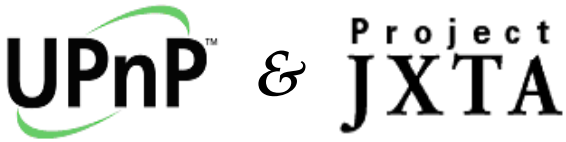
UPnP-JXTA bridging

Bogers, G.J.C.

Award date:
2004

UPnP – JXTA Bridging



Confidential

Master Thesis
Gertjan Bogers

UPnP – JXTA Bridging

Master Thesis

Author:	G.J.C. Bogers
Student ID:	431513
TU/e Department:	Mathematics and Computer Science
TU/e Research group:	System Architecture and Networking
Supervisors TU/e:	Dr. J.J. Lukkien Y. Mazuryk
Company:	Philips Research Eindhoven
Company Supervisors:	Ir. I.W.F. Paulussen Ing. W.H.G.M. van den Boomen
Graduation Period:	15 September 2003 till 30 June 2004

Abstract

Whether it is to share files, to share an Internet connection, or to play digital content throughout the house, networking has become part of everyday life for many home and small-business users. Despite this, networks have not gotten a whole lot easier to set up and to configure. This networked lifestyle increasingly demands more self-configuring networks that allow devices to easily join and leave networks and to learn about other connected devices. At the moment, multiple technologies are being developed to make networking-configuration hassles a thing of the past. Examples of these technologies are Universal Plug and Play (UPnP) and JXTA.

The graduation assignment is fulfilled as a result of two European Research projects, Spation and Share-It. Spation focuses on storing and retrieving information in a home network of connected Consumer Electronic (CE) devices and finds solutions to help the user to move, retrieve, and organize this information. Share-It allows consumers to publish and distribute content to other consumers. The project builds on existing technologies of peer-to-peer networks and broadband connections. Within Spation the applicability of UPnP is investigated. Share-It on the other hand uses JXTA as a building block.

This master thesis provides an architecture to connect both UPnP and JXTA together. Before connecting both technologies together the strengths and weaknesses of each technology will be discussed separately. As a guideline for the master thesis three usage scenarios are developed, which contain a number of scenario specific requirements. These requirements are needed to investigate which techniques can be used to connect both technologies together and which consequences a certain technique can have (for example at overall performance). After this an in-depth overview of the specific network requirements is given. At the end, an architecture proposal will be highlighted. Also the advantages and disadvantages with respect to the architecture proposal will be discussed.

Samenvatting

Of het nu gaat om het delen van bestanden, een internet verbinding of het afspelen van digitale content door het hele huis, netwerken behoren tegenwoordig voor veel mensen tot het alledaagse leven. Ondanks dit alles is het tot op heden niet veel gemakkelijker geworden om netwerken op te zetten en te configureren. Deze levensstijl vereist in steeds grotere mate zelf instelbare netwerken. Deze moeten ervoor zorgen dat apparaten op een gemakkelijke wijze aan het netwerk toegevoegd of van het netwerk verwijderd kunnen worden. Momenteel wordt er gewerkt aan meerdere technologieën om deze problematiek op te lossen. Voorbeelden hiervan zijn Universal Plug and Play (UPnP) en JXTA.

De afstudeeropdracht is uitgevoerd in het kader van twee verschillende Europese onderzoeksprojecten, te weten Spation en Share-It. Spation richt zich op het opslaan en het terugvinden van informatie in een thuis netwerk dat bestaat uit met elkaar verbonden consumenten elektronica (CE). Het geeft oplossingen om de gebruiker te helpen met het verplaatsen, terug vinden en organiseren van deze informatie. Share-It staat gebruikers toe om gegevens te publiceren en te distribueren. Dit gebeurt met behulp van reeds bestaande technologieën omtrent peer-to-peer netwerken en breedband verbindingen. Binnen het Spation project wordt de toepasbaarheid van UPnP onderzocht. Dit in tegenstelling tot Share-It, wat gebruik maakt van JXTA als communicatie-technologie.

Dit afstudeerverslag draagt een oplossing aan om beide technologieën met elkaar te verbinden. Echter voordat de verbinding gelegd kan worden zullen eerst alle sterke en zwakke punten van beide technologieën afzonderlijk besproken worden. Tevens zullen er een drietal scenario's worden opgezet die als richtlijn kunnen dienen om beide technologieën met elkaar te verbinden. Deze scenario's kunnen gebruikt worden om te onderzoeken wat de gevolgen zijn (bijvoorbeeld op de algehele prestatie) bij het gebruik van een bepaalde techniek. Ook zal er verder ingegaan worden op de specifieke netwerk vereisten. Als laatste zal een architectuur voorstel worden besproken met alle bijkomende voor- en nadelen.

