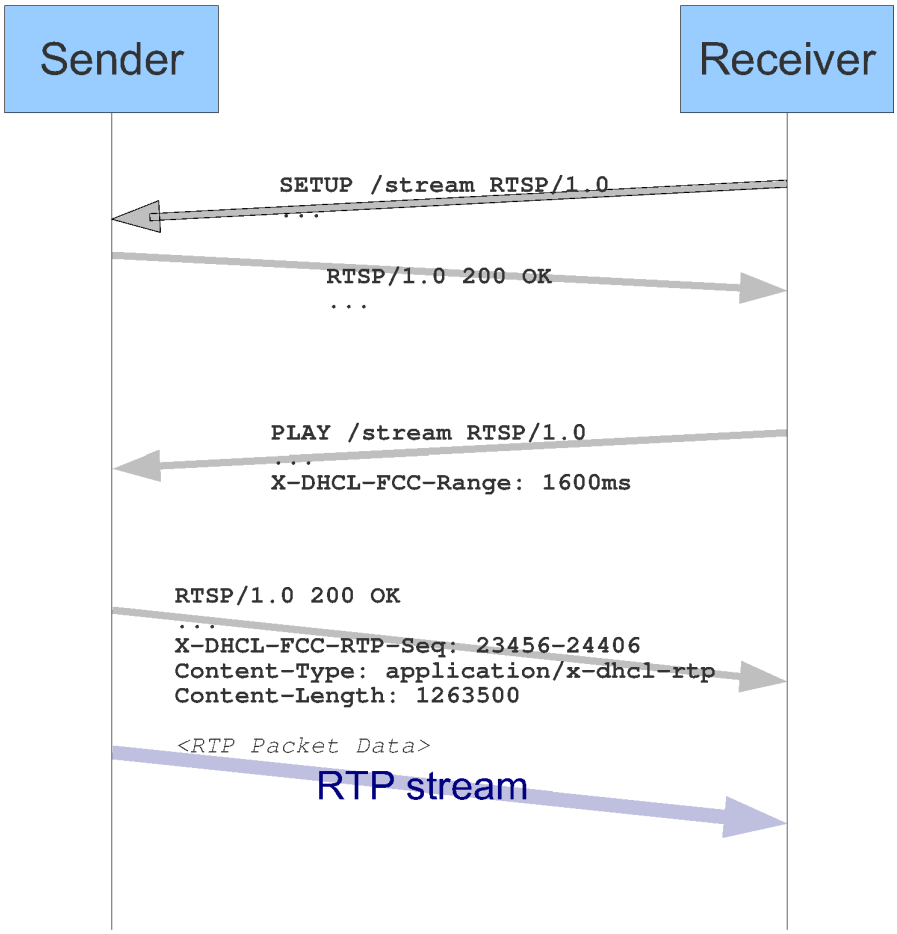

Figure 5.2: RTSP-based fast channel change handshake

The client chooses to specify the requested buffer size either in net payload bytes or as a rendering time duration in milliseconds. Examples:

```
X-DHCL-FCC-Range: 1048576bytes
X-DHCL-FCC-Range: 1600ms
```

`X-DHCL-FCC-RTP-Seq` : In the RTSP `PLAY` response, the server indicates the RTP sequence number of the first and the last RTP packet in the returned buffer.

```
      RTP packet range ::= "x-dhcl-fcc-rtp-seq" ":" LWS dhcl-fcc-rtp-range
    dhcl-fcc-rtp-range ::= rtp-seqno "-" rtp-seqno
             rtp-seqno ::= 1*5DIGIT
```

Payload format

The content type "`application/x-dhcl-rtp`" indicates raw RTP packets:

```
Content-Type: application/x-dhcl-rtp
```

The body consists of raw RTP packets, each prefixed by a 16-bit packet length field in network byte order.

Switching to the RTP session

Immediately after receiving the buffer fill payload, the client starts receiving the RTP data. Depending on network conditions and client/server performance, there is a gap in RTP data between the buffer fill data and the first received RTP packet. This gap can be filled using ARQ error correction.

As the FCC scheme is triggered using vendor-specific headers in the RTSP handshake, it would be backwards-compatible to clients and servers not capable of this FCC scheme.

6 Time synchronization

One major issue while streaming broadcast media over data networks is time synchronization. On a legacy DVB set-top-box, receiver, decoder and display components share the same clock, synchronized to the reference clock of the received channel. In a networked environment, these roles are usually split: The broadcast receiver runs on an expansion card in the media server, which transmits the received data over an IP network using RTP, which is in turn received by a separate media player.

In this scenario, we have three independent clocks:

- The reference clock provided by the tuned station, contained in the received data stream
- The local clock in the media server, to which the stream transmission is synchronized
- The local clock in the media player, controlling the playback speed.

These clocks do not necessarily run at the same speed, especially in cheap consumer products, the clocks are influenced by temperature, radio interference and aging.

This will sooner or later lead to buffer over- or underruns in the receiver, resulting in the render skipping or dropping the signal in a way noticeable by the user.

In a one-to-many scenario, it is clear that the first clock, the stations reference clock, cannot be changed. Thus the two other clocks must be synchronized to the reference clock. In practice, both synchronize themselves on the program clock reference (PCR) timestamps contained in the MPEG transport stream by the means of a software PLL (Phase Locked Loop).

6.1 The phase locked loop

A classic PLL is an analog circuit used to recover a reference clock signal from an analog signal. It consist of a phase detector, a loop filter and a voltage controlled oscillator (VCO). The phase detector compares the phase of the input signal and that of the VCO, and provides an error signal depending on the phase difference. To reduce jitter, this error signal is first filtered through the loop filter (usually an IIR lowpass filter), and then fed into the VCO, correcting the VCO's speed to minimize the phase error.

The software PLL used in this project works in a similar manner, just the VCO (which is an analogue component) is replaced by an Numerically Controlled Oscillator (NCO), its digital, discrete counterpart.

With each received PCR value, the phase detector compares the NCOs current value with the PCR value, providing a phase difference. This is filtered through a 2nd order IIR filter. The filtered phase error is then used to compute the new speed factor of the NCO relative to the system's clock.

6.2 Software PLL implementation

In the PLL implementation, all clock values are normalized to the "wallclock" frequency of 1Hz. The calculations are done with double precision floating point numbers to provide sufficient precision for processing 27MHz extended PCR timestamps. This allows for simpler calculations, as no frequency conversions are necessary (All participating components such as PCR, NCO and system clock now have comparable values).

6.2.1 Initialization

With the first received PCR timestamp t_0, the PLL is initialized. The NCO is initialized with a frequency of 1.0Hz (same as the system clock) at time $T_0 = 0$.

6.2.2 PLL adjustment

Whenever a subsequent PCR timestamp t_n is received at NCO time T_n, the phase error is calculated:

$$E_n = (t_n - t_0) - (T_n - T_0)$$

The phase error is filtered through the loop filter[1]:

$$E = \begin{pmatrix} E_n, E_{n-1}, E \end{pmatrix} \begin{pmatrix} 0.00496 \\ -0.0012 \\ 0.99 \end{pmatrix}$$

From this filtered phase error (see figure 6.1 for an example), the new speed factor for the NCO is calculated. As the NCO has a base frequency of 1.0 Hz, the speed factor is equivalent to the NCO frequency f. To achieve a smooth correction response, a *NCO gain factor* G_{NCO} is introduced. Experiments with DVB-T transmissions have shown that $G_{NCO} = 0.05$ is a good compromise between a smooth clock value and quick locking to the PCR frequency.

