# IAEMStream - an On-Demand Streaming Server Product

Georg Holzmann
http://grh.mur.at
mail: grh@mur.at

Abstract

IAEMStream is an on-demand streaming server product for the open source content management system Zope/Plone. It is possible to handle audio files in various codecs and streams them in MP3 format over the RTSP protocol, using the open source Helix DNA Streaming Server. Additionally it takes over the right-management of Plone, therefore one has a fine-grained control over who is allowed to stream or download specific media files. This report discusses the underlying technologies and the IAEMStream product itself.

February 2008

Institute of Electronic Music and Acoustic (IEM)
University of Music and Dramatic Arts Graz, Austria

Advisor: Ao.Univ.-Prof. DI Winfried Ritsch

## Contents

# 1 Introduction

IAEMStream is an on-demand streaming server product for the open source content management system Zope/Plone[1], written in python.

The initial motivation was to provide students an access to the Archive of Electronic Music at the Institute for Electronic Music [2], University of Music and Dramatic Arts Graz. This archive consists also of material which is unfortunately not copyright-free, therefore the streams must not be available freely for the public. IAEMStream takes over the right-management of Plone and so it can be well defined who is able to download or stream specific media files. The RTSP protocol (real time streaming protocol) is used for on-demand streaming, which makes it legal for students to access the library over the internet in a comfortable way.

This document tries to give an overview of underlying streaming technologies (codecs, protocols and etc.) in section 2. Then existing open source streaming servers are discussed and evaluated in section 3 and finally the IAEMStream product is explained in section 4.

# 2 Streaming Technologies

This chapter describes some commonly used streaming technologies: multimedia codecs, streaming methods and relevant network protocols for streaming. It is thought as an overview of some possibilities and for more detailed information the given references should be considered. In general these technologies are relevant for audio and video streaming, but the focus in the report is on audio streaming.

## 2.1 Multimedia Codecs

Streaming media needs network bandwidth and as audio data is quite big it needs some compression. To be able to receive a realtime stream, the datarate of the compressed audio file must be at least smaller than the network bandwidth.

There is a distinction in "lossless" and "lossy" compression algorithms. Lossless compression algorithms use methods (e.g. Huffman coding) which can be exactly reconstructed, whereas lossy algorithms throw away information which is e.g. not perceived by humans. Some examples of lossy audio codecs are Dolby Digital (AC3), MPEG 1 Audio Layer III (MP3), Advanced Audio Codec (AAC), MPEG 2 and 4 standards, Ogg-Vorbis, ...

In the next two subsections the wellknow MPEG 1 Audio Layer III (MP3) codec and the newer, open-source Vorbis audio codec is presented.

### 2.1.1 MPEG 1 Audio Layer III

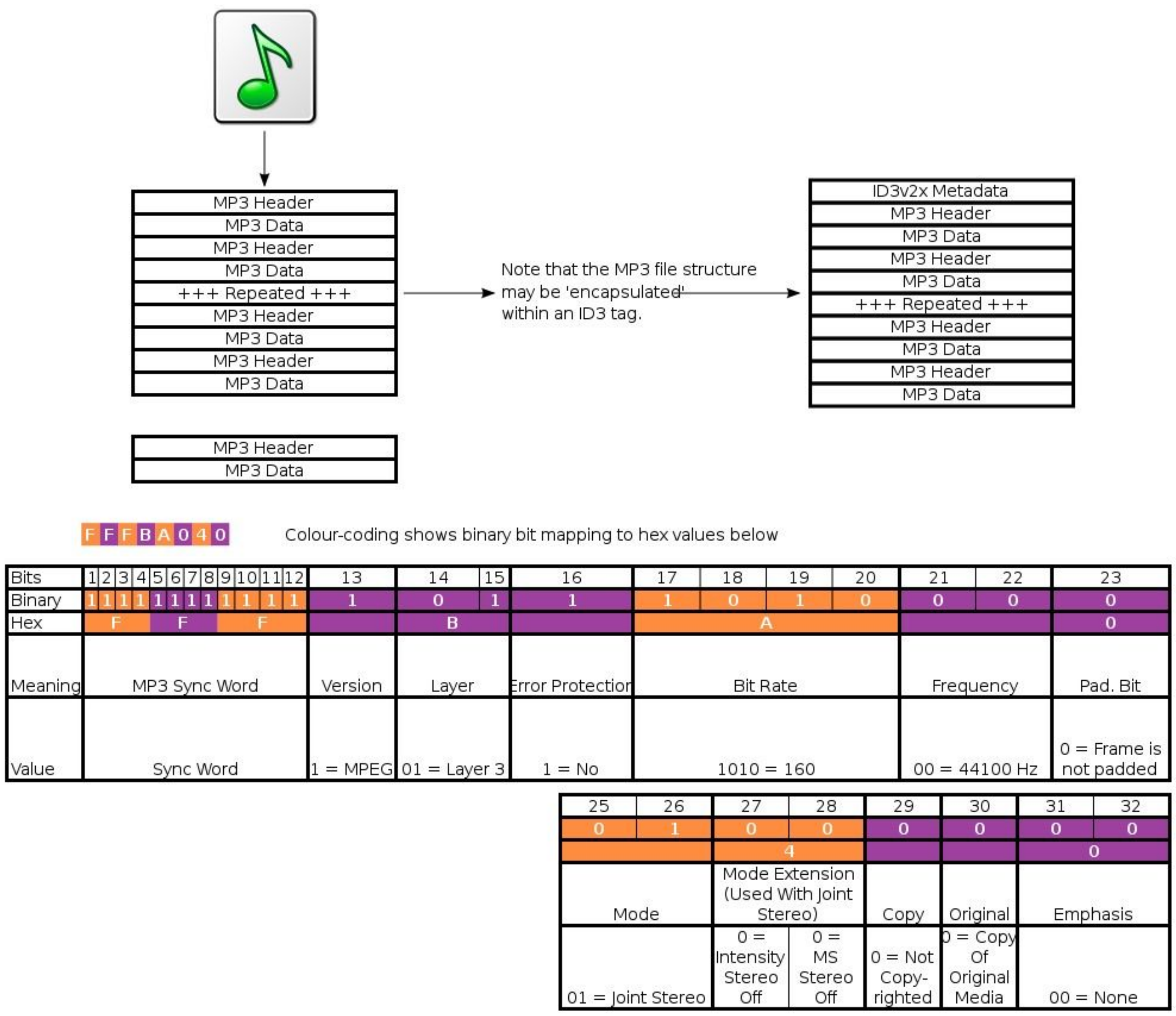
The biggest advantage of MP3 is that it is very widespread, especially in the internet. MP3 compression works by reducing the accuracy of certain parts of the sound (see [10]). These parts are not perceived by most people, therefore this method is commonly referred to as perceptual coding. Additionally a modified discrete cosine transform (MDCT) is used to increase the resolution in high frequency bands (see [5]).

There are many different MP3 encoders available, each producing files of differing quality, because the MPEG-1 standard does not include a precise specification for an MP3 encoder.

