Final Thesis

An Internet content overview and implementation on an IP based set-top box

by

Linus Widborg

LITH-IDA-EX-05/089-SE

2005-12-14
Final Thesis

An Internet content overview and implementation on an IP based set-top box

by

Linus Widborg

LITH-IDA-EX--05/089--SE

2005-12-14

Supervisor: Henrik Carlsson
Examiner: Lena Strömbäck
This thesis covers the investigation of different content sources on the Internet and the analysis of the requirements they put on a set-top box. It also covers the adaptation of the set-top box to one of these sources.

An IP based set-top box (IP-STB) is mainly constructed for access to TV and video distributed over a high speed network. The IP-STB is also connected to the Internet and it potentially has access to all of the Internet based content. This could provide the user of the IP-STB with a lot of new services and was the starting point for my thesis work.

The goal of this thesis was to make a broad investigation of different content sources on the Internet and to adapt an IP-STB to one of them.

The investigation focused on streaming media content and how it was distributed over the Internet. It resulted in a representative overview of the streaming media content available and how the content is distributed. The IP-STB was adapted to handle the Streaming HTTP protocol which made it possible to listen to web radio transmissions from Sveriges Radio and other providers using the same protocol for their distribution.
Abstract

This thesis covers the investigation of different content sources on the Internet and the analysis of the requirements they put on a set-top box. It also covers the adaptation of the set-top box to one of these sources.

An IP based set-top box (IP-STB) is mainly constructed for access to TV and video distributed over a high speed network. The IP-STB is also connected to the Internet and it potentially has access to all of the Internet based content. This could provide the user of the IP-STB with a lot of new services and was the stating point for my thesis work.

The goal of this thesis was to make a broad investigation of different content sources on the Internet and to adapt an IP-STB to one of them.

The investigation focused on streaming media content and how it was distributed over the Internet. It resulted in a representative overview of the streaming media content available and how the content is distributed. The IP-STB was adapted to handle the Streaming HTTP protocol which made it possible to listen to web radio transmissions from Sveriges Radio and other providers using the same protocol for their distribution.

Keywords

IP based media systems, Set-top box, Embedded systems, Streaming media, Streaming HTTP, Internet content.
1 Introduction ........................................................................................................ 1
   1.1 Background .................................................................................................. 1
   1.2 Problem specification ................................................................................ 1
   1.3 Work process ............................................................................................. 2
      1.3.1 Organization ........................................................................................ 2
      1.3.2 Subgoals ............................................................................................. 2
   1.4 Document overview ................................................................................... 2
   1.5 Reading instruction .................................................................................... 3
   1.6 Glossary ..................................................................................................... 3

2 IP-STB System .................................................................................................. 5
   2.1 IP-STB ....................................................................................................... 5
   2.2 Hardware ................................................................................................. 6
   2.3 Software .................................................................................................... 6
      2.3.1 Media Service ....................................................................................... 9
      2.3.2 Streamer ............................................................................................. 9

3 Internet content overview ............................................................................... 11
   3.4 Content type .............................................................................................. 11
   3.5 Method ....................................................................................................... 11
      3.5.1 Limitations .......................................................................................... 12
   3.6 Providers .................................................................................................... 12
      3.6.1 Sveriges Radio, www.sr.se .................................................................. 12
      3.6.2 Sveriges Television, www.svt.se ......................................................... 13
      3.6.3 SHOUTcast, www.shoutcast.com ....................................................... 13
      3.6.4 MSN Radio, radio.msn.com ................................................................. 13
      3.6.5 Napster, www.napster.com ................................................................. 13
      3.6.6 TV4, www.tv4.se ................................................................................ 13
      3.6.7 MTV, www.mtv.com .......................................................................... 14
      3.6.9 SF anytime, www.sf-anytime.com .................................................... 14
      3.6.11 NRJ, www.nrj.se ............................................................................... 14
      3.6.13 TV3, www.tv3.se ............................................................................... 15
      3.6.14 Kanal5, www.kanal5.se ..................................................................... 15
      3.6.15 ZTV, www.ztv.se ............................................................................... 15
      3.6.16 Movielink, www.movielink.com ........................................................ 15
   3.7 Conclusion .................................................................................................. 15

4 Problem analysis ............................................................................................. 17
   4.1 Content analysis .......................................................................................... 17
   4.2 IP-STB system analysis ............................................................................... 18
   4.3 Choosing content source ........................................................................... 18
   4.4 Analysing SR webradio ............................................................................... 19
   4.5 SR webradio on the IP-STB ....................................................................... 19
   4.6 Requirements ............................................................................................. 21
      4.6.1 Communication .................................................................................... 21
      4.6.2 Streaming HTTP data support ............................................................ 22
      4.6.3 Demonstration HTTP data application ............................................. 23
5 Technology ............................................................................................................. 25
  5.1 cURL .................................................................................................................. 25
    5.1.1 Command line tool, curl ............................................................................. 26
    5.1.2 libcurl ......................................................................................................... 26
  5.2 Streaming HTTP protocol ............................................................................... 27
    5.2.1 Communication ......................................................................................... 27
    5.2.2 Server sent ASF header ............................................................................. 28
    5.2.3 Client sent HTTP header ............................................................................ 29
    5.2.4 Server sent HTTP body data ..................................................................... 31
  5.3 Advanced system format, ASF ......................................................................... 33
    5.3.1 ASF file structure ...................................................................................... 33

6 Implementation .................................................................................................... 35
  6.1 Communication ............................................................................................... 35
    6.1.1 Identifying Streaming HTTP ....................................................................... 36
    6.1.2 Streaming HTTP header request ............................................................... 36
    6.1.3 Parse Streaming HTTP header ................................................................... 38
    6.1.4 Streaming HTTP data request ................................................................... 38
  6.2 Streaming HTTP data support ......................................................................... 39
    6.2.1 Parse Streaming HTTP data ....................................................................... 41
    6.2.2 Process ASF data ....................................................................................... 41
  6.3 Demonstration application. .............................................................................. 42

7 Conclusion & Further development .................................................................. 45
  7.1 Conclusion ........................................................................................................ 45
    7.1.1 Investigation and analysis .......................................................................... 45
    7.1.2 Adaptation .................................................................................................. 45
    7.1.3 Streaming HTTP support .......................................................................... 45
    7.1.4 Media player .............................................................................................. 46
    7.1.5 Demonstration application ....................................................................... 46
  7.2 Further development ....................................................................................... 46
    7.2.1 Codec’s ...................................................................................................... 46
    7.2.2 Transport protocols ................................................................................... 47
    7.2.3 Streaming HTTP ....................................................................................... 47
    7.2.4 Streamer .................................................................................................... 47

8 References .......................................................................................................... 49

A Glossary .............................................................................................................. 50
This chapter will give a short introduction to my master thesis project.

1.1 Background

Today consumers can use their computer to access a lot of different content on the Internet. But the computer is only one unit, which makes it a constant source of conflict. This could easily be avoided by buying one computer for every member of the family. On the other hand, if you could provide support for Internet content on the IP based set-top box (IP-STB) a real value to the consumers will be added since it would be one less computer to buy and maintain.

IP-STBs are mainly constructed to access TV and video on demand (VoD) from an operator controlled IP network. The audio and video streams, sent by the operator, are today mainly compressed with MPEG-2. Due to that the IP-STB is connected to an IP network. It also has a connection to the Internet and hence all of it's content. But for the IP-STB to be able to handle the variety of Internet content, among other things support for additional codec's and transport formats is needed.

1.2 Problem specification

The purpose of this thesis is to investigate different content sources on the Internet that today is consumed via a computer or another unit and analyse the requirements they put on the IP-STB. One of the sources will be chosen and necessary adjustments will be implemented on the IP-STB. An application for demonstration purposes will also be built.

The main problem is to:
Investigate and analyse different Internet content sources and adapt the IP-STB to one of them.
The problem will be divided into three subproblems:
1) An investigation and analysis of the Internet content has been made were information about techniques that it uses and the organization behind it has been collected.

2) One source has been chosen and the techniques that it uses have been implemented on the IP-STB.

3) A demonstration application has been built that demonstrates that the IP-STP can access the Internet content source.

1.3 Work process

This section will cover how the thesis work will be organized and subgoals will be specified.

1.3.1 Organization

The work can be divided into two distinct parts:

1) Investigation and analysis of available content.

In this part a broad investigation of the available Internet content will be performed. The result of the investigation will be a list of possible content sources. The list will be analysed and requirements put on the IP-STB, by the sources, will be added. In this step the specific source that will be implemented is selected.

2) Implementation of the Internet content source.

The source selected in the first step will be implemented in this step. It will first be done by adapting the IP-STB to the requirements put on it by the source, and second to build a demonstration application that will use the source.

1.3.2 Subgoals

The subgoals are coherent with the three subproblems specified in section 1.2 Problem specification. The subgoals will be fulfilled when the following have been done:

1) An investigation and analysis of the Internet content has been made. And when information about techniques that the source uses, and the organization behind it have been collected.

2) One source have been chosen and the techniques that it uses have been implemented on the IP-STB.

3) A demonstration application have been built that demonstrates that the IP-STP can access the Internet content source.

1.4 Document overview

This section explains the structure of this report. It will also explain what content the chapters contain.

Chapter 2 presents the IP-STB on which the system will be built. It will give an overview of the whole system and a detailed description on the parts that will be used or altered in this thesis.

Chapter 3 presents the Internet content overview that was made in the beginning of my thesis. The overview presents different internet content providers and specifies what content they provide and how it’s distributed.

Chapter 4 analyses the content and the IP-STB and specifies what source to be implemented and how it should be done. It will also define the requirements put on the system.
Chapter 5 presents cURL, Streaming HTTP and ASF which is the technologies used to analyse and implement the Internet content support.

Chapter 6 describes how the communication, Streaming HTTP support and the Demonstration application were implemented on the IP-STB.

Chapter 7 concludes the work and presents how the system could be developed further.

Chapter 9 contains a list of references.

Appendix A contains a glossary.

The report will start by giving a overview of the IP-STB analysis in chapter 2 and the content investigation in chapter 3, this will provide chapter 4, problem analysis, with the information base needed to select a content source. Depending of the source chosen, chapter 5 will present technology relevant to that source. The information about the technologies is used in chapter 6, implementation, to specify how the different parts should be implemented. The report is then concluded and further development is presented in chapter 7. Finally there are references and an appendix.