If the phase error is positive, the NCO runs too slow, and must be sped up w.r.t. the wallclock with a speed factor $f > 1$, while it has to be slowed down ($0 < f < 1$) if the phase error is negative. For a stable operation, the transfer function must have several properties:

- It must be always positive, the NCO driven clock running backwards would cause unexpected results within the player backend.
- If the phase error is 0, the system clock runs exactly as fast as the PCR timestamps. Thus it must yield exactly 1.0.
- If the phase error is negative, it must yield a value less than 1.0, i.e. an NCO frequency slower than the system clock.
- If the phase error is positive, it must yield a value greater than 1.0, i.e. an NCO frequency faster than the system clock.

[1] The filter coefficients are taken from the lecture script "Future Media Internet": http://www.nt.uni-saarland.de/education/FMI-LectureWS09-10/LectureNotes/

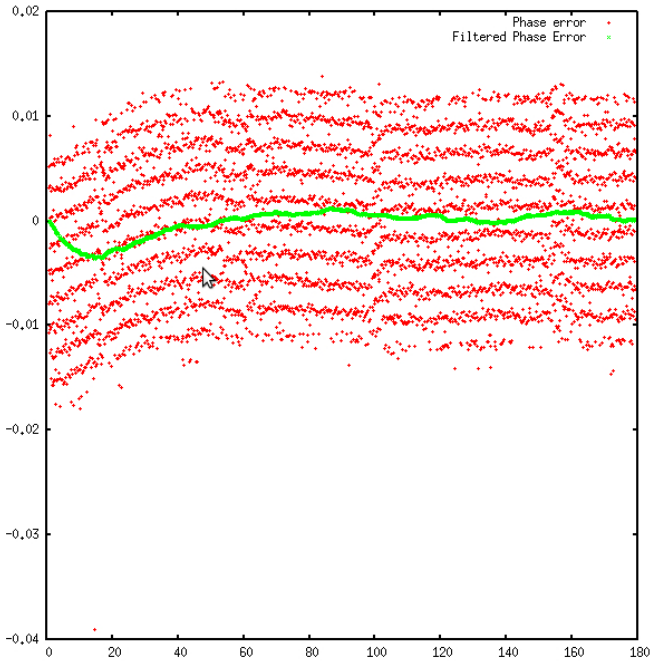

Figure 6.1: Raw phase error and filtered phase error (seconds) vs. wallclock time (seconds) while receiving a DVB-T signal.

The exponential function has all properties necessary for this task:

$$s = \exp(E \cdot G_{NCO}) \tag{6.1}$$

The NCO is then set to the new frequency $f = s * 1.0Hz$. Figure 6.3 illustrates this. See 6.2 for a real-world plot of the NCO.

6.2.3 The NCO

The NCO is implemented as a variable-speed clock linked to the system's clock. Upon Initialization, the speed factor is set to 1.0 (same speed as the system clock), the *initial wallclock time* T_0 is recorded and the *pivot point*[2] T_p is set to 0.0.

Whenever the speed factor is updated, the pivot point is moved to the current NCO time/Wallclock time position, and the NCO now runs from there (See figure 6.4). This way, there are no jumps or discontinuities in the NCO.

To adapt the rendering speed in the client to the PCR clock, the playback system just has to feed each PCR timestamp into the DPLL, and use the resulting NCO-driven clock instead of the system clock. We will see in section 8.2.3 that the xine-based playback engine allows that with reasonable effort.

[2]The "pivot point" concept is taken from the libxine[37] metronom class

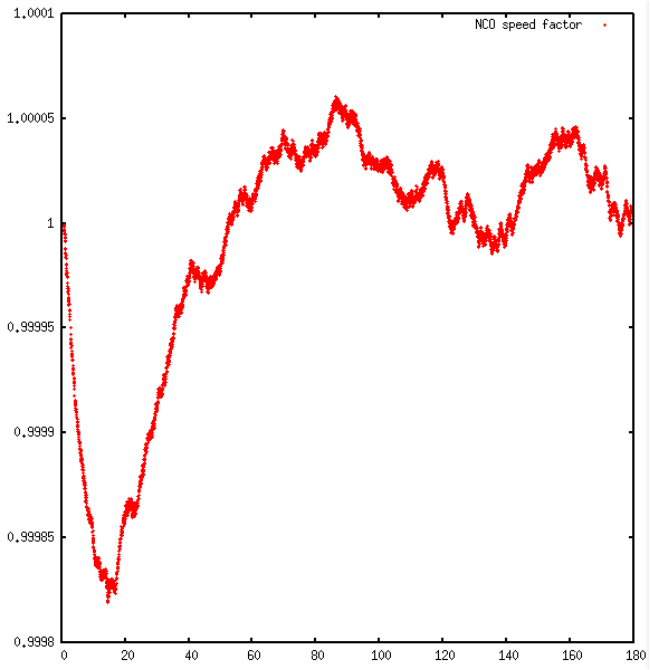

Figure 6.2: NCO frequency (Hz) vs. wallclock time (seconds) for a DVB-T signal.