Table of Contents

1. Introduction

All sorts of devices – Personal Computers (PCs), mobile phones, cameras, handheld computers and so on - are increasingly connecting to networks, and they are using a multitude of connectivity methods to do so. This trend increases the need for self-configuring networks that allow devices to easily and automatically join and leave networks and to learn about other connected devices. Home networks, automotive networks and similar environments demand new technologies that can automate device and service discovery and control, obviating the need to administer these networks.

This graduation assignment deals with two different technologies. The first one is Universal Plug and Play (UPnP) [1]. This technology enables pervasive peer-to-peer network connectivity of PCs of all form factors, intelligent appliances and wireless devices. It is a distributed, open networking architecture that leverages TCP/IP and Web technologies to enable seamless proximity networking in addition to control and transfer data among networked devices in the home, office, and everywhere in between.

The second technology is JXTA [2]. JXTA is a standards-based, peer-to-peer technology that supports collaboration and communication on networked devices, including cell phones, pagers, handhelds, PCs and servers. Sun Microsystems [3] released JXTA to the open-source community in April 2001. The technology runs across multiple platforms. JXTA is short for "Juxtapose", as in side by side. It is a recognition that peer-to-peer is juxtapose to client server or Web based computing—*"what is considered today's traditional computing model"* according to the Project JXTA Web site.

The graduation assignment has been carried out at Philips Research in Eindhoven. Specifically the assignment is done within the group Storage Systems and Applications.

1.1 Group: Storage Systems and Applications (SSA)

The group Storage Systems and Applications aims at innovations at the system level of storage devices to enable increasingly advanced applications of data storage, primarily in the home consumer environment.

This department considers storage systems in their context, which is determined by among other Input / Output (I/O) devices, networks and data processors. Now that all information, like audio and video has become digital, the target is to identify and exploit the multitude of options this creates for enhanced and new storage enabled applications.

Video recording is traditionally the most storage intensive application in the home and it naturally represents one of the main topics of the group. The recording, editing and management of compressed digital video in multiple streams on removable optical disks and fixed magnetic disks poses enormous challenges with respect to scheduling, defect management, data allocation and the like.

Further they witness an explosive growth of available content combined with enormous increases in storage capacities. More content means less value if the user cannot find its way through it. Data searching and retrieval mechanisms are studied in the group and integrated into the storage systems. Future storage systems all interact with networks, enabling among others distributed storage. This is also investigated in the group, both for the home environment and for mobile applications. Much of the output of the work finds its way into new Integrated Circuits (IC's) and software running on those IC's. System architecture and system integration are therefore key capabilities in the group.

1.2 Graduation Assignment

The group SSA of Philips Research in Eindhoven offers a graduation assignment on the topic of UPnP-JXTA bridging. Both technologies have their own advantages in their specific areas.

The graduation assignment is fulfilled as a result of two European Research projects, Spation and Share-It. The objective of Spation is to find solutions for the problem how to move, organize and retrieve information (music, video, pictures, documents, etc.) in a heterogeneous home network. Within Spation the applicability of UPnP in a home network has been investigated and how to build applications on top of it.

Share-It allows consumers to publish and distribute content to other consumers. The project builds on existing technologies of peer-to-peer networks and broadband connections. The devices establish an authenticated connection to exchange rights information to ensure that content is only distributed when rights allow it. The JXTA technology is used as a building block for this.

1.2.1 Problem Situation

It is possible to assume that in the near future all Consumer Electronic (CE) devices in the home will become interconnected. Users will be faced with the issues of storing and retrieving content in a distributed system. It is also interesting to share content on a peer-to-peer basis between homes. For in the home the UPnP middleware standard is used, the JXTA standard is used between homes.

For Philips both in-home and home-to-home networking is important. UPnP gets more and more momentum in the CE market and it looks like UPnP will be the de-facto standard in in-home networking for networked enabled CE-devices. JXTA provides a solution for home-to-home networking where UPnP is not applicable because of its scalability.

1.2.2 Assignment

The assignment is to investigate both UPnP and JXTA and try to develop a solution to bridge both technologies. The word bridging in this context can be misleading with respect to the architecture that is needed for this assignment. Bridging does not automatically imply that a hardware bridge is needed, but is more a general term to define the binding between two or more technologies.

1.2.3 Goal of the assignment

The goal of the assignment is to find an answer for the question, “how to bridge UPnP and JXTA”. To answer that question, both technologies need to be investigated to determine what these standards itself offer to solve this problem. At the end of the project the results need to be implemented. This should make it possible to show in a small demonstration the working of the bridge.