[1]Plone Webpage: http://plone.org/
[2]Institute of Electronic Music Graz: http://www.iem.at

E.g. LAME [3] is a free software application used to encode audio into the MP3 file format, which is also used by the IAEMStream product to encode files. Decoding, on the other hand, is carefully defined in the standard.

An MP3 file is structures in multiple MP3 frames, which consist of the MP3 header and the MP3 data (see figure 1). This sequence of frames is called an elementary stream. The MP3



| Bits | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Binary | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| Hex | F |  |  |  | F |  |  |  | F |  |  |  |  | B |  |  | A |  |  |  |  |  | 0 |
| Meaning | MP3 Sync Word |  |  |  |  |  |  |  |  |  |  |  | Version | Layer |  | Error Protection | Bit Rate |  |  |  | Frequency |  | Pad. Bit |
| Value | Sync Word |  |  |  |  |  |  |  |  |  |  |  | 1 = MPEG | 01 = Layer 3 |  | 1 = No | 1010 = 160 |  |  |  | 00 = 44100 Hz |  | 0 = Frame is not padded |

| Bits | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
|---|---|---|---|---|---|---|---|---|
| Binary | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Hex | 4 |  |  |  | 0 |  |  |  |
| Meaning | Mode |  | Mode Extension (Used With Joint Stereo) |  | Copy | Original | Emphasis |  |
| Value | 01 = Joint Stereo |  | 0 = Intensity Stereo Off | 0 = MS Stereo Off | 0 = Not Copy-righted | 0 = Copy Of Original Media | 00 = None |  |

Figure 1: Diagram of the structure of an MP3 file. The table shows the bits and their meanings of the frame header. (picture from [10] and modified)

header consists of a sync word (Bit 1 to 12), which is used to identify the beginning of a valid frame.

Additionally it is possible to add ID3 metadata, which is stored before or after the MP3 frames. ID3 is a metadata container and allows information such as the title, artist, album, track number, or other information about the file to be stored in the file itself.

The most relevant problems with the MP3 codec, at least for our usage, are license problems. A large number of different organizations, like the Frauenhofer Gesellschaft, have claimed

[3]LAME project page: `http://lame.sourceforge.net/`

ownership of patents necessary to implement the algorithms. The various patents claimed to cover MP3 by different patent-holders have many different expiration dates, ranging from 2007 to 2017 in the U.S. according to [7].

There are also several design limitations inherent to the MP3 format that cannot be overcome by any MP3 encoder (see [10]). Theses issues are fixed in newer audio compression formats such as Vorbis (section 2.1.2) and AAC.

### 2.1.2 Vorbis

Vorbis is a free and open source, lossy audio codec project headed by the Xiph.Org foundation and intended to serve as a replacement for MP3. It is most commonly used in conjunction with the Ogg container and is therefore called Ogg Vorbis. The development of this codec was initiated in 1998 after a letter of the Frauenhofer Gesellschaft where they announced plans to charge licensing fees for the MP3 audio format.

For many applications, Vorbis has clear advantages over other lossy audio codecs in that it is patent-free and has free and open-source implementations and therefore is free to use, implement, or modify as one sees fit, yet produces smaller files than most other codecs at equivalent or higher quality (from [12]).

Vorbis also uses the modified discrete cosine transform (MDCT) for converting sound data from the time domain to the frequency domain. The resulting frequency-domain data is broken into noise floor and residue components, and then quantized and entropy coded using a codebook-based vector quantization algorithm. The decompression algorithm reverses these stages. The noise floor approach gives Vorbis its characteristic analog noise-like failure mode (when the bitrate is too low to encode the audio without perceptible loss), which many people find more pleasant than the metallic warbling in the MP3 format (from [12]). Vorbis also inherently uses variable bitrates (VBR), so bitrate may vary considerably from sample to sample.

Various listening tests were performed to find the best quality lossy audio codecs at certain bitrates. Some conclusions from [12] are the following (see also the references in [12]):

- Low bitrates (less than 64 kbit/s): the most recent public multiformat test at 48 kb/s shows that Vorbis (aoTuV) has a better quality than WMA and LC-AAC, the same quality of WMA Professional, and a lower quality than HE-AAC.

- Mid to low bitrates (less than 128 kbit/s down to 64 kb/s): private tests at 80 kb/s and 96 kb/s shows that Vorbis (aoTuV) has a better quality than other lossy audio codecs (LC-AAC, HE-AAC, MP3, MPC, WMA).

- Mid bitrates (128kb/s): most recent public multiformat test at 128 kb/s shows a four-way tie between Vorbis (aoTuV), LAME-encoded MP3, WMA Pro, and QuickTime AAC, with each codec essentially transparent (sounds identical to the original music file).

- High bitrates (more than 128 kb/s): most people do not hear significant differences. However, trained listeners can often hear significant differences between codecs at identical bitrates, and Vorbis (aoTuV) performs better than LC-AAC, MP3, and MPC.

Although Vorbis has many advantages compared to MP3 it definitely did not replace the MP3 codec, which is still the most used lossy audio codec in the internet. However, in some areas, like e.g. in computer games [4], it is quite popular.

Of course it was also our goal to use Vorbis as codec for the IAEMStream product, but unfortunately all streaming servers which support Vorbis have some other major disadvantages. Therefore we had to use MP3.

## 2.2 Streaming Methods

In principle there are two basic scenarios of streaming: live, multicast and on-demand, unicast streaming.

Live streaming works basically like radio or television. There is one sender which transmits exactly the same stream to all clients (see figure 2). This method is of course much easier

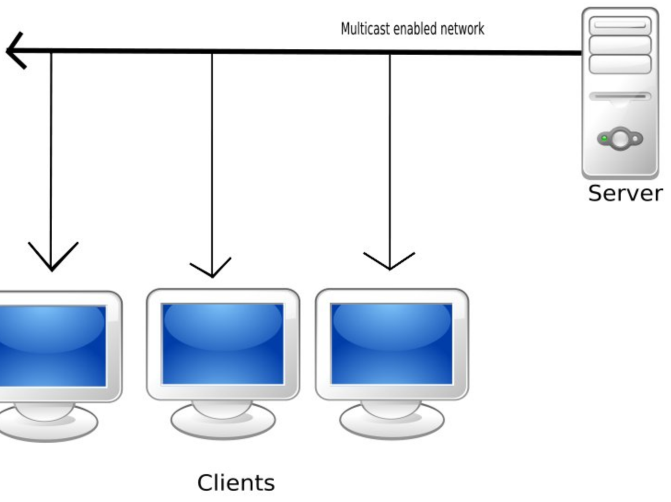


Figure 2: Multicast Streaming: a server sends the same data to all clients. (picture from [11])

to handle for the server, because it can send the same data to all clients (therefore the term multicast), which saves network bandwidth. Usually the clients don't have possibilities to influence the stream, besides turning it on or off.

With on-demand streaming each user can receive the data he likes at the time he wants, like it is possible to choose an audio CD and listen to it at any time. Therefore the server has to send a separate data stream to each client (unicast - see figure 3). In terms of difficulty of implementing technically, unicast protocols are the most simplistic. At the cost of this simplicity, there can be massive duplication of the data being sent on the network. Usually the clients can also influence the stream they receive: choosing the stream, seeking in the stream and etc. On-demand streaming is also the use-case of the IAEMStream product: we have lots of audio files on the server and each client should be able to listen to them at any time.