1.5 Reading instruction

To get a quick overview of this report, you should read the introduction chapter together with chapter 4, Problem analysis. To get a deeper understanding of the thesis you should read chapter 2, 3 and 6 which will cover the analysis of the Internet content providers, the IP-STB and the implementation. To get a description of the technologies used for the thesis, you should read chapter 5, Technology. Chapter 7, Conclusion & Further development gives you my reflections on the thesis work and is an essential chapter for anybody that wants to further develop the system.

1.6 Glossary

A glossary is available in Glossary.
This chapter will give you a description of the IP based set-top box (IP-STB) system used, especially the parts that will affect this thesis.

The chapter describes the system that this master thesis is built on. It will start with a general description of the IP-STB and the environment that it operates in. This is followed by a more detailed description of the hardware and software components that the system is built on.

### 2.1 IP-STB

An IP-STB is a device that makes digital multimedia services, distributed over a broadband IP network, available to a TV receiver. The IP-STB has its main focus on receiving digital TV, but it also has a variety of additional services. Today it has additional services such as Video on demand (VoD), games and browsing the Internet. Figure *System overview* gives an overview of the environment in which the IP-STB operates. The environment consists of a Configuration server and Operator components. The Configuration server distributes and updates basic software to the IP-STB. The basic software components consists of the Hardware Abstraction Layer (HAL), the Linux Operating System, drivers, platform and standard applications, more about them in section 2.3 *Software*. The Operator components are the services and software provided and developed by the operator. These services include among other things network access with DNS and DHCP, TV broadcasting, VoD broadcasting and web portals. The components, servers and boxes in the environment are connected to each other and to the Internet through a high-speed IP network.

When a user switches the IP-STB on, and connects it to the network, it starts to download software from the Configuration server and the Operator. It will also download and configure components from the Operator. The software and components will reside in the memory until the IP-STB is rebooted or the power is turned off, more about the memory in section 2.2 *Hardware*. When the box has finished its boot process the user can start to use the services, e.g. watch TV and browse the Internet. The user has the possibility to make some individual changes to the IP-
STB, e.g. choose language. The changes made are stored on a flash memory and will hence survive a reboot. A remote control or a keyboard is used to control the box.

The fact that the IP-STB has a connection to the Internet makes a lot of new services available to it. And it’s these services that are the starting point for this thesis. The new services are among other things web radio, web television, games, chats and bloggs.

![Figure 1: System overview](image)

### 2.2 Hardware

The IP-STB can be viewed as a simple computer with only a CPU, a DRAM and a flash memory. There are also some special purpose hardware for MPEG decoding, TV processing and an IR receiver for controlling the STB. The CPU manages the GUI, the networking and the administration of the Linux OS. The DRAM is used to hold the downloaded software components described in section 2.1 IP-STB. The flash memory holds the boot process and the user configurations. The MPEG decoder is used to decompress the TV transmissions without involving the CPU. The reason for using a hardware decoder is to offload the CPU from the computational intensive operation of decompressing.

### 2.3 Software

The software for the STB is developed around a Linux kernel, which makes software development very similar to Linux/Unix software development when it comes to available software libraries and compilation environment. The software architecture is designed for a hardware environment with very limited resources (memory and CPU). It’s also designed for a specific user and applica-
The Software architecture is layered, i.e. it’s structured into layers to get a nice abstraction between applications, services and hardware. The architecture is divided into three layers: Application layer, Platform layer and Hardware Abstraction Layer (HAL). The figure Software layers provides schematic over the layer structure.

![Software layers diagram]

On the top we find the Application layer, which is used to develop functionality for the users of the IP-STB. The Application layer is connected to the Platform layer through the TV Open Interface (TOI). The TOI separates the platform from the application and provides the applications with access to Platform services. The Platform layer supplies the applications with services and a management system for the platform. The Platform services supplies the applications with lifecycle handling, i.e. it handles the registration, the installation/uninstallation and the activation/deactivation of applications. It also provides the application with audio/video handling. I.e. it handles the decoding, mixing and volume of media streams for the applications. The management system controls the services. It includes the starting/stopping, the activation/deactivation, the supervision and the fault handling of the services. HAL provides the Platform layer with an interface against the IP-STBs hardware. It consists of the Linux OS, drivers and libraries. The figure IP-STB Software Architecture shows a more detailed description over the architecture. At the top, applications in the Application layer are shown. The applications provide the users with the possibility to navigate applications with the Navigation application, Watch TV or Video on Demand with the TV/VoD application, configure the box with the Settings application or browse the Internet with the Browser application (a light version of Mozilla). It’s possible for the operators to develop applications for the IP-STB, and they often implement a Portal application where their users receive information and operator specific services. To summarize it, if the feature requires a user interface
it’s implemented as an application. The application uses the platform layer to perform different tasks, through the services it provides:

Name Service
Looks up service interfaces using the service name.

Application Service
Handles the lifecycle of the applications. This includes registration, installation/uninstallation, start/stop, activation/deactivation and input/output of the application.

Uniform Resource Identifier (URI) Loader Service
Dispatches an URI for the application.

Information Service
Handles the distribution of information between applications and the user settings.

Media Service
Handles the playback of audio and video streams.

Video Mixer Service
Is responsible for mixing video and graphics. This includes the handling of alpha windows.

Volume Service
Handles the volume. This includes the increase, the decrease and the mute of the volume.

Display Service
User input/output control.

The Platform services use HAL to access the hardware. HAL is a collection of Linux commands, drivers and libraries.

The following section describes the Media Service. It is used by the demonstration application to control the media stream. A discussion, over the use of the Media service for the demonstration application, can be found in section 4.5 SR webradio on the IP-STB.

Figure 3: IP-STB Software Architecture

The Platform services use HAL to access the hardware. HAL is a collection of Linux commands, drivers and libraries.

The following section describes the Media Service. It is used by the demonstration application to control the media stream. A discussion, over the use of the Media service for the demonstration application, can be found in section 4.5 SR webradio on the IP-STB.
2.3.1 Media Service

The media service provides the application layer with an interface for controlling media streams. It’s capable of handling media content in different formats, delivered in different ways. The Media Service can do the following:

- **Stream Sources**
  It gets media streams from the network, that’s delivered by either the HTTP or the RTP protocol.

- **Stream Formats**
  It can handle all stream formats in use for MPEG1 and MPEG2 encoded content, e.g. the MP3 format. It can also handle the WMA and ASF format.

- **Conditional Access**
  It can be configured to handle conditional access systems.

- **Multiple application instances**
  It can handle more than one application concurrently. Each instance is completely separated from the other and has the capacity of handling one media stream.

An application that wants to use the service can do so in two ways: either uses it for playing a media stream, or listen to a stream that is already playing. The media service is always in a well defined state and responds differently to application commands depending on the state.

The component that’s responsible for processing the media stream in the media service is the streamer. A description of the streamer is given in the Streamer section.

2.3.2 Streamer

The streamer component is used by the Media Services to process media streams. It’s by extending the streamer that new features (e.g. codec’s and transport protocols) can be added. The streamer is built up with components called elements, where an element is an entity that can process stream data. The Streamer core is responsible for organizing the elements into a pipeline. The elements are connected in the pipeline with pads. It’s on these pads that the elements put data that has been processed, and get data to be processed. Figure Streamer pipeline example shows a simple pipeline.

![Streamer pipeline example](image)

When a media stream has processed through the pipeline it will be in a hardware readable format. Today the streamer supports the HTTP and RTP transport protocols, which are used to handle the Internet connection. These protocols are supported with the help of the Curl library libcurl. The streamer can also decode MPEG1, MPEG2 and WMA coded content as well as the ASF format.
An Internet content overview and implementation on an IP based set-top box
Internet content overview

In this chapter the Internet content overview will be presented. It will describe the method used, present the investigated providers and finally a conclusion will be given.

The chapter presents the Internet content overview that was the first part of this master thesis. It will start with a presentation and a discussion of the type of content that was included in the overview. It continues with a presentation of the method used to gather and analyse the content. Then the list of gathered content is presented, with added information from the analysis. Finally the overview is concluded and the result of the overview is presented.

3.4 Content type

Internet content is a broad term that could describe a lot of different content types, e.g. it describes everything that could be presented on a web page. This thesis will focus on streaming media content, e.g. audio and video content that’s delivered using streaming media techniques. The reason for delimit the Internet content overview to streaming media, is to make the thesis work manageable. That is to adapt it to the time and work extent expected for a master thesis. An IP-STB device is also mainly built for handling streaming audio and video content, which makes the choice of the streaming audio and video content natural.

3.5 Method

The investigation of streaming media content started by listing companies that are known media providers, e.g. media companies as SVT and BBC. The list was then extended through internal discussion at the company and with sites that were linked to, from already listed sites.

An analysis of the listed sites was then performed, with the result that irrelevant sites was complemented with an explanation on why they were irrelevant and relevant ones was complemented with information about organization, content and techniques. When deciding if a site was relevant or not, the following, questions were taken into account:
1) Is there any streaming media content?
   The question filters out media companies that don’t publish media content on
   the Internet.
2) Is the content usable?
   This question filters out sites that contain content but the content is not usable. E.g. if the
   content is only available in the U.S.A. it will be difficult to use it in this thesis, if the
   quality of the content is poor it will not appeal to the user and is not that interesting to
   implement.

The information added for the relevant sites is a short presentation of the company and a descrip-
tion of how the content is published and distributed on the Internet. The resulting list is presented
in section 3.6 Providers.

3.5.1 Limitations

The emphasis of the search was to make a broad inventory of the media content on the Internet.
I.e. as many sources as possible were looked at. But the fact that investigation is based on my own
and people at the company’s knowledge about media content sources, will limit it. Another limita-
tion is that the Internet is a nearly unlimited source of content, which makes it necessary for me to
sift out content that seems relevant. Section 3.4 Content type presents this limitation further.
These limitations could result in that “more interesting” content is missed.

The search was also limited to different types of streaming media sources, such as live or recorded
audio streams and live or recorded video streams. The limitation was enforced by limitation in the
IP-STB and the fact that the IP-STB is built for handling streams of media content.

3.6 Providers

In this section the result of the Internet content search will be presented. For relevant content pro-
deriver a short presentation of the organisation, what content they provide and what techniques they
use to deliver it, will be given. For irrelevant content providers an explanation of why they are
irrelevant is given.