1.2.4 Report Structure

This Master Thesis gives an overview of the graduation assignment fulfilled as a result of the Spation and Share-It projects. The focus is on how the UPnP technology, used within the Spation project, can be connected (bridged) to the JXTA technology, used within the Share-It project. To reach this goal the following structure in the Master Thesis is used:

- Chapter 2 focuses on the specification of the graduation assignment;
- Chapter 3 contains the usage-scenarios which are used throughout the assignment as a guideline;
- Chapter 4 contains the basics of the UPnP standard;
- Chapter 5 contains the basics of the JXTA standard;
- Chapter 6 gives an in-depth overview of the network setup needed;
- Chapter 7 covers the details involved for the UPnP-JXTA architecture;
- Chapter 8 presents a list of conclusions and
- Chapter 9 discusses the future work.

2. Specification

In this chapter the operation boundaries of the graduation assignment are defined. This needs to be done because the graduation assignment has a relatively broad scope and needs to be limited. The first question that arises is how to bridge two different technologies? Referring to the problem situation, mentioned in the previous chapter, this question can be formulated more specifically as: how to bridge two UPnP home networks using JXTA? To answer this question many other questions can arise, like:

- What does bridging mean?
- What are the possibilities and limitations of both technologies?
- What do transparency and scalability mean for home networks?

To answer the first question, the dictionary is consulted to get the meaning of the word “bridge”. According to the dictionary it means:

Bridge A structure spanning and providing passage over a gap or barrier.

When in this explanation “structure” is replaced by “device” and the word “technical” is put in, the next sentence appears: a device spanning and providing passage over a technical gap or barrier. This already makes the link between the word bridge and the first question. But still bridging technologies has a broad scope and in the context of the graduation assignment it is still too vaguely described. To narrow this broad scope, a more technical definition needs to be consulted for the definition of bridging.

Bridging is also defined as:

Bridging allows devices to interoperate with devices that use other protocols.

This simple definition cover the entire range of systems in which networked devices can usefully be deployed. A key question is: what does interoperability mean in the context of the graduation assignment? According to the Institute of Electrical and Electronics Engineers (IEEE) [4] interoperability means:

Interoperability is the ability of two or more systems or components to exchange information and to use the information that has been exchanged.

Applying this definition to the graduation assignment reduces the scope to:

Use JXTA as a bridge to exchange information between two or more UPnP networks and use this information that has been exchanged.

A simple introduction to both technologies learns that UPnP is intended to be used for peer-to-peer (P2P) networking. Peer-to-peer networking is a serverless networking technology that allows several network devices to share resources and communicate directly with each other. Particularly devices are implemented as a control point or as a device. An important aspect of peer-to-peer networking is the ability to discover peers on the network. UPnP employs a broadcasting mechanism to do so. However, if the number of peers is relatively big, the traffic generated by the broadcasting is significant. Therefore, UPnP is limited with respect to scalability.

JXTA can be used for different kind of networks. It can use UDP broadcasting techniques and/or HTTP to discover other JXTA compliant devices. Another aspect of JXTA is that specifications of JXTA protocols are not bound to a specific transport method (e.g. UDP, HTTP). JTXA also contains a number of techniques to pass through firewalls and in this way to enhance its scalability.

With respect to the gradation assignment the UPnP network will be called the home-network, where JXTA will be used to make a connection between different home-networks. The home-network consists of a range of heterogeneous components. This means that the home-network is made up of different kind of devices. All these devices need to communicate with each other. For this interconnection multiple solutions are available, and the network in the home will be a mixture of solutions, such as wired network connections, wireless network connections, Bluetooth, etc. Combining these devices creates a network that allows users to control devices in the home. Together with JXTA as communication technology between home-networks figure 1 depicts a home-to-home network.

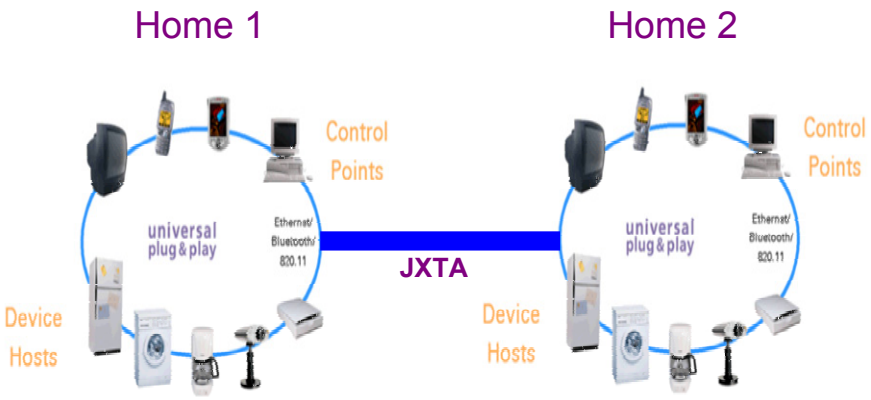


Figure 1: Home to home network

An important aspect of this configuration is to hide the fact that its processes and resources are physically distributed across multiple networks. A configuration that is able to present itself to users and applications as if it were only a single network is said to be transparent. To accomplish this kind of transparency the JXTA part needs to be configured in a way that it is possible to scale up the home-network (by connecting one or more other home networks). This is the point where scalability comes in. With this configuration the network is enlarged in such a way that multiple home networks are connected together as if it is one (big) network.

