Final thesis

Design and implementation of a prototype home media system for an IP-based set-top box

by

Robert Bo Johansson

LITH-IDA/DS-EX--04/058--SE

2004-06-01
Final thesis

Design and implementation of a prototype home media system for an IP-based set-top box

by

Robert Bo Johansson

LITH-IDA/DS-EX--04/058--SE

2004-06-01

1.0

Supervisor: Henrik Carlsson Kreatel Communications AB
Examiner: Juha Takkinen Department of Computer and Information Science
Titel
Design och implementation av en mediasystemprototyp för en IP-baserad set-top-box i hemmet

Författare
Robert Bo Johansson

Sammanfattning
Abstract
This thesis covers design and implementation of a media system solution for home networks with personal computers and a set-top box.

In a home there are effectively two independent media systems with the same purpose: the personal computer and the digital set-top box, with the purpose of delivering digital media in form of audio and video to the consumer.

The goal of the thesis work was to implement a solution that bridges the gap between the two systems, so that the user, from the set-top box, can play back media that is actually stored on one or several personal computers.

Our solution is based on UPnP technology, which is used for service discovery and control. The choice of UPnP is motivated by an evaluation of discovery protocols, which concludes that UPnP is the most suitable technology in this particular system. Also, an evaluation of suitable transport protocols was done. Here, HTTP was used.

For the personal computer, a media server and a graphical user interface for configuring the media server were created. For the set-top box, a media client, and a graphical user interface for browsing the content of the media server, were created. In conclusion, the creation of the prototype was successful and the set-top box was able to playback media that had been shared by the PC on the network.

Nyckelord
Keyword
home media system, set-top box, UPnP, service discovery, digital media, streaming
Abstract

This thesis covers design and implementation of a media system solution for home networks with personal computers and a set-top box.

In a home there are effectively two independent media systems with the same purpose: the personal computer and the digital set-top box, with the purpose of delivering digital media in form of audio and video to the consumer.

The goal of the thesis work was to implement a solution that bridges the gap between the two systems, so that the user, from the set-top box, can play back media that is actually stored on one or several personal computers.

Our solution is based on UPnP technology, which is used for service discovery and control. The choice of UPnP is motivated by an evaluation of discovery protocols, which concludes that UPnP is the most suitable technology in this particular system. Also, an evaluation of suitable transport protocols was done. Here, HTTP was used.

For the personal computer, a media server and a graphical user interface for configuring the media server were created. For the set-top box, a media client, and a graphical user interface for browsing the content of the media server, were created. In conclusion, the creation of the prototype was successful and the set-top box was able to playback media that had been shared by the PC on the network.

Keywords

home media system, set-top box, UPnP, service discovery, digital media, streaming
Design and implementation of a prototype home media system
for an IP-based set-top box

Robert Bo Johansson
Acknowledgements

I wish to thank Kreatel Communications AB for giving me the opportunity to carry out this thesis. In particular, a great thanks goes to my supervisor Henrik Carlsson, for giving me free hands when working on the solution, and providing me with helpful directions when necessary. Not to forget is the helpful information provided by Kreatel developers Dag, Daniel and Mankan. Professional skill are the words that come to mind.

I am also very thankful for the extensive and valuable feedback on the report that my examiner Juha Takkinen has given me. And thanks, Henrik Leion, for the attentive scrutiny of the system’s design, which you performed as opponent.

Finally, I would like to thank Per Andersson and David Haglund; having you fellows around definitely made the thesis term unforgettably enjoyable!

In a sunny Linköping, June 6th 2004

Robert Bo Johansson
Design and implementation of a prototype home media system
for an IP-based set-top box

Robert Bo Johansson
1 Introduction .................................................. 1
  1.1 Background ................................................. 1
  1.2 Problem ...................................................... 1
  1.3 Example scenario for a home media system .............. 2
  1.4 Expected outcome and delimitation of thesis work .... 3
  1.5 Method of thesis work .................................... 4
    1.5.1 Philosophy ........................................... 4
    1.5.2 Work process ........................................ 4
  1.6 Document overview ....................................... 5
  1.7 Reading instructions ..................................... 5
  1.8 Glossary ................................................... 5

2 Problem analysis ............................................. 7
  2.1 Vision of operation ...................................... 7
  2.2 Use cases .................................................. 7
    2.2.1 Select media files to share ......................... 8
    2.2.2 Play back media files ............................... 8
  2.3 Needed modules .......................................... 8
  2.4 Network topology ....................................... 10
  2.5 Preliminary requirements analysis ....................... 10
  2.6 Other media systems .................................... 12

3 System requirements ......................................... 13
  3.1 Base requirement level .................................. 13
  3.2 Extra requirement level ................................ 13
  3.3 Home Media Client ....................................... 14
  3.4 Home Media Server ...................................... 14
  3.5 Home Media Browser .................................... 15
  3.6 Audio formats ............................................ 16
  3.7 Operating systems ...................................... 16

4 Choice of protocols ......................................... 17
  4.1 Introduction to discovery protocol ..................... 17
  4.2 Discovery protocol selection criteria .................. 18
  4.3 Comparison of discovery protocols ..................... 19
    4.3.1 Jini .................................................. 20
    4.3.2 Salutation .......................................... 22
    4.3.3 Service Location Protocol .......................... 23
    4.3.4 Universal Plug and Play ............................ 24
  4.4 Discussion of discovery protocols ...................... 25
    4.4.1 Salutation .......................................... 25
    4.4.2 Jini versus SLP and UPnP .......................... 25
    4.4.3 Software reliability ................................. 27
4.4.4 Interoperability .......................... 27
4.5 Discovery protocol conclusion .................. 27
4.6 Choice of transport protocol .................. 28
4.7 New TCP- or UDP-based protocol ............... 28
4.8 RTP ............................................. 28
4.9 HTTP ............................................. 29
4.10 Transport protocol conclusion .................. 29

5 Home Media System architecture ............... 31
  5.1 UPnP details ..................................... 31
  5.2 UPnP A/V Architecture .......................... 32
  5.3 Mapping of UPnP A/V to the Home Media System ........ 34

6 Home Media System prototype ................... 37
  6.1 The client C++ classes and layers ............... 37
  6.2 Design patterns used ............................ 40
    6.2.1 Facade .................................... 40
    6.2.2 Composite .................................. 40
    6.2.3 Chain of responsibility ....................... 40
    6.2.4 Singleton .................................. 41
    6.2.5 Delegate .................................... 41
  6.3 Client-side software development ............... 42
  6.4 A note on server-side software development .......... 42
    6.4.1 Implementation note ........................ 43

7 User interfaces .................................. 45
  7.1 Home Media Browser Interface ............... 45
    7.1.1 Interaction analysis ........................ 45
    7.1.2 Visual appearance ........................... 46
  7.2 Home Media Manager Interface ................ 50

8 Evaluation ...................................... 53

9 Conclusion ...................................... 55
  9.1 Future work .................................... 55

10 Personal reflections ............................. 59
  10.1 Software reuse and time ........................ 59
  10.2 Significance of design patterns ............... 59
  10.3 Development model ............................ 59

References ........................................ 61

A Glossary ........................................ 67

B Functional requirements .......................... 69
C Non-functional requirements .................. 77
D Propagation of browsing invocation ............. 79
1 Introduction

1.1 Background

The amount of digitally distributed and digitally stored media in the form of pictures, audio and video is rapidly increasing (DHWG 2003). It is not uncommon for people to have their own private collections of media files stored on their home computers. It is also increasingly common to have a home network to enable the computers to share files, printers and other resources. Media files are typically viewed directly on a computer or on a nearby TV set by using the TV out connection of the computer’s graphics card and the line out of the sound card.

In the transition from analog to digital video broadcasting, set-top boxes based on IP/Ethernet standards are gradually being introduced and deployed in people’s homes (DHWG 2003). Consequently, in a home there can be two or more separate and non-interoperable systems with the same purpose: Consuming digital media. If home computers and set-top boxes somehow could be brought together, all the digital media available in a home could be accessed in a unified and convenient way. This report explores how this can be achieved.

This thesis work has been carried out at Kreatel Communications AB, a company that provides software platforms and set-top box solutions for delivery of TV-centric services in broadband networks.

The term set-top box, often abbreviated STB, can be described as a digital device connected to both a TV-set and a broadband network. The task of the STB is to present the services of the network to the user. Typical fields of application of a STB include watching digital television broadcasts, order and view video on demand, surf the web, and utilize other internet-based services.

1.2 Problem

As suggested in above, one simple way to bring a home computer together with the TV set, is to simply connect the TV out and line out connections of the computer’s graphics card to the TV-set. One problem with that approach is that only one computer at a time can be connected to the TV set and the sound system, unless some kind of audio/video switch is used. Audio/video switches, however, are uncommon in home environments. In addition to that, far from all computers have a TV out connection. Also, if cables carrying analog video and audio signals are used to connect the PC with the STB, quality decreases with increasing cable length. If the PC is in another room, controlling the player application via a remote control may not be possible.
Another way is to burn a CD or DVD with the desired files and play them back on a DVD-player connected to the TV. However, over time, storage, labelling and physical handling of the discs can become a frustration.

In a home with an IP-based STB in place new possibilities emerge. The STB, having an RJ-45 jack (Twisted Pair cable, Category 5) and an existing TCP/IP stack implementation, could potentially act as a client to home PCs which share their media files: Virtually all new PCs are equipped with a network card. In other words, it would be possible to view and listen to content served from one or more PCs on the TV set.

The benefits of the described idea are multiple:

- Video and audio quality does not degrade with the distance from the media file server.
- Typically, the STB is connected to the best TV set and sound system in the home, allowing higher quality playback than possible from a typical home computer. It is also probable that the TV set is located in a comfortable environment suited for viewing.
- The STB is noiseless.
- Multiple PCs can act as servers.
- Existing network equipment can be used, such as switches and cables.
- No rewiring is necessary to use different sources.
- We will get a dedicated interface for accessing and viewing media files.
- The same remote control can be used for controlling channels and served media content.

### 1.3 Example scenario for a home media system

To give a picture of what consumer-needs a home media system can satisfy, we will look at an example scenario.

Meet Martina. She has just found out about the endless possibilities of all the new digital consumer devices available. Being a hobby photographer she bought herself a brand new digital camera. Most of all she likes to take pictures of her friends or the scenic countryside. After a holiday trip around Europe, Martina invites her friends for dinner. They eat, drink, have some good laughs, and eventually get settled in Martina’s living room to chill out.

Martina has already downloaded all her holiday pictures to her PC. All she has to do is to turn on the TV set, go to the media browser and slide through the pictures with the remote control while telling her friends about her adventures.
As the party moves on they all feel like grooving to some music. Martina has a big collection of MP3 files on her PC. From the media browser she can directly select among the music files on the PC and play her and her friends’ favourite songs.

1.4 Expected outcome and delimitation of thesis work

To bring the scenario presented above to reality, the following high-level goals are set for this project:

• Create a way of configuring what media files to share from a PC.
• Make a PC somehow share the selected content on a network.
• Extend the functionality of the STB to be able to locate and retrieve shared content on PCs.
• Create a graphical user interface on the STB which allows the user to browse and play back the content.

To make the problem more manageable, browsing pictures will not be considered at all for the prototype being created. The focus will be to make browsing and playing back music work.
1.5 Method of thesis work

We will now look at the philosophy and the work process used.

1.5.1 Philosophy

The philosophy that will guide the decisions during the project is based on two principles:

- Simple solutions.
  When confronted with alternative solutions, prefer those that apply more specifically to the problem at hand, rather than choosing overly general and extensible solutions. The system will be easier to survey and problems easier to track down.

- Software reuse.
  Reuse existent reliable software where possible. By not reinventing the wheel, time can be saved. Apart from the already existing software for the STB platform, open-source software will be regarded in particular.

1.5.2 Work process

The work process can be summarized in the following main stages:

1. Identify the problem and subproblems.
2. Evaluate protocols
3. Choose available and reusable software components
4. Design necessary new software
5. Implement the system
6. Evaluate the outcome

Steps 4 and 5 will be carried out in several iterations after first having done one initial comprehensive design stage. By doing so, the final goal will be clear from the beginning. The biggest obstacles can be foreseen and avoided, while details can be worked out as potential problems arise. Also, most of the time there will be a working version of the software, providing feedback on how the system performs. The idea of frequent small releases is taken straight from the extreme programming work paradigm (Beck 2000). However, the method used is far from “real” extreme programming. The most notable differences from stringent extreme programming are that no pair programming is carried out and no automated tests are written. Also, the design and code is being commented and documented.
1.6 Document overview

The first few chapters of this report cover the pre-study phase of the project. The problem is analyzed in chapter 2, requirements are set in chapter 3 and different potential solutions are compared in chapter 4.