The providers can be divided into three different categories: Sveriges radio and television are pub-
clic providers that are controlled by the Swedish government, SHOUTcast is a non commercial
provider and MSN, Napster, TV4, MTV, ITunes, SF anytime, Live365, NRJ, Nordic Web Radio,
TV3, Kanal5, ZTV and Movielink are all commercial providers. The channels where selected as
described in section 3.5 Method.

3.6.1 Sveriges Radio, www.sr.se

Sveriges Radio is a non commercial radio station from Sweden. It’s financed through a radio and
TV fee. SR has four main channels P1, P2, P3, P4 and twelve extra channels. All the channels are
available live on the Internet and most of them have recorded material that you can listen to.
To be able to listen to the live broadcast you need an installation of a media player that supports
the .wma or .rm format and a browser that supports popup windows, javascripts and cookies. To
be able to listen to the recorded material you need installation of a media player that supports the
.rm format. The content is transported and controlled with the RTSP protocol or HTTP protocol.
3.6.2 Sveriges Television, www.svt.se

Sveriges television is a non commercial TV station from Sweden. It’s financed through a radio and TV fee. SVT has two main channels Svt1, Svt2 and four extra channels. The material consists mainly of recorded programs from the latest broadcast and an archive that holds a lot of recordings of old programs. There is also some live broadcasting where the extra channel Svt24 is sent on the Internet. Some special events (mainly sport events) are also broadcasted live.

To be able to watch the content (both recorded and live) you need an installation of a media player that supports the .wmv or .rm format and a browser that supports popup windows, javascripts and cookies. The content is transported and controlled with the RTSP protocol.

3.6.3 SHOUTcast, www.shoutcast.com

SHOUTcast is a radio portal that provides a collection of free radio stations. They also provide software for broadcasting radio. Their collection consists of almost 9000 radio stations. SHOUTcast is a service of the Nullsoft Corporation. They are also on the verge of providing a similar service for live video broadcast.

To be able to listen to the live broadcast you need an installed media player that supports the .mp3 format. The content is transported and controlled with the HTTP protocol.

3.6.4 MSN Radio, radio.msn.com

MSN Radio is a portal for over 3000 different radio stations. The portal is controlled by the Microsoft Corporation. There are some free stations, but most of them demands that you register for an account. A lot of the content only becomes available after a payment is made.

To be able to listen to the content you need an installed media player that supports the .wma format and a browser that supports javascripts and cookies. The content is transported and controlled with the RTSP protocol.

3.6.5 Napster, www.napster.com

Napster delivers an on-line music service where you buy music and download it to your computer. To be able to buy, download and listen to the music you need to download and install Napster’s own software and have an installed Windows Media Player version 7.1 or later. Napster is only available in the United States, Canada and the United Kingdom.

3.6.6 TV4, www.tv4.se

TV4 is a commercial TV station from Sweden. They have one main channel TV4 and three extra channels. The content is a selection from the four channels and you need to register for an account and in some cases pay to get hold of the content.

To be able to watch the content you need an installed media player that supports the .wmv format and a browser that supports popup windows, javascripts and cookies. The content is transported and controlled with the RTSP protocol.
3.6.7 MTV, www.mtv.com

MTV is a commercial TV station from the United Kingdom, which broadcasts in Europe. They offer both live radio and recorded TV shows.

To be able to watch or listen to the content you need an installed media player that supports the .wma, .wmv or .rm format and a browser that supports popup windows, javascripts and cookies. The content is transported and controlled with the RTSP protocol.


iTunes is a music service that allows you to buy and download music files. It’s also possible to listen to radio broadcasts. The service is controlled by the Apple Corporation.

To be able to use the service you need to download and install their software. The buy and download service is available in Europe and North America. The radio service is a cooperation with live365 (see 3.6.10).

3.6.9 SF anytime, www.sf-anytime.com

SF anytime is a movie rental service from Svensk Filmindustri, SF. The idea is that you pay for a movie and get it streamed to your computer.

To watch the movie you need an installation of a media player that supports the .wmv format and a browser that supports flash 6.0. The content is transported and controlled with the RTSP protocol.


Live365 is a commercial radio station portal with thousands of radio stations. To get hold of most of the content you need to register for an account. If you upgrade your account and pay a fee you will get the content free from commercials and with higher quality.

To be able to listen to the radio stations you need an installed media player that supports the .mp3 format and a browser that supports javascripts. The content is transported and controlled with the RTSP protocol.

3.6.11 NRJ, www.nrj.se

NRJ is a commercial radio station that is owned by MTG and is broadcasted in Europe. MTG also controls the stations Rix FM, Lugna favoriter, Svenska favoriter and Bandit 106,3. NRJ and the other stations have live broadcasting and recorded material on the internet. They use a system from Nordic Web Radio (see 3.6.12) to broadcast there material.


Nordic web radio supplies interactive services with focus on web radio. They are behind the radio portals Miniradio and SPRAYdio. Radio stations that are using NWR’s system is Radio City, Studio 107,5, Rock Klassiker, NRJ, Mix Megapol, Vinyl 107, The Voice, Rix FM, Bandit Rock 106,3, Lugna Favoriter, Metro FM, Radio Match, City 106,5, Gold FM, IDG.se, B-2000 and Boogie 75.

To be able to listen to the content you need an installed media player that supports the .wma format and a browser that supports cookies, javascripts and popup windows. The content is transported and controlled with the RTSP protocol.
3.6.13 TV3, www.tv3.se

TV3 is a commercial TV station from Sweden that is controlled by the Modern Times Group. They don’t publish their media content on the Internet, which makes them irrelevant as a possible content source.

3.6.14 Kanal5, www.kanal5.se

Kanal5 is a commercial TV station from Sweden. They don’t publish their media content on the Internet, which makes them irrelevant as a possible content source.

3.6.15 ZTV, www.ztv.se

ZTV is a commercial TV station from Sweden that is controlled by the Modern Times Group. They don’t publish their media content on the Internet, which makes them irrelevant as a possible content source.

3.6.16 Movielink, www.movielink.com

Movielink is an online movie rental service from U.S.A. It’s only available in the U.S.A., which makes it irrelevant as a possible content source.

3.7 Conclusion

Most of the sites included in the overview use similar solutions for broadcasting their material on the Internet. The solutions includes a server that stores and broadcasts the content in an Internet friendly format, a client that uses a browser that handles popup windows, and/or a site specific software (actually not a big difference, because you always need to download a media player if you don’t have one) to access the content. There were also some sites that required you to register and some that wanted payment. The table Internet content overview summary gives a summary of the investigated sites and the information collected.

The servers often use the RTSP protocol for controlling and transporting the media stream, but there are cases of using the HTTP protocol for the control and transport.

On the client side, a support for the codec’s Windows Media and Real Media were often demanded, and in some cases support for the MP3, OGG vorbis and Quick Time codec’s were needed. There were also requirements put on the clients to have a browser that handles popup windows, cookies and javascripts.
### Table 1: Internet content overview summary

<table>
<thead>
<tr>
<th>Name</th>
<th>Provider category</th>
<th>codec’s</th>
<th>Transport protocol</th>
<th>Browser requirements</th>
<th>Registration</th>
<th>Installation of Software</th>
<th>Relevant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sveriges Radio</td>
<td>Public</td>
<td>WMA, RM</td>
<td>RTSP, HTTP</td>
<td>Popup window, Javascript, Cookie</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Sveriges television</td>
<td>Public</td>
<td>WMV, RM</td>
<td>RTSP</td>
<td>Popup window, Javascript, Cookie</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>SHOUTcast</td>
<td>Non-commercial</td>
<td>MP3</td>
<td>RTSP</td>
<td>_ (1)</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>MSN Radio</td>
<td>Commercial</td>
<td>WMA</td>
<td>RTSP</td>
<td>Javascript, Cookie</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Napster</td>
<td>Commercial</td>
<td>_ (2) _ (2) _ (1)</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TV4</td>
<td>Commercial</td>
<td>WMV</td>
<td>RTSP</td>
<td>Popup window, Javascript, Cookie</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>MTV</td>
<td>Commercial</td>
<td>WMA, WMV, RM</td>
<td>RTSP</td>
<td>Popup window, Javascript, Cookie</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>ITunes</td>
<td>Commercial</td>
<td>_ (2) _ (2) _ (1)</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>SF anytime</td>
<td>Commercial</td>
<td>WMV</td>
<td>RTSP</td>
<td>Flash 6.0</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Live365</td>
<td>Commercial</td>
<td>MP3</td>
<td>RTSP</td>
<td>Javascripts</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>NRJ</td>
<td>Commercial</td>
<td>WMA</td>
<td>RTSP</td>
<td>Popup window, Javascript, Cookie</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Nordic Web Radio</td>
<td>Commercial</td>
<td>WMA</td>
<td>RTSP</td>
<td>Popup window, Javascript, Cookie</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>TV3</td>
<td>Commercial</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>No</td>
</tr>
<tr>
<td>Kanal5</td>
<td>Commercial</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>No</td>
</tr>
<tr>
<td>ZTV</td>
<td>Commercial</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>No</td>
</tr>
<tr>
<td>Movielink</td>
<td>Commercial</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>No</td>
</tr>
</tbody>
</table>

(1) Don’t apply to the content source.  
(2) No information.  
(3) No content found.  
(4) Only available in the USA
This chapter will analyse the problem, on the basis of the Internet content overview and the IP-STB System study. It will also generate a list of requirements.

The chapter will start with a general analysis of the content and the IP-STB, this is done to support the content source choice. The chosen source is then analysed and the needed adaptation to the box is established.

4.1 Content analysis

This section analyses the investigated content, presented in the 3 Internet content overview chapter. It will work as a foundation for the choice of content source, made in section 4.3 Choosing content source, together with section 4.2 IP-STB system analysis.

From a user point of view, most of the content sites work very similarly. You enter the site from a PC, follow the link to the content and a pop-up window appears and the content is played. One obstacle a user may encounter is, that some sites demand a user account and in some cases payment. Another could be, that you need to install software to adapt to a different codec, browser and/or service specific requirement. A requirement a user puts on a content source, is availability. It’s very annoying for a user if he for example wants to listen to a radio station but can’t, because the station is currently not available. Even the obstacles presented earlier in this part, is about availability. Cause if a user comes across to many obstacles, he tends to loose interest.