In figure 2 an overview of the network stacks is given. On the left and the middle the OSI and TCP/IP protocol stacks are shown. Compared to the OSI stack the TCP/IP stack is simpler, i.e. the presentation and session layer are not used. Originally the OSI and TCP/IP application layer consisted of protocols like TELNET, FTP, etc. Currently applications moved to a higher level and make use of an extra layer called middleware. The right side of the picture makes clear that middleware in this case is built on top of the used network stack and includes functionality that applications have in common. Examples of applications that can be placed on top of the middleware layer are video streaming, browsing remote databases, or showing a user interface on a display in the house. Functions that middleware stacks typically take care of are for example, easily adding of new devices with new services. Depending on the combination of the physical network connection, the network stack and the middleware stack, the home network can handle adding of these new devices and services.

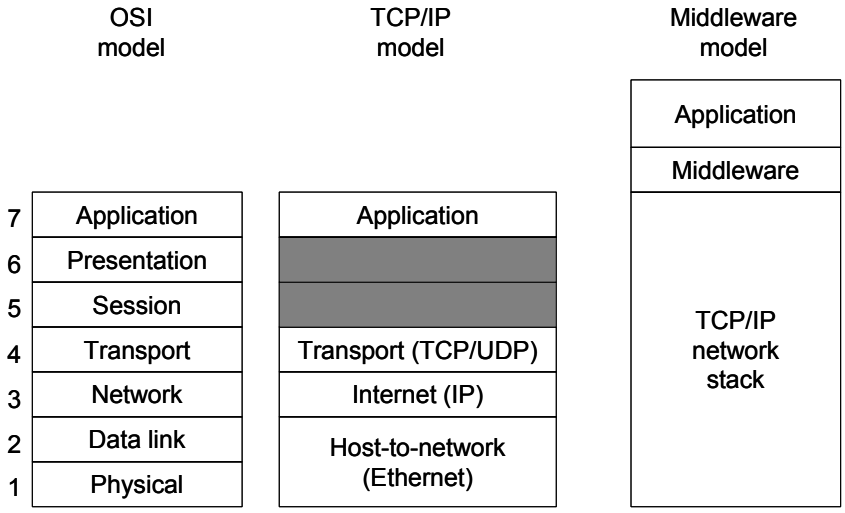


Figure 2: Comparison between the OSI model and the TCP/IP model

JXTA is designed to be independent of programming languages, system platforms, and networking platforms. It is designed to provide a layer on top of which services and applications are built. It is a set of simple, open peer-to-peer protocols that enable any device on the network to communicate, collaborate, and share resources.

JXTA protocols establish a virtual network on top of existing networks, hiding their underlying complexity. Unlike traditional client-server networks, which rely on a centralized point of message transport, a JXTA virtual network allows any peer to interact with any other peer or resource directly, even across firewalls or different physical network boundaries. Thus, access to the resources of the network is not limited by platform incompatibilities or the constraints of the hierarchical client-server architecture (figure 3).