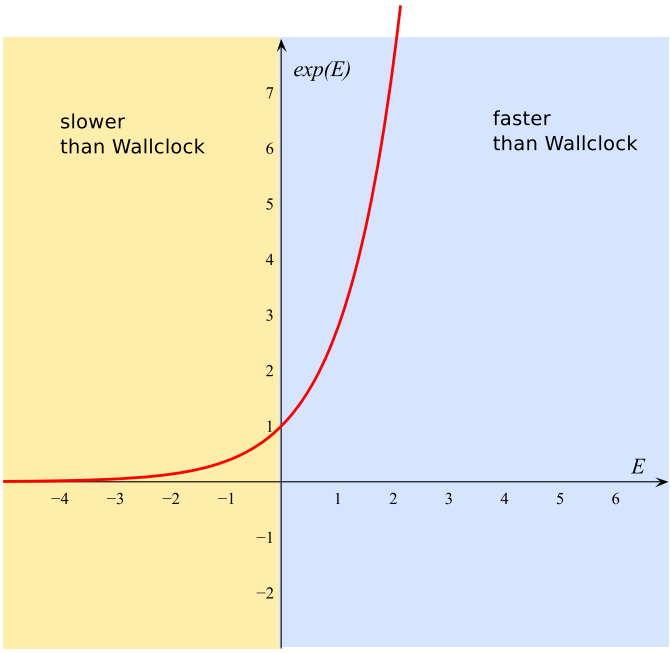

Figure 6.3: Filtered phase error vs. new NCO speed factor

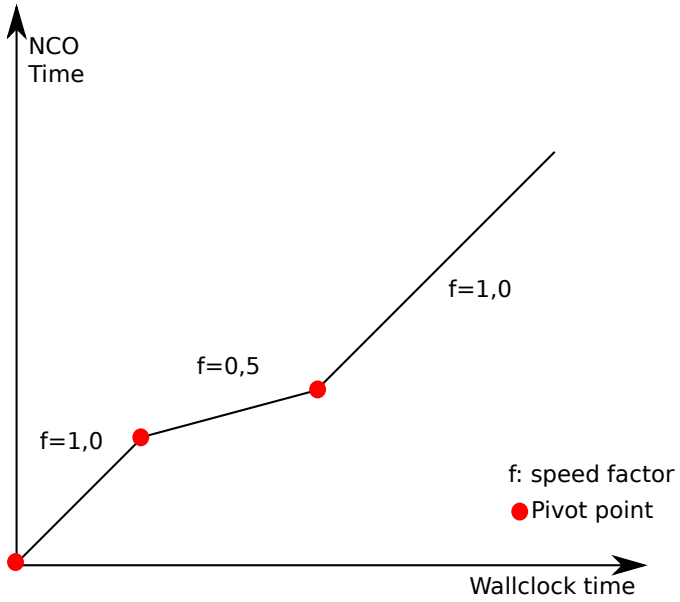

Figure 6.4: NCO pivot point illustration

7 EPG and Service Information

A user-friendly home network device should be able to discover the available broadcast services automatically (compared to a "channel scan" by an analog receiver). Current DVB standards provide service information (containing the names of the stations) and an EPG, listing the programming schedule of each TV and radio station including a short description. This is a major improvement over analog broadcast, which carried no metadata at all[1].

7.1 DVB-SI

DVB Service Information[14] is contained within the MPEG transport stream as additional tables similar to the PSI tables (See section 3.3.1. Each station or program in the stream is called a *service*.

The most important additional tables are:

- Network Information Table (NIT): Contains information about the available transport streams on the current network[2], including physical access parameters (e.g. channel frequency, modulation, etc.). As each network has a unique ID, each transport stream can be uniquely identified by network ID and transport stream ID.
- Bouquet Association Table (BAT) listing the name of a Bouquet and its associated services. This is helpful if some services of the Bouquet are not on the current transport stream.
- Service Description Table (SDT): contains names of services and their service providers.
- Event Information Table (EIT) contains about information events (e.g. TV shows, movies) with their names, start and end dates, duration, etc.
- Running Status Table (RST) indicates whether an event is running or not. This can be used to trigger recording of a TV show as soon as its status changes to "running".

From these tables, user friendly information about the available TV and radio stations, as well as a list of current and upcoming shows EPG can be generated. All the tables are broadcast regularly on DVB-S/C/T.

Using this data, for example a DVB-S receiver can bootstrap the complete service list from one single transponder frequency, by extracting DVB-SI NIT information from the available transponder stream and filling the channel list iteratively.

[1] Apart from Radio Data System or Teletext

[2] A network in this case is the collection of all transport stream on one delivery system, e.g. all channels on one satellite network like Astra 1 on 19.2°E.

7.2 DVB-SD&S

DVB Service Discovery And Selection (SD&S)[17] is used by DVB-IPTV devices (including DVB-BGDs) for the discovery of the services provided.

7.2.1 Discovery

The discovery of the SD&S data source (the *SD&S entry point*) is attempted in this order via

- a DHCP option containing the domain of the service provider (set by the DHCP server on the access network or in the DNG). The SD&S server(s) are acquired via a DNS service location lookup (for *example.com*, the SRV records would be *_dvb-servdsc._tcp.example.com* or *_dvbservdsc._udp.example.com*).
- joining a IANA-registered multicast group (224.0.23.14, *DvbServDisc*)
- via DNS service location with the service provider domain of *services.dvb.org*
- User interaction, if the other methods fail.

The service provider (SP) discovery information is then retrieved in pull mode (explicit request via TCP/UDP) or push mode (on the multicast channel). The discovery information contains a list of DVB-IPTV offerings made by this service provider, and information on how to retrieve them (push address or pull URL).

Each offering may be a

- Broadcast discovery record (TS Full SI): it describes a live media broadcast (LMB) stream containing complete DVB-SI data (thus “TS full SI”). The endpoint can access this stream and parse the DVB-SI data.
- Broadcast discovery record (TS Optional SI): Just as above, except that the LMB does only contain MPEG PSI, but no DVB-SI (“TS optional SI”). Here, the discovery record contains the SI data.
- "Service From other Services Providers" record: references other SPs.
- Package discovery record, grouping several services to a single entity.
- Broadband Content Guide record: Describing how to access guides listing live content or content-on-demand (CoD).

In the multicast case, a special UDP-based protocol called DVB SD&S Transport Protocol (DVBSTP) is used. In the unicast case (pull mode), the SP Discovery Information and the Service Discovery Information is retrieved via HTTP.

7.2.2 Selection

For Service Selection (meaning accessing the content), DVB-IPTV offers two alternatives: The first is IGMP[2]; the endpoint just joins an existing multicast channel for live content. The second one is RTSP[29], where the client negotiates delivery of the content with the service providers' server. Transfer of the content in this case can be multicast, too (for live content requiring some accounting or billing), or unicast for Video-on-Demand services and Trick Mode applications.

7.3 Translating DVB-SI and DVB-SD&S to Content Directory items

A DVB-UGD translates from DVB-S/C/T broadcast (providing DVB-SI data), a DVB-BGD from a DVB-IPTV delivery network (providing SD&S) to the DVB-HN compliant home network. This translation includes transforming the metadata received in the form of SI or SD&S data into UPnP ContentDirectory entries for each service (channel).

The DVB-HN specifications[18] contain translation tables between SI/SD&S entries and ContentDirectory attributes. For example, the *upnp:channelName* field of a channel's CDS entry is extracted from the DVB SD&S `IPService/TextualIdentifier@ServiceName` field, or from the DVB-SI SDT entry "service name".

EPG data is incorporated into the ContentDirectory, too, in the form of an *object.item.epgItem* class item for each program. A client endpoint can parse the epgItems from the CDS and present them to the user in the accustomed timetable form.

In the end, both channel information (in the form of a channel list) as well as upcoming program information (in the form of EPG data) is available in the Content Directory.

8 Implementation

The software stack developed during this thesis implements most of a typical home entertainment network stack. The `rtpserver` DVB-MS/DVB-UGD developed during [13] has been extended with native DVB card access, and a corresponding DMP has been implemented. The common components used by both programs were split off into library packages that can be reused in other projects (See section 8.4.1), including basic I/O and multithreading, network communication, HTTP/RTP/RTSP protocol handling and media processing.

8.1 Media transport

The basic *pipeline* principle was already explained in [13]: A chain of pipeline objects (A source, possibly multiple translators, and a sink) processes the media stream received from the server. Each Source produces a stream in a specific *content type*, such as MPEG TS, AVI or MPEG ES, in the form of payload units in a specific *payload format*, e.g. MPEG TS frames or MPEG PS packs. This distinction is necessary, as a source may provide a known format (e.g. MPEG transport stream), but in a raw payload format, but not (yet) need in discrete payload units (packets, frames, etc.). Similarly, each sink class accepts a certain set of *content types* and *payload formats*. A translator class is simply both sink and source, and transforms the stream in a certain way (decoding, demultiplexing, etc.).