From an implementer’s point of view, there are more things to take into account. You have to adapt the system to different transport systems (RTSP, HTTP, MMS) and codec’s (WMA, RAM, MP3). You might also need to handle the communication and authentication to servers who distribute the media content.
4.2 IP-STB system analysis

This section analyses the IP-STB system, presented in the 2 IP-STB System chapter. It will work as a foundation for the choice of content source, made in section 4.3 Choosing content source, together with section 4.1 Content analysis.

The IP-STB has support for the RTP and HTTP protocols and the MP3 and WMA codec’s (this includes support for the ASF package format). It uses Mozilla for accessing the World Wide Web and has its own media player for playing the media content. There’s no support for downloading and installing software on the system. This means that if support for another codec or software is needed, it has to be implemented on the system. It can’t be installed by a user.

The HTTP protocol support is done by using the cURL library. A typical communication cycle is that the box sends a HTTP HEAD request to determine if the web server is up. If an answer is received a HTTP GET request is sent.

4.3 Choosing content source

In deciding which source to adapt the IP-STB to, the previous two sections were used as a base. The emphasis for the decision will be on the user, since the IP-STB is a consumer product and everything implemented on it will benefit the user. The following criteria were taken into account when deciding the content source:

1) Content availability
   For the user, availability is essential. As a user you tend to lose interest if the content is difficult to use or if it’s not available. For the IP-STB, availability is about finding a content source that doesn’t need specific software to be installed. This is because of the limitations in the hardware, which makes it impossible to install external software.

2) Support for the codec
   Implementing support for a new codec is an excessive job and there will not be enough time, within this thesis, to implement support for a new codec. This makes it necessary to find a source that’s coded with a supported codec.

The first criteria states that if the content is easy to use and access, it will be premiered over content that needs registration and/or payment. It also states that if the content site demands installation of specific software, it will be excluded. The second criteria states that if the content is coded with a format that the IP-STB don’t handle, it will be excluded.

If the providers, listed in section 3.6 Providers, are matched against these criterias the following selection can be made:

Excluded providers
   To be able to use Napster and ITunes, a download and installation of their software is necessary and it will violate the availability criteria. The content provided by Sveriges Television, TV4 and SF anytime is coded with the WMV format, which is not supported by the IP-STB and hence violates the Support for the codec criteria.

Possible providers
   Sveriges Radio, SHOUTcast, MSN Radio, MTV (the radio content), Live365 and Nordic Web Radio are all possible content providers for this project. For MSN Radio and Live365 you need to register and/or pay if you want full access to the content, this will make them less interesting than Sveriges Radio, SHOUTcast, MTV and Nordic Web Radio.

From the list of possible providers I will choose one to adapt the IP-STB to. I will exclude MSN Radio and Live365, because of the registration and payment obstacle. From the remaining four providers I chose Sveriges radio’s (SR) live web transmission. This choice was mainly based on
Chapter 4 Problem analysis

4.4 Analysing SR webradio

Sveriges Radio uses the WMA codec and HTTP transport for their live streaming (3.6.1 Sveriges Radio, www.sr.se). The WMA coded content is packaged in the ASF container format (Microsoft 2004). SR has outsourced the operation and maintenance of the content to a company called Qbrick (Qbrick 2005). This means that when you try to listen to any SR.se content you will be connecting to a Qbrick server.

Qbrick have a couple of servers that hold the streaming media and use a load balancing system to divide the workload evenly between the servers. The servers use the Microsoft media service to control the media streams.

The analysis started by analysing the transmission between the SR webradio on the Qbrick server and the Windows media player. This was accomplished with free software called Ethereal (Ethereal 2005). By listening to the transmission, it was established that the media stream was transported with the HTTP protocol, that the communication were in two steps (a header request followed by a media stream request) and that a couple of special header fields were needed. The analysis continued with a comparison between a streaming media ASF file from SR, and an ordinary ASF file stored on a computer. This was accomplished by dumping the ASF stream from SR on to the computer with the Curl software (Curl 2005) and then hex comparing it with the ordinary ASF file. The comparison showed that the streaming ASF file from SR had some extra objects added to it. These objects were identified as streaming HTTP objects. The conclusion is that the Qbrick server’s uses streaming HTTP to distribute the streaming media packaged in an ASF format. Chapter 5 Technology describes the streaming HTTP protocol and the ASF format in detail.

4.5 SR webradio on the IP-STB

This section describes the adaptation needed to implement a SRs webradio support on the IP-STB. It also describes the work process used to establish where in the IP-STB the changes are needed.

In section 4.4 Analysing SR webradio, SRs webradio transmission were analysed. The analysis stated that SR uses the streaming HTTP protocol to distribute their live, WMA coded, radio content in an ASF package. The IP-STB is already adapted to the WMA codec and the ASF package. It only lacks support for the streaming HTTP protocol.

A deeper analysis of the IP-STB was made to establish where in the media stream processing the support for streaming HTTP was needed. The entity responsible for the media stream processing is the streamer, which is described in section 2.3.2 Streamer. The first step of the analysis was to play an ASF packaged WMA stream, which was transported with pure HTTP. I.e. the ASF file was put on to an ordinary webserver and then contacting it from the IP-STB. The IP-STB used the media player to process the ASF file and the following streamer pipeline was built.

![Figure 5: ASF streamer pipeline](image)
The second step was to try to connect to an ASF stream that was transported with streaming HTTP, and compare it to the previous step. The result was that the streamer crashed when it tried to frame the data from the HTTP Source Element which indicates that a support for streaming HTTP is needed in the HTTP Source Element. The HTTP Source Element is used by the streamer to collect HTTP data from a source and process it to the Framing element. The collection of data is done by using the Curl library.

To adapt the HTTP Source Element to the Streaming HTTP protocol the following must be done:

Adapt the Curl library
The Curl library must be adapted to handle the Streaming HTTP communication, section 5.2 Streaming HTTP protocol describes the communication in detail.

Process the Streaming HTTP data
The HTTP Source Element must be able to process the Streaming HTTP data. This will be done by handling the different Streaming HTTP objects described in section 5.2 Streaming HTTP protocol.

Figure IP-STB, SR interaction gives an overview of the system and how the different parts interact. It also shows where the Streaming HTTP support is implemented. The left box represents the

![Diagram](image)

**Figure 6: IP-STB, SR interaction**
parts that will be active when a user connects to SRs webradio on the right. The Demonstration application uses the Media player to connect to and play the webradio transmission. When the Media player gets a request to connect to the webradio, it will start building a streamer pipeline where the first element is the *HTTP Source Element*. The *HTTP Source Element* then uses the Curl library to connect to the Qbrick servers and later on, to transport the ASF data. The Curl library handles all of the Internet communication and will provide the Media player with the Streaming HTTP data.

### 4.6 Requirements

This section will define requirements put on the implementation, in order for the IP-STB to be fully adapted to the SRs webradio. The requirements are based on the adaptation described in section 4.5 *SR webradio on the IP-STB*. The implementation can be divided into three distinct parts, Communication, Streaming HTTP support and Demonstration application.

The following example shows the notation used to specify the requirements.

**R 0.0 Example requirement**

<table>
<thead>
<tr>
<th>Description</th>
<th>A description of the requirement.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motivation</td>
<td>An explanation of why this requirement is stated.</td>
</tr>
<tr>
<td>Influences</td>
<td>Reference to requirements that are influenced by this requirement.</td>
</tr>
<tr>
<td>Influenced by</td>
<td>Reference to requirements that influence this requirement.</td>
</tr>
</tbody>
</table>

#### 4.6.1 Communication

The communication between the client and the server in a streaming HTTP session is described in the 5.2.1 *Communication*. The communication will be done in four steps, the first step is to identify a Streaming HTTP connection, the second step is to do a streaming HTTP header request, the third step is to parse the Streaming HTTP header and finally a Streaming HTTP data request is done. The following requirements will be put on the Communication parts:

**R 1.1 Identifying Streaming HTTP**

<table>
<thead>
<tr>
<th>Description</th>
<th>The Communication is able to identify an attempt to connect to a Streaming HTTP source.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motivation</td>
<td>We need to identify that the IP-STB is trying to connect to a Streaming HTTP source, to be able to adapt and parse the communication. This is also necessary because, the IP-STB needs to be able to handle other communication techniques.</td>
</tr>
<tr>
<td>Influences</td>
<td>R 1.2, R 1.3, R 1.4</td>
</tr>
<tr>
<td>Influenced by</td>
<td>-</td>
</tr>
</tbody>
</table>
Problem analysis

Chapter 4

4.6.2 Streaming HTTP data support

To implement a Streaming HTTP support it’s necessary to handle the data objects added by the Streaming HTTP. This will be done in two steps, parsing Streaming HTTP data and processing the ASF data. An extensive description of the HTTP streaming protocol can be found in the chapter 5.2 Streaming HTTP protocol and the requirements it puts on the Streaming HTTP data support are:

<table>
<thead>
<tr>
<th>R 1.2 Streaming HTTP header request</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Motivation</strong></td>
</tr>
<tr>
<td><strong>Influences</strong></td>
</tr>
<tr>
<td><strong>Influenced by</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R 1.3 Parse Streaming HTTP header</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Motivation</strong></td>
</tr>
<tr>
<td><strong>Influences</strong></td>
</tr>
<tr>
<td><strong>Influenced by</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R 1.4 Streaming HTTP data request</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Motivation</strong></td>
</tr>
<tr>
<td><strong>Influences</strong></td>
</tr>
<tr>
<td><strong>Influenced by</strong></td>
</tr>
</tbody>
</table>
**R 2.1 Parse Streaming HTTP data**

| **Description** | The Streaming HTTP data support is able to parse and handle the Streaming HTTP data objects. |
| **Motivation** | The Streaming HTTP data contains information about how the ASF data is to be handled. This information needs to be collected and used. And is hence the purpose of this requirement. |
| **Influences** | R 2.2 |
| **Influenced by** | - |

**R 2.2 Process ASF data**

| **Description** | The Streaming HTTP data support is able to process valid ASF data to the next element in the streamer. |
| **Motivation** | Only valid ASF data should be processed to the next element in the streamer pipeline, which is controlled by this requirement. |
| **Influences** | - |
| **Influenced by** | R 2.1 |

**4.6.3 Demonstration application**

To demonstrate that you could listen to webradio on the STB, a Demonstration application will be built. It must handle the connection and the controlling of the media stream. The requirements put on the Demonstration application are:

**R 3.1 Connecting to SRs webradio**

| **Description** | The Demonstration application is able to connect to SR’s webradio transmission. |
| **Motivation** | To be able to demonstrate that the IP-STB can connect to SR’s webradio transmission. |
| **Influences** | R 3.2 |
| **Influenced by** | - |
### R 3.2 Controlling the media stream

<table>
<thead>
<tr>
<th>Description</th>
<th>The Demonstration application is able to control the media stream. E.g. Play and Stop.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motivation</td>
<td>To be able to demonstrate that the IP-STB can control the media stream.</td>
</tr>
<tr>
<td>Influences</td>
<td>-</td>
</tr>
<tr>
<td>Influenced by</td>
<td>R 3.1</td>
</tr>
</tbody>
</table>
This chapter will present some of the technologies used in the thesis.