With that done, the report gradually goes into more detail. First the general architectural solution is described in chapter 5, next chapter 6 presents and motivates the actual software design of the system. Finally, the planned user interface is developed in chapter 7.

The last part of the report is mostly concerned with looking back at what has been achieved. In chapter 8 the system is evaluated, after which chapter 9 draws the conclusions and presents ideas of what future work could be done. Chapter 10 then rounds off with some personal thoughts on the work done.

1.7 Reading instructions

In order to just get a quick overview on the topic it is sufficient to have read this introductory chapter. Optionally, reading section 2.3, ”Needed modules” will provide a more visual overview of the system.

For a more deep understanding of the problems involved, without digging too deep into technical details, reading chapter 2, ”Problem analysis” is recommended.

More in-depth coverage of the software design can be found in chapter 6, ”Home Media System prototype”.

1.8 Glossary

This document contains several acronyms. It is highly recommended to scan through appendix A, which lists the ones used in the document.
Chapter 1: Introduction

Design and implementation of a prototype home media system for an IP-based set-top box

Robert Bo Johansson
2 Problem analysis

In the strive for the best acceptable solution possible, the problem is here analyzed from different points of view.

2.1 Vision of operation

Our vision of operation is as follows: When Martina (read more about her in section 1.3, “Example scenario for a home media system”) starts the media browser on the STB, it scans the network for media servers. All available media servers respond making the STB aware of them. Next, the STB queries each media server for information about what resources (i.e. folders and files) they share. The retrieved information is transformed and shown in an informative way in the media browser user interface.

When Martina chooses to view or playback a specific resource (e.g. an MP3 file), a new connection is set up from the STB to the media server sharing that resource. The connection will feed the media file data to the media player on the STB.

2.2 Use cases

To get a start in understanding what problems await, we begin with identifying the high level use cases. As can be seen in figure 1 the two main activities of the user are selecting what media files to share and playing back media files.

![Figure 1: High level use cases](image)

*Figure 1: High level use cases*
2.2.1 Select media files to share

On the PC the user should be able to select which folders to make available to the STB. Media files located in the selected folders will be possible to play back on the STB.

2.2.2 Play back media files

On the STB the user should be able to play back available media files.

2.3 Needed modules

The STB at hand has several internal services (Kreatel 2003). These include, among others, media player service, information service and application registration service. Whenever possible and appropriate, the available services should be reused. The modules communicate with a variant of XML-based remote procedure calls (RPCs) (Tanenbaum 2003). This allows loose coupling between the modules. A high level overview of the existing STB architecture is shown in figure 2 (Kreatel 2003). The applications are at the top and communicate with the service layer modules via the RPC C++ interface. The web browser, the TV- and the video-on-demand- (VOD-) viewers are example of applications. Below the service layer there is a hardware abstraction layer (HAL), and beneath that the Linux operating system and the STB hardware.

![Figure 2: STB architecture overview](image)
The existing architecture makes it appropriate to add the new functionality in form of modules. An extended and simplified version of figure 2 is shown in figure 3. The bright boxes represent the modules that need to be added to create the Home Media System. On the PC there is the Home Media Manager, which is an application that allows the user to configure the Home Media Server. The Home Media Server in turn is an application without a graphical user interface, that makes media files available to Home Media Clients. The Home Media Browser is the graphical user interface built upon the Home Media Client and allows a user to select and playback the media files.

![Diagram of existing and planned modules](image-url)

**Figure 3: Existing and planned modules**

Describing the process of defining, designing and implementing the four modules shown in figure 3 is in essence what the rest of this document will be about.
2.4 Network topology

The above description assumes there is a functioning network available. One question that soon arises is: What does that network look like?

In a common deployment of Kreatel STBs, a fast network link goes from the operator to each home installation, where a switch enables multiple devices to utilize the link. There is also a DHCP server (Tanenbaum 2003) on the network, typically located at the operator, which assigns IP addresses to network hosts.

![Network topology diagram]

In another variant of network topology, the link contains two or three virtual networks. One of the virtual networks is dedicated to the STB, one to IP telephony and another for all other general purpose data traffic. In such a setup the switch has special capabilities to be able to separate the virtual networks and route them to different connector jacks. There are other possible setups as well, but the only one considered in this thesis is the one shown in figure 4. This setup is such a simple case, that it can form a basis to more complex setups.

The assumption is made that the STB and the PC are located on the same subnet (i.e. abc.def.ghi.0/255.255.255.0), that they use the same DHCP server and that both have acquired an IP address.

2.5 Preliminary requirements analysis

A preliminary analysis of the requirements for the Home Media System gives us the following issues:
Dynamic network population
It is in the nature of a network that devices are added and removed. In the case of wireless LANs it is more a rule than an exception that devices come and go, rather than stay for a long period of time. Therefore, the system should handle addition and removal of Home Media Servers gracefully and definitely not fail upon such events.

Device and service discovery
Even in the case that devices are permanently connected to the network, at some point they are stopped and restarted. When a Home Media Server device comes up on the network, Home Media Client devices must be able to find them. Therefore, the system must contain some kind of device and service discovery functionality. Device discovery protocols are covered in chapter 4, "Choice of protocols".

Client and server quantitative relationships
In the simplest case, there is one Home Media Server and one STB on a network, that is, a 1:1 relationship. With more Home Media Servers on the network there will be an n:1 relationship. In the most general case, there is more than one STB in the home network and the relationship becomes n:m. Both n and m are relatively small numbers. Certainly n + m <= 254 since that is the maximum number of hosts in an IPv4 subnet (Tanenbaum 2003). The worst-case scenario from one Home Media Server’s bandwidth point of view is when all STBs simultaneously fetch content from it. However, in a typical home the number of STBs is very unlikely to be greater than 2 or 3. An example calculation gives the following: A 120 minute (= 7200 seconds) movie with a size of about 1300 MB (approximately 13000 Mbit) requires a bandwidth of approximately 13000 Mbit / 7200 seconds = 1.8 Mbit/s. Three such streams only sum up to 3 * 1.8 Mbit/s = 5.4 Mbit/s. This should not be a concern on today’s or tomorrow’s networks.

Limited resources
The Home Media Client and Home Media Browser must have a limited memory footprint and be efficient because of the limited memory and processing power available.

The Home Media Server and Home Media Manager do not have these constraints on a regular PC. However, if the Home Media Server and Home Media Manager were able run on other popular devices such as PDAs it would increase the value of the Home Media System for the end-user. Even though PDAs currently have limited resources, they are quickly getting more powerful. The memory-capacity and processing-power issues should be kept in mind when designing the server-side software.

Portability
The Home Media Client and Home Media Browser do not have a high need for portability since they are only going to be run on the
STB, which is based on Linux.

In contrast, the Home Media Server and Home Media Manager software should be portable since there is a future possibility that they will be ported to other platforms than a stationary PC.

2.6 Other media systems

There are some consumer products that from a user's perspective provide more or less the functionality that we strive for and that was described in section 1.3, “Example scenario for a home media system”. There is also a number of projects that do not deal with device discovery but nonetheless provide audio and video streaming frameworks.

For example, a company called Oregan develops middleware solutions for digital home products (Oregan 2004). Their software uses Universal Plug and Play (UPnP) for communication between different networked devices (Microsoft 2002). The server software running on a PC can be configured to make media files available to other devices. Several video, audio and image formats are supported (Oregan 2004).

Another product, "MediaMVP", from manufacturer Hauppauge consists of a minimal embedded PowerPC-based system running Linux (Hauppauge 2004). It has a network connection and is able to play video and audio and also show photos. The user interface is created dynamically in HTML by the server application installed on the PC when the user initiates an action, and is then transferred to and rendered on the box (Hauppauge 2004).

There is also an open-source project called VideoLAN (VideoLan 2004). It has both a framework and applications for playing back audio and video over a network. It does not, however, provide any service discovery. Parts of the system could potentially be reused for stream coding/decoding and data transport (VideoLAN 2004). While the two previously mentioned products are available as combined hardware and software products, VideoLAN is a software-only product.

RealNetworks has a software-only client/server media streaming solution called Helix. It has support for coding/decoding of several audio and video formats and data transport (Helix 2004).

The strength of a set-top box with an integrated home media system such as ours, is the combination of broadcast media, on-demand media and personal content available through one single point of access.
3 System requirements

Now that the problem is known in more detail we can set up requirements for the system.

The actual requirements for our system are quite lengthy to read, so the full listings are therefore found in appendix B, ”Functional requirements”, and appendix C, ”Non-functional requirements”. In this chapter, only the functional base requirements are listed. Now, let us first look at how the requirements are defined.

The following example shows the notation with which each requirement is specified.

<table>
<thead>
<tr>
<th>Level</th>
<th>B or E level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>An informative description of what the requirement means.</td>
</tr>
<tr>
<td>Motivation</td>
<td>Explanation of why this requirement is stated.</td>
</tr>
<tr>
<td>Influences</td>
<td>References to other requirements which are influenced by the current requirement.</td>
</tr>
</tbody>
</table>

3.1 Base requirement level

Requirements which have been classified at the base requirement level must be met by the system. Solutions needed to meet these requirements are designed for before the implementation starts. Base requirements are fulfilled and implemented before work on the extra requirements start. In the requirements specification appendices this level will be abbreviated as B.

3.2 Extra requirement level

Requirements which have been classified at the extra requirement level are merely desired ones. Solutions for requirements at the extra level are not designed at beforehand, but on an ”as needed” base just before implementation. In the requirements specification appendices this level will be abbreviated as E.

Extra requirements may be implemented if they can be done without breaking the schedule and interfering with the realization of the base requirements.
### 3.3 Home Media Client

These are the base requirements for the Home Media Client.

**F 1.1 Single Home Media Server discovery**

<table>
<thead>
<tr>
<th>Level</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>If at least one Home Media Server is connected to the network and an Home Media Client starts, the Home Media Client discovers at least one of the Home Media Servers and finds out its services.</td>
</tr>
</tbody>
</table>
| **Motivation** | For zero configuration need from the user’s point of view.  
The system should be dynamic and allow servers to be removed and inserted into the network without any need for reconfiguration in the Home Media Client. |
| **Influences** | F 1.3, F 1.3, F 1.4, F 3.1 |

**F 1.2 Single Home Media Server support**

<table>
<thead>
<tr>
<th>Level</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>One device on the network can act as Home Media Server.</td>
</tr>
<tr>
<td><strong>Motivation</strong></td>
<td>At least one device (e.g. PC) must be present and act as Home Media Server for the system to work.</td>
</tr>
<tr>
<td><strong>Influences</strong></td>
<td></td>
</tr>
</tbody>
</table>

### 3.4 Home Media Server

These are the base requirements of the Home Media Server.

**F 2.1 Listing of available folders and files**

<table>
<thead>
<tr>
<th>Level</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Upon request from a Home Media Client the server will respond with a set of available folders and files.</td>
</tr>
<tr>
<td><strong>Motivation</strong></td>
<td>For the user to be able to browse through the list of files available via the Home Media Browser, the set of files must be communicated to the Home Media Client.</td>
</tr>
<tr>
<td><strong>Influences</strong></td>
<td></td>
</tr>
</tbody>
</table>
F 2.2  **Single stream of MP3 audio file**

<table>
<thead>
<tr>
<th>Level</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>The server is able to stream one MP3 file to one STB at a time.</td>
</tr>
<tr>
<td><strong>Motivation</strong></td>
<td>Minimal prototype requirement.</td>
</tr>
</tbody>
</table>

F 2.3  **Basic audio playback control**

<table>
<thead>
<tr>
<th>Level</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>The server can receive and effectuate start, stop and pause audio playback commands.</td>
</tr>
<tr>
<td><strong>Motivation</strong></td>
<td>In analogy with well-known devices such as VCRs or CD-players. Expectation from the user.</td>
</tr>
<tr>
<td><strong>Influences</strong></td>
<td>F 2.4</td>
</tr>
</tbody>
</table>

3.5  **Home Media Browser**

The requirements of the Home Media Browser state what the user can do in the user interface.