Additionally, the pipeline offers signaling of broadcast events. Any node can indicate when the stream has reached the end, or when an unrecoverable error occurs. The pipeline nodes can also send out-of-band information down the pipeline, e.g. PCR timestamp updates or PID changes.

To create a pipeline, a *pipeline specification* object is needed. This contains parameters such as the content URL, and flags e.g. whether to request a multicast or unicast transmission from an RTP sender.

The pipeline is then created using this specification object and a sink node. Depending on the stream URL, the matching source object is created, then the necessary translators are inserted between source and sink using a set of rules. For each content type, the rule set decides which translator needs to be added to get closer to the sink's requirements. This rule set is currently implemented as a static decision tree, if e.g. the current source produces MPEG TS frames and the

8.1.1 Pipeline creation and media transport

As we focus on today's DVB broadcasters' content format, which is a MPEG Transport Stream (TS) containing several elementary streams (MPEG2 video[23], MPEG Layer II audio[1]).

To illustrate, we take a look at the pipeline created both in the Media Server and the Media Player for the before mentioned use case. We have the following preconditions and requirements:

- Stream is received via DVB using a DVB tuner card.
- On the network, the stream is to be transmitted as an MPEG transport stream.
- Transport to the client via RTSP/RTP protocol.
- Playback on the client using the *xine-lib*-based playback engine (see chapter 8.2)
- playback engine requires demultiplexed MPEG elementary streams for audio and video

Client transport initiation

When the playback request is initiated by the user, the Media Player creates the XinePlayerSink object and (based on the *rtsp://* URL) the RTSPSource object. The latter is responsible for connecting to the content RTSP URL (on the Media Server) and receiving the RTP stream.

As we use the UPnP AV/DLNA infrastructure with its descriptive Content Items[5], the client already knows many properties of the stream even before issuing the RTSP DESCRIBE request: From the content type *video/mpeg* and the `DLNA_PN` field of `MPEG_TS_SD_EU`, the client knows what to expect: A PAL MPEG transport stream with MPEG audio and video in SD resolution (See section 2.3.3. This way, the complete content autodetection that usual streaming media clients have to do can be skipped.

Thus the client can now initiate content transfer using the standard RTSP handshake: The DESCRIBE request returns a SDP document which also contains the audio and video PIDs of the streams.

The PID numbers of the streams are necessary for the client to demultiplex the transport stream, but they are not contained in a DLNA-specified Content Item[2]. Thus we decided to place the PIDs in vendor-specific attributes in the SDP document: The attribute `apid` contains the audio PID(s), the attribute `vpid` contains the video PID (which is also the PCR PID). Table 8.1 contains an example.

Server pipeline creation

During the SETUP request, the server finally creates the server-side pipeline. The server only knows that the client needs a certain content item (here, a DVB television channel) by its URL in a certain container format (MPEG TS). This is to be transmitted to the client via RTSP/RTP.

[1] processing of other audio formats like LPCM or ATSC A/52 a.k.a. Dolby Digital is out of scope for now

[2] The DLNA specification authors intended that the implementers rely on client-side-autodetection of the PIDs via PAT/PMT, see section 2.3.3

```
v=0
o=- 1287058073 1287058073 IN IP4 127.0.0.1
s=3sat
c=IN IP4 239.35.129.11
t=0 0
m=video 0 RTP/AVP 33
a=apid:562
a=control:rtsp://127.0.0.1:10554/content/dvb/tv/T-8468-514-515?apid=562&\
        audio=MP2&container=MPEGTS&ppid=561&video=MPEG2&vpid=561
a=vpid:561
```

Table 8.1: Session Description Document of a DVB TV channel. Note the `apid` and `vpid` attributes with the stream PIDs.

Now the server creates both a source object (*DVBSource*) and a sink object (*RTPSink*). The DVB source provides a PID-filtered MPEG TS stream, which is already frame-aligned (i.e. it produces distinct MPEG-TS frames with 188 bytes each). This is exactly what the *RTPSink* requires, thus the server-side pipeline is now complete (figure 8.1).

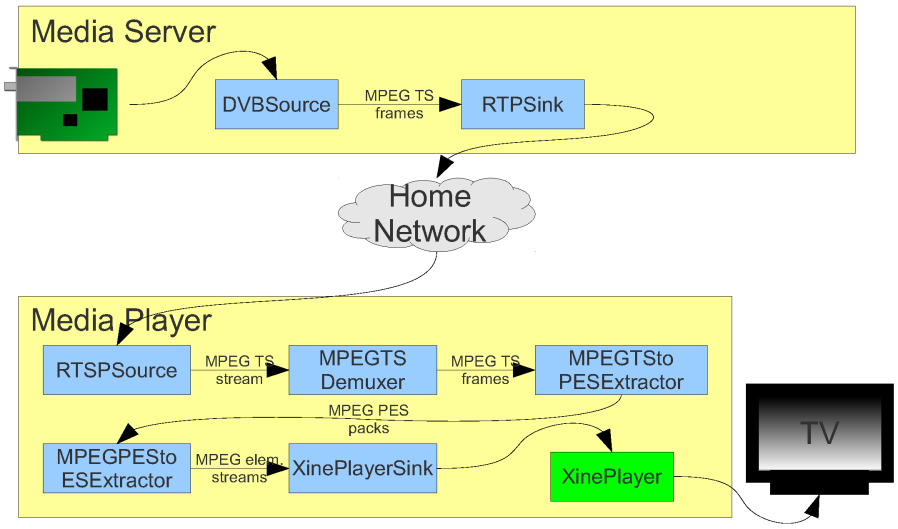

Figure 8.1: Server and client pipeline overview

Client pipeline creation

Meanwhile, the client creates its pipeline. While the RTSP source produces raw MPEG TS data (we do not assume that all RTP streaming servers available produce RTP packets with frame-aligned TS frames), the *XinePlayerSink* needs MPEG elementary streams. After applying the pipeline creation rules, the client pipeline consists of

- The *RTSPSource*
- the *MPEGTSPacketizer*: synchronizes itself on the MPEG TS sync byte and produces distinct MPEG TS frames. It also feeds the PCR timestamps into the software PLL (See chapter6).
- the *MPEGTStoPESExtractor*: demultiplexes the TS into the single MPEG PES (Packetized Elementary Streams) based on the stream PIDs.
- the *MPEGPEStoESExtractor* extracts single MPEG elementary stream frames from the PES packets.
- the *XinePlayerSink* feeds these to the *xine-lib*-based player.

Streaming

Now the client can *start* the pipeline. The *RTSPSource* issues the RTSP PLAY request, and the server commences streaming. See figure 8.1 for an overview.

Changing content

When the currently played content item is changed by the user, it is usually not economic to tear down the complete pipeline including the sink. Thus the pipeline allows the content to be changed on-the-fly, reusing the sink object. All other nodes are rebuilt to match the new content's properties. This saves time and avoids flickering of disappearing and reappearing video decoder windows.

Ending a streaming session

When the user terminates the streaming session, all pipeline components are stopped and destroyed. The end of a streaming session can be apparent if the HTTP connection is closed or the RTSP client sends a `TEARDOWN` request, but if the RTSP session is just ended and the client exits, the server still assumes that a client is listening. To detect such dead clients, the server requires a regular "watchdog" signal from the client in the form of an RTSP `OPTIONS` request. If this request is not received within a 65 second timeframe, the server assumes that the client is no longer listening and stops the RTP transmission.