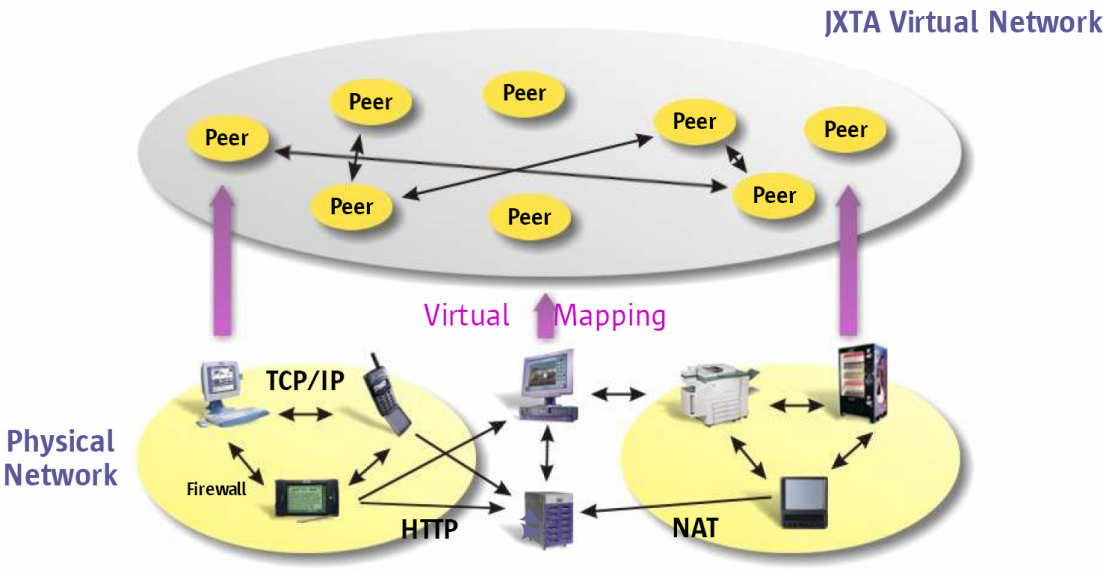


Figure 3: JXTA Virtual Network

In this chapter it became clear that there are several important aspects where the graduation assignment should focus on. The most important aspects are:

- Home network bridging;
- Heterogeneity;
- Scalability;
- Transparency.

By addressing these important aspects, some insights where gained about the boundary conditions of the graduation assignment. These operation boundaries bring focus in the graduation assignment and help to solve the main problem of how to set up a bridge between two or more UPnP home networks using JXTA as communication technology.

3. Usage Scenarios

Three usage scenarios are introduced to limit the domain of the graduation assignment any further. Each of these scenarios gives another description of a home-to-home network. These scenarios are short stories of envisioned real life situations, which describe a user group in a specific contextual setting interacting with specific devices to fulfill specific tasks and intentions. It illustrates in short the usage, the functionality and purpose of novel devices and services in daily life. It does not prescribe literally the research issues that should be realized or what the final result of the graduation assignment should look like.

The *analysis* of a scenario provides room for any discussion and aligns expectations and mindsets of the different aspects of the assignment. The goal of analyzing the scenarios is to reveal how they could be interpreted. This results in user-requirements and a more technical analysis.

The *user-requirements* specify the criteria that should be met to make the scenario usable and useful. The *technical analysis* specifies a more thorough analysis of the different aspects relevant for the different scenarios.

There are some user requirements, which apply for all usage scenarios. These general user requirements refer to the contextual setting and the intended user groups in which all the scenarios are situated.

Intended User Group
The intended users are families. The age of the intended users range from 3-80 years. Home usage implicates and in fact even requires that users do not need to be experienced users of (new) electronic devices or computers. The intended user group does not have to be technology-minded.

Contextual Setting
The context-of-use is multiple networked environments in and between different (home) networks. Users can go everywhere in the house without losing control upon the networked environment. Even outside the house it is possible to set up a connection to the home network. Related issues to mobility concern the portability of devices and the location and security of data.

For the usage scenarios a handheld device mainly establishes the interaction with the home environment. This device has a direct connection to the network (wired or wireless) and is capable of controlling the other devices available within the network.

Users do not want to be bothered with the need to search every device separately to find the content of interest but want to have one central place where they can access their content. The user can access in this central place all the content that is distributed over a lot of devices within the home. This central place should shield the actual storage locations for the user. When the access to the content is centralized and the real storage location of the content is shielded, users get the idea that they access one virtual storage place. From now on this central place will be called the *home cabinet*.

To make sure that all the content in the home network can be accessed in this home cabinet, searching for content descriptions is needed because the content can be stored at different locations in the network. These content descriptions usually referred to as metadata, need to be combined, stored and retrieved. This prevents that users need to interact directly with the device containing the content.

General User Requirements
The following user requirements are essential for all scenarios:

U1 *Transparent use*
Performing operations that are yet unknown to many users such as those that extend over multiple devices (e.g. transferring data from one device to another) should be self-explanatory or guided. After the first "getting-to-know" phase, this should go quickly and hassle-free.

U2 *Transparent control*
The interaction of home devices should be transparent, meaning that users are able to perceive the most effective and efficient ways to complete their tasks successfully at a glance. The user is in control.

U3 *Identification*
When using one of the handheld devices, personal settings should be easy to set or select. For personalization purposes, the user can be automatically and easily identified. The user should only have to perform a minimum number of actions to make him/her self known on all systems (e.g. the neighbor's network and his/her own network).

U4 *Easy to learn*
Since first time users attempt to operate a device immediately without the aid of instructions, home devices need to be easy to learn, at least the basic operations.

3.1 Scenario 1: Listening Music

Kevin is a music addict, just like his neighbor Peter. Kevin has already stored all of his music on the home cabinet, though the music is located at different places in the house. He always has his own handheld with him to listen to the music and to switch between songs.

One day he buys a new rock-cd on the Internet. After he has paid the cd he is able to download the content to his home network. When he received the whole cd it will be automatically put on the home cabinet to make it available for everybody in the house. Then his friend Peter calls and ask him to come over.

At Peter's house Kevin asks Peter if he wants to listen to the new cd he has just downloaded. Of course Peter wants to do that. Kevin takes his handheld and browses to the new music, which just has been made available on his home cabinet. He selects a song from the new cd and they both listen to the music on Peter's audio system.