There exist also streaming media servers which try to combine unicast and multicast solutions to both cut down on the bandwidth requirements and provide users most of the on-demand functionality of a pure unicast.

[4]games that use vorbis: http://wiki.xiph.org/index.php/Games_that_use_Vorbis

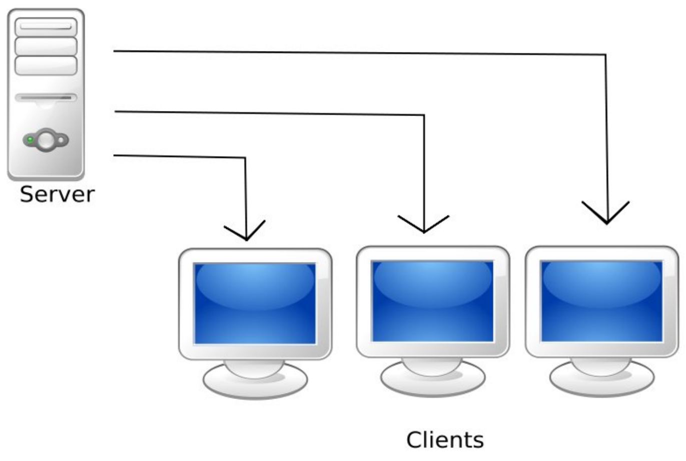


Figure 3: Unicast Streaming: the server sends a separate data stream to each client. (picture from [11])

## 2.3 Streaming Protocols

Network protocols are needed to transmit data between different computers. It is an agreement, how the communication between two or more computers should happen. Special network protocols suitable for streaming can be used.

Networks are usually described with the OSI reference model, which is a layered abstract description for communications and computer network protocol design (see figure 4).

For the actual data transfer of a media stream the transport control protocol (TCP) or the user datagram protocol (UDP) can be used, which are part of the layer 4, the transport layer. TCP provides a reliable transmission of data packets. Lost packets are recognized and sent again, making it suitable for applications like file transfer and e-mail. UDP on the other hand does not guarantee reliability or ordering in the way that TCP does. Data-packets may arrive out of order, appear duplicated, or go missing without notice. Avoiding the overhead of checking whether every packet actually arrived makes UDP faster and more efficient. Time-sensitive applications, like streaming media, usually use UDP because dropped packets are preferable to delayed packets (from [5]).

The next subsections discuss some protocols of the application layer, which are often used for streaming media.

### 2.3.1 HTTP

HTTP is maybe one of the best known protocols because it was designed for the transfer of information on intranets and the world wide web. Its original purpose was to provide a way to publish and retrieve hypertext pages over the internet. HTTP needs a reliable transport protocol and therefore almost always uses TCP.

The communication between a client and a server consists of two parts: a request of the client and a following response from the server. After a successful response, the client can send

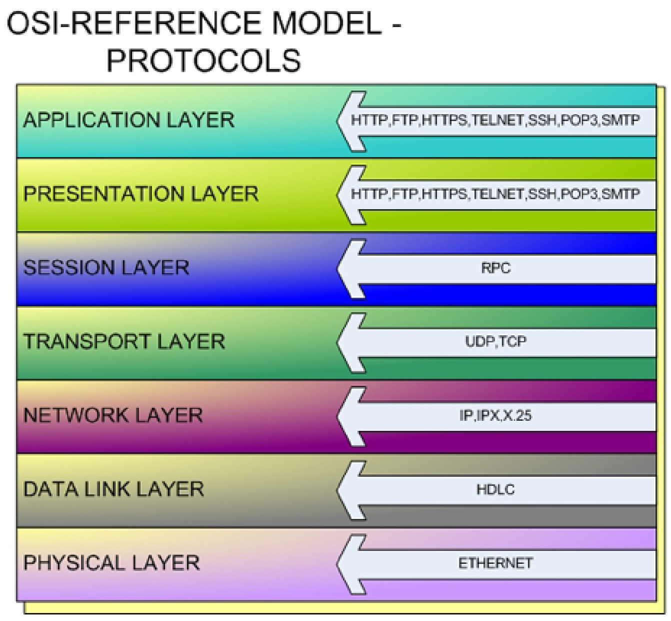


Figure 4: The OSI model: a layered abstract description of communications and computer network protocol design. Each layer describes a specific functionality.

further requests until the communication is terminated.

Although HTTP was designed for file transfer it is also used for streaming media. In the on-demand case the media file is transmitted like in a regular download and afterwards it can be used like a regular file. Usually it is not possible to seek in such an HTTP-stream, but some webservers (like for example apache) have special modules so that also seeking is somehow possible.

As explained in [6] HTTP inherits the properties of the transport protocol TCP. This means that if some packets are lost they will be sent again and the client has to wait for them, which can result in dropouts. To decrease the number of dropouts data buffers are used. However, as HTTP was not designed for realtime streaming other protocols should be considered.

### 2.3.2 RTP/RTCP

The Real-Time Transport Protocol (or RTP) defines a standardized packet format for delivering audio and video over a network. It is often used in audio- and video-conferences and other real time applications such as live streaming.

RTP can use UDP or TCP as transport protocol, but in real time applications latency is important and therefore usually UDP is used. With UDP some data packets might get lost, but the latency is lower and it's also possible to send a stream to only one receiver (unicast) or to many receivers (multicast).

RTCP (Real-Time Control Protocol) is the control protocol of RTP. Via RTCP the transmitter and receiver can send feedback about the quality of the data stream. Parallel to RTP the server and clients send RTCP packets in regular intervals. Applications can use these RTCP information to change datarate, resolution and etc. of the stream and adapt them to the requirements of the clients. RTCP uses a different network port than RTP, usually the next higher one (from [5]).

The RTP header of each packet also includes a timestamp and a sequence number. Since with UDP it is not guaranteed that each packet will receive the client, a media player can reorder the received packets according to their sequence number (see figure 5).

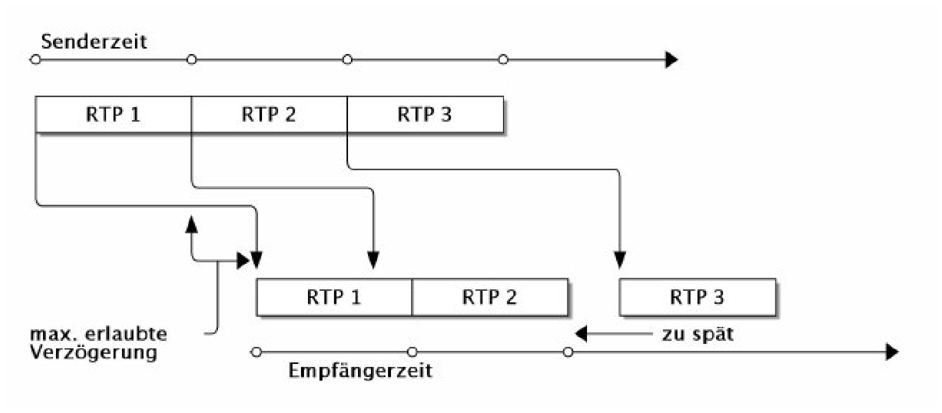


Figure 5: This picture shows three RTP packages on the sender (above) and on the receiver (bottom). Packet RTP2 receives the client too early and is delayed a little bit. Packet RTP3 comes too late, therefore it can't be played and will be skipped. (picture from [6])

### 2.3.3 RTSP

The Real-Time Streaming Protocol (RTSP) is used to control RTP data streams. It can be compared to a remote control of a video recorder, where one can send start, stop and seek commands. The media stream itself is usually transmitted via RTP and not via RTSP.