F 3.1  **Audio file selection**

<table>
<thead>
<tr>
<th>Level</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Via the STB the user can select from the set of available audio files, which one to control. (See F 3.2)</td>
</tr>
<tr>
<td><strong>Motivation</strong></td>
<td>Expectation from the user.</td>
</tr>
</tbody>
</table>

F 3.2  **Basic audio playback control**

<table>
<thead>
<tr>
<th>Level</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>From the Home Media Browser it is possible to start, stop and pause audio playback.</td>
</tr>
<tr>
<td><strong>Motivation</strong></td>
<td>In analogy with well known devices such as VCRs or CD-players. Expectation from the user.</td>
</tr>
</tbody>
</table>
3.6 Audio formats

There is only one base requirement for audio formats.

F 4.1 Support for MP3 files

<table>
<thead>
<tr>
<th>Level</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Playback of files in MP3 format is possible.</td>
</tr>
</tbody>
</table>

3.7 Operating systems

Different parts of the Home Media System run on different operating systems. The following requirements state this.

F 5.1 Home Media Client and Home Media Browser host OS

<table>
<thead>
<tr>
<th>Level</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>The Home Media Client and Home Media Browser run on Linux.</td>
</tr>
<tr>
<td><strong>Motivation</strong></td>
<td>The client and browser are located on the STB which runs Linux.</td>
</tr>
<tr>
<td><strong>Influences</strong></td>
<td>F 1.1, F 1.3, F 1.3, F 1.4</td>
</tr>
</tbody>
</table>

F 5.2 Home Media Server and Home Media Manager host OS

<table>
<thead>
<tr>
<th>Level</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>The Home Media Server and Home Media Manager run on Windows 2000/XP</td>
</tr>
<tr>
<td><strong>Motivation</strong></td>
<td>Most home PCs run Windows.</td>
</tr>
<tr>
<td><strong>Influences</strong></td>
<td></td>
</tr>
</tbody>
</table>
4 Choice of protocols

In this chapter we will decide which protocols to use for communication between the Home Media Client and the Home Media Server. We start off with deciding upon a discovery protocol and conclude with the choice of transport protocol.

4.1 Introduction to discovery protocol

When Home Media System compatible devices, be it STBs, PCs, PDAs or any other, are plugged into the home network they somehow have to discover each other’s presence and capabilities. This way the user does not have to manually configure anything. For an end-user device this self configuration ability is quite important since users in general do not want to read manuals (Faulkner 2000). They expect things to “just work” right out of the box.

We start this chapter with describing the four candidate protocols for device discovery. They are compared against each other and one of them is chosen for usage in the Home Media System.

However, first a few related service-announcement technologies, and also the SMB protocol, are discussed and motivations are presented of why they do not qualify as candidates for the final comparisons.

- **UDDI and Web services**
  UDDI stands for Universal Description, Discovery and Integration and is a specification that describes how web service based business applications can find each other (UDDI 2002). The quantitative relationship between services and clients is \( n:m \), where both \( n \) and \( m \) can be large, several thousands or more. UDDI is very generic. However, the server-side software in a UDDI/Web services system is intended to run on medium- to large-scale web servers, not on home PCs or PDAs.

- **Session Announcement Protocol**
  The session announcement protocol, often abbreviated as SAP, is formally identified as RFC 2974 (RFC 2974). The services announced by SAP are primarily multicast multimedia conferences. In other words, the service provided is not an on-demand service requested by the user. Rather, the user can search for ongoing or announced conferences and then optionally join one of them. A related technology, Internet Media Guide, proposed by the Internet Engineering Task Force, makes use of SAP for announcing media broadcasts (IETF 2003). This is quite different from the Home Media System case in which it is the user who initiates any activity.

- **DVB-SI**
  The Digital Video Broadcasting Project has a standard called DVB-SI
where SI stands for Service Information and is transmitted together with the digitally broadcast video itself (ETSI 2003; onTV 2003). The transmitted data contains information about current program title, duration and classification. This is a one-way type of communication that does not require (or allow) any requests from the receiving units. Because of this DVB-SI is not suitable for the Home Media System case, which is more of an on-demand type of system.

- **DVB-IPI**
  IPI stands for Internet Protocol Infrastructure. Complementary to the DVB-SI, DVB has defined standards for IP-based communications in general, which apply to home networks in particular. Unfortunately, lack of openly available information at the time of writing (May 2004) make this candidate fall away. Also, the DVB-IPI is somewhat biased towards video services, whereas the Home Media System is more general and could include image services.

- **Peer-to-peer technologies**
  File-sharing systems of peer-to-peer type have grown common. In such a system the file-sharing application acts both as a server and a client. Apart from the debate of the legality of file sharing there is a technical obstacle. The memory of the STB is limited and it is erased at each reboot. Since the STB can not store a lot of media files it is not likely to be able to join file-sharing peer-to-peer networks, which often require the peers to share a minimum (and often large) amount of data to join. Also, if the STB succeeds to join a network and the user finds an interesting file, it is not guaranteed that the file will be available at a future point in time.

- **Server Message Block protocol**
  The Server Message Block protocol, or SMB for short, is a file- and printer-sharing protocol used in Microsoft Windows 95 and later (Microsoft 2004). There is a UNIX implementation as well called Samba (Samba 2004). The SMB protocol is, however, proprietary and Samba was created by reverse-engineering SMB. Using it would be against the spirit of the open standards philosophy stated in section 1.5, ”Method of thesis work”. The aim of SMB is not to be a generic service discovery protocol. It focuses on file and printer sharing. Therefore, features such as event notification and callbacks, which may be necessary in our project, are not among its primary objectives.

### 4.2 Discovery protocol selection criteria

In the process of selecting the most appropriate discovery protocol a number of criteria were set up. For each protocol and for each criterion, positive and negative aspects were considered.

The selected discovery method should meet the following criteria:
- Be well-defined and documented.
- Be expected to become a de facto standard. The reason is to increase the chance of being compatible with devices from other parties.
- Have open specifications.
- Be easy to implement on the client side on the STB platform.
- Be portable, so as to allow the server program to run on different platforms.
- Be interoperable with implementations on different platforms.
- Be suitable for its deployment context (IP/Ethernet-based home network).
- Be able to work in an ad-hoc environment without a central server.
- Have a low-cost licence, so as to reduce the cost of mass-deployment of STBs.

When choosing the most suitable protocol the home pages of each candidate protocol were consulted. The comprehensive documentation available was studied. Also the web was searched using a search engine (Google) to find complementary material either contradicting or strengthening the statements and the official propaganda of the candidate home pages.

No implementation tests were made for the different candidates and possible combinations of them. Only when a final protocol was chosen tests were made to ensure that communication worked.

### 4.3 Comparison of discovery protocols

The first candidate is Jini. Jini is a Java-based technology developed and promoted by Sun Microsystems (SUN 2003c).

The second candidate Salutation. Behind Salutation stands the Salutation Consortium, a group of large companies in the software and hardware field (Salutation 2002).

The third candidate is Service Location Protocol (SLP for short) which is entirely based on RFCs.

The fourth and last candidate is Universal Plug and Play, or UPnP for short. The UPnP Forum is the organization behind UPnP, among whose members the most prominent are Microsoft, Intel and IBM (UPnP 2004).

What these protocols all have in common is that they provide ways of locating and describing services provided by devices on a network. We now examine the details of each candidate protocol.
### 4.3.1 Jini

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification</td>
<td>Usage and function of Jini is very well documented in several official documents (SUN 2004a).</td>
</tr>
<tr>
<td>Adoption</td>
<td>While Jini was first developed and launched as a service discovery protocol for small embedded devices, it has had its greatest success in the business framework area (Murphy 1999; SUN 2003c).</td>
</tr>
<tr>
<td>Openness</td>
<td>Specifications are open and available on the Jini website. Also, a lot of examples are available, which ease understanding. However, creating non-commercial implementations of Jini from scratch is prohibited by Sun Microsystems (Edwards 1999; SUN 2003b).</td>
</tr>
<tr>
<td>Implementation</td>
<td>First of all, Jini requires a JRE. Apart from SUN's implementation, a company called PsiNaptic offers a product called CMatos which is a C implementation of the Jini framework (PsiNaptic 2004). It has the advantage of having a very small footprint, about 60 KB. It is also expected to run faster than a Java counterpart. There are several open-source JRE implementations, of which Kaffe (Kaffe 2004) and Jikes (Jikes 2004) are the most prominent. None of these, however, provide complete implementations of the Java API and cannot be guaranteed to work with Jini libraries. Yet another possibility is to use Jini and compile using the GNU Java Compiler (GCJ 2004), which can produce machine-native code, improving execution speed. There are no open-source implementations of Jini since making new non-commercial implementations from scratch is prohibited by Sun (SUN 2003b).</td>
</tr>
<tr>
<td>Portability</td>
<td>Jini is strongly tied to the Java language. Provided that there is a JRE for the host platform, the server can be used directly on the host platform without any need to rewrite source code.</td>
</tr>
</tbody>
</table>

*Table 1: Jini aspects*
<table>
<thead>
<tr>
<th>Aspect</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>License</td>
<td>Sun JREs can be bundled and distributed with no license fees, provided that the bundled software does not replace any functionality of the JRE (SUN 2003a). Commercial use of Jini requires special license from Sun (SUN 2003b). CMatos also has a licensing cost associated with commercial usage.</td>
</tr>
<tr>
<td>Other</td>
<td>Java development typically involves fewer memory-related bugs. This saves implementation time.</td>
</tr>
</tbody>
</table>

*Table 1: Jini aspects*
### 4.3.2 Salutation

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification</td>
<td>Salutation Architecture Specification 2.0c was released in June 1999 and is thorough.</td>
</tr>
<tr>
<td>Adoption</td>
<td>The adoption of Salutation is unclear. While the Salutation web site (Salutation 2002) lists about 30 IT companies that have contributed to the development of the protocol, it is hard to find any evidence of widespread usage by searching the companies’ web sites. There are, however, some known implementations for fax machines and Lotus Notes (Bettstetter &amp; Renner 2000).</td>
</tr>
<tr>
<td>Openness</td>
<td>Specifications are available on the Salutation homepage (Salutation 2002).</td>
</tr>
<tr>
<td>Implementation</td>
<td>There is an incomplete reference implementation called Salutation-Lite available.</td>
</tr>
<tr>
<td>Portability</td>
<td>The specification is not tied to any platform. The reference implementation is written in ANSI C, which with good probability makes it portable to different platforms.</td>
</tr>
<tr>
<td>License</td>
<td>No royalties or license fees are required.</td>
</tr>
</tbody>
</table>
| Other       | Salutation has a very broad area in which it can be applied. It is not tied to any language or transport protocol. Compared to the other candidates, it is more based on academic research and has its strengths in coordination rather than in device discovery (CSWL 2002).  
The Salutation web site shows an increasing lack of maintenance over time and there are no signs of update since around may 2003, which might indicate that the Salutation consortium does not put effort into the project anymore.  
The Salutation Consortium staff has not answered to our e-mail requests regarding Salutation’s status and future plans. |

*Table 2: Salutation aspects*
### 4.3.3 Service Location Protocol

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification</td>
<td>SLP is defined in RFC 2608, RFC 2609, RFC 2610, and RFC 2614 (IETF 2004).</td>
</tr>
<tr>
<td>Adoption</td>
<td>SLP seems to have gained acceptance in the Linux domain, mainly as a protocol used by various service location daemons for printers. Axis and Sun Microsystems provide implementations (OpenSLP 2003).</td>
</tr>
<tr>
<td>Openness</td>
<td>RFCs are accessible without restrictions.</td>
</tr>
<tr>
<td>Implementation</td>
<td>Full open-source implementations (OpenSLP) exist for C under Linux and for Java.</td>
</tr>
<tr>
<td>Portability</td>
<td>The Java implementation is portable to any platform having a JRE.</td>
</tr>
<tr>
<td>License</td>
<td>SLP is provided with a BSD-style license</td>
</tr>
<tr>
<td>Other</td>
<td>-</td>
</tr>
</tbody>
</table>