3.1.1 Analysis

The main focus of this scenario is on audio retrieval, content compilation and content sharing:

- *"...all of his music on the home cabinet, though the music is located at different places in the house."* : distribution of the content among interconnected devices.
- *"...whole cd, it will be automatically put on the home cabinet..."* : automatic or semiautomatic method to convert legacy material in digital format and an Internet service which provide the music.
- *"...Kevin takes his handheld and browses to the new music, which just has been made available on his home-cabinet."* : interconnection between houses and devices (authentication, authorization and security), sharing of content.

3.1.2 User Requirements

U1 *Knowledge*
Knowledge of content descriptions (e.g. titles, artist names) is hard to make explicit in search and retrieval, unless the information is already known (e.g. famous artists). Interaction should support recognition of information rather than the need of recollection and recall.

U2 *Skills*
The user does not need to have special skills to retrieve and share content between different locations.

3.2 Scenario 2: Synchronization

Most of the time dad has a very busy schedule, but sometimes he comes back early from his work. Just like today. His wife is still at work and the kids are not back from school yet. When he is preparing the meal for his wife and two children he watches his appointments for that evening on TV. Normally he would do this on his handheld, but this time he left it in his car. This is not so unusual for dad because he often forgets it.

He reads on the TV that he has an appointment at 8 pm. with his colleague. They will meet in the pub. After dinner with his wife and children he cleans up a little bit and goes to the pub. Of course this time he takes the handheld with him.

When dad meets his colleague they start talking and having fun. After some time they decide to go home but first they make another appointment. They pick a date and time. Dad decides to put it directly in his handheld to make sure he will not forget. Even if he forgets his handheld again, just like today, he will be notified because all the information is also stored automatically on his home cabinet.

3.2.1 Analysis

The main focus of this scenario is on content sharing, interchangeability and interoperability:

- *"...he watches his appointments for that evening on TV"* : interoperability between devices.
- *"...Of course this time he takes the handheld with him"* : interaction with handheld devices.
- *"...Dad decides to put it directly in his handheld to make sure he won't forget. Even if he forgets his handheld again, just like today, he will be notified because all the information is also stored automatically on his home cabinet."* : interconnection and synchronization between multiple networks, content sharing, dynamic network.

3.2.2 User Requirements

U1 *Security*
Every time the handheld establish a connection to a network (in-home or out-home) the data sent to the network must be confidential. This is especially important for private data.

U2 *Skills*
The user does not need to have special skills to retrieve and share content between different devices and locations.

3.3 Scenario 3: IR-device

Last year the whole family went on a holiday to France. Of course dad took for this occasion a photo camera with him to shoot some nice pictures of all the beautiful things they wanted to see. It was a trip they will never forget.

When they are back home dad wants to see all the nice pictures as a slide show on his computer so he can make a nice presentation of it. When he brings the camera inside the home and turns on his computer he is able to see the pictures as a slide show. To view the slide show he uses a remote control to navigate through the pictures.

A couple of days later they visit the neighbors. After a lot of talking about the holiday the neighbors want to see some pictures. Gladly dad has expected this and he already has his handheld with him. He takes his handheld and starts the slideshow, which is stored somewhere on his home-network. When the first picture appears on the television he uses some remote control to navigate through the pictures.

3.3.1 Analysis

The main focus of this scenario is on content sharing, metadata, interoperability and interchangeability:

- *"...When he brings the camera inside the home and turns on his computer he is able to see the pictures as a slide show."* : distribution of the content among interconnected devices.
- *"...To view the slide show he uses a remote control to navigate through the pictures."* : Interaction with a connected device.
- *"...He takes his handheld and starts the slideshow..."* : interaction with a handheld device, dynamic network.
- *"...which is stored somewhere on his home-network."* : search and retrieval of heterogeneous multimedia content, metadata.
- *"...When the first picture appears on the television he uses some remote control to navigate through the pictures."* : interaction with multiple interconnected devices.

3.3.2 User Requirements

U1 *Easy connection*
Devices may come and go. They can be turned on or off or taken from or brought into the network. Devices must easily 'find each other and the services available' without any user-assistance.

U2 *Transparency*
From a user point of view, it should appear as if storage and the network configuration of devices are centrally organized. However, in reality, this will be done in a distributed manner, since devices can come and go in the network and a registering facility should take notice of that.

U3 *Interconnection*
Other devices also connected to the network can use and manage the capabilities of other devices, without explicit user-interaction.