RTSP, in contrary to RTP, most often uses TCP as transport protocol and knows several states as shown in figure 6:

- *Init*: Client sends a SETUP request and waits for an answer of the server.
- *Ready*: Client received the answer to its SETUP request or to a PAUSE request in playing state.
- *Playing*: Server received PLAY request of the client and is now sending data.
- *Recording*: Server received RECORD request of the client and is recording data.

For reliable on-demand streaming it was important for us to use the RTSP protocol for the IAEMStream product.

## 3 Evaluation of Existing Streaming Servers

Choosing the right streaming server was not as easy as thought before. Quite some solutions exist in the open source world, but it was not clear from the beginning which one is most appropriate for the IAEMStream product.

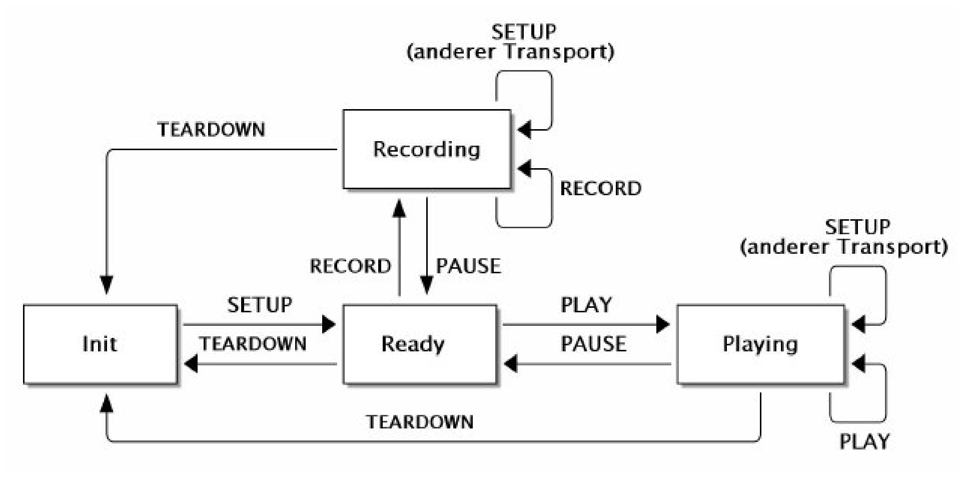


Figure 6: States of the RTSP protocol. For description see text. (picture from [6])

To fulfill our desire a streaming server should have the following properties:

- Stable open source streaming server which runs under Linux
- Support for on-demand streaming
- RTSP/RTP as protocol
- Seeking in the on-demand stream should be possible
- Vorbis as audio codec
- Commonly used, stable and open source clients should exist for Linux, Windows and Mac, which are able to receive the stream
- Stable browser plugins should exist for Mozilla, Internet Explorer and Safari to be able to play and control the stream directly in the content management system

In the following subsections existing streaming servers are discussed and problems we experienced are listed. However, we made our experiments at the beginning of the year 2007, therefore have in mind that everything could be quite different already at the time of this writing.

## 3.1 Icecast

Icecast [5] is a free streaming media project maintained by the Xiph.org Foundation. It was created in December 1998/January 1999 to provide an open source audio streaming server.

The Icecast server is capable of streaming content as vorbis, MP3, theora video, AAC and NSV over the HTTP protocol. The streams have to be encoded with external programs, called source clients (from [9]).

[5]Icecast Webpage: http://www.icecast.org/

Icecast is very common in the open source community and is used for many free internet radio/tv stations. However, it is not able to stream media on demand.

Problems with Icecast:

- No on-demand streaming, which is mandatory for the IAEMStream product.
- Streaming protocol is HTTP, no RTSP possible.

## 3.2 Darwin Streaming Server

The Darwin Streaming Server is a full featured open sourced RTSP/RTP media streaming server and capable of streaming a variety of media types including H.264/MPEG-4 AVC, MPEG-4 Part 2 and 3GP. It is developed by Apple and Mac OS X's QuickTime Streaming Server is based on the Darwin Streaming Server (see [1]).

For a reliable transmission to many clients the stream can be distributed over multiple servers (see figure 7), so that the network transmission at each server can be reduced.

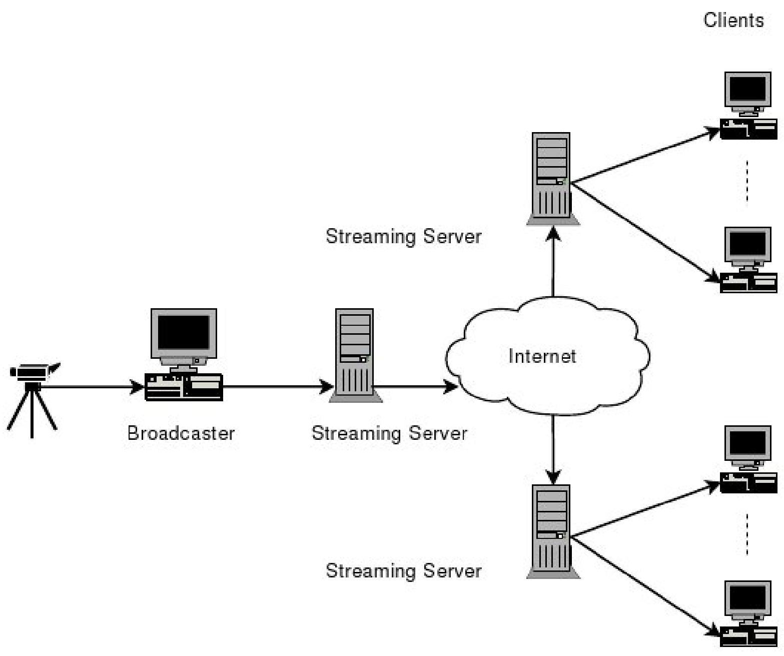


Figure 7: Darwin Streaming Server: distribution of a stream over multiple servers. (picture from [5])

The Darwin Streaming Server is able to stream media on-demand or live and common clients (e.g. quicktime or VLC player) are able to receive the stream.

Problems:

- If one wants to stream only audio in MP3 format, the MP3 file has to be embedded in a MPEG-4 video file. Then it can be streamed within playlists, but no seeking is possible.

- No possibility to stream ogg vorbis.