*Table 3: Service Location Protocol aspects*
### 4.3.4 Universal Plug and Play

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification</td>
<td>The UPnP Device Architecture 1.0.1 was released in December 2003 and is thorough.</td>
</tr>
<tr>
<td>Adoption</td>
<td>UPnP is gaining ground and has a lot of corporate backing (Home Networking News 2000). The UPnP web site states that as of December 31, 2003, there are 648 member vendors of which many are influencing actors in the hardware and software field.</td>
</tr>
<tr>
<td>Openness</td>
<td>Specifications are available on the UPnP web site (UPnP 2004). There is also at least one good book available; UPnP Design by Example (Jeronimo &amp; Weast 2003).</td>
</tr>
<tr>
<td>Implementation</td>
<td>There are two major open-source UPnP implementation projects called Libupnp (Libupnp 2004) and CyberLink (CyberLink 2004). Libupnp was created by Intel and is written in C. CyberLink is available both as C/C++ code and Java code. Microsoft provides an SDK for UPnP for Windows platforms (Microsoft 2003).</td>
</tr>
<tr>
<td>Portability</td>
<td>The specification is not tied to any language or platform. However, many implementations are written in C or C++. Using Libupnp together with Minimalist GNU for Windows (MinGW 2004) makes the Libupnp library likely to be reusable on Windows platforms. The Java version of CyberLink is portable to any platform which has a JRE.</td>
</tr>
<tr>
<td>License</td>
<td>Both Libupnp and CyberLink have a BSD-style license.</td>
</tr>
<tr>
<td>Other</td>
<td>UPnP is specifically targeted towards home networking. The UPnP forum has published a standard called the “UPnP Audio Video Architecture” (Microsoft 2002), which defines a system for browsing, searching and controlling playback of media content on a local network. Each UPnP device has a built-in HTTP server (Jeronimo &amp; Weast 2003).</td>
</tr>
</tbody>
</table>

*Table 4: UPnP aspects*
4.4 Discussion of discovery protocols

4.4.1 Salutation

Salutation is the only initiative of the four in which the organization behind it is completely non-profit. Salutation falls short on some critical points. It is not clear if it is being actively developed anymore, which makes it less probable that new devices will be based on the specification. Even though the reference implementation is available with no royalties, it is incomplete, which means that if used, more time will have to be spent implementing the Home Media Server and the Home Media Client. Also, it is not clear whether Salutation is actually used in many devices as of January 2003. Because of the overall uncertainty Salutation was dropped as a candidate.

4.4.2 Jini versus SLP and UPnP

Java technology in general has since its introduction had a wide adoption. As mentioned in section 4.3.1, “Jini”, Java development in general tends to be quite free of memory-related bugs, which saves both implementation and testing time. Jini, however, does not seem to be intended for consumer devices the same way UPnP is. Commercial use of Jini also requires a license fee.

The availability of open-source implementations for both Linux and platform-independent Java makes a UPnP or an SLP solution appealing.

To ease development and maintenance of Home Media Server and Home Media Client, using the same libraries on both sides is desirable. There are four main possibilities to get the same libraries on both server and client sides:

1. Using Sun JRE and Jini
   Both Home Media Client and Home Media Server use Sun JRE and Jini. This is the easiest solution, which is expected to give little hassle. However, as mentioned above, Jini is not as widely adopted among embedded-device vendors, as for example UPnP, and commercial use implies paying a license fee.

2. Using Jini and compile with GCJ
   As mentioned in section 4.3.1, “Jini”, GCJ (GCJ 2004) can compile both Java source code and byte code to native machine instructions on both Linux and Windows platforms, which would mean faster execution. However, Jini is not guaranteed to work when compiled to native machine code with GCJ (GNU 2004). Also, this does not remove the Jini license fee problem.

3. Using Libupnp
   On the STB it is possible to use Libupnp directly since it is a dedicated Linux project. There is a theoretical possibility to port parts of
Libupnp to Windows and compile with MinGW. To do so, however, is expected to take too much coding and testing effort. Also, with the server software written in C++, this solution is a little less portable to other devices than a Java-written server.

4. Using OpenSLP

The OpenSLP project has implementations of SLP for both C under Linux and for Java. Using the native Linux implementation on the STB and the Java implementation on the server is a reasonable idea.

The two most appealing alternatives are 1 and 4 on account of their simplicity. Choosing between these, alternative 4 wins because of the availability of stable open-source implementations.

Looking at alternatives where the same libraries do not necessarily have to be used on both sides, an interesting option turns up. On the STB the native C/C++ Libupnp implementation can be used, while the Java CyberLink implementation can be used for the Home Media Server on the PC. Using this solution, the portability of the Home Media Server can be greatly enhanced and, at the same time, a fast and resource saving Home Media Client can be obtained on the STB.

The final alternative protocols thus are UPnP represented by Libupnp and CyberLink implementations, and SLP, represented by the OpenSLP implementation. The critical point is their respective tool support and purpose. While SLP gives a good general impression, UPnP shines with good tool support, the Intel UPnP tools (Intel 2004), and excellent documentation (Jeronimo & Weast 2003). UPnP also has a superset, the UPnP Audio Video architecture, which specifies media content handling in a network environment (Microsoft 2002). As an added bonus, UPnP by judgement seems to be the protocol that has had the greatest breakthrough and which will continue to be the most common protocol in the embedded device area (DHWG 2003; Oregan 2004).

It is important to note that supporting any of the candidate protocols does not enable the STB to download or stream files from Home Media Servers. None of the protocols deal with data transport, only with service discovery. Choice of transport protocol is dealt with in sections 4.6 through 4.10.
4.4.3 Software reliability

An important question at this point is: How functional and stable is the Libupnp and CyberLink software? Both projects are hosted by SourceForge.net, a large open-source collaboration site (SourceForge 2004). On SourceForge.net, each project has a status level indicating how far it has come. The levels are:

1. Planning
2. Pre-alpha
3. Alpha
4. Beta
5. Production/Stable
6. Mature
7. Inactive

The Libupnp project has status level 6 and the CyberLink project has status level 5, meaning that both are in a good shape for use in a reliable UPnP solution.

4.4.4 Interoperability

Both projects contain a light-bulb control example system showing the capabilities of the respective implementation. The light-bulb example of both projects is an implementation of a standard UPnP device definition called LightSwitch. The system consists of two parts: A UPnP device, represented by a virtual light-bulb that exposes an on-off service and a dimmer service, and a UPnP Control Point, which is used to control the virtual light-bulb.

The CyberLink light-bulb was possible to control from the Libupnp Control Point and the Libupnp light-bulb was possible to control from the CyberLink Control Point. By performing this test we verified that CyberLink and Libupnp are able to work together.

4.5 Discovery protocol conclusion

The conclusion of our investigation is as follows:

- Libupnp, a native Linux implementation of UPnP will be used in the Home Media Client on the STB.
- CyberLink, a Java implementation of UPnP will be used in the Home Media Server on the STB.
4.6 Choice of transport protocol

In the previous sections of this chapter we discussed different discovery protocols and finally decided upon which discovery protocol to use. We will now move on to discussion of transport protocols.

Once a media file has been chosen by the user in the Home Media Browser, it somehow has to be transferred from the Home Media Server to the Home Media Client for playback or viewing. Therefore, a suitable transport protocol has to be chosen.

Some things are important to have in mind when making this choice:

- The STB already has support for fetching content via HTTP as well as RTSP (Kreatel 2003; RTSP 2004).
- Even though only audio playback is the primary goal for the prototype in our current work, it should be extensible, allowing video and image fetching in the future. See appendix B, “Functional requirements” and requirements F 4.1, “Support for MP3 files”, F 3.4, “Viewing of MPEG-2 video file” and F 3.5, “Viewing of image files”.
- All UPnP devices have a built-in HTTP server (Jeronimo & Weast 2003).

4.7 New TCP- or UDP-based protocol

TCP and UDP are the most basic ways of transferring data (Tanenbaum 2003). In practice, using TCP or UDP would mean inventing yet another application layer protocol, or worse, reinventing an existing one because information such as MIME type, message length and search commands needs to be transmitted as well as the data itself. On the plus side, a customized protocol based on TCP or UDP can be highly efficient for its purpose. In addition, RTSP, which is supported by the STB, can use UDP as transport protocol (RTSP 2004).

4.8 RTP

The Realtime Transfer Protocol, or RTP (IETF 2004; RFC 1889) for short, is a UDP-based protocol intended for audio and video data transfer. If used in conjunction with RTSP (RTSP 2004), it allows absolute positioning within the audio or video file played back (Schulzrinne 2003). In other words, seeking forwards and backwards in a file is no problem.

RTP is made for audio and video content. It can not be used to transfer only images. There seems to be no reliable open-source implementation available, which means that the server side would have to be implemented from scratch.
4.9 HTTP

The Hypertext Transfer Protocol, abbreviated HTTP, is a simple and well understood application layer protocol (Tanenbaum 2003). It allows transfer of audio, video and images. This means that the same software on the server side can deliver both audio, video and image content. HTTP also allows manifestation of MIME types.

Searching in audio and video files is not considered by HTTP.

4.10 Transport protocol conclusion

In section 4.5, “Discovery protocol conclusion”, UPnP was chosen. Since each UPnP Device has a built-in HTTP server anyway, using HTTP as the transfer protocol reduces the complexity of the UPnP device. Because of that, and its general-purpose nature, HTTP will be used as transfer protocol for content that is being sent from Home Media Servers to Home Media Clients.
Design and implementation of a prototype home media system for an IP-based set-top box
Robert Bo Johansson
5 Home Media System architecture

Having all the protocol issues out of the way, it is time to see what the Home Media System as a whole will look like. We will identify exactly what components are needed and how they communicate with each other on a high level. However, to ease the understanding of the client design described in chapter 6, a little more detailed description of UPnP will be given.

5.1 UPnP details

A UPnP device holds a self-describing document, which among other things contains a list of the services the device provides. A service is like an interface that exposes a number of methods with a certain signature. There can be both input and output arguments. With UPnP nomenclature, methods are called actions. The description documents and action invocations are all encoded with XML. More specifically, the actions are encoded with the Simple Object Access Protocol protocol or SOAP for short (W3C 2003), which is XML-based. Each service has a Control URL to which action invocations can be sent. This means that all UPnP devices have a mini HTTP server installed.

Each UPnP device has a specific device type value associated with it and each service has a service type value. Device type names and service type names have a standardized format. There are also a couple of type names reserved by the UPnP forum. For example, a media server device has the device type urn:schemas-upnp-org:device:MediaServer:1, and its embedded Content Directory service has the service type urn:schemas-upnp-org:service:ContentDirectory:1. See section 5.2, “UPnP A/V Architecture”, for an explanation of content directories.

When a UPnP device connects to a network it first tries to determine its IP address. It first tries to acquire its address from a DHCP server. If there is no DHCP server the device will try to determine if there is an ad-hoc network, in which case it will look for a free address. The case where no DHCP server is present is, however, not considered since the assumption was made in section 2.4, “Network topology”, that there is one available. Next, the device sends out a notification broadcast telling all other devices of its presence.

The Control Point is a piece of software that operates on the actions exported by UPnP devices. A Control Point is not a UPnP device, though a physical device can contain both UPnP devices and a Control Point. Since a Control Point is not a UPnP device it does not send out any notification message upon initialization. Instead, it sends out a
search broadcast. The broadcast contains the device type that the Control Point is interested in. Only the devices of the requested type will respond.

It is also possible for Control Points to subscribe to events. Events are triggered by state changes in devices. A Control Point may for instance be interested in monitoring a certain variable of a device. The Control Point will in an initial stage subscribe to the device. The device will keep a list of all Control Points that have it subscribed, and as soon as it changes it will send out a notification to all of them.

5.2 UPnP A/V Architecture

When investigating the UPnP protocol it is inevitable to come across the UPnP Audio Video (A/V) Architecture (Microsoft 2002). It describes how media content can be browsed, searched and played back via a local network using UPnP for both device discovery and device control. This is exactly what we are trying to achieve. Therefore, the UPnP A/V architecture will be used in the solution. The following section will give a brief overview of the UPnP A/V Architecture. For full information, see the specification referenced above.

Three logical entities are defined in the UPnP A/V Architecture: The Media Server, the Media Renderer and the Control Point (Microsoft 2002; Intel 2003a; Intel 2003b). These logical entities can be implemented separately in three different physical devices or can be combined in one or two physical devices.