3.4 Technical Analysis

Till now the different scenarios have been analyzed and some user requirements are defined. The most important aspects of the analysis will now be discussed in more detail. Furthermore the infrastructure belonging to these specific aspects will be discussed. Because of the overlap between the different scenarios the most important aspects (applicable to all scenarios) are put in one table:

	Detailed	**Infrastructure**
Distributed Home Cabinet	A device that contains the information about the location of all content available in the home network (even if devices are not connected)	Device (e.g. PC) connected to the home network, capable of handling different requests from other devices.
Legacy Conversion	Conversion method to convert material (e.g. music) to digital format and vice versa.	Some device (hardware) with the functionality integrated to convert legacy material into digital format.
Interconnection between homes	The connection of two or more different (home) networks, both capable of handling the authentication, authorization and security issues needed to set up a connection.	Two or more distinct networks, which are both connected to the Internet through a gateway or router.
Interoperability	Sharing of content between different devices, both connected to one or more (home) networks.	Two or more devices directly connected (wired or wireless) to a network, capable to announce its presence, share services and content.
Home Network	Two or more devices interconnected to form a local area network (LAN).	Two or more devices directly connected (wired of wireless) to a network. The network could be managed, i.e. a DHCP [5] server is available, or the network is unmanaged.

Table 1: Technical Analysis Overview

The definitions defined in this overview can be used throughout the document as a reference.

4. UPnP Basics

As mentioned in chapter 2 – Specification, UPnP operates at the middleware layer. A middleware layer takes care that e.g. adding new devices with new services is easy because this is standardized. How well a home network can handle the adding of new devices and services depends on the combination of the physical network connection, the network stack and the middleware stack.

The UPnP Forum is, as stated on their website, a group of companies and individuals across multiple industries that intend to play a leading role in the authoring of specifications for UPnP devices and services. Formed in June 1999, the Forum is a non-profit association of more than 687 consumer electronics, computing, home automation, home security, appliances, printing, photography, computer networking, mobile products and other leading companies working together in an open process to design schema and protocol standards for the UPnP initiative.

The goals of the Forum are to allow devices to connect seamlessly and to simplify the implementation of networks in the home and corporate environments. The Forum will achieve this by defining and publishing UPnP device control protocols built upon open, Internet-based communication standards.

The aim of UPnP is to create a plug and play network where no configuration is needed; in this way users do not need to have any network knowledge. Just plug in your device and use it. The device tells the network what it can do and it learns from the other available devices what they can do. This makes it possible for devices to use each other's functionality. As easy as it is to plug in a device, it can also be removed from the network without causing major trouble in the network. Some characteristics of UPnP are:

- It is an open standard based on widely known Internet technology.
- It offers a good balance between standardization and implementation freedom.
- It has relatively low implementation complexity and cost.

UPnP defines a way for multiple machines to interact with each other over a network. It is a stack of protocols that starts at the IP level, where it defines how to receive an IP address. Most of UPnP is based on existing standards such as TCP/IP, HTTP, XML, SSDP, SOAP etc. In a UPnP based network two main device categories are distinguished, devices and control points.

A control point can be implemented on a device, such as a handheld device, from where control needs to take place. The control point can search for devices in the network that it can control. The control point might look for a certain type of device, such as a CD player. When it finds a device it is interested in, it can query which services the device offers. One of these services could be a rendering control service, which contains functions like setting the volume, muting the sound etc.

The control point could query the current volume setting and display it in the user interface. When the user changes the volume, the control point can send a command to the device to set the volume to the new value. When multiple control points are used, these control points can reflect the change in their user interfaces, since UPnP allows sending event notifications for this purpose. The control point coordinates these devices using UPnP *action commands* to initialize, configure and to make devices ready to transfer content from amongst each other. The devices do not interact directly with each other and use a non-UPnP "out-of-band" communication protocol [6]. The control point is not directly involved in the actual transfer of the content. In Figure 4 an example of UPnP interaction between standardized devices is shown. The control point initiates the transfer from a *MediaServer* acting as a source to a *MediaRenderer* acting as a sink.

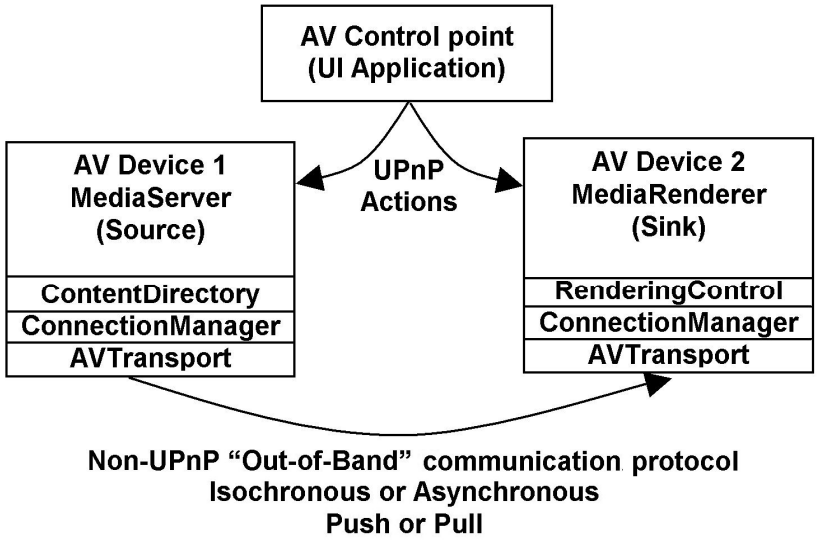


Figure 4: Interaction between devices in UPnP.