## 3.3 VLC

VLC [6] is the VideoLAN open source media player which can be used as a server and a client to stream and receive network streams. VLC is able to stream all that it can read (see figure 8). There exists also VLS (VideoLAN Server), which can stream MPEG-1, MPEG-2 and MPEG-4 files, DVDs, digital satellite channels, digital terrestrial television channels and live videos on the network in unicast or multicast. However, most of the VLS functionality is already integrated in VLC and usage of VLC instead of VLS is advised (from [8]).

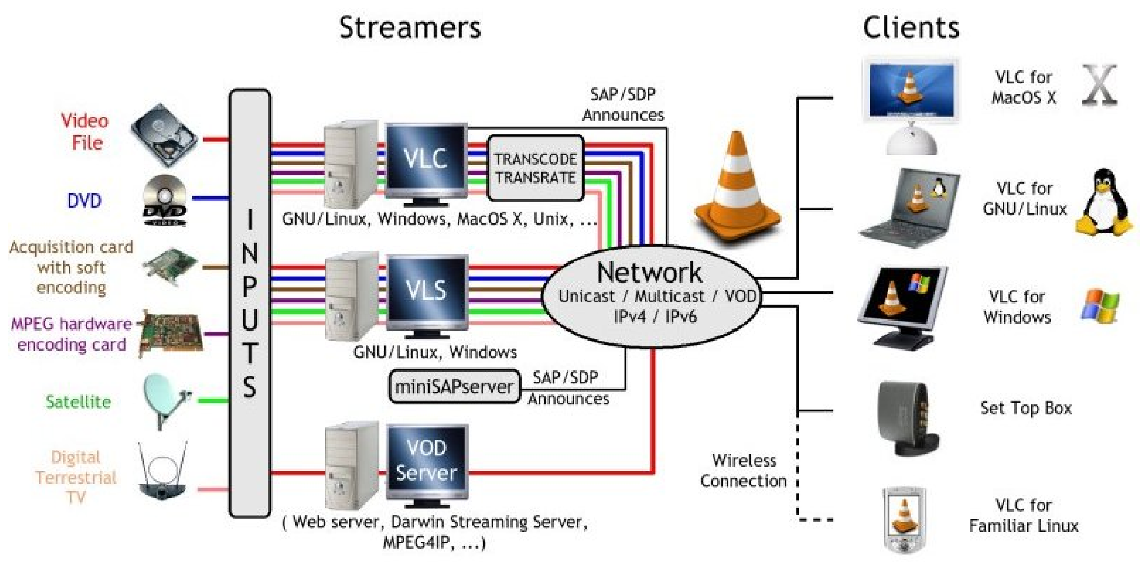


Figure 8: VideoLAN (VLC) streaming solution overview. (picture from [8])

VLC is able to stream live and on-demand and over various network protocols like HTTP, RTP/RTSP or MMS. The server is also controlable with a telnet interface or HTTP, which is quite handy.

Altogether the VLC solution is quite attractive because it really supports many codecs and protocols, therefore it exists also a branch in the SVN repository of the IAEMStream product, which uses VLC. However, at the time of our experiments we had the following problems and stopped the development of the VLC branch:

- With RTSP streaming we had problems reading the stream from other clients than VLC and the VLC mozilla plugin was very buggy.
- Seeking was not always possible in the VLC mozilla plugin with RTSP streams.
- Ogg vorbis streaming over RTSP was not possible, only HTTP.

[6]VLC Webpage: http://www.videolan.org/

## 3.4 Helix DNA Streaming Server

Helix DNA Server [7] is the open source version of Real Networks commercial streaming server (Helix Server). By releasing the source of the Helix Server, Real Networks expected a better market position compared to the growing competitors Apple and Microsoft.

The Helix Server supports on-demand and live streaming of RealVideo, RealAudio or MP3 to any device or application that supports the HTTP or RTSP streaming protocols and other codecs/protocols can be implemented as plugins. It exists also a plugin to stream ogg vorbis.

The stream of the Helix Server can be distributed over multiple servers (see figure 9). The distributed Helix system can use three kinds of special servers: advanced, relay and edge servers. An advanced server sends the encoded data to relay servers, afterwards the relay servers can distribute the stream to various edge server. Finally these edge servers, which are usually located close to the desired clients, stream the data to them (from [5]).

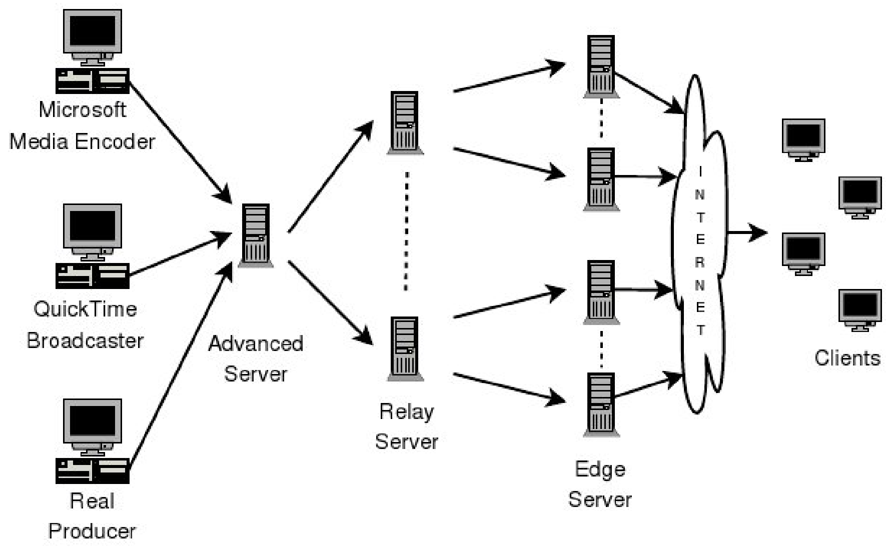


Figure 9: Helix Server: distribution of a stream over multiple servers. For description see text. (picture from [5])

The commercial Helix Server has some more features like Quicktime, MPEG-4 and Windows Media streaming. For a complete feature comparison see [3].

Many clients can receive the helix stream such as real player, helix player (open source version of real player), VLC player and more. Browser plugins for Mozilla, Internet Explorer and Safari exist for the real player and also for VLC.

Helix DNA Server problems:

- Ogg vorbis streaming over RTSP is not possible yet, only HTTP.

[7] Helix DNA Server Webpage: https://helix-server.helixcommunity.org/

## 3.5 Conclusion and Choice for IAEMStream

As pointed out in the previous subsections, no streaming server solution fullfilled all our wishes. No server we are aware of was able to stream ogg vorbis audio over the RTSP/RTP protocol, therefore we decided to use MP3 as codec.

We started the development with the VLC server, but then switched to the Helix DNA Server because of stability problems. Much more clients can play the stream generated by Helix DNA Server and also stable browser plugins exist for the real player.

The Helix Server has also an ogg vorbis plugin. However, it is not yet capable of streaming vorbis over RTP, but according to some discussions on mailinglists not much is needed to get this working. We also thought about implementing this feature but skipped it due to time constraints. It would be of course easy to integrate it into the IAEMStream product when the vorbis plugin gets finished.