8.2 The playback engine

Implementing the Media Server developed during the underlying bachelor thesis[13] had the advantage that no direct user interface was necessary. The biggest challenge during the development of the Media Server component was playing back the received DVB channel while keeping in control of the playback clock.

At first, the available playback libraries had to be evaluated. The most prominent available choices were:

- The *FFmpeg*[35] library, used in mplayer[24] and as a backend in many multimedia applications.
- *xine-lib*[37], used among others in the vdr-xine output plugin for VDR[27] and also as a backend in many multimedia frameworks and applications.
- Using an available media player directly by creating a child process.

The last option was not feasible, as the available players don't offer any control over playback timing, and only limited control over content detection. FFmpeg lacked both a stable API[3] and an up-to-date API documentation. *xine-lib* at least provided introductory documentation and up-to-date sample code, somewhat easing the access to writing own applications. So, *xine-lib* became the library of choice for implementing the Media Player.

8.2.1 *xine-lib* overview

xine-lib[37] is a versatile multimedia playback library built using an extensible plugin architecture. This allows applications to influence the playback process at various stages, making it the best choice for the task at hand.

The typical xine playback pipeline consists of the following components:

- `input plugin`: Responsible for getting the stream from the outside, i.e. a file on disk or a network stream
- `demuxer plugin`: Demultiplexes the system stream into its individual (audio, video or subtitle) streams, and also feeds the following plugin with timing information (see below).
- `decoder plugins`: For each elementary stream, the appropriate decoder plugin is loaded which decodes individual frames to an uncompressed format understood by the output plugin.
- `output plugins`: finally render the content (e.g. video to the screen, audio through the sound card). Many output plugins are available, including those that support hardware-accelerated decoding.

Usually, the output plugins are responsible for rendering the streams at the right time and pace. The overall xine playback system timing is controlled by what is called `metronom`, a monotonous clock providing 90kHz timestamps.

[3] the so-called stable releases by the FFmpeg team are usually just snapshots of their current development trunk

8.2.2 xine playback

One way of improving the channel switching delay is to make use of the knowledge about the incoming media stream. Thus any form of content autodetection in *xine-lib* is to be bypassed. As the stream is already demultiplexed at the point when it is to enter the xine playback sink, the input plugin stage of the xine architecture is completely bypassed[4]. Instead, the MPEG elementary streams are handed directly to the demuxer plugin, which in turn distributes them among the decoder plugins.

8.2.3 xine timekeeping

In the xine playback system, all timing is controlled by 90kHz presentation timestamps (PTS). Each playback content unit (e.g. a video or audio frame) is marked with a PTS when it is due to be displayed. This is controlled by a unique time source, the metronom, which is in turn controlled by SCR (system clock reference) plugins. During normal playback from a local disk, the time source (SCR plugin) is usually[5] the local system clock.

As the stream speed is now no longer controllable by local means (i.e. reading faster or slower from disk), the playback speed has to be synchronized to the stream. Thus the standard Unix clock SCR is replaced by a SCR plugin controlled by the DPLL, which is in turn controlled by the PCR timestamps of the incoming stream. Figure 8.2 shows a block diagram.

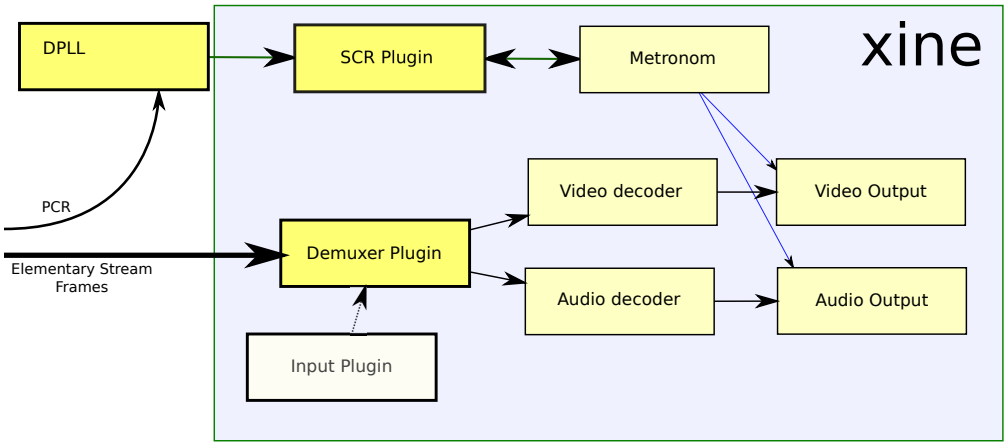

Figure 8.2: Xine interface block schematic

8.2.4 Matching the playback speed

As the playback clocks (graphics card refresh rate, sound card DAC clock) are now potentially deviating from the stream's intended playback speed, the question arises on how to align them. For the video/subtitle part, delaying or advancing single frames is usually not detectable for the

[4] In fact, a dummy input plugin doing nothing was necessary, as xine always expects an input plugin to be present

[5] The *xine-lib* distribution also includes SCR plugins e.g. for hardware MPEG decoder cards.

human eye. For audio on the other hand, playing too fast or too slow would result in playback buffer underruns or overflows. The standard handling in xine in this case is to either drop audio samples (if the stream is to be played faster) or to insert silence (null) samples. For sufficiently small speed deviations, this is tolerable, but for higher deviations, xine does this in rather big chunks, making it audible. A solution could be “real” audio time-shifting algorithms.

All of this of course only works for audio formats that can be processed at sample-level precision. Compressed multi-channel audio (A.52, dts) cannot be stretched/upset this way.

8.3 DVB card management

During the bachelor thesis, a running VDR[27] instance was necessary to provide access to the DVB cards. Now, `rtpserver` itself accesses the DVB hardware directly via the Linux DVB API. This requires an intelligent management of the available DVB tuners, to maximize the number of concurrent streaming operations w.r.t. the number of physical tuners.

8.3.1 Cards and Frontends

In a well-equipped Media Server, there can be more than one tuner, possibly even tuners for different broadcasting systems (e.g. 1 DVB-S, 1 DVB-S2 and 1 DVB-T card). Each card can have multiple frontends, for example there are many twin-tuner cards on the market. And finally, the DVB standards usually provide several *services* (programs) on the same frequency (called a Transponder on DVB-S, or a Multiplex on DVB-T). So the same tuner can serve two different TV stations, if they are located on the same frequency[6].

We present an intelligent DVB card management that keeps track of all available tuners including a priority system.

8.3.2 DVB card inventory

The top level object is a singleton object called *DVBCardInventory*. Upon creation, it detects the available tuners. On Linux systems, the cards usually provide their device nodes under `/dev/dvb/adapter`*N*. This directory is searched for available adapters and each one found is stored in the database. It keeps track of the current card usage.

8.3.3 Card access control

Other classes do not access the cards directly, but rather request a *DVBCardHandle* from the inventory by providing the channel to be tuned and an integer *priority* value ranging from 1 to 100. Similar to VDR[27]s internal card management, the higher priority always wins, allowing e.g. for scheduled recordings to override a running live TV streaming operation.