The Media Server, just as the name suggests, makes media content available. The Media Server has a ContentDirectory service which takes care of browse and search actions, processes them and sends back the result. The Content Directory is a hierarchical representation of all the files shared. Elements in the hierarchy tree can either be containers or items. Containers are like folders and can hold other containers or items. Items are the actual files being shared. Each element has a unique id. The root element in the hierarchy is always a container and has element id 0. An example hierarchy is shown in figure 5.
Browsing a Content Directory is very straightforward. All that is needed is to send a browse action with the desired container id as argument to the Control Url (see section 5.1, "UPnP details" and 5.3, "Mapping of UPnP A/V to the Home Media System") of the Content Directory. The Content Directory will respond with a listing of all sub elements and their id numbers. Not only the id numbers but also information, such as name and whether each element is a container or an item, is sent. If the element is an item, an url is given that indicates where the item can be downloaded. Having received the new element id numbers, it is then possible to continue browsing sub containers.

An example of the communication that takes place between the Control Point and the media server when a browse action takes place, is shown in figure 6. For reasons of clarity, we only show a fraction of the XML code that is sent between the parties. What we see is a Home Media Client that wants to find out what is inside the object with id 1 on the server. As we can see from figure 5, there are three items with id numbers 3, 4 and 5 respectively inside the object with id 1, which is a container. The Home Media Server will respond with a list of all items, and an access url for each item. With this access url, the Home Media Client is able to retrieve the file at a later time. In the figure, only the part of the server’s response that concerns the item with id 2 is shown. In the response, we also see that the server has provided the title of the file, “Diggeloo”, which can be used by the graphical user interface when presenting the available file choices to the user.
The third device, the Media Renderer is the device that processes and displays the content. All actions are initiated by the Control Point, which is the user’s way of interacting with the system.

5.3 Mapping of UPnP A/V to the Home Media System

The mapping of the UPnP A/V Architecture to the Home Media System in our work is straightforward. The Home Media Server corresponds to the Media Server, the media player of the STB corresponds to the Media Renderer, and finally, the Home Media Client and the Home Media Browser correspond to the Control Point. It should be noted that the media player is not a real UPnP media renderer device, since it cannot receive commands from an arbitrary Control Point on the network. It is only accessible by the Control Point on the STB. A device in which the Control Point and the Media Renderer have been combined together, as in our system, is called a “Combo Device” (Microsoft 2002).

The communication between a Media Client Combo Device and a Media Server can be divided in four main stages (Microsoft 2002). Figure 7 shows this communication, applied to our system. The stages that will require most attention and client design is this work are stages three and four. How browse- and file transfer facilities are served to the Home Media Browser is described in section 6.2.1, “Facade”.

In the first stage, the Home Media Client sends out a search request on the network. The search specifies that the Home Media Client is interested in Home Media Servers. All available Home Media Servers will respond to the Home Media Client, making it aware of their presence.
In the second stage, the Home Media Client asks each server about more specific information, such as the Control Url of the server. When the Home Media Client has obtained the Control Url of a Home Media Server, it is able to go to the next stage in the communication.

In the third stage, the Home Media Client is able to invoke actions to the Control Url of the Home Media Server. Actions typically are browse requests and searches.

In the fourth and final stage, the user has chosen a file to play back, and thus the file has to be transferred from the Home Media Server to the Home Media Client. This transfer takes place out-of-band from a UPnP point of view. An external protocol, in our case HTTP (see section 4.6, “Choice of transport protocol”), is used.
Figure 7: Communication between client and server in our Home Media System.
6 Home Media System prototype

This chapter discusses the internal design of the Home Media Client only.

All the work done on the Home Media Server, Home Media Manager and Home Media Browser was straightforward implementation work using already existing interfaces and frameworks, and is therefore not covered in detail. However, we make a note on the choice of implementation platform for the server, later in the chapter.

The Home Media Client’s task is to provide the overlying application, the Home Media Browser, with an easy way of accessing the content of Home Media Servers.

In figure 3 on page 9 the Home Media Browser and Home Media Client are shown as separate modules talking to each other via remote procedure calls or library calls. For simplicity, these two modules will be combined into one application. However, as will be explained, the Home Media Browser is loosely coupled to the Home Media Client anyway, allowing to break out the Home Media Client to a separate library or module in the future, if needed.

6.1 The client C++ classes and layers

The class diagram shown in figure 8 displays the structure of the classes making up the Home Media Client. Method signatures have been omitted as well as methods of less crucial importance.

The client classes can be divided into three layers. At the lowest level there are a couple of classes which act as wrappers for the Libupnp library. Their names are prefixed with “Libupnp”. Above them the core classes of the Home Media Client can be found. They contain most of the Home Media Client’s logic. Finally, on top of them there is the BrowseUtility class which is the interface towards the Home Media Browser application.

The two library wrapper classes LibupnpMediaServerFinder and LibupnpMediaServer do the actual work of talking to the Home Media Server. They are completely dependent on the Libupnp library. Therefore it is desirable that these two classes together with the other library wrapper classes are as loosely coupled as possible to the core classes. The motivation for this is that if the Libupnp library is replaced by another library in the future, as little as possible of the application should have to be rewritten. The loose coupling is achieved by having two interfaces, IMediaServerFinderImpl and IMediaServerImpl which are then implemented by LibupnpMediaServerFinder and LibupnpMediaServer. If another library is introduced and used
instead of Libupnp, the new wrapper classes conform to the IMediaServerFinderImpl and IMediaServerImpl interfaces the rest of the application remains unaffected.

Figure 8: Home Media Client class diagram
The `BrowseUtility` isolates the core classes from the Home Media Browser application. Even if the core classes change, `BrowseUtility` will remain the same so that an application using `BrowseUtility` will not have to be rewritten.

Each `PropertyNode` (see figure 8), including all subclasses, has a `Properties` object attached. The `Properties` class manages a simple database consisting of `<key, value>` pairs. This makes it possible to store properties in each node in the content hierarchy. This mapping is mainly used to store properties extracted from the XML-coded responses of the browse actions. Some examples follow to give an idea of how the `Properties` class is used.

- Each `MediaServer` has a property pair `<contentDirectory, URL>` where the URL value holds the Content Directory Control Url.
- All `PropertyNode` objects have an `<id, ID>` pair where ID is an integer value uniquely identifying the node on the media server.
- All `PropertyItem` objects have a `<name, NAME>` pair where NAME contains the filename of the item.

Other properties, such as MIME media type and date of creation, can be added if needed.

The drawback of using `<key, value>` mappings is that the library wrapper classes and the core classes have to agree on the names of the keys, since there are no explicit access methods for getting and setting each property. Instead, all properties are set via a common method `setProperty(string key, string name)`. If a new library wrapper class, for example, would fail to set the key string of the ID property correctly, the core classes would not be able to browse the media server. To partly remedy this potential problem some common standard key strings are defined in the `Properties` class. These strings should be used by all library wrapper classes.

An alternative solution would be to have explicit set and get methods for the bare minimum properties needed, ID and NAME, and have optional properties being set via the `setProperty()` and `getProperty()` methods of the `Properties` class.
6.2 Design patterns used

Five design patterns (Gamma et al. 1995) are used in the client, and in this section we explain where they are used. One definition of patterns goes: "... simple and elegant solutions to specific problems in object-oriented software design" (Gamma et al. 1995). In other words, they can be seen as a software counterpart of standard problem solutions in mathematics; they are well understood and are known to give a “correct answer” when used on an applicable problem. In the software world a “correct answer” would be properly functioning programs. So the choice to use design patterns was not really a choice. Why reinvent the wheel if you already know about it? Using design patterns also sounds well with the project principals set up in section 1.5.1, "Philosophy”.

6.2.1 Facade

The first design pattern is the facade pattern. The BrowseUtility acts as a facade of the Home Media Client against the Home Media Browser. Its task is to act as the single entry point for the Home Media Browser when operating on the Home Media Client. It provides facilities such as walking up and down a Media Server file hierarchy and getting an item’s url. The BrowseUtility in turn operates directly on several of the Home Media Client classes. This way the application’s dependencies on internal Home Media Client classes is kept to a minimum. If the internal classes were to change, it would not have an impact on the application code as long as the BrowseUtility interface stays the same.

6.2.2 Composite

The second pattern used is composite. The three classes PropertyNode, PropertyContainer and PropertyItem together form the composite design pattern. The use of this pattern comes quite naturally from the way content is organized in UPnP A/V media servers. A media server exports one or several content directories which in turn can contain other content directories or files.

6.2.3 Chain of responsibility

The third pattern is chain of responsibility. MediaServerFinder and MediaServer are specializations of a BrowsableContainer. MediaServerFinder’s browse() method searches for media servers and MediaServer’s browse() method browses a media server. An ordinary BrowsableContainer has none of these capabilities. Instead when its browse() method is invoked it passes on the request to its parent. The request is passed on that way until it reaches a MediaServer object, which carries out the request.
The decision to use the described approach has more logical reasons than technical. The MediaServer object is not much more complex than a BrowsableContainer, so in fact all BrowsableContainer objects could be replaced by MediaServer objects with little memory usage penalty. However it is easier to think in terms of “dumb” BrowsableContainer objects and “smart” MediaServer objects with distinct roles in the content hierarchy.

The sequence diagram in appendix D illustrates how a descend() method invocation from the client application propagates through the client classes.

### 6.2.4 Singleton

The next to last pattern is the singleton pattern. The LibupnpMediaServerFinder and LibupnpMediaServer need to initialize the Libupnp library before certain functions can be used. Initialization and finalization of the Libupnp library must not happen more than once per program run. This resource management takes place in the LibupnpManager class which is a singleton class. When LibupnpMediaServerFinder and LibupnpMediaServer want to use the functionality provided by LibupnpManager they call LibupnpManager::GetInstance(), which is used instead of the default constructor.

### 6.2.5 Delegate

The last pattern is the delegate pattern. When a MediaServer object’s browse() method is invoked it delegates the actual work of talking to the Home Media Server to the library wrapper class LibupnpMediaServer. The MediaServerFinder and LibupnpMediaServerFinder behave the same way. As already discussed in section 6.1, “The client C++ classes and layers”, the reason for delegating is to make the core classes independent of the communication libraries used.
6.3 Client-side software development

The Libupnp library operates on the lowest layer on the client side. On top of that is the Home Media Client middleware. Even though the Home Media Browser and the Home Media Client will be one single application, the intent is to make the Home Media Client code as independent as possible of both the UPnP library and the graphical user interface used. If designed well, it will be easier to separate the two modules in the future. The graphical user interface in this case is the Home Media Browser, which is described in chapter 7.

For simplicity, the UPnP A/V Renderer device (see chapter 5) will not be implemented. Instead, when the Control Point wants to initiate playback of a media file, the internal media player service of the STB at hand will be invoked (see section 2.3). The consequence of this is that no external Control Point on the network will be able to start and stop playback on the STB. This is good from a security point of view, since the user must have physical access to the remote control of the STB in order to control playback. However, it makes the solution less general.

Most of the already available software in our STB is written in C or C++ for reasons of efficiency. The Libupnp library is written in C. In the UPnP domain there are a number of concepts such as media servers, containers and media items. To model these concepts, the object-oriented paradigm will suit us very well. Therefore, the choice of development language for the client software will be C++.

The Home Media Browser user interface will be created using the GIMP Toolkit version 1.2, more commonly known as GTK (GTK 2004). The reasons for this choice are that GTK is already available on the STB. Also, by using it, a more consistent look and feel is obtained, with regard to existing applications.

6.4 A note on server-side software development

The original idea was to use the open-source project Jetty (Jetty 2004) as the HTTP server component in our system. By using Jetty, a highly modular and customizable HTTP/1.1 server component can be obtained. Of course there is a certain cost in integrating Jetty into the system.

CyberLink, the UPnP software chosen in section 4.5, has a superset called CyberMediaGate, which is a partial MediaServer implementation. Right at the end of our prestudy phase it became apparent that the HTTP server built into CyberMediaGate not only had support for message sending over HTTP, but also basic HTTP/1.0 file transfers. To minimize the components involved and thereby lightening the strain of software integration, Jetty was abandoned in favour of the built-in
HTTP server. Also, if the internal HTTP server is used instead of Jetty, the memory footprint size of the server is reduced, which is desirable (see section 2.5, “Preliminary requirements analysis”).

The Home Media Manager becomes a GUI working on top of the CyberMediaGate framework. Again, the philosophy of the simplest solution (see section 1.5.1) guided the decision. In any case, the use of CyberMediaGate implies that Java will be our development language. The widget framework chosen is Swing (SUN 2004b).

The Home Media Manager and the Home Media Server will be implemented as one single application for simplicity. But any new functionality introduced will be placed in the right module, keeping the roles of the Home Media Server and the Home Media Manager modules strictly apart. This will make it easier to separate the modules in the future.