This master thesis uses a couple of technologies, for the analysis and implementation of the system. The technologies are: Curl, ASF and Streaming HTTP. The analysis described in section 4.4 Analysing SR webradio, uses Curls command line tool, curl, and the ASF specification for the analysis of Sveriges Radio’s webradio transmission. The choice of using curl for the analysis was obvious, since the IP-STB uses the Curl technique to communicate over an IP network. Sveriges Radio sends its content ASF packaged which means that the ASF specification is needed in the analysis of the content to separate the ASF data from the Streaming HTTP data. The implementation described in chapter 6, uses the Curl library, libcurl, to communicate with Sveriges radio’s content servers and is adapted to handle the Streaming HTTP protocol. The result of the Streaming HTTP data parsing, described in section 6.2.1 Parse Streaming HTTP data, should be an ASF data stream. This is validated using the ASF specification.

5.1 cURL

Curl is free and open software originated by the Haxx organisation. The software is a multi protocol file transfer library, supporting FTP, FTPS, HTTP, HTTPS, Gopher, Telnet, Dict, File and LDAP. I.e. you can use Curl for data transfer over the Internet. Curl comes in a library called libcurl and a command line tool called curl.

The cURL software is used in a couple of different ways in the thesis. The command line tool is used in the analysis of Sveriges Radio’s webradio transmission, presented in section 4.4 Analysing SR webradio. The libcurl library is used by the IP-STB to connect to the Internet. It will be extended to handle the Streaming HTTP communication. Section 6.1 Communication explains how the extension is implemented.
This section will only describe the parts of the Curl library that are of interest in the thesis. All information presented in this section is gathered from their web site. To view this information and for further information, visit their web site curl.haxx.se.

5.1.1 Command line tool, curl

curl is a command line tool used to transfer data between a server and a client on the Internet. It’s available under a wide variety of operating systems. curl uses libcurl for all data transfer that it performs.

The tool synopsis is:

```
curl [option] [URL]
```

There are a wide variety of options available, the ones relevant to this thesis will be explained here. For further information please visit curl.haxx.se or look at manual pages.

- `-A/--user-agent <agent string>`

  Specifies the User-Agent field in the HTTP header.

5.1.2 libcurl

There’s two interfaces for the library, the synchronous easy and the asynchronous multi. In this thesis only the easy interface will be used and described.

To initiate an Internet transfer you will need to create a handle with the `curl_easy_init()` function. The handle is a logical entity that holds information and settings about the transfer. It’s important to have one handle for every thread used for transferring. Threads should never share a handle.

To set options in the handle you need to use the `curl_easy_setopt(<handel>, <option>, <value>)` method. The option is used to set options of the type behaviour, callback, error, network, authentication, http, ftp, protocol, connection, security or telnet. The value is used to specify the value of the option and is of the type string or a pointer to data.

If you e.g. want to receive data from a URL you need to set the url option to specify which URL you want to receive data from. It’s done with the following call to `curl_easy_setopt`:

```
curl_easy_setopt(handle, CURLOPT_URL, "http://domain.com/");
```

If you then want to handle the data received from the server, you need to set the `CURLOPT_WRITEFUNCTION` option to the name of a function that is to be used for data processing. If this is not set the curl will print it to the stdout. It’s done with the following call to `curl_easy_setopt`:

```
curl_easy_setopt(handle, CURLOPT_WRITEFUNCTION, <function name>);
```

To modify the HTTP header you need to set the option `CURLOPT_HTTPHEADER`. The way to this is to create a `curl_slist` and append the header values with a call to:

```
curl_slist_append(<header>, User-Agent: NSPlayer);
```

Then it’s just to add the header to the handle:

```
curl_easy_setopt(easyhandle, CURLOPT_HTTPHEADER, headers);
```

When all options are set, the `curl_easy_perform(<handle>)` function is used to execute the transfer.
5.2 Streaming HTTP protocol

The Streaming HTTP protocol is a real time streaming protocol from Microsoft that’s based on the HTTP 1.0/1.1 transfer protocol. It’s used by servers that use the Microsoft Media Service (MMS) for distributing streaming media content. There’s no publicly available documentation, so the information is collected from SDP team’s (SDP team 2003) home page.

The following sections describe the communication between clients and servers and the data sent between them. This description will serve as a specification for the implementation parts communication and Streaming HTTP data support.

Sveriges Radio’s webradio transmission is distributed using the Streaming HTTP protocol, which is described in section 4.4 Analysing SR webradio.

5.2.1 Communication

This section describes the communication between the client and the server. It will also describe the headers and data sent between them. This information will be used when implementing the communication and the Streaming HTTP data support, in chapter 6. The communication starts with a HTTP 1.0/1.1 request/response session and continues with a streaming media session. A HTTP streaming protocol session exchange is shown in figure HTTP streaming session.

The first step of the exchange is for the client to connect to the server and establish a connection with its TCP socket. Note! If the server is up and running it will respond with a HTTP OK statement. But if you use the HTTP HEAD (HEAD is used to just receive the header) request the server might answer with an error message, i.e. it’s safer to do a HTTP GET request to avoid any confusion. When a connection is established the client requests an ASF header from the server. The server responds with an ASF header containing the Streaming HTTP information that’s needed by the client for further communication, section 5.2.2 Server sent ASF header describes the header and the information in detail. After the server has sent the header it will close the connection, this is to prevent the client from getting streaming data that it can’t handle. After the client have parsed the header information, i.e. when it has extracted the header information that it needs for further communication with the server, it will reconnect with the server and make a request for a stream with appropriate header values set, section 5.2.3 Client sent HTTP header describes the stream request header. The server will then respond with the ASF header followed by some media packets and then continue to transfer data packet until it reaches the end of the stream and send the client an End Of Media (EOM) command. When the client gets an EOM command it stops the media player. The media and data packets are described in section 5.2.4 Server sent HTTP body data.
5.2.2 Server sent ASF header

This section provides a description of the ASF header that the server sends as a response to the clients ASF header request. The header is parsed by the client to collect information about the ASF stream and the information is used by the client to make a stream request. The information presented in this section is used for the implementation of the Streaming HTTP header parsing functionality that is described in section 6.1.3 Parse Streaming HTTP header. It's used to identify and collect the different ASF header values that the server sends.
The **Server sent ASF header sample** table, presents a sample of an ASF header and is followed by an explanation of the header values and what they are used for.

### Table 2: Server sent ASF header sample

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-Type:</td>
<td>application/vnd.ms.wms-hdr.asfv1</td>
</tr>
<tr>
<td>Server:</td>
<td>Cougar/9.00.0.3101</td>
</tr>
<tr>
<td>Content-Length:</td>
<td>1765</td>
</tr>
<tr>
<td>Date:</td>
<td>Mon, 15 Aug 2005 12:07:35 GMT</td>
</tr>
<tr>
<td>Pragma: client-id</td>
<td>78563219</td>
</tr>
<tr>
<td>Pragma: features</td>
<td>“broadcast, playlist”</td>
</tr>
<tr>
<td>Pragma: x-wms-content-size</td>
<td>673291</td>
</tr>
</tbody>
</table>

The field **Content-Type** is used to show which MIME type the data has. Typical values for this protocol are `application/x-mms-framed`, which is the MIME type for ASF media packets using MMS style pre-headers.

The field **Server** is used to show the server name and version.

The field **Content-Length** is used to show the total length of the HTTP packet.

The field **Date** is used to show the current connection date and time.

The field **Pragma: Client-ID** is used to give the client a unique identifier.

The field **Pragma: feature** is used to set the number and type of streams sent in the ASF transport stream. Values can be `broadcast`, `playlist`, `seekable` and `stridable`.

The field **Pragma: x-wms-content-size** is used to set the total length of the media type in bytes.

When the client requests the media stream it must specify the number of streams that it wants to receive. The field **Pragma: feature** contains information about the amount and type of streams available. The information in this field will be collected and then used for the media stream request. Section 5.2.3 **Client sent HTTP header** describes how the collected information is used in the request.

#### 5.2.3 Client sent HTTP header

This section provides a description of the header sent by the client in the media stream request. The content of the header is dependent on what information the client got from the server sent ASF header, described in the previous section. The information presented in this section is used to implement the Streaming HTTP data request described in section 6.1.4 **Streaming HTTP data request** and it shows an example of the values set in the header. This information is vital for the communication since if the values are not properly set we will not receive the stream in the order that we would expect or even worse we will not receive the stream at all.
The Client sent HTTP header table, presents a sample of the header sent in the media stream request and it’s followed by an explanation of the header values and what they are used for.

Table 3: Client sent HTTP header

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept:</td>
<td><em>/</em></td>
</tr>
<tr>
<td>User agent:</td>
<td>NSPlayer/10.0.0.3802</td>
</tr>
<tr>
<td>Host:</td>
<td>sr-wm.qbrick.com</td>
</tr>
<tr>
<td>Pragma: version11-enable</td>
<td>1</td>
</tr>
<tr>
<td>Pragma: no-cache</td>
<td>-</td>
</tr>
<tr>
<td>Pragma: rate</td>
<td>1.000000</td>
</tr>
<tr>
<td>Pragma: request-context</td>
<td>2</td>
</tr>
<tr>
<td>Pragma: max-duration</td>
<td>0</td>
</tr>
<tr>
<td>Pragma: xPlayStrm</td>
<td>1</td>
</tr>
<tr>
<td>Pragma: xClientGUID</td>
<td>3300AD50-2C39-46c0-AE0A-9D3EC3C32549</td>
</tr>
<tr>
<td>Pragma: stream-switch-count</td>
<td>2</td>
</tr>
<tr>
<td>Pragma: stream-switch-entry</td>
<td>ffff:1:0 ffff:2:0</td>
</tr>
</tbody>
</table>

The field User-Agent is used to show the client player version and type.