When a tuner for the selected channel is available, the *DVBCardInventory* hands out the *DVBCardHandle* which is already tuned for the channel.

The card inventory also has the possibility to "revoke" a granted card handle. On the next read attempt by the controlling component, it is notified of that and can end its operation gracefully.

[6] of course including the same parameter such as symbol rate, FEC etc.

8.3.4 Card sharing

Every concrete *DVBCard* object (directly visible only to the *DVBCardInventory*)tracks its usage count on a per-PID-basis. If the application requests a card for a certain channel, the inventory checks where an active card can provide this channel, e.g. because both the new and the already active TV station are broadcast on the same Transponder or Multiplex.

8.4 Components

8.4.1 Class libraries

During the development of both the Media Server and the Media Player, several libraries have been created, which contain generic classes for various purposes. These could also be used in other projects. As this project runs under the "Digital Home Compliance Lab" moniker, all libraries carry the "dhcl" abbreviation in their name.

libdhcl-common

This provides basic infrastructure classes for logging, file and network I/O, multithreading and thread synchronization and various helper methods. It also includes a basic HTTP server implementation. The dependency on 3rd-party libraries is minimal to ease usage in embedded environments.

libdhcl-rtp

libdhcl-rtp covers the RTP/RTSP-related classes, e.g. RTP packet handling, the RTSP server and state handling, SDP (session description protocol) message parsing and rendering.

libdhcl-media

Here, all media stream related classes are contained. It includes the media pipeline framework with most Sources, Sinks and Translators, and the framework to access the Linux DVB API, including classes tracking the usage of all available DVB cards. The glue code to *xine-lib* is included here, too.

8.4.2 libupnp++

libupnp++, started during the *rtpserver*[13] bachelor thesis, already contained everything necessary to implement a UPnP[7]/DLNA[9] compatible UPnP AV[6] Media Server (DMS). During the master thesis, it was augmented with all the functionality necessary to implement a UPnP AV Media Player(DMP), Control Point or any other UPnP device.

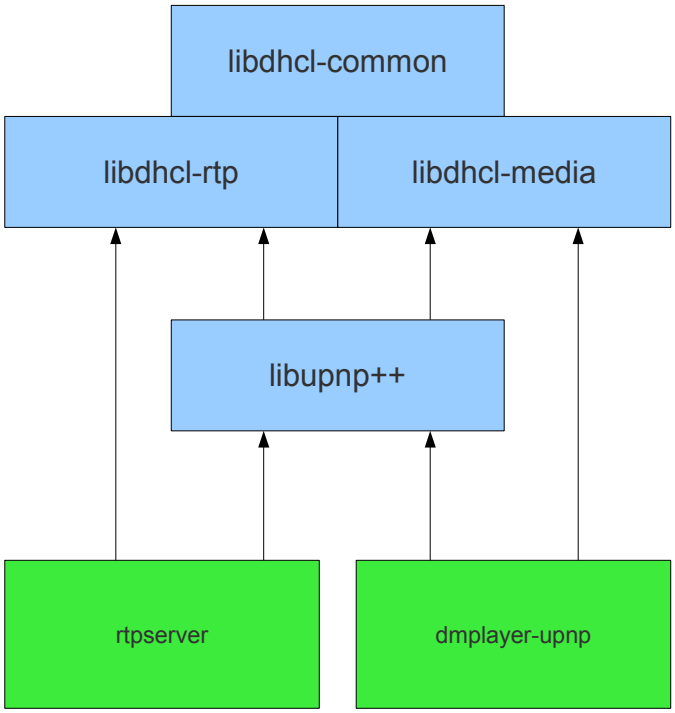

Figure 8.3: Project components

8.4.3 rtpserver

This is the UPnP/AV and DLNA compatible Media Server with live DVB broadcast support, developed during the preceding bachelor thesis. During this master thesis, it has been extended with native Linux DVB support via *libdhcl-media* and newly-implemented RTP support.

8.4.4 dmplayer-upnp

The UPnP/AV compatible Media Player application developed during this master thesis. It is based on dmplayer[12], a file-only media-player frontend.

From dmplayer to dmplayer-upnp

As this work should also provide a Media Player application, there was the need for a graphical user interface that fits the constraints in the living room (TV screen far away, only an IR remote control for interaction). Instead of reimplementing everything, the existing dmplayer[12] was forked, to reuse the GUI and remote control components.

Originally, dmplayer was used to play back audio and video files from network shares mounted by the underlying Linux operating system, using mplayer[24] as a backend, so the media transport itself was beyond its scope.

During this thesis, the software was extended to be a UPnP AV Media Player implementation that uses the *libdhcl-media* pipeline for playback.

User interface

dmplayer-upnp is operated using directional keys (cursor keys, PgUp, PgDown), Enter, Back (Esc) and four color keys (F1-F4 on the PC keyboard). Using them, discovered media servers can be selected and their provided content items are browsed.

Device discovery

After startup, dmplayer-upnp discovers the available UPnP/DLNA Media Servers on the local network. Other devices (e.g. printers) are ignored. Pressing the blue (F4) key triggers a manual discovery. As new devices announce themselves automatically upon entering the network, this is usually not necessary.

Content enumeration

The content objects provided by the media servers can be browsed in their native tree structure.

Additionally, pressing the green (F2) key on a Media Server queries it for all broadcast TV stations and displays them in a list. This is possible using the ContentDirectory Search method, which allows complex queries based on Content Object properties. For example, the search query

```
ContentDirectory:Search(0, "*", 0, 0, "upnp:class derivedfrom
\"object.item.videoItem.videoBroadcast\"", "+upnp:channelNr,+dc:title")
```

filters out all broadcast TV channels on the server, sorted by their channel number and title.

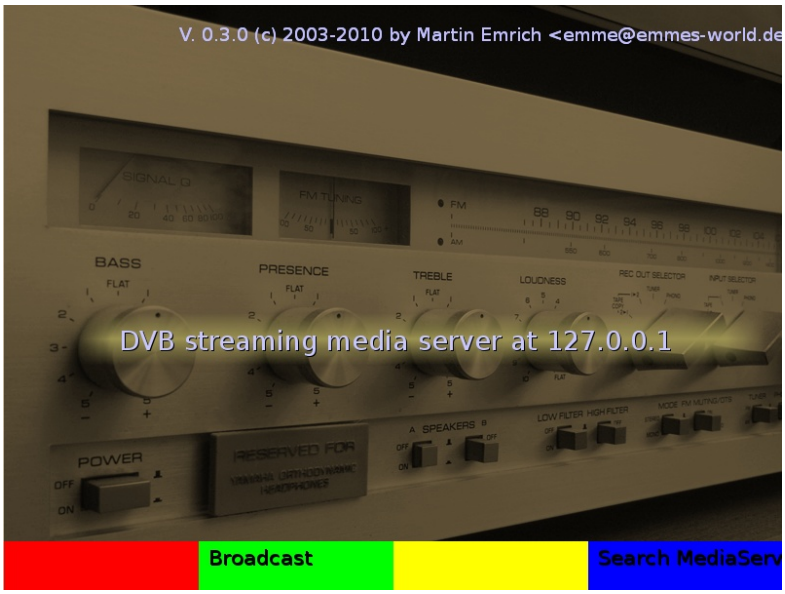

Figure 8.4: dmplayer-upnp showing the list of discovered UPnP Media Servers

Pressing Enter on any of the returned TV stations starts playback. Pressing PgUp/PgDn during playback switches to the next/previous channel (“zapping”). This is again implemented using UPnP ContentDirectory Search queries, by querying for the channel with the lowest channel number higher than the current one. This allows both for gaps in channel numbering as well as a changing set available channels in the Content Directory. Note that the list changes *not* due to changing availability of tuners.