### 6.4.1 Implementation note

When the implementation started it became apparent that the chosen HTTP server, the built in CyberMediaGate one, had one major deficiency. In the available implementation a file was read in its whole to memory before being transferred. This lead to memory problems as soon as the file in question was large, as is the case with audio and video files. Therefore the use of Jetty was reconsidered. The choice stood between improving the CyberMediaGate HTTP server or incorporating the Jetty HTTP server, despite of the reasoning in section 6.4, "Server-side software". The two options both had their advantages and disadvantages.

Patching the CyberMediaGate server to support transfer of arbitrarily large files would involve reading and understanding the CyberMediaGate code in detail and then adding and modifying code as appropriate.

After examining the CyberMediaGate code, the estimation was made that adding support for large file transfers would be feasible within a reasonable amount of time. The necessary changes were therefore made.

Later, another issue with the CyberMediaGate HTTP server was found. When the HTTP server receives a request for only transmitting the HTTP header, it actually sends the whole file. This problem was also resolved by modifying and adding code, so the CyberMediaGate server was our final choice.
In an end-user product the user interface is of importance. The user expects the system to be responsive, easy to learn and efficient in terms of number of needed button presses to carry out actions (Faulkner 2000). Even though the focus of this project is more on making the technical aspects, e.g device discovery and communication to work, some effort is also put into creating a draft user interface. The user interface in our system consists of the interface of the Home Media Browser on the client and the interface of the Home Media Manager on the server. These two are described below.

7.1 Home Media Browser Interface

The tasks that the user will perform in the Home Media Browser are:

- Browse a list of files
- Control playback of a file

File hierarchies are commonly used to browse large set of files, and so is also the case in the UPnP A/V Architecture, as described in section 5.2. As described in section 5.2, an element in can be either a container or an item. In the case of a hierarchy, the browsing task can be broken down into the following subtasks:

- Select the next element in the current hierarchy level
- Select the previous element in the current hierarchy level
- Descend one level (Enter a child container)
- Ascend one level (Go to the parent container)
- Execute element (Enter a container or play an item)

Playback control is performed with the dedicated Play and Stop buttons on the remote control of the STB, which do not require further explanation.

7.1.1 Interaction analysis

The next design problem is how to map the above browse actions to the STB controls. The STB lacks a mouse for navigation and selection. Even though there is a good joypad on the keyboard, efficient navigation and interaction will probably not be achieved by using it only. The reason is that it involves placing the mouse cursor on the desired UI element and then pressing the OK button to fire an action, which is cumbersome. Instead, the arrow buttons on the remote control, along with the OK button, should be used as extensively as possible. The buttons are in a near
proximity of each other (see figure 9). This enables the user to navigate easily without ever having to look on the remote control in order to relocate the finger for each touch.

The browsing subtasks described earlier are therefore mapped to the remote control buttons as can be seen in figure 9.

![Figure 9: Navigation with remote control](image)

### 7.1.2 Visual appearance

A low-fi prototype of the user interface is shown in figure 10. The view is dominated by the item list. It has a scrollbar to allow listings that contain more items than physically can fit into the area. Containers are shown first, followed by media files. To the right there is a preview area which can show video. Right below there is a shortcut indicator to start the preview. Below there is a group of buttons that allows the user to select what kind of files to show in the item list. Finally, at the bottom, there are three coloured shortcut indicators. Each colour on the shortcut indicators corresponds to a special button on the remote control with the same colour.

The interface has been created with Nielsen’s heuristics in mind (Faulkner 2000). The following walkthrough of the ten guidelines shows how they have been applied to the user interface.

1. Simple and natural dialogue.
   The dialogue contains a small amount of elements. One could argue that the video preview is too much. However, it is a balance between simplicity and consistency; the already existing video on-demand application on the STB has a preview video area.

2. Speak the user’s language.
   No acronyms or technical jargon is used. The most technical word in the dialogue is the word “file”, a word that a user with a home computer should be familiar with.

3. Minimize the user’s memory load.
   The only thing the user has to remember when switching from the
main dialogue to the search dialogue is what keyword she wanted to search for.

The same word is used everywhere to denote the same thing.

5. Feedback.
Scanning the network for media servers may take up to three seconds. During this time the small information dialogue shown in figure 11 pops up to inform the user of what is happening.

6. Clearly marked exits.
The exit is not marked at all in the main dialogue. However, just as with other applications on the STB, the back or menu buttons on the remote control will bring the user back to the STB main menu. The search dialogue has a clearly marked way of going back to the main dialogue.

7. Shortcuts.
The mapping of buttons to actions described in section 7.1.1, “Interaction analysis” is probably very close to the most accelerated way of navigating the hierarchy, given the physical button layout on the remote control. Novices are likely going to find the learning curve to be very lean.

8. Good error messages.
It was decided that finding all possible sources of error and creating error messages for them was out of the scope of prototyping.

The actions that the user can undertake are very restricted, so normal use will not cause errors. An example of this is the dialogue shown in figure 12, which restricts the possible user actions to inputting keywords, searching, selecting an item and going back to the main dialogue. However, network errors, file corruption and other errors may cause the system to behave in unexpected ways.

In both the main dialogue and the search dialogue, access to context sensitive help is clearly marked.
Figure 10: Low-fi prototype of the Home Media Browser main dialogue

Figure 11: Low-fi prototype of feedback dialogue
Figure 12: Low-fi prototype of search dialogue
7.2 Home Media Manager Interface

The interface of the Home Media Manager tries to be as minimalistic as possible. At the top of the window there is a toolbar with three icons. Clicking the leftmost icon, a dialogue pops up which allows the user to select a new folder to share. A click on the middle button makes the selected shared folder and its contained files unavailable, after first having displayed a confirmation question. The rightmost icon saves the changes and exits the program. To further guide the user, each button has a tooltip that gets displayed when the mouse pointer rests over the button (Tooltips are informative text strings displayed when the mouse pointer rests for a while over a UI element).

Figure 13: Home Media Manager interface

Again, Nielsen’s heuristics are used (Faulkner 2000).

1. Simple and natural dialogue.
   The main window shows an overview of what folders have been shared. The different actions possible are represented by buttons in the toolbar.

2. Speak the user’s language.
   The folder selection dialogue is of Java standard type, which is very similar to, for example, Windows file selection dialogues. Tooltips
explain in a clear way what will happen when pressing a toolbar icon.

3. Minimize the user’s memory load.
   The interface is very minimal and there is not much at all that has to be remembered. When going from the main window to the folder selection dialogue the user only has to remember what folder she wanted to share.

   The GUI elements and dialogues used are of standard type that the user is likely to recognize.

5. Feedback.
   As soon the user has added or removed a folder, the change is immediately seen in the folder list.

6. Clearly marked exits.
   Apart from the standard way of closing windows via the top right cross button, the user can also exit the program by pressing the button with the tick mark.

7. Shortcuts.
   Shortcuts in the form of key combinations are missing.

8. Good error messages.
   Just as with the client interface (see section 7.1.2), it was decided that determining all possible errors was out of the scope of prototyping.

   Handling of some errors are prevented, such as program crashes due to folders having been moved or renamed while the program was not running.

    Detailed help text is missing. However, tooltips give a short description of the consequence of the button actions.
8 Evaluation

How well did the system fulfil the requirements that we had set up? Tests of the functional requirements were done by starting the system and carrying out actions that verified that the system behaved as stated in the “Description” field of each requirement (See appendix B, “Functional requirements”).

The actions verified that the following base level functional requirements were fulfilled (see chapter 3, “System requirements”):

F 1.1, “Single Home Media Server discovery”
F 1.2, “Single Home Media Server support”
F 2.1, “Listing of available folders and files”
F 2.2, “Single stream of MP3 audio file”
F 2.3, “Basic audio playback control”
F 3.1, “Audio file selection”
F 3.2, “Basic audio playback control”
F 4.1, “Support for MP3 files”
F 5.1, “Home Media Client and Home Media Browser host OS”
F 5.2, “Home Media Server and Home Media Manager host OS”

In addition, the following extra level functional requirements were fulfilled:

F 1.3, “Multiple Home Media Servers discovery”
F 1.4, “Multiple Home Media Servers support”
F 3.4, “Viewing of MPEG-2 video file”
F 6.1, “Folder sharing”
F 6.2, “Folder un-sharing”
F 6.3, “Effect of folder sharing”
F 6.4, “Effect of folder un-sharing”
F 5.3, “Home Media Server and Home Media Manager portability”

Looking at the non-functional requirements, it is more of a judgement if they were fulfilled or not. A usability walkthrough (Faulkner 2000) was made, but not all planned GUI parts were implemented. Therefore the only non-functional requirement fulfilled was:

N 2.1, “Open standards”
To summarize, all base requirements were fulfilled and 8 out of 14 extra requirements were also met. 1 out of 3 non-functional requirements were also completely implemented.
9 Conclusion

We have successfully designed and implemented a prototype Home Media System. On the way, the four discovery protocols Jini, OpenSLP, Salutation, and UPnP were evaluated. Of them, UPnP was chosen. This does not mean that UPnP is the most suitable in all situations, but together with the UPnP  Audio Video architecture its capabilities matched the purpose of this project well. An evaluation of transport protocols was also performed, which resulted in HTTP being chosen.

As the basis for the server software, CyberMediaGate was chosen. Even though some problems that needed to be resolved were found, using this software saved a significant amount of design and development time. On top of CyberMediaGate a GUI for managing what media files to share was built.

Most of the design and implementation effort was put on the client side. The STB had no software for media handling via UPnP. The general UPnP library Libupnp from Intel was used for the low-level communication. A small framework for browsing media servers was designed and implemented. Finally, a graphical user interface for the STB using the developed framework was created.

A quick look back on section 1.4, “Expected outcome and delimitation of thesis work”, confirms that the high level goals that were set up were fulfilled:

- There is a graphical user interface on the server, which can be used to configure what media files to share from a PC.
- A PC can share the selected media files on the network.
- The set-top box is able to locate and retrieve the shared media files.
- A graphical user interface has been created on the client (the set-top box), so that the user can browse and play back the shared media files.

9.1 Future work

During the project many ideas were spawned on how the system could be improved. Though most ideas would have been very useful if implemented in the thesis work, they were not central to the main task of creating a prototype media management system, and were therefore left out. The following sections present some of these ideas.
Virtual file hierarchies

Classifying media files by their physical place in the file hierarchy may not always be the best way. Users may want to sort files by artist, album, year or other. One way to solve this would be to have the server retrieve all metadata information from the media files and create virtual folders in which it places the media files depending on the current view mode.

Module separation

The amount of time spent on developing the Home Media Browser was significantly smaller than that spent on the Home Media Client. Therefore, there is great room for improvement on its internal structure. Redesigning the program to use the model-view-controller design pattern is a top improvement candidate (Baray 1999; eNode 2002).

Also, in the current solution the Home Media Server and Home Media Manager are combined into a single application. A more flexible solution would be to separate the two and let the Home Media Server be configurable through a well-defined interface. This would decrease the memory consumption when the Home Media Manager is not running and allow different front-ends (Home Media Managers) to be developed, possibly integrating them with existing media players on the host platform.

In chapter 6 it was decided that the Home Media Browser and Home Media Manager should be combined into a single application for simplicity. To make the capabilities of the Home Media Client available to other applications than the Home Media Browser as well, the client could be made to a platform service on the STB with C++ and/or JavaScript interfaces. Because of the already loose coupling between the Home Media Client and the Home Media Browser, this can be done without big complications.

Viewing photos

The Home Media Browser can be modified to support browsing and viewing of digital photographs. This also includes modifying the Home Media Server so that it exports files of the desired type.

Other transport protocols

In the current solution only HTTP is supported as the transport protocol. Another useful protocol to support would be RTSP. In order to support RTSP, mainly the Home Media Server would need to be modified, since the STB already can fetch content using RTSP.
Personal video recording

Contemporary PCs have a large amount of hard disk memory space, and could therefore be used as personal video recorders (PVRs for short). A UPnP server device could be developed for the STB, which publishes the broadcast digital video. A PC client could then connect to such a device, get the data and save it on the hard disk.

Usability

The implemented user interface is very basic and needs improvement in a number of areas:

- Feedback. When an activity that takes time begins, such as searching for media servers, the user should see a dialogue informing her about the activity so that she knows what is going on and does not think that the system has hung. The suggested solution with a feedback dialogue presented in chapter 7, “User interfaces”, would be one way to support this.