The field Host is used to show the host the client is trying to connect to.

The field Pragma: version11-enable is probably used to tell the server that we can handle HTTP version 11 packets. I haven’t found a lot of information about it, but through what I have read on Microsoft.com and what I have observed in the data sent, this seems to be a satisfiable conclusion.

The field Pragma: no-cache is used to indicate if no cache is to be used.

The field Pragma: rate is always set to 1.000000.

The field Pragma: stream-time is used to set the offset time into the media. 0 indicates the start of the media stream.

The field Pragma: stream-offset is used to set the offset point into the media. 0:0 indicates the start of the media stream.

The field Pragma: request-context is used to set what type of content the client wants. The client can use 1 for initial header request, 2 for pre-recorded or live media packets requests and 4 for seek into media requests.

The field Pragma: max-duration is used to set the maximum time allowed for streaming. After this time, the server will disconnect from the client. As default this value is set to 0, which represents no limit.

The field Pragma: xClientGUID is set by the client upon connection to the server and changes for each new connection made.
The field *Pragma: xPlayStream* is always set to 1.

The field *Pragma: stream-switch-count* is used to set the number of streams that are requested. The value is parsed from the *Server sent HTTP body data Pragma: feature*.

The field *Pragma: stream-switch-entry* is used to select or deselect the requested streams. It’s set with the following syntax `fff:ID:state`, where ID is the stream id (i.e. the first stream has ID 1, the second has ID 2 etc.) and the status is one of the following:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Full frame rate (ON).</td>
</tr>
<tr>
<td>1</td>
<td>Key frames only.</td>
</tr>
<tr>
<td>2</td>
<td>Deselected (OFF).</td>
</tr>
</tbody>
</table>

The server’s response to this request is, to start sending data. A description of the server sent data is presented in the next section.

### 5.2.4 Server sent HTTP body data

This section describes the data that the server sends as a response to the client’s media stream request. The information presented in this section is used by the *6.2.1 Parse Streaming HTTP data* section in the implementation chapter. When the Streaming HTTP objects are removed from the body data, they will contain the actual data and payloads of the ASF format.

The body data starts with 4 bytes of header structure, divided in the following way:

<table>
<thead>
<tr>
<th>Field name</th>
<th>Field type</th>
<th>Size (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>WORD</td>
<td>16</td>
</tr>
<tr>
<td>Length</td>
<td>WORD</td>
<td>16</td>
</tr>
</tbody>
</table>

**Command**

The command value can have the following meanings:

<table>
<thead>
<tr>
<th>Hex</th>
<th>ASCII</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x2448</td>
<td>$H</td>
<td>ASF header follows</td>
</tr>
<tr>
<td>0x2444</td>
<td>$D</td>
<td>ASF data packets follows</td>
</tr>
<tr>
<td>0xA444</td>
<td>?D</td>
<td>ASF data packets follows</td>
</tr>
</tbody>
</table>
The length value indicates the total length in bytes of the packet. Starting from the first byte after this header. For the End of stream and Changing media command the length is the length of this header, i.e. 0x08.

After the header there is a type object structure, divided in the following way:

**Table 7: Type Object**

<table>
<thead>
<tr>
<th>Field name</th>
<th>Field type</th>
<th>Size (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence number</td>
<td>DWORD</td>
<td>32</td>
</tr>
<tr>
<td>ID</td>
<td>BYTE</td>
<td>8</td>
</tr>
<tr>
<td>Flags</td>
<td>BYTE</td>
<td>8</td>
</tr>
<tr>
<td>Length</td>
<td>WORD</td>
<td>16</td>
</tr>
</tbody>
</table>

### Sequence number
The sequence number is a number that increments for every ASF packet sent. If it’s a pre-recorded media stream it starts from 0 and if it’s a live media stream it will start on the number that it happens to have when you tune in.

### ID
The ID value tells you if it’s live- or pre-recorded media stream. 0x00 for live and 0x01 for pre-recorded.

### Flags
Flags are used to indicate where you are in a stream. 0x00 indicates the middle of a packet’s series, 0x04 indicates first packet of a series, 0x08 indicates last packet of a series and 0x0C indicates that this is the only packet of a series.

### Length
The length value indicates the total length, in bytes, of this packet including the type object structure. I.e. it should be equal to header length value.

After the type object the ASF header object or data packet follows, more about ASF in 5.3 *Advanced system format, ASF.*
5.3 Advanced system format, ASF

ASF is a file format that is designed to store and transport media data. It’s developed and distributed by Microsoft. The information presented in this section originates from the ASF Specification (Microsoft 2004).

An ASF file can contain one or more media streams. The file header specifies the properties of the entire file and the streams carried. It supports live and on-demand media content and is specifically designed for streaming and/or local playback.

The IP-STB already supports the ASF format which means that no further support for it needs to be implemented. But this specification was used to analyse Sveriges Radio’s webradio transmission, in section 4.4 Analysing SR webradio and to validate that the Streaming HTTP data support, described in section 6.2 Streaming HTTP data support, delivers correct ASF data to the next element in the streamer pipeline. The section will only cover the basic file structure and not go into any details, further information is found in the ASF specification (Microsoft 2004).

5.3.1 ASF file structure

ASF files are divided into three parts, the header object, the data object and the index objects. The header object is mandatory and must be placed at the beginning of the file. It contains information about the ASF file and the streams that it transports. The data object is also mandatory and must follow the header object. It contains the data and payloads from the streams. The index objects are optional and can be used to provide time-based access into the ASF file. This is illustrated in figure ASF file structure.

```
<table>
<thead>
<tr>
<th>Header Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Properties Object</td>
</tr>
<tr>
<td>Stream Properties Object 1</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>Stream Properties Object N</td>
</tr>
<tr>
<td>&lt;Other header objects&gt;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Packet 1</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>Data Packet M</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Index Object 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
</tr>
<tr>
<td>Index Object K</td>
</tr>
</tbody>
</table>
```

Figure 8: ASF file structure
This chapter will cover how the system was implemented to fulfill the requirements. The implementation will consist of three parts: Communication, Streaming HTTP data support and Demonstration application.

The implementation is done in three parts: Communication, Streaming HTTP data support and Demonstration application. The Communication part handles the initial connection to the content source, the Streaming HTTP data support handles the parsing and processing of data and the Demonstration application part implements an application used for demonstration purposes. Requirements on the parts are specified in section 4.6 Requirements and will be used to ensure that a complete Streaming HTTP support is implemented. The three parts are implemented in the IP-STB as shown in figure IP-STB, Sveriges Radio interaction. It shows how the parts work together when a user uses the system. The demonstration application will use the media player to play the content that the user selects. The media player will use the Curl library to communicate with the server holding the content. The figure also shows where the implemented parts fit in the system.

6.1 Communication

The communication between the client and the server in a streaming HTTP session is described in the 5.2.1 section and needs to be implemented to support the initial communication between the client and the server. The Curl library is used by the IP-STB for communicating with the Internet and it’s in the Curl library the Communication will be implemented.

Section 4.6.1 Communication specifies the requirements for the Communication implementation. The requirements are, Identifying Streaming HTTP, Streaming HTTP header request, Parse Streaming HTTP header and Streaming HTTP data request. A description of how these requirements are implemented follows.
6.1.1 Identifying Streaming HTTP

Implementation of the Identifying Streaming HTTP functionality is controlled by the R 1.1 Identifying Streaming HTTP requirement.

The Curl library handles all Internet requests on the IP-STB. This means that it’s necessary to recognize when there’s a HTTP streaming request. This is done by searching the URL for occurrences of HTTP:// and MSWMExt=.asf. The HTTP:// indicates that this is a HTTP request to a web server and the MSWMExt=.asf tells the server that it shall set the mswnext parameter to .asf in the server script (Microsoft, 2003). The CurlMultiDocument member function OpenHeader is responsible for the initial URL request and the identify support is implemented there.

When the request has been identified as a Streaming HTTP request it will be processed by the Streaming HTTP header request functionality, described below.

6.1.2 Streaming HTTP header request

Implementation of the Streaming HTTP header request functionality is controlled by the R 1.2 Streaming HTTP header request requirement.
The `CurlMultiDocument` member function `OpenHeader` is used by the STB as a “Check if the web server is up routine” and uses a HTTP HEAD request to achieve it. This is a bit of a problem, because the HTTP streaming server could refuse the HTTP HEAD request and send a HTTP error 501 (The request not implemented) in response. If the type of answer weren’t of any interest it could have been ignored and it could have been stated that the server is responding. But in this case it’s necessary to parse the answer for future communication, which makes a HTTP GET request to the server vital. This is achieved by changing the `CURLOPT_NOBODY`, in the handle, to false. It will make Curl use the HTTP GET request.

```c
curl_easy_setopt(easyhandle, CURLOPT_NOBODY, false);
```

To be able to parse the answer, the `CURLOPT_WRITEHEADER` option is set to `parseHttpStreamHeader`.

```c
curl_easy_setopt(easyhandle, CURLOPT_WRITEHEADER, parseHttpStreamHeader);
```

It’s also necessary to set the HTTP Pragma field’s `no-cache, rate, stream-time, stream-offset, request-context, max-duration, xClientGUID` and the `User-Agent` field in the HTTP header. This is done by creating a `curl_slist` and appending the fields with the `curl_slist_append` function and the header should look like this:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept:</td>
<td><em>/</em></td>
</tr>
<tr>
<td>User agent:</td>
<td>NSPlayer/10.0.0.3802</td>
</tr>
<tr>
<td>Host:</td>
<td>sr-wm.qbrick.com</td>
</tr>
<tr>
<td>Pragma: no-cache</td>
<td>-</td>
</tr>
<tr>
<td>Pragma: rate</td>
<td>1.000000</td>
</tr>
<tr>
<td>Pragma: stream-time</td>
<td>0</td>
</tr>
<tr>
<td>Pragma: stream-offset</td>
<td>0:0</td>
</tr>
<tr>
<td>Pragma: request-context</td>
<td>1</td>
</tr>
<tr>
<td>Pragma: max-duration</td>
<td>0</td>
</tr>
<tr>
<td>Pragma: xClientGUID</td>
<td>3300AD50-2C39-46c0-AE0A-9D3EC3C32549</td>
</tr>
</tbody>
</table>

A detailed description of the fields can be found in the 5.2.3 Client sent HTTP header section.