8.5 Standard compliance status

One of the primary goals of this project is to provide software products that adhere to various standards in the field of networked home entertainment. While testing for a certain level of compatibility and interoperability is simple, determining real compliance is either hard due to costly certifications or simply not possible.

8.5.1 Universal Plug'n'Play

As the formal body issuing the UPnP standards does not offer certification, one can only "declare" that both `rtpserver` and `dmplayer-upnp` are UPnP compatible to the UPnP Device Architecture 1.0[7] and adhere to the specifications of the UPnP AV Architecture 1.0[6]. But the following standards do build upon UPnP AV and some offer validation methods.

`rtpserver` : **compatible**. `dmplayer-upnp` : **compatible**

8.5.2 DLNA

The Digital Living Network Alliance[9] offers a Compliance Test Tool (CTT) to its members. Thanks to the Intel affiliation, we had access to both the specifications and the CTT, and thus can assess the level of compatibility of `rtpserver` and `dmplayer-upnp` to the standards up to a certain level. Unfortunately, the CTT was not designed with broadcast content in mind, which is usually infinite: Many of the scripted test cases will not terminate until the content has been streamed completely, which is of course never the case with live broadcast content.

To attain official "compliance", a costly certification process is still necessary, so in the course of this master thesis, achieving real *DLNA compliant* software products is not possible.

`rtpserver` : **compatible**. `dmplayer-upnp` : **unknown**

8.5.3 DVB-HN

`rtpserver` implements most of the 2008 DVB-HN draft specifications[16], but not all of the current (2010) version[18]. As of now, `dmplayer-upnp` is not a full DVB-MR (DVB-MR), as it does not expose any UPnP *device* to be controlled by an external control point. Instead, it has a built-in control point. Note that at the time of writing, no devices were on the market implementing DVB-HN[7].

`rtpserver` : **compatible**. `dmplayer-upnp` : **compatible**

[7] At least none were found on the web, any reasonable web search for DVB-HN devices results only in links to the DVB-HN standards themselves.

9 Conclusion

Integrating broadcast content in today's home entertainment networks is possible and desirable. The software components presented here provide a minimal, extensible infrastructure for a two-box scenario which provides DVB television in the UPnP AV/DLNA/DVB-HN home network.

The DLNA standard provides a basic infrastructure for networked home entertainment devices, although focused on stored content. DVB-HN extends this to live media broadcast content, including service information and EPG. Both DLNA and DVB-HN only make minimal but constraining provisions regarding error correction. Minimal changes to the specifications would allow integrating an advanced but backwards-compatible AHEC protection.

A simple, TCP-based fast-channel-change scheme together with retransmission-based error correction could reduce the initial rendering delay of RTP-based streaming sessions.

With these proposals, an IP-based home infrastructure could deliver the same quality of experience end users are accustomed to from existing digital broadcast.

List of Figures

Glossary

Bouquet A logical collection of multiple broadcast stations, usually (but not always) bundled on the same frequency. They usually belong to the same broadcasting company. See Transponder or Multiplex. 39

Multiplex In DVB-T broadcasting, a Multiplex is a set of services (TV, radio, ...) broadcast on the same frequency as one single MPEG transport stream.. 49, 57

SOAP Originally called Simple Object Activation Protocol, SOAP allows remote procedure calls (RPC) using a HTTP and XML based network protocol. Object requests are sent as HTTP requests with an XML body, return values are returned as XML documents.. 14

Transponder In terms of DVB-S broadcasting, transponder is a certain set of tuning parameters (frequency, symbol rate, FEC, ...) which carries an MPEG transport stream. Multiple services (TV, radio, ...) can be carried on a transponder.. 49, 57

Trick Mode Trick Mode describes playback of streams in a nonlinear fashion, including pausing the stream or jumping backwards/forwards in the stream.. 41

Acronyms

AHEC An adaptive combination of FEC and ARQ. 22–25, 59

ARQ Automatic Repeat Request: Error correction by retransmission of lost packets. 21–23, 58

DHCP A DHCP server assigns IP addresses to DHCP client hosts.. 13

DMC Digital Media Controller: A UPnP AV control point in DLNA context.. 15

DMP Digital Media Player: In DLNA and DVB context, a network node capable of discovering content and playing it back. 15, 43

DMR Digital Media Renderer: In DLNA and DVB context, a network node capable of playing back content, but without the capability to discover the content. Has to be controlled by a compatible Control Point. 15

DMS Digital Media Server: In DLNA and DVB context, a network node capable of serving content items. 15

DNG In DVB terms, a device connecting the home network to a delivery network. This is usually a broadband router. 17

DVB The European standard for digital TV and radio broadcasting. 17

DVB-BGD A DVB Bidirectional Gateway Device connects the home network to a DVB delivery network carrying DVB-IPI, translating DVB-IPI protocols to DVB-HN/DLNA conforming home network protocols. 17, 40, 41

DVB-MR A DMR that also adheres to the DVB-HN specifications.. 53

DVB-MS A DMS that also adheres to the DVB-HN specifications.. 17, 43

DVB-UGD A DVB Unidirectional Gateway Device connects the home network to a receive-only access network such as DVB-S/T/C. 17, 30, 41, 43

DVBSTP DVB SD&S Transport Protocol. 40

EPG A text-based service usually transmit together with the broadcast signal describing the current and upcoming TV program with show names, description and additional metadata.. 12, 39

ETSI European Telecommunications Standards Institute: The European standardization institution for electronics and telecommunication related topics.. 17

FEC Correction of transmission errors by adding redundancy packets to the data. 21, 25, 58

GOP Group Of Pictures: In an MPEG video stream, a GOP is the smallest payload unit that can be decoded independently. 28

HND In DLNA context, regular device classes in the home entertainment network. 15

HNED In DVB terms, a Home Network End Device is any device receiving DVB transmissions over the home network, such as set-top-boxes or DVB-HN-compatible TV sets.. 18, 24

HTTP TCP-base protocol primarily used on the World Wide Web.. 19

IGMP Subscription to multicast channels are managed via the Internet Group Management Protocol[2].. 17, 20

MHD In DLNA context, device classes with different content format requirements than a NHD. 15

NCO Numerically Controlled Oscillator: A frequency generator where the frequency can be controlled numerically (digital). 33

PCR Program Clock Reference: A 27MHz reference clock signal embedded in a MPEG transport stream. It is split into a 90kHz base clock and a 27MHz extension.. 20, 34, 43

PID Program Identifier: Identifies a unique program (usually a single audio or video elementary stream) in a MPEG transport stream.. 20, 43–46, 50

PRRT Predictable Reliable Realtime Transport: A IP protocol implementing AHEC.. 23

PSI Program Specific Information: In a transport stream, the data that describes the contained programs.. 20, 29

RTCP Real Time Control Protocol[28]. 19

RTP RTP: A Transport Protocol for Real-Time Applications[28]. 17, 19, 33

RTSP Real Time Streaming Protocol[29]. 19

SDP Session Description Protocol: A text-based format for describing RTP session parameters.. 19, 23, 24, 29, 44

SSDP Simple Service Discovery Protocol: A protocol to discover other SSDP-enabled devices on an IP network.. 13

SSRC Synchronization Source: 32-bit identifier of a single stream source in a RTP session. 19

UPnP Universal Plug'n'Play: A standard for discovering and controlling devices on a network without configuration. 13

Bibliography

[1] A. Li, Ed. RTP Payload Format for Generic Forward Error Correction, December 2007. http://www.ietf.org/rfc/rfc5109.txt.