- Navigation speed. In an end-user device, such as an STB, the user expects the system to be responsive and unobtrusive. Currently when descending or ascending a container in the Home Media Browser there can be a delay of up to half a second between the button being pressed and the new container being displayed. This delay is due to network communication latency and processing time on the Home Media Server.

One way to decrease the delay is preloading of subdirectories. In other words, while the user looks at the current item list, all immediate subdirectories are loaded in the background. When the user finally decides to enter a subdirectory, the action can be carried out immediately since the directory has already been loaded into memory.

- Fast forward and rewind with HTTP. As mentioned in section 4.9, HTTP does not support searching in media streams. However, by clever usage of the HTTP/1.1 Content-Range header (RFC2616 1999) it is possible to achieve searching within an audio or data stream. This requires the Home Media Server to support HTTP/1.1.

An outline of the solution looks as follows: The client has a counter that keeps track of how far into the current stream it has consumed data. When the client wants to search it closes the connection to the server. Depending on whether a fast forward or rewind command was issued the counter is either increased or decreased. The client then sends an HTTP/1.1 GET request with the Content-Range header start value set to the value of the counter and the end value set to the length of the file being transmitted. The server will then
send the file starting from the position of the counter and the system will effectively appear to search within the transmitted file.

Security issues

No security impact analysis has been carried out. Questions to investigate include: Can non-media files be accessed, say executables? Does the server and client software cause any security breaches in their respective operating systems? How is user privacy protected? Do other people, who are not on the same local network, have access to the files; what if each home installation does not have its own subnet, e.g. they share the same IP subnet address space?

Arbitrary Control Point location

In the chosen solution, the UPnP A/V Control Point and Media Renderer device are combined into one, the Home Media Browser. The consequence of this is that one must physically be near the STB to be able to control what content is being played back. If instead a separate Control Point application is created, it is no longer necessary to be in the proximity of the STB to control it. Then, the Home Media Browser could for example be controlled from a PDA over a WLAN.

PDA adaptation

As handheld devices grow more powerful, a natural consequence is that more and more software that was initially developed for larger computers makes its way into such devices. The memory capacity also increases, which allows the user to store digital music, videos and images on a PDA. The Home Media Server and Home Media Manager could be adapted to run on, for instance, PDAs. This probably mainly means making the Home Media Manager interface fit on a small display.

Realtime video format conversion

All video formats are not supported by the STB. Since the video decoding is done in dedicated hardware, it is not possible to upgrade the decoder once the STB is deployed. If the computer that runs the Home Media Server is powerful enough, it could convert unsupported file formats in real time to a format recognized by the STB.
10 Personal reflections

10.1 Software reuse and time

The major concern I had in the beginning of this project was how to bring together all the pieces necessary to make the whole system work. There was the Home Media Server, the Home Media Manager, Home Media Client, and the Home Media Browser. All these pieces were quite different from each other, and all pieces had to work, or the others would be useless. Without a working client a server makes no sense and vice versa. It was therefore very fortunate that there were some open-source software components available that I was able to reuse. Without them, there is no doubt that it would not have been possible to create the prototype within the scheduled time.

10.2 Significance of design patterns

One of my objectives before starting the project was to practice the use of design patterns as much as possible (Gamma et al. 1995). The main use for the design patterns that I had in mind was, quite naturally, as part of the design process. Not very surprisingly they helped a lot. I found it much easier to visualize different solutions in my mind. Before starting the project I did not really know that I was going to find and use the open-source components I did use. I therefore had to read and understand more code than expected, and design less. Reading the code, another benefit of design patterns manifested itself: There were several occasions when I was reading the code that I realized that a certain design pattern was being used. As a consequence, I had to read much less code, since I already knew on a high level what would be there. It also helped in giving a bigger picture of how the software was structured.

10.3 Development model

In the very beginning the idea was to have very separate project phases, much like the waterfall model (Pfleeger 2001). Soon I discovered that I would not be able to design some things completely in advance, before first having tried other things out in code, just to see if things would work. I was first frustrated because I felt I was leaving a more “pure” development model for something that reminded of undirected ”hacking”. However, after having sorted details out, I was able to go back with greater confidence and do my design. At this point I started to read about extreme programming (Beck 2000). While I was not convinced about all aspects of it, for example the lack of separate documentation and a big comprehensive design plan, I was attracted by the idea of
almost always having a working version of the software. My decision therefore was to, at any point, do as much design as possible that I knew would not change, and leave the design of parts that I did not know enough about to a later point in time.
References

Technologies, digital media and protocols


CSWL - California Software Labs (2002). *UPnP, Jini and Salutation - A look at some popular coordination frameworks for future networked devices*


Faulkner, Xristine (2000). *Usability Engineering*; Palgrave Macmillan


Kaffe (2004). *Kaffe.org* [www]

Libupnp (2004). *Linux SDK for UPnP Devices 1.2.1* [www]

Microsoft (2002). *UPnP AV Architecture*

Microsoft (2003). *Universal Plug and Play* [www]
Accessed: 2002-06-14


onTV (2003). *DVB-SI Metadata for digital television* [www]
Accessed: 2004-02-10

OpenSLP (2003). *An Introduction to SLP* [www]
Accessed: 2004-02-16

Oregan (2004). *End-To-End Digital Home Entertainment Middleware*

PsiNaptic (2004). *Products/Solutions* [www]
Accessed: 2004-02-06

Salutation (2002). *Salutation Consortium* [www]

Samba (2004). *opening windows to a wider world* [www]

Schulzrinne (2003). *Some Frequently Asked Questions about RTP* [www]
Accessed: 2004-02-16
<http://sourceforge.net/>

SUN (2003a). *J2RE v1.4.2_x license* [www]
<http://java.sun.com/j2se/1.4.2/j2re-1_4_2_04-license.txt>
Accessed: 2004-02-17

SUN (2003b). *Jini General FAQs* [www]
<http://www.sun.com/software/jini/faqs/#1q3>
Accessed: 2004-02-06

<http://wwws.sun.com/software/jini/whitepapers/>
Accessed: 2004-02-05


SUN (2004b). *Java Foundation Classes (JFC/Swing)* [www]
<http://java.sun.com/products/jfc/>

UDDI (2002). *UDDI Executive Whitepaper* [www]
Accessed: 2004-02-23

<http://www.upnp.org/>

<http://www.videolan.org/>
Accessed: 2004-05-14

<http://www.w3.org/TR/soap12/>

**Software engineering and design**

Baray, Cristobal (1999). *The Model-View-Controller (MVC) design pattern* [www]
Accessed: 2004-04-14

Beck, Kent (2000). *Extreme Programming Explained; Addison Wesley*

eNode (2002). *Model-View-Controller Pattern* [www]
<http://www.enode.com/x/markup/tutorial/mvc.html>
Gamma, Erich; Helm, Richard; Johnson, Ralph; & Vlissides, John (1995). Design Patterns: Elements of Reusable Object-Oriented Software; Addison Wesley


**RFCs**

IETF (2004). IETF repository retrieval [www]
<http://www.ietf.org/rfc.html>


RFC 2609 (1999). Service Templates and Service: Schemes

RFC 2610 (1999). DHCP Options for Service Location Protocol

RFC 2614 (1999). An API for Service Location

RFC 2616 (1999). Hypertext Transfer Protocol HTTP/1.1


**General reading**

DHWG (2003). Digital Home Whitepaper [www]

Home Networking News (2000). New “Simple Control Protocol” is Link between Internet and the Light Switch [www]
<http://www.homenetnews.com/pages/Archives/H0007SCP.htm>
Accessed: 2004-02-06

Intel (2003a). Designing a UPnP AV MediaServer

Intel (2003b). Designing a UPnP AV MediaRenderer

<http://www.intel.com/technology/upnp/>
Accessed: 2004-06-03


Murphy, Kieron (1999). Beaming to the enterprise [www]
<http://www.developer.com/tech/article.php/10923_609401_2>
Accessed: 2004-02-14

Tanenbaum, Andrew S (2003). *Computer Networks; US Imports & PHIPEs*
Design and implementation of a prototype home media system
for an IP-based set-top box
Robert Bo Johansson
## A Glossary

<table>
<thead>
<tr>
<th>Concept</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DHCP</td>
<td>Dynamic Host Configuration Protocol. A way of automatically assigning IP numbers to hosts on a network. This lightens some of the configuration burden off of network administrators’ shoulders.</td>
</tr>
<tr>
<td>HAL</td>
<td>Hardware Abstraction Layer. Software which has the purpose of hiding hardware details from the applications of the system.</td>
</tr>
<tr>
<td>Home Media Browser</td>
<td>The user interface of the Home Media Client with which the user interacts. The Home Media Browser may sometimes be referred to as just “browser”.</td>
</tr>
<tr>
<td>Home Media Client</td>
<td>The software that runs on the STB and communicates with the Home Media Server. If no confusion with other clients is possible, an Home Media Client may be referred to as just ”client”.</td>
</tr>
<tr>
<td>Home Media Manager</td>
<td>An administrative program which allows the user to configure an Home Media Server.</td>
</tr>
<tr>
<td>Home Media Server</td>
<td>A program running on a PC which makes media files available to Home Media Clients. If no confusion with other servers is possible, an Home Media Server may be referred to as just ”server”.</td>
</tr>
<tr>
<td>Home Media System</td>
<td>The compound system consisting of Home Media Servers, Home Media Managers and a Home Media Client with the Home Media Browser.</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol. A general-purpose protocol for transferring data in the Internet.</td>
</tr>
<tr>
<td>JRE</td>
<td>Java Runtime Environment. Implementation of a Java Virtual Machine for a specific platform allowing execution of platform independent Java programs.</td>
</tr>
<tr>
<td>MIME Media Type</td>
<td>Multipurpose Internet Mail Extensions Media Type. A way of classifying electronically transferred documents.</td>
</tr>
<tr>
<td>MinGW</td>
<td>Minimalist GNU for Windows. GNU tools and libraries ported to the Windows platform, which enable creation of native Windows applications using code written for the GNU set of tools.</td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>--------</td>
<td>------------</td>
</tr>
<tr>
<td>MP3</td>
<td>MPEG-1 layer 3. MP3 is one of the MPEG sub-standards (see below), intended specifically for audio encoding.</td>
</tr>
<tr>
<td>MPEG</td>
<td>Motion Picture Experts Group. MPEG is the name of a collection of standards used for coding audio and visual information (like movies, video and music) in a compressed format. There are different revisions of the standard with different purposes. The revisions are denoted by appending -1, -2, -3 or -4 to the MPEG name.</td>
</tr>
<tr>
<td>PDA</td>
<td>Personal Digital Assistant. A small handheld computer device with calendar, notes, address book, and other useful applications.</td>
</tr>
<tr>
<td>RPC</td>
<td>Remote Procedure Call. A way of doing function or method calls between program entities over a network. This makes it possible for programs running on physically disjoint machines to invoke each other.</td>
</tr>
<tr>
<td>SDK</td>
<td>Software Development Kit. Tools and development libraries provided by a software house or group, which aid third-party developers in developing extensions for the software provided by the software house or group.</td>
</tr>
<tr>
<td>STB</td>
<td>Set-top box. A device connected to a TV set which is able to decode digital broadcast streams. More advanced types of STBs may be IP-based, meaning that the broadcast streams are carried over the internet protocol. Such a box also has the potential to connect to various internet services and perform tasks such as web surfing.</td>
</tr>
<tr>
<td>Virtual network</td>
<td>A network running on top of another (host) network. Typically the virtual network has a different address space than the host network, and many virtual networks can coexist on a host network.</td>
</tr>
<tr>
<td>WLAN</td>
<td>Wireless Local Area Network. A computer network in which the signals are not carried by cable, but instead propagate through the air as short wave electromagnetic signals.</td>
</tr>
</tbody>
</table>
## B Functional requirements

### B.1 Home Media Client

#### F 1.1 Single Home Media Server discovery

<table>
<thead>
<tr>
<th>Level</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>If at least one Home Media Server is connected to the network and an Home Media Client starts, the Home Media Client discovers at least one of the Home Media Servers and finds out its services.</td>
</tr>
<tr>
<td>Motivation</td>
<td>For zero configuration need from the user’s point of view. The system should be dynamic and allow servers to be removed and inserted into the network without any need for reconfiguration in the Home Media Client.</td>
</tr>
<tr>
<td>Influences</td>
<td>F 1.2, F 1.3, F 1.4, F 3.1</td>
</tr>
</tbody>
</table>