## 4 IAEMStream Product

Finally the IAEMStream product is presented in the current section. After the conclusions from 3.5 we started to integrate the Helix DNA Server in a plone product.

Our Use-Case:

- If a user wants to listen to a specific file and has the required permissions, then a "hear-channel" will be openend for him with a random generated stream URL.
- Now she/he can open the stream in an external audio player or in the browser plugin and pause, seek, etc. in it.
- After some time this random generated stream URL will be removed from the server.

In subsection 4.1 the internal structure and procedures of the product are described and subsection 4.2 shows how to use and configure it in a plone system or as a client.

### 4.1 IAEMStream Internals

#### 4.1.1 Product Structure

The product is build out of two classes, as shown in figure 10. Class IAEMMediaData represents one media file on the server and class IAEMStream is the interface to the Helix streaming server. Methods marked with $\ll action, view \gg$ have an additional HTML-interface in plone.

An instance of IAEMMediaData is generated for each file that is uploaded and included in the content management system. The method *iaemmedia_view()* is the default view of an uploaded file and displays the name, length and some more metadata (see also figure 12). No random stream URL or "hear-channel" is generated while this method is executed. By calling the *stream()* method an random URL is generated as described in 4.1.2 and the user gets to the streaming interface (realplayer plugin) with seek, play and pause commands (see also figure 13). The other methods are used to query metadata, displaying some help and adding/converting files as discussed in 4.1.3.

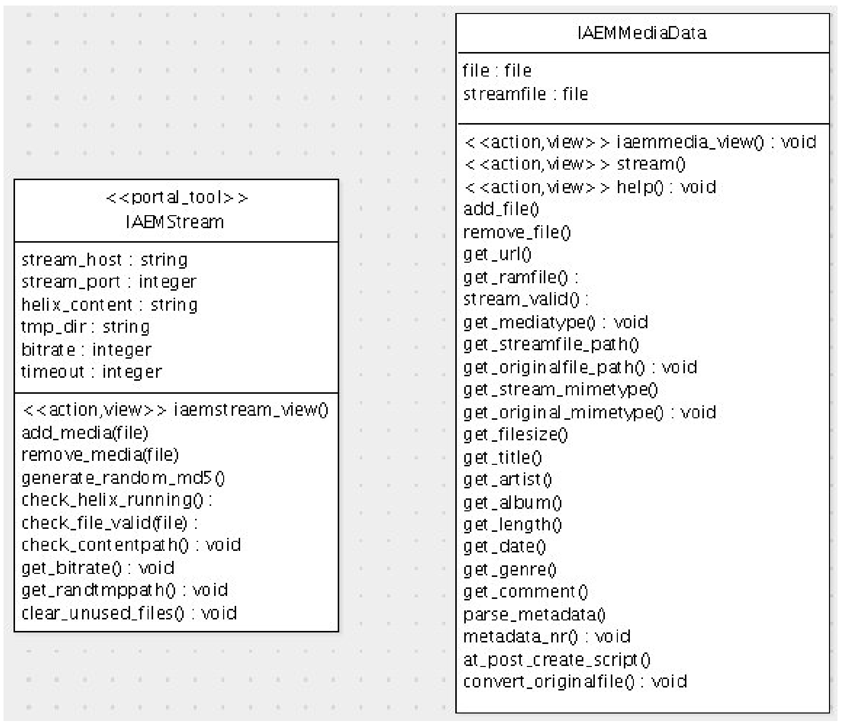


Figure 10: Class diagram of the IAEMStream product. Description see text.

Class IAEMStream exists only once per zope instance and is the interface to the streaming server. It is implemented as a *portal tool*, which is in plone something like a singleton design pattern and often used for the configuration of products. Instances of IAEMMediaData can query the IAEMStream instance to get information about the server, add files to it, generate a random URL etc. The method *iaemstream_view()* is used as a *configlet* to set the streaming server configuration for the whole plone instance (see also figure 11). The other methods are used to add/remove files, to generate a random URL as discussed in 4.1.3 and for various checks.

### 4.1.2 Generation and Lifetime of a Stream URL

The generation and lifetime of a "hear-channel" happens in the following steps:

- An IAEMMediaData instance requests the unique IAEMStream instance and asks for a "hear-channel".
- Then the IAEMStream instance generates a random mountpoint, which is constructed out of an MD5 sum of random numbers.
- A link in the helix servers content path, with the random MD5 sum, to the actual location of the media file is created and delivered to the client.
- Finally a timer is started, which removes again the random generated link after a specific timeout.

The stream URL is generated in the following format: *rtsp://stream_host:stream_port/random_MD5_sum.mp3*. In addition also a so called ram file is automatically created by the Helix server, which usually just has one line with the full stream URL. It is used as a link on the HTML interface, because it automatically launches an external audio player and connects to the stream. For more information about the ram file consider [4] chapter 21.

### 4.1.3 Media File Conversion

Audio files in various codecs can be added to the system. At the time of this writing the supported file formats are ogg, mp3, wav, aiff, au, gsm and voc. However, since the Helix streaming server is only able to stream mp3, other fileformats have to be converted.

After a new IAEMMediaData instance is created the method *at_post_create_script* will be called. This method first extracts all the metadata of the uploaded file with the help of the mmpython module [8] and then converts all files to the MP3 format if needed.

The conversion is done in the following steps:

- A separate thread, with a lockfile for synchronization, is started on the server for the conversion.
- The uploaded file is converted to an uncompressed WAV file with the open source program SoX [9] into a temp directory (attribute *tmp_dir* of class IAEMStream).
- Afterwards this temporal file is encoded to MP3 with the open source encoder LAME [10] and the desired bitrate as specified in the *bitrate* attribute of class IAEMStream.
- Finally the temporal file is deleted and the generated MP3 file is stored in the current IAEMMediaData instance. However, it is not added to the ZODB (Zope Object Database) and instead stored directly on the filesystem with the help of the ExternalStorage product [11].