The request will result in that a Streaming HTTP header will be sent. The information in the header is used to request the Streaming HTTP data and must hence be parsed. How this is implemented is described in the next section.

Table 8: A HTTP header for a header request
6.1.3 Parse Streaming HTTP header

Implementation of the Parse Streaming HTTP header functionality is controlled by the R 1.3 Parse Streaming HTTP header requirement.

To be able to send a Streaming HTTP data request, it’s necessary to parse information from the Streaming HTTP header, received with the Streaming HTTP header request. This is done by searching header for information and then extracting it. The information needed is, the number and type of streams included and the Pragma: feature field. This information is used to set the stream-switch-count and stream-switch-entry fields in the Streaming HTTP data request.

6.1.4 Streaming HTTP data request

Implementation of the Streaming HTTP data request functionality is controlled by the R 1.4 Streaming HTTP data request requirement.

When making the request for the HTTP stream we need to add the information extracted in 6.1.3 in the stream-switch-count and stream-switch-entry fields. The Pragma field’s version11-enable, no-cache, rate, stream-time, stream-offset, request-context, max-duration, xPlayStrm, xClient-GUID and the User-Agent field are also needed. Which fields to set and the value of them are decided by what type of stream that is requested. The headers for a live and a pre-recorded media stream look as below:

Table 9: A HTTP header for a live stream request

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept:</td>
<td><em>/</em></td>
</tr>
<tr>
<td>User agent:</td>
<td>NSPlayer/10.0.0.3802</td>
</tr>
<tr>
<td>Host:</td>
<td>sr-wm.qbrick.com</td>
</tr>
<tr>
<td>Pragma: version11-enable</td>
<td>1</td>
</tr>
<tr>
<td>Pragma: no-cache</td>
<td>-</td>
</tr>
<tr>
<td>Pragma: rate</td>
<td>1.000000</td>
</tr>
<tr>
<td>Pragma: request-context</td>
<td>2</td>
</tr>
<tr>
<td>Pragma: max-duration</td>
<td>0</td>
</tr>
<tr>
<td>Pragma: xPlayStrm</td>
<td>1</td>
</tr>
<tr>
<td>Pragma: xClientGUID</td>
<td>3300AD50-2C39-46c0-AE0A-9D3EC3C32549</td>
</tr>
<tr>
<td>Pragma: stream-switch-count</td>
<td>2</td>
</tr>
<tr>
<td>Pragma: stream-switch-entry</td>
<td>ffff:1:0 ffff:2:0</td>
</tr>
</tbody>
</table>
An Internet content overview and implementation on an IP based set-top box

The main difference is in the *stream-time* and *stream-offset* fields. These fields are only valid for pre-recorded streaming media.

### 6.2 Streaming HTTP data support

To implement a Streaming HTTP support it’s necessary to handle the extra headers and information that the streaming HTTP server adds to the ASF stream. An extensive description of the Streaming HTTP protocol can be found in the chapter 5.2 *Streaming HTTP protocol*. The Streaming HTTP protocol adds the Meta data, the Header, the Data, the End of stream, the Changing media and the Packet pair objects and it’s necessary to remove them from the ASF media stream. It’s also necessary to add padding data to the ASF Data packets, because of a difference in the size of the expected and the actual received data packets. The module best suited to introduce streaming HTTP support is the HTTP Source Element module of the streamer, because this object will process all data received by the HTTP transfer. The function that’s in charge of processing the data is *CurlWriteDocument* and the functionality for parsing and processing the HTTP

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept:</td>
<td><em>/</em></td>
</tr>
<tr>
<td>User agent:</td>
<td>NSPlayer/10.0.0.3802</td>
</tr>
<tr>
<td>Host:</td>
<td>sr-wm.qbrick.com</td>
</tr>
<tr>
<td>Pragma: version11-enable</td>
<td>1</td>
</tr>
<tr>
<td>Pragma: no-cache</td>
<td>-</td>
</tr>
<tr>
<td>Pragma: rate</td>
<td>1.000000</td>
</tr>
<tr>
<td>Pragma: stream-time</td>
<td>0</td>
</tr>
<tr>
<td>Pragma: stream-offset</td>
<td>4294967295:4294967295</td>
</tr>
<tr>
<td>Pragma: request-context</td>
<td>2</td>
</tr>
<tr>
<td>Pragma: max-duration</td>
<td>0</td>
</tr>
<tr>
<td>Pragma: xPlayStrm</td>
<td>1</td>
</tr>
<tr>
<td>Pragma: xClientGUID</td>
<td>3300AD50-2C39-46c0-AE0A-9D3EC3C32549</td>
</tr>
<tr>
<td>Pragma: stream-switch-count</td>
<td>2</td>
</tr>
<tr>
<td>Pragma: stream-switch-entry</td>
<td>ffff:1:0 ffff:2:0</td>
</tr>
</tbody>
</table>

The main difference is in the *stream-time* and *stream-offset* fields. These fields are only valid for pre-recorded streaming media.

---

1. In the ASF header it’s stated that an ASF data packet should be of a specific size. But in reality the data packet is 3 bytes less than that. This is an unexplained phenomenon (no support for this have been found in any specification) that occurs when we get the data from a streaming HTTP server.
stream to an ASF stream, is added there. The HTTP support will be achieved in two steps, parsing and processing. A flow chart of the streaming HTTP support can be viewed in figure *Streaming Http flow chart*.

![Streaming HTTP flow chart](image)

**Figure 10: Streaming HTTP flow chart**

The Buffer IN is where the received data is buffered. The parsing step checks the first two bytes for what type of an object it has received. If the data isn’t recognized it will be sent to the buffer out, the buffer out is where the next element in the stream will get its data. If the data is to be put on the buffer out the streaming HTTP objects will be removed and padding data will be added.

Section 4.6.2 *Streaming HTTP data support* specifies the requirements for Streaming HTTP data support implementation. The requirements are: Parse Streaming HTTP data and Process ASF data. A description of how these requirements are implemented follows.
6.2.1 Parse Streaming HTTP data

Implementation of the Parse Streaming HTTP data functionality is controlled by the \textit{R 2.1 Parse Streaming HTTP data} requirement.

This step will identify the different HTTP streaming objects and set three general and two object specific parameters that will affect the processing step. The three general parameters are:

\textbf{asfDataToBuffer} \\
The flag is set if the ASF data following the streaming HTTP object should be processed to the buffer out.

\textbf{streamingHttpObjectSize} \\
The parameter is set to the size of the streaming HTTP object.

\textbf{MetaDataFlag} \\
The flag is set if the streaming HTTP object is of the type Meta Data and unset if it’s not. This will ensure that no Meta Data will be processed to the buffer out.

The two object specific parameters are:

\textbf{dataPacket} \\
The flag is only set if the streaming HTTP object Data Packet is found. This will be used by the processing step to decide if padding data should be added.

\textbf{firstDataPacket} \\
The parameter is a state machine for the streaming HTTP Data Packets and is defined as follows:

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure11.png}
\caption{Data Packet state machine}
\end{figure}

\begin{itemize}
\item \textbf{State 0} \\
Indicates that padding data is to be added and will continue adding until a Header Packet arrives.
\item \textbf{State 1} \\
Waits for the first Data Packet to arrive.
\item \textbf{State 2} \\
Waits for the second Data Packet to arrive.
\end{itemize}

This state machine were introduced to control that padding data is added after the first Data Packet.

6.2.2 Process ASF data

Implementation of the Process ASF data functionality is controlled by the \textit{R 2.2 Process ASF data} requirement.

In this step the data will be processed and only valid ASF data is passed to the next module in the streamer. It will be achieved by using the parsed information from the previous step. The asf–
DataToBuffer will tell if the data is to be passed to the buffer out. The dataPacket and the firstDataPacket will decide if any padding data is needed.

If a changing data object is found the buffer out will be emptied to enable a new media transfer.

### 6.3 Demonstration application.

To demonstrate that you could listen to webradio on the IP-STB, a Demonstration application will be built. Section 4.6.3 Demonstration application specifies the requirements for the Demonstration application. The requirements are: Connecting to SR’s webradio and Controlling the media stream. There is already an application built for the IP-STB that can handle these two requirements, it’s called TOI tutorial and is a very simple application that is used as a programming tutorial. When the tutorial is finished, the application is able to connect and control a media stream. This suits the Demonstration application well and it will be used to demonstrate that we can listen to Sveriges Radio’s webradio transmissions. The only change to it, is that a list of stations are added. The Demonstration applications GUI has a screen to display video content, a drop down list were a content source URL can be chosen and four control buttons (Start, Stop, Play and Pause) as shown in figure Demonstration application.

![Demonstration application](image)

**Figure 12: Demonstration application**

To play a content source the user needs to do the following:

1. Choose content source with the drop down list.
2. Press the START button to connect to the source.
3. Press the PLAY button to play the content.
4. Press the STOP button to disconnect from the source.

A state machine for the Demonstration application is shown in figure State machine for the Demonstration application and it will be in the Choose source state when it’s started. The state machine follows the interaction from the user and will change state when a button is pressed.
Figure 13: State machine for the Demonstration application
Conclusion & Further development

This chapter gives a conclusion to this thesis work. It will give an analysis of the work done on the different parts. The chapter should be consulted before any further development on this work is done.

7.1 Conclusion

In this section I will conclude my thesis work and give a reflection over the different parts.

7.1.1 Investigation and analysis

The aim for the investigation was to make it broad and thorough and look at as many sources as possible. I soon felt that this aim was a bit excessive and tried to narrow the search down to include only well established media content providers. Even though I made this limitation I felt that I spent to much time in the investigation part. But the investigation covered some of the most important Internet content providers on the market, which was the most important.

The choice of SR webradio as media source was quite easy. They were the provider that best met the requirements.

7.1.2 Adaptation

The adaptation to Sveriges Radio’s web transmissions was a bit tricky at the beginning because I thought that the stream was delivered with pure HTTP and that it just started with some meta data at the beginning. But when I discovered that the stream was delivered with streaming HTTP I got the adaptation on the way.