[2] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan. Internet Group Management Protocol, Version 3, October 2002. http://www.ietf.org/rfc/rfc3376.txt.

[3] S. Cheshire, B. Aboba, and E. Guttman. Dynamic Configuration of IPv4 Link-Local Addresses. RFC 3927 (Proposed Standard), May 2005. http://www.ietf.org/rfc/rfc3927.txt.

[4] J. Cohen and S. Aggarwal. General Event Notification Architecture Base. IETF Draft (expired), July 1998. http://quimby.gnus.org/internet-drafts/draft-cohen-gena-p-base-01.txt.

[5] Contributing Members of the UPnP Forum. ContentDirectory:1 Service Template Version 1.01, June 2002.

[6] Contributing Members of the UPnP Forum. UPnP AV Architecture:0.83 v1.0, 2002.

[7] Contributing Members of the UPnP Forum. UPnP Device Architecture v1.0, July 2006.

[8] Contributing Members of the UPnP Forum. Universal Plug and Play forum website, November 2010. http://www.upnp.org.

[9] Digital Living Network Alliance. Digital Living Network Alliance: official web site. http://www.dlna.org.

[10] Digital Living Network Alliance. DLNA Networked Device Interoperability Guidelines Volume 1: Architecture and Protocols, October 2006.

[11] Digital Living Network Alliance. DLNA Networked Device Interoperability Guidelines Volume 2: Media Format Profiles, October 2006.

[12] Martin Emrich. DMPlayer: A diskless media player, 2003. http://www.emmes-world.de/dmplayer.

[13] Martin Emrich. Implementing a DLNA-compliant UPnP AV MediaServer with DVB and RTSP/RTP support, April 2009.

[14] ETSI. ETSI EN 300 468 V1.11.1: Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems. http://www.etsi.org/deliver/etsi_en/300400_300499/300468/01.11.01_60/en_300468v011101p.pdf, April 2010.

[15] European Telecommunications Standards Institute. ETSI TS 102 034 V1.3.1: Digital Video Broadcasting (DVB); Transport of MPEG-2 TS Based DVB Services over IP Based Networks, 2007.

[16] European Telecommunications Standards Institute. DVB-HN (Home Network) Reference Model Phase 1 (Draft 2008, 2008. Draft from 2008.

[17] European Telecommunications Standards Institute. ETSI TS 102 034 V1.4.1: Digital Video Broadcasting (DVB); Transport of MPEG-2 TS Based DVB Services over IP Based Networks, August 2009. http://www.etsi.org/deliver/etsi_ts/102000_102099/102034/01.04.01_60/ts_102034v010401p.pdf.

[18] European Telecommunications Standards Institute. DVB-HN (Home Network) Reference Model Phase 1 (2010). http://etsi.org/deliver/etsi_ts/102900_102999/102905/01.01.01_60/ts_102905v010101p.pdf, May 2010.

[19] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1, June 1999. http://www.ietf.org/rfc/rfc2616.txt.

[20] Yaron Y. Goland, Ting Cai, Paul Leach, Ye Gu, and Shivaun Albright. Simple Service Discovery Protocol/1.0 Operating without an Arbiter. IETF Draft (expired), October 1999. ftp://ftp.pwg.org/pub/pwg/ipp/new_SSDP/draft-cai-ssdp-v1-03.txt.

[21] Jochen Grün and Manuel Gorius. Predictably Reliable Realtime Transport.

[22] M. Handley and V. Jacobson. SDP: Session Description Protocol, April 1998. http://www.ietf.org/rfc/rfc2327.txt.

[23] ISO/IEC. ISO/IEC 13818-1: Information technology — Generic coding of moving pictures and associated audio information: Systems, December 2000.

[24] MPlayer Team. MPlayer, 2000. http://www.mplayerhq.hu.

[25] J. Ott, S. Wenger, H. Sato, C. Burmeister, and J. Rey. Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF), July 2006.

[26] J. Rey, D. Leon, A. Miyazaki, V. Varsa, and R. Hakenberg. RTP Retransmission Payload Format, July 2006.

[27] Klaus Schmidinger. Video Disk Recorder. http://www.cadsoft.de/vdr.

[28] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications, July 2003. http://www.ietf.org/rfc/rfc3550.txt.

[29] H. Schulzrinne, A. Rao, and R. Lanphier. Real Time Streaming Protocol (RTSP). RFC 2326, April 1998. http://www.ietf.org/rfc/rfc2326.txt.

[30] Guoping Tan and Thorsten Herfet. RTP-Level Hybrid Error Correction for DVB Systems in Wireless Home Networks. In *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*. IEEE, April 2006.

[31] Guoping Tan and Thorsten Herfet. The Optimization of RTP Level Hybrid Error Correction Scheme for DVB Systems Over Wireless Home Networks Under Restrict Delay Constraint. In *IEEE Transactions on Broadcasting, Vol 53, Issue 1, Part 2*, pages 297–307. IEEE, March 2007.

[32] Guoping Tan, Thorsten Herfet, and Manuel Gorius. Evaluation of the Performance of a Hybrid Error Correction Scheme for DVB Services over IEEE 802.11a. In *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*. IEEE, 2007.

[33] The DVB Project. DVB-HN (Home Network) Reference Model Phase 1. `http://www.dvb.org/technology/standards/a109.tm3690r2.DVB-HN_ref_model.pdf`, February 2007.

[34] The DVB Project. A152 Server-Based Fast Channel Change for DVB-IPTV Systems. `http://www.dvb.org/(RoxenUserID\x3d8f405878aa20d5b6a74baf2c98db08c2)/technology/standards/a152_server_based_FCC.pdf`, August 2010.

[35] The FFmpeg developers. FFmpeg. `http://www.ffmpeg.org`.

[36] The Linux Kernel developers et al. Linux man-pages, December 2010. `http://kernel.org/doc/man-pages/`.

[37] The xine developers. xine - a multimedia software library. `http://www.xine-project.org`.

[38] B. VerSteeg, A. Begen, T. VanCaenegem, and Z. Vax. Unicast-Based Rapid Acquisition of Multicast RTP Sessions. IETF Draft (expires on Nov 30 2010), May 2010. `http://tools.ietf.org/pdf/draft-ietf-avt-rapid-acquisition-for-rtp-10.txt`.