## 4.2 IAEMStream Configuration and Usage

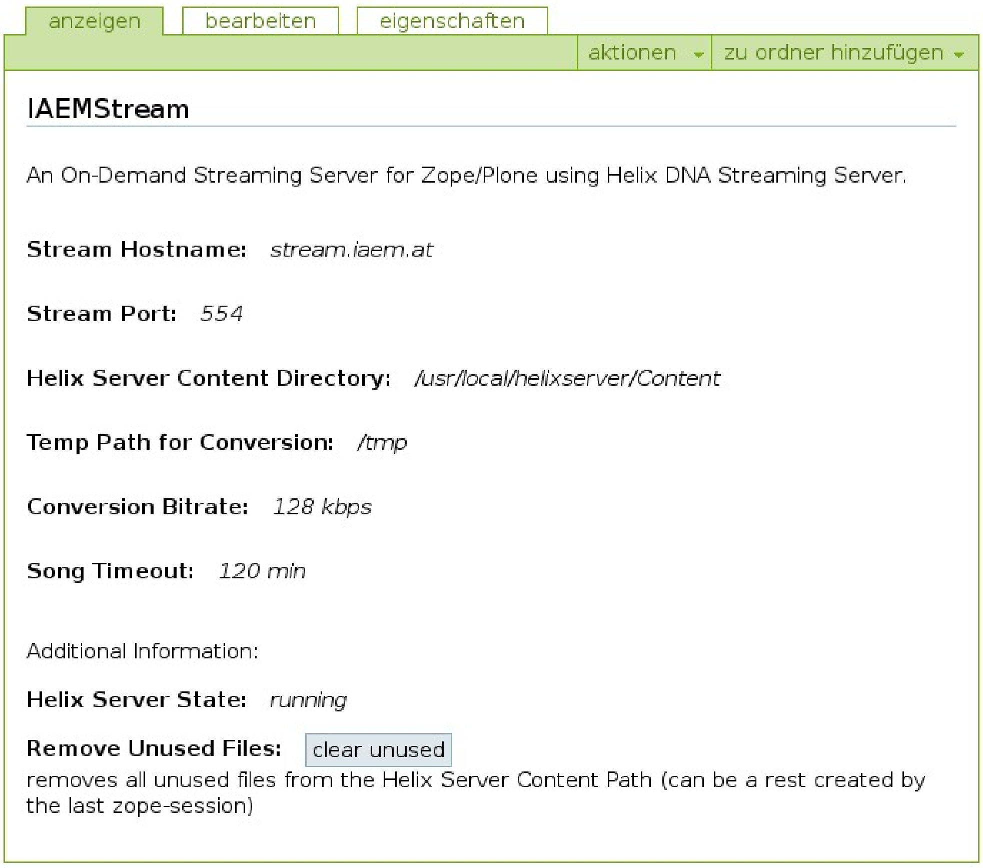
This subsection should give a small overview of how to configure and use the IAEMStream product. It does not contain any installation instructions. Consider the file README.txt in the IAEMStream source distribution instead, which contains step by step installation instructions for server and clients. For more information on how to setup and use your plone system have a look in the Plone Book [2].

### 4.2.1 Product Configuration

Once the product is installed, the IAEMStream Configuration Configlet is visible in the Plone Control Panel. This Configlet (method *iaemstream_view()* of class IAEMStream) is shown in figure 11.

Here it's possible to set the basic configuration of the streaming server:

[8] mmpython Webpage: `http://mmpython.sourceforge.net/`
[9] SoX Webpage: `http://sox.sourceforge.net/`
[10] LAME project page: `http://lame.sourceforge.net/`
[11] ExternalStorage Webpage: `http://plone.org/products/externalstorage/`



IAEMStream

An On-Demand Streaming Server for Zope/Plone using Helix DNA Streaming Server.

**Stream Hostname:** *stream.iaem.at*

**Stream Port:** *554*

**Helix Server Content Directory:** */usr/local/helixserver/Content*

**Temp Path for Conversion:** */tmp*

**Conversion Bitrate:** *128 kbps*

**Song Timeout:** *120 min*

Additional Information:

**Helix Server State:** *running*

**Remove Unused Files:** clear unused
removes all unused files from the Helix Server Content Path (can be a rest created by the last zope-session)

Figure 11: IAEMStream Configuration HTML Interface.

**Stream Hostname:** Hostname of the computer where the Helix DNA Streaming Server runs.

**Stream Port:** Port of the stream. This is defined in the configuration file of the Helix Server. Usually the RTSP port is 554.

**Helix Server Content Directory:** Content Directory of the Helix Server. Here the random stream links are generated, as presented in 4.1.2. This path is usually */path/to/helixserver/Content*.

**Temp Path for Conversion:** Temporal directory for the media file conversion as discussed in 4.1.3. You need to have read and write permissions.

**Conversion Bitrate:** Other media files are converted to MP3 files with the bitrate specified here, as described in 4.1.3.

**Song Timeout:** If an user requests a stream the generated random URL will be removed after this timeout, see section 4.1.2.

Additionally the Configlet provides some status information if the Helix Server is running. The button *clear unused* removes unused links from the content directory. Unused links can be a remainder from the last zope session if it was killed in a hard way before the song timeout.

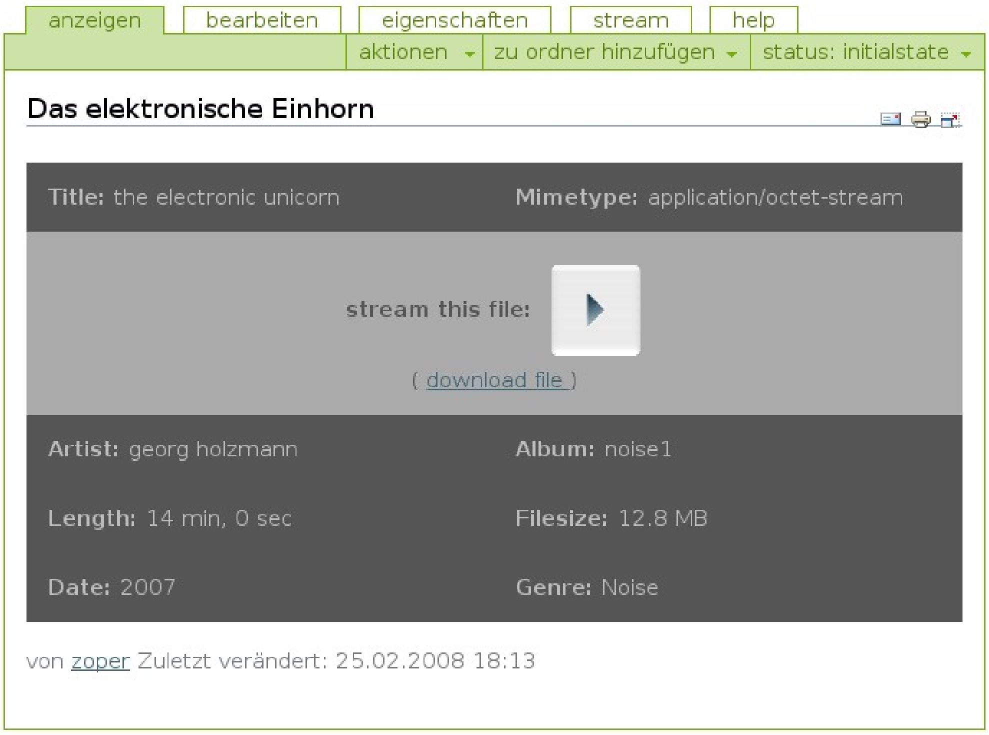
### 4.2.2 Permissions and Security

The IAEMStream product comes with two additional permissions: *IAEMStream download permission* and *IAEMStream stream permission*. They can be set for groups or individual media files in the Zope Management Interface (ZMI), consider the Plone Book [2] for more information.

*IAEMStream stream permission* allows users to listen to the RTSP stream and with *IAEMStream download permission* it's also able to download the whole media file.