7.1.3 Streaming HTTP support

Due to the delay in the beginning of the implementation phase there were not enough time to do a complete streaming HTTP support. The adaptation lacks support for Packet Pair and Changing Media objects. But because of the need for a streaming HTTP protocol support to listen to SR’s webradio the IP-STB has support for all media content delivered with the streaming HTTP proto-
col. This was tested by adding NRJ’s webradio transmission to the demonstration application and playing it.

### 7.1.4 Media player

I made my investigation on a Windows media player 10 and took the request parameters from that communication, it’s a possibility that the streaming HTTP server could assume that the media player on the IP-STB could handle things that a Windows media player 10 normally can. I.e. it could be a source for instable performance.

### 7.1.5 Demonstration application

The demonstration application is very simple and it plays a windows media stream, which is delivered with the streaming HTTP protocol. To make this into an end user application, some important design decisions are needed.

1) Continuous media player updates

The Internet content sites assume that you have or can get a specific media player. This fact will require a continuous update of the media player and it will require that the media player can handle a variety of codec’s and transport formats.

2) Stand alone application vs. browser plug-in

When a GUI is implemented there are two options: build a stand alone application or build a media player plug-in for the browser. The biggest advantage with a browser plug-in is that the Internet content sites are developed to be accessed with a browser. Another advantage is that the users have access to what ever content source they want to and is not limited to a list of predetermined content providers. The biggest disadvantage with browser plug-in solution is that the browser uses a lot of the CPU and memory which could slow down the entire system. Another problem is that it’s up to the user to determine what content they can use and what content they can’t use. This problem could result in a lot of confusion and discontent from the user. The stand alone application is a much more memory and CPU efficient solution. But it’s much less flexible when it comes to the content it can access. It will be limited to a predetermined list of content sources which limits the usability of the system. The administration of the list is problematic and will need some decisions of its own. This will be discussed further in section three Static list of content vs. a non static one.

3) Static list of content vs. a non static one

If a stand alone application is to be implemented a decision upon how the list of available Internet content is to be administrated. The easiest solution is to implement a static list that is not updated. The obvious draw back is that such a list will soon be out of date and the service will stop working. The non static list will probably not go out of date but demands some sort of administration. A general draw back with the list solution is that the users don’t have the option to choose what content he/she wants to access.

### 7.2 Further development

This section will cover how this thesis work could be developed further to enhance the system.

#### 7.2.1 Codec’s

If more codec’s were implemented in the system it would make it more flexible and usable. There is a direct connection between the number of supported codec’s and the number of supported Internet content sources. E.g. a support for Real Medias codec’s would drastically increase the number of supported content sources since a lot of providers use Real Medias codec’s.
7.2.2 Transport protocols

If more transport protocols were to be implemented it would increase the number of supported content sources. There is a direct connection between the number of supported transport protocols and the number of supported content. E.g. a support for the MMS transport protocol would increase the number of supported content sources since it’s used by some providers today and it will probably be more common in the future. This is because it’s a relatively new system and it’s developed and marketed by Microsoft.

7.2.3 Streaming HTTP

The streaming HTTP support would need some further development to work properly and it’s highly recommended to get the Streaming HTTP specification from Microsoft. In this thesis I used the collected knowledge of the SDP team, but their specification is nothing but a good guess on how the Streaming HTTP protocol works, which makes it a bit error prone. To make the Streaming HTTP support complete a implementation of the Packet Pair and Changing Media objects are needed.

7.2.4 Streamer

My approach was to extend the HTTP streamer element with streaming HTTP support. But at the end of the implementation phase I realized that it would have been much better approach to create a streaming HTTP element in the streamer to handle the streaming HTTP support. But then there weren’t enough time to change it, which resulted in the implemented solution.
Conclusion & Further development

Chapter 7
Curl (2005), Information about the Curl software, <http://curl.haxx.se>
Accessed 09/08 2005

Accessed 05/07 2005

Accessed 05/10 2005

Kanal5 (2005), Information about the content provider, <http://www.kanal5.se>
Accessed 05/10 2005

Accessed 05/10 2005

Microsoft (2003), Playing a URL Involving Server-Side Processing Fails,
<http://support.microsoft.com/default.aspx?scid=KB;EN-US;Q236895&>

Microsoft (2004), Advanced System Format Specification,

Microsoft (2005), Microsofts Media Services,

Movielink (2005), Information about the content provider, <http://www.movielink.com>
Accessed 05/10 2005

Mozilla (2005), Information about the Mozilla software, <http://www.mozilla.com>
Accessed 05/10 2005

MSN (2005), Information about the content provider, <http://www.msn.com>
Accessed 05/10 2005


NRJ (2005), *Information about the content provider*, <http://www.nrj.se> Accessed 05/10 2005


Sveriges television (2005), *Information about the content provider*, <http://www.svt.se> Accessed 05/10 2005


This appendix will list and describe all acronyms and technical terms that will be used in this thesis. The descriptions are mostly gathered from Wikipedia (2005).

ASF
Advanced System Format (ASF) is an audio/video container format developed by Microsoft. A detailed description about ASF is given in section 5.5 Advanced system format, ASF.

Bit rate
Is the frequency (bit/s) at which bits of data are passed through a system.

Codec
Describes a device or program that’s capable of Coding or Decoding a data stream. The codec’s are used to code raw data into a manageable format and to decode it when it’s used.

Container format
Is a file that combines audio/video data, tags, menus and subtitles. For example the ASF format.

Cookies
Is a packet of information that is sent by a server to a web browser when it contacts the server for the first time. The Cookie is then used by the browser when it contacts the same server again.

CPU
A Central Processing Unit is the part of the computer that carries out instructions from contained in the software.
Flash memory

Is a rewritable memory that holds its memory content without the need of a power supply.

HTTP

*Hyper Text Transport Protocol (HTTP)* is a request/response protocol between clients and servers that mainly focuses on the transport of information. It was jointly developed by World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF) and it resulted in a couple of RFCs, where the RFC 2616 is for HTTP/1.1 and RFC 1945 is for HTTP/1.0.

Java script

Is a object-based scripting programming language based on the concept of prototypes. It’s often used for websites and access to objects embedded in applications.

IETF

The *Internet Engineering Task Force* is in charge of developing and promoting the Internet.

Container format

Is a file that combines audio/video data, tags, menus and subtitles. For example the ASF format.

MMS

Microsoft’s streaming servers uses the Microsoft Media Service protocol to transport data. This is used to extend the HTTP protocol with streaming media services.

MP3

MP3 is a digital audio encoding and compression format. It’s designed to greatly reduce the amount of data needed to represent audio, without too much loss in the audio quality. MP3 is based on the standard encoding technique MPEG-1 Audio Layer 3. There is also some implementations of MP3 with the newer standard encoding technique MPEG-2 Audio Layer 3. The *MPEG-1 Audio Layer 3* is a lossy compression codec that is used to store audio data.

MPEG

The *Moving Picture Experts Group* is in charge of developing video and encoding standards.

MPEG-1

Is a group of audio and video coding standards from the Moving Pictures Experts Group (MPEG).

MPEG-2

Is a group of audio and video coding standards from the Moving Pictures Experts Group (MPEG).

Multicasting

Is the delivery of information to a group of destinations.

OGG

Is an open source container format designed for efficient streaming and file compression.
Quick time

Quick Time is a multimedia technology developed by Apple Computer. QuickTime functions as a multimedia container, i.e. it can hold any type of multimedia content (audio, video, image, text). It is a container format developed by Apple Computer.

RAM

The Random Access Memory is used by computer to hold program code during execution. It’s possible to both read and write to it and it looses its content when the power is shut off.

Real Media

Is a container format created by Real Networks and holds Real Video and/or Real Audio formats. Real Media is a digital audio and video format developed by Real Networks.

Real Audio

Real Audio is a digital audio format developed by Real Networks. It’s designed to handle low-bandwidth networks, which makes it a good Internet streaming media format. Real Audio have the following file extensions, .rm, .ra or .ram. The later versions of RealAudio are compressed using the MPEG-4 AAC and/or the RealAudio lossless format.

Real Video

Real Video is a digital video format developed by Real Networks. It’s designed to handle low-bandwidth networks and optimized for the RTSP protocol, which makes it a good Internet streaming media format. Real Video is often paired with Real Audio to make a Real Media file with the file extension .rm.

RDT

The Real Data Transport protocol is used to transport audio and video over the Internet. It was developed by Real Networks.

RFC

The Request for Comments documents are a gathering of informational documents and standards, for the Internet community. The documents are published by the Internet Society (ISOC) and its technical standards-setting bodies.

RTP

The Real Time Transport Protocol defines a standardized packet format for delivering audio and video over the Internet. It was developed by the IETF.

RTSP

The Real Time Streaming Protocol is a protocol developed by the IETF. The protocol is used for streaming media over an IP network. It allows clients to remotely control the media stream with commands such as play and pause. RTSP is a client/server multimedia presentation control protocol, designed to address the needs for efficient delivery of streamed multimedia over IP networks. It was jointly developed by Real Networks, Netscape Communications and Columbia University and became an IETF standard 1998, RFC 2326. The protocol uses the RTP or the RDT transport protocol which is built on top of the UDP protocol.

Streaming

Allows data to be transferred in a stream of packets over an IP connection. The packet are handled as they arrive.
Streaming media system

A streaming media system is a combination of a player, a server and a codec that must have the same mechanisms. The codec is needed to put the content into an “usable” format, the server is needed to store the coded content and make it available to the Internet and the player is needed to present the content to the user. The “glue” that puts it all together is the transport and control protocols. There are two techniques used for transporting and controlling the media stream, RTSP and HTTP.

TCP

The Transmission Control Protocol is used to create connection between two computers on an IP network.

VoD

Video on Demand is systems that allows users to interactively choose and watch video content over an IP network.

Windows Media

Is a framework for media creation and distribution developed by Microsoft.

WMA

Windows Media Audio is a compressed audio file format developed by Microsoft. A WMA file is often encapsulated in an Advanced System Format (ASF) file. WMA supports digital rights management and specific codec’s for lossless, multi-channel surround sound and voice encoding. It has the file extension .wma. or .asf.

WMV

Windows Media Video is a compressed video file format developed by Microsoft. It’s based on the MPEG-4 standard and often combined with a WMA stream. The WMV stream is often found in a container format as an AVI or an ASF.
På svenska

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extra-ordinära omständigheter uppstår.
Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.
Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsmann i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.
För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

In English

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.
The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.
According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.
For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© [Linus Widborg]