#### F 1.2 Single Home Media Server support

<table>
<thead>
<tr>
<th>Level</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>One device on the network can act as Home Media Server.</td>
</tr>
<tr>
<td>Motivation</td>
<td>At least one device (e.g. PC) must be present and act as Home Media Server for the system to work.</td>
</tr>
<tr>
<td>Influences</td>
<td></td>
</tr>
</tbody>
</table>

#### F 1.3 Multiple Home Media Servers discovery

<table>
<thead>
<tr>
<th>Level</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The Home Media Client is able to discover “all” (at least four) Home Media Servers connected to the network.</td>
</tr>
<tr>
<td>Motivation</td>
<td>It is common to have many devices (e.g. PCs) connected to a network and it is probable that each of them has its own content to share.</td>
</tr>
<tr>
<td>Influences</td>
<td>F 1.4, F 3.1</td>
</tr>
</tbody>
</table>
### F 1.4 Multiple Home Media Servers support

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
<th>Motivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>Multiple devices on the network can act as Home Media Servers concurrently.</td>
<td>It is common to have many devices (e.g. PCs) connected to a network and it is probable that each of them has its own content to share.</td>
</tr>
</tbody>
</table>

### B.2 Home Media Server

#### F 2.1 Listing of available folders and files

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
<th>Motivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Upon request from a Home Media Client the server will respond with a set of available folders and files.</td>
<td>For the user to be able to browse through the list of files available via the Home Media Browser, the set of files must be communicated to the Home Media Client.</td>
</tr>
</tbody>
</table>

#### F 2.2 Single stream of MP3 audio file

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
<th>Motivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>The server is able to stream one MP3 file to one STB at a time.</td>
<td>Minimal prototype requirement.</td>
</tr>
</tbody>
</table>

#### F 2.3 Basic audio playback control

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
<th>Motivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>The server can receive and effectuate start, stop, and pause audio playback commands.</td>
<td>In analogy with well-known devices such as VCRs or CD-players. Expectation from the user.</td>
</tr>
</tbody>
</table>

Influences: F 2.4
### F 2.4 Extended audio playback control

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
<th>Motivation</th>
<th>Influences</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>The server can receive and effectuate seek forward and seek backward audio playback commands.</td>
<td>In analogy with well known devices such as VCRs or CD-players. Expectation from the user.</td>
<td></td>
</tr>
</tbody>
</table>

**Motivation**

In analogy with well known devices such as VCRs or CD-players. Expectation from the user.

**Influences**


### F 2.5 ID3 tag awareness

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
<th>Motivation</th>
<th>Influences</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>The server can read ID3 tags from MP3 files.</td>
<td>Useful when presenting music file choices to the user in the Home Media Browser</td>
<td></td>
</tr>
</tbody>
</table>

**Motivation**

Useful when presenting music file choices to the user in the Home Media Browser.

**Influences**


### F 2.6 MIME type awareness

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
<th>Motivation</th>
<th>Influences</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>The server can make distinctions between file types and classify them by their MIME-type.</td>
<td>Future extensibility of the Home Media System platform.</td>
<td></td>
</tr>
</tbody>
</table>

**Motivation**

Future extensibility of the Home Media System platform.

**Influences**


### B.3 Home Media Browser

#### F 3.1 Audio file selection

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
<th>Motivation</th>
<th>Influences</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Via the STB the user can select from the set of available audio files, which one to control. (See F 3.2)</td>
<td>Expectation from the user.</td>
<td></td>
</tr>
</tbody>
</table>
### F 3.2 Basic audio playback control

<table>
<thead>
<tr>
<th>Level</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>From the Home Media Browser it is possible to start, stop and pause audio playback.</td>
</tr>
<tr>
<td>Motivation</td>
<td>In analogy with well known devices such as VCRs or CD-players. Expectation from the user.</td>
</tr>
<tr>
<td>Influences</td>
<td></td>
</tr>
</tbody>
</table>

### F 3.3 Extended audio playback control

<table>
<thead>
<tr>
<th>Level</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>From the Home Media Browser it is possible to seek backward and seek forward in audio playback.</td>
</tr>
<tr>
<td>Motivation</td>
<td>In analogy with well known devices such as VCRs or CD-players. Expectation from the user.</td>
</tr>
<tr>
<td>Influences</td>
<td></td>
</tr>
</tbody>
</table>

### F 3.4 Viewing of MPEG-2 video file

<table>
<thead>
<tr>
<th>Level</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>From the Home Media Browser the user can select to view MPEG-2 video served from the Home Media Server.</td>
</tr>
<tr>
<td>Motivation</td>
<td>Value adding product feature for the user.</td>
</tr>
<tr>
<td>Influences</td>
<td></td>
</tr>
</tbody>
</table>

### F 3.5 Viewing of image files

<table>
<thead>
<tr>
<th>Level</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>From the Home Media Browser the user can select to view image files served from the Home Media Server.</td>
</tr>
<tr>
<td>Motivation</td>
<td>Value adding product feature for the user.</td>
</tr>
<tr>
<td>Influences</td>
<td></td>
</tr>
</tbody>
</table>
### B.4 Audio formats

#### F 4.1 Support for MP3 files

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
<th>Motivation</th>
<th>Influences</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Playback of files in MP3 format is possible.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### F 4.2 Support for ogg files

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
<th>Motivation</th>
<th>Influences</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>Playback of files in ogg format is possible.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### B.5 Operating systems

#### F 5.1 Home Media Client and Home Media Browser host OS

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
<th>Motivation</th>
<th>Influences</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>The Home Media Client and Home Media Browser run on Linux.</td>
<td>The client and browser are located on the STB which runs Linux.</td>
<td>F 1.1, F 1.2, F 1.3, F 1.4</td>
</tr>
</tbody>
</table>

#### F 5.2 Home Media Server and Home Media Manager host OS

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
<th>Motivation</th>
<th>Influences</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>The Home Media Server and Home Media Manager run on Windows 2000/XP</td>
<td>Most home PCs run Windows.</td>
<td></td>
</tr>
</tbody>
</table>
### F 5.3 Home Media Server and Home Media Manager portability

<table>
<thead>
<tr>
<th><strong>Level</strong></th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>The server and manager should be designed in such a way that they are easily portable to other platforms. (e.g. PDAs or PCs running Linux)</td>
</tr>
<tr>
<td><strong>Motivation</strong></td>
<td>Value adding feature for the user. Larger potential user base.</td>
</tr>
<tr>
<td><strong>Influences</strong></td>
<td>F 2.3, F 3.4, F 3.5</td>
</tr>
</tbody>
</table>
### B.6 Home Media Manager

#### F 6.1 Folder sharing

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
<th>Motivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>The user is able to select which folders should be made available to the STB. In other words which folders and files the Home Media Server will make available to the Home Media Client.</td>
<td>Media files can be located in various locations on the device (e.g. PC). The user knows best what media files she wants to share.</td>
</tr>
</tbody>
</table>

#### F 6.2 Folder un-sharing

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
<th>Motivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>The user is able to make previously shared folders unavailable to Home Media Clients.</td>
<td>Media files can be located in various locations on the device (e.g. PC). The user knows best what media files she wants to share.</td>
</tr>
</tbody>
</table>

#### F 6.3 Effect of folder sharing

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
<th>Motivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>When a folder has been added to the set of available folders the Home Media Server is able to announce the folder’s availability and serve the folder’s files upon request.</td>
<td></td>
</tr>
</tbody>
</table>
F 6.4 Effect of folder un-sharing

<table>
<thead>
<tr>
<th>Level</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>When a folder is removed from the set of available folders any new requests from an Home Media Client to access files in that folder will fail. The folder will no longer be announced as available to Home Media Clients.</td>
</tr>
<tr>
<td>Motivation</td>
<td></td>
</tr>
<tr>
<td>Influences</td>
<td></td>
</tr>
</tbody>
</table>
## C Non-functional requirements

### C.1 Usability

#### N 1.1 Home Media Browser usability

<table>
<thead>
<tr>
<th>Level</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The Home Media Browser is intuitive, easy to use and adheres to accepted usability guidelines.</td>
</tr>
<tr>
<td>Motivation</td>
<td>The Home Media System is an end-user product which makes usability an important success factor.</td>
</tr>
<tr>
<td>Influences</td>
<td>F 3.1, F 3.2</td>
</tr>
</tbody>
</table>

#### N 1.2 Home Media Manager usability

<table>
<thead>
<tr>
<th>Level</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The manager is intuitive, easy to use and adheres to accepted usability guidelines.</td>
</tr>
<tr>
<td>Motivation</td>
<td>The Home Media System is an end-user product which makes usability an important success factor.</td>
</tr>
<tr>
<td>Influences</td>
<td>F 6.1, S</td>
</tr>
</tbody>
</table>

### C.2 Standards

#### N 2.1 Open standards

<table>
<thead>
<tr>
<th>Level</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>As much as possible of the Home Media System should be based on open standards.</td>
</tr>
<tr>
<td>Motivation</td>
<td>Facilitate interoperability with other devices and make the Home Media System more versatile.</td>
</tr>
<tr>
<td>Influences</td>
<td></td>
</tr>
</tbody>
</table>
D Propagation of browsing invocation
Appendix D: Propagation of browsing invocation
A
access url ...........................................33
ascend ..................................................45
B
backward .............................................71, 72
C
CMatos ................................................20
Combo Device .......................................34
composite ..........................................40
Content Directory ............................31, 39
Control Point ....................................27, 31, 32, 33, 34, 42, 58
Control Url ........................................31, 39
CyberMediaGate ...............................42
D
DCHP ...................................................10
delegate ...............................................41
descend ............................................41, 45
device type .........................................31
DHCP ..................................................10, 67
DVB .....................................................67
E
extreme programming ...................4, 59
F
facade .................................................40
forward ............................................71, 72
G
GCJ .....................................................20
GNU ....................................................67
GTK ....................................................42
H
HAL .....................................................8, 67
Home Media Browser ..................9, 14, 15, 16, 28, 37, 42, 45, 67
Home Media Client .........................9, 11, 14, 16, 25, 28, 33, 37, 67
Home Media Manager ..................9, 16, 37, 45, 50, 67
Home Media Server ......................9, 11, 14, 28, 29, 33, 37, 67, 69
Home Media System ....................9, 10, 16, 17, 31, 34, 55, 67
J
Jetty ......................................................42
Jini .....................................................19, 25
JRE .....................................................20, 67
K
Kaffe ......................................................20
Kreatel ...................................................1
L
layer ....................................................42
loose coupling .................................8, 37, 56
M
MediaServer ......................................40
MediaServerFinder ............................40
MIME ..................................................39, 67
MinGW ..............................................24, 26, 67
model-view-controller ......................56
MP3 ....................................................3, 7, 15, 16, 70, 73
MPEG ...............................................68, 72
O
ogg .....................................................73
OpenSLP ..............................................23, 26
open-source .....................4, 12, 20, 23, 24, 25, 26, 27, 28, 42, 59
P
pause ..................................................15, 70, 72
PDA .......................................................68
playback ..........................................15, 70, 72
properties ...........................................39
PropertyContainer .........................40
PropertyItem ....................................39, 40
PsiNaptic .........................................39, 40
PsiNaptic .........................................20
R
RPC .................................................68
S
Salutation ..........................................19
SDK ...................................................24, 68
seek ...................................................71
<table>
<thead>
<tr>
<th>Term</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>service type</td>
<td>31</td>
</tr>
<tr>
<td>singleton</td>
<td>41</td>
</tr>
<tr>
<td>SLP</td>
<td>19, 25</td>
</tr>
<tr>
<td>SOAP</td>
<td>31, 68</td>
</tr>
<tr>
<td>start</td>
<td>15, 70, 72</td>
</tr>
<tr>
<td>STB</td>
<td>1, 68</td>
</tr>
<tr>
<td>stop</td>
<td>15, 70, 72</td>
</tr>
<tr>
<td>UPnP</td>
<td>12, 19, 24, 25</td>
</tr>
<tr>
<td>virtual network</td>
<td>10, 68</td>
</tr>
<tr>
<td>waterfall model</td>
<td>59</td>
</tr>
<tr>
<td>WLAN</td>
<td>68</td>
</tr>
<tr>
<td>wrapper</td>
<td>37</td>
</tr>
</tbody>
</table>