### 4.2.3 User Interface

When a client selects a media file the method *iaemmedia_view()* of class IAEMMediaData is called and generates the HTML interface as shown in figure 12. This view displays some



Das elektronische Einhorn

**Title:** the electronic unicorn **Mimetype:** application/octet-stream

**Artist:** georg holzmann **Album:** noise1

**Length:** 14 min, 0 sec **Filesize:** 12.8 MB

**Date:** 2007 **Genre:** Noise

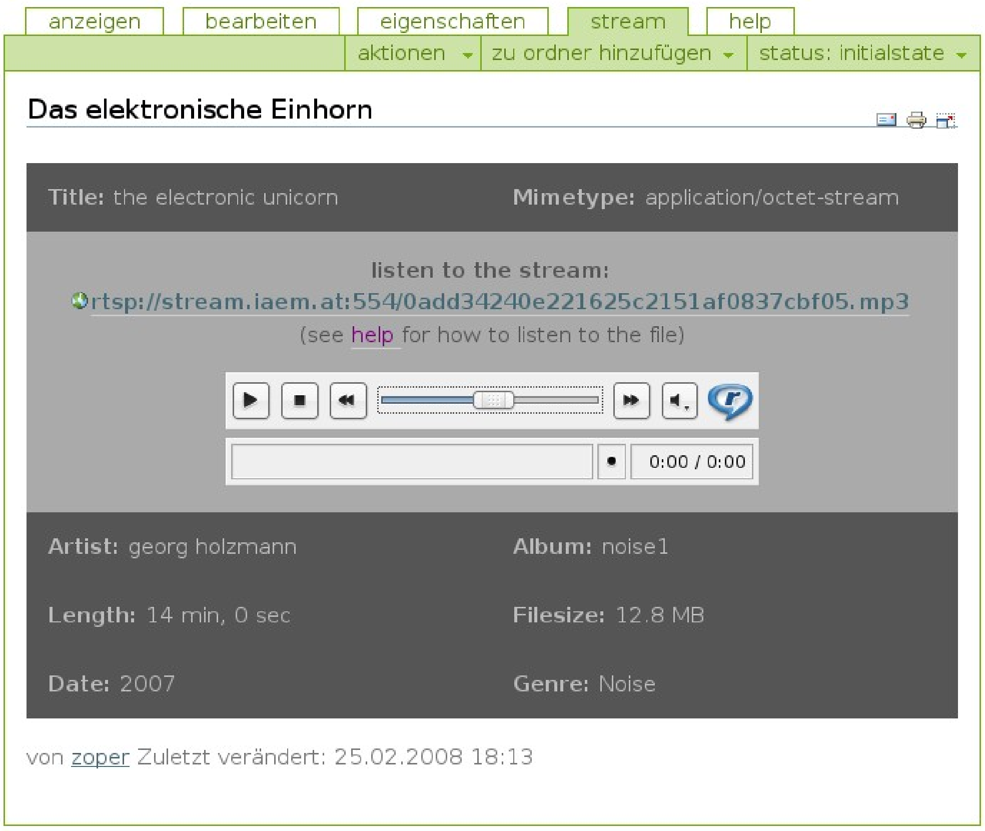
von zoper Zuletzt verändert: 25.02.2008 18:13

Figure 12: HTML interface of a media file. The stream is generated by clicking on the play button.

metadata of the media file, a play button and a link to download the file if one has the right permissions.

By clicking on the play button the method *iaemmedia_stream()* of class IAEMMediaData is called and generates the random stream URL as discussed in 4.1.2. Additionally the HTML streaming interface is presented to the client and shown in figure 13.

The streaming interface shows some metadata, the realplayer browser plugin and the generated stream URL. With the browser plugin it is possible to directly control and seek in the stream. The stream URL links to a ram file generated by the Helix Server (see 4.1.2)


anzeigen | bearbeiten | eigenschaften | stream | help

aktionen | zu ordner hinzufügen | status: initialstate

Das elektronische Einhorn

**Title:** the electronic unicorn **Mimetype:** application/octet-stream

**listen to the stream:**
rtsp://stream.iaem.at:554/0add34240e221625c2151af0837cbf05.mp3
(see help for how to listen to the file)

0:00 / 0:00

**Artist:** georg holzmann **Album:** noise1

**Length:** 14 min, 0 sec **Filesize:** 12.8 MB

**Date:** 2007 **Genre:** Noise

von zoper Zuletzt verändert: 25.02.2008 18:13

Figure 13: HTML streaming interface with realplayer browser plugin.

and therefore automatically starts realplayer or an other media player if it is not installed. Additionally it is possible to open the RTSP-URL in an external media player like VLC or realplayer. Instructions on how to install and use client players are given in the link *help*.

## 5 Conclusion and Future

In this report IAEMStream, a streaming server for the content management system Zope/Plone was presented. It uses the Helix DNA Server and streams audio in the MP3 format over RTSP. The report has given also an overview over streaming technologies and existing open source implementations.

Besides various bug fixes the following features would be nice to have in future:

- **Video Streaming:** Class IAEMMediaData is already designed with a possible extension to video streaming in mind. For example it contains already the method *get_mediatype()* which returns *audio* or *video*. However, it would be necessary to choose a video codec which can be streamed by the open source Helix DNA Server and to implement conversion routines for other codecs.

- **Ogg Vorbis Streaming:** As mentioned in 3.5 it already exists an ogg plugin for the Helix Server, which has its project page at https://xiph.helixcommunity.org/Index.html. It would be necessary to extend this plugin so that the server is also able to stream ogg over RTSP. Then video (theora) and audio (vorbis) could be streamed with ogg.

For any further questions, suggestions or bug reports do not hesitate to contact me!

## References

[1] Apple. Darwin streaming server, accessed 20.2.2008. http://developer.apple.com/opensource/server/streaming/index.html.

[2] Andy McKay. The definitive guide to plone, accessed 20.2.2008. http://docs.neuroinf.de/PloneBook.

[3] Real Networks. Helix dna server feature comparison, accessed 20.2.2008. https://helix-server.helixcommunity.org/2006/devdocs/helix_server_comparision.html.

[4] Real Networks. Realnetworks production guide, accessed 20.2.2008. http://service.real.com/help/library/guides/ProductionGuide/prodguide/realpgd.htm.

[5] Eric Peters. Entwurf und implementierung eines verteilten multimedia streaming servers auf basis der netzwerk-integrierten multimedia middleware (nmm).

[6] Daniel Schreiber. Peer-to-peer-videostreaming.

[7] tunequest. A big list of mp3 patents (and supposed expiration dates), accessed 20.2.2008. http://www.tunequest.org/a-big-list-of-mp3-patents/20070226/.

[8] VideoLAN. Vlc streaming server, accessed 20.2.2008. http://www.videolan.org/vlc/streaming.html.

[9] Wikipedia. Icecast, accessed 20.2.2008. http://en.wikipedia.org/wiki/Icecast.

[10] Wikipedia. Mp3, accessed 20.2.2008. http://en.wikipedia.org/wiki/Mp3.

[11] Wikipedia. Streaming media, accessed 20.2.2008. http://en.wikipedia.org/wiki/Streaming_media.

[12] Wikipedia. Vorbis, accessed 20.2.2008. http://en.wikipedia.org/wiki/Vorbis.