The previous small explanation introduces the basic functionality of UPnP. In the following sections each aspect will be described in detail.

4.1 UPnP Architecture

UPnP is set up in different levels, stacked on top of each other. All levels make use of existing technology as far as possible (Figure 5). As shown in the figure, UPnP lays over IP. Depending on the actions being performed, one or another higher level protocol is used (SSDP, HTTP, GENA or SOAP). After these standard protocols, there are two additional layers that give information about the specific type of device. The UPnP Device Architecture [7] defines a schema or template for creating device and service descriptions for any device or service type. Individual committees agree on standardization on various device and service types and create a template for each individual device or service type. Finally, a vendor fills in this template with information specific to the device or service, such as the device name, model number, manufacturer name and an URL to the service description.

Then the data is encapsulated in UPnP specific protocols. The required UPnP specific information is inserted into all messages before they are formatted using the Simple Service Discovery Protocol (SSDP), General Event Notification Architecture (GENA), and Simple Object Access Protocol (SOAP). Which format to use depends on the different phases of the UPnP architecture, as will be discussed later on. Every message is delivered via HTTP, HTTPU, or HTTPMU, either a Unicast or Multicast variety of HTTP running over UDP, or the standard HTTP running over TCP. Ultimately, all messages above are delivered over IP.

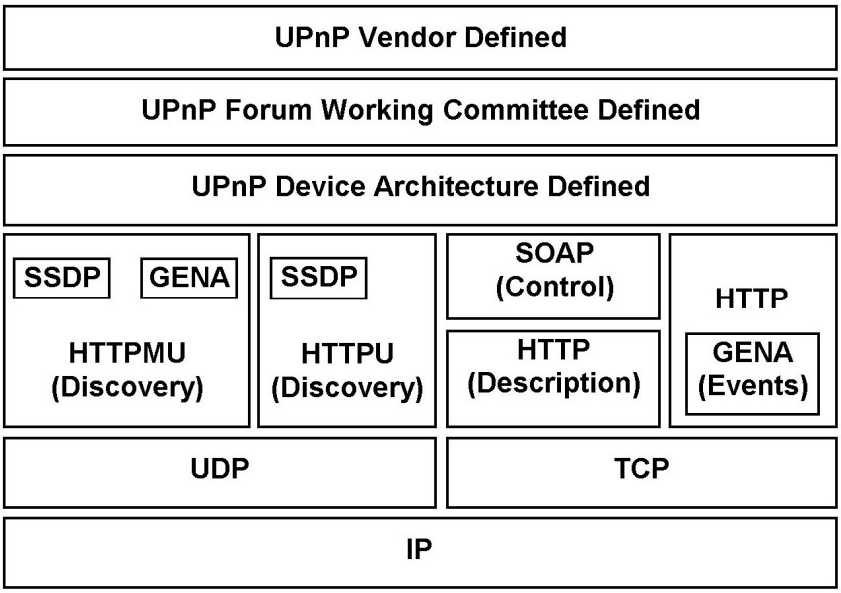


Figure 5: UPnP protocol stack

Discussed so far, in a network of UPnP devices, control points can discover devices, invoke actions on a device's services, and subscribe to events. Devices, on the other hand, respond to invoked actions and send events when state variables change. To make this basic functionality possible, all UPnP devices follow the same basic pattern, or phases, of operation:

- *Addressing*, devices and control points obtain IP addresses using DHCP (Dynamic Host Configuration Protocol) or AUTO-IP [8] to participate in the network.
- *Discovery*, when a device is connected to the network it sends an advertisement to inform the control points its availability in the network. When a control point is connected to the network it can search for devices of interest. Both use SSDP.
- *Description*, control points learn details about devices and their services using HTTP.
- *Control*, control points invoke service actions on devices using HTTP and SOAP.
- *Eventing*, devices notify control points of changes in their state using HTTP and GENA.
- *Presentation*, Control points control devices and/or view a device status using an HTML User Interface.

Together, these steps define how all UPnP devices behave within a network. Figure 6 shows the UPnP phases and their dependencies. A device first acquires an address, and then is able to provide a description of its capabilities to control points that have discovered it. Once a control point has discovered a device and retrieved the description of its services, it is able to either control the device, request that it receive notification of events, or an administrator may manually monitor or configure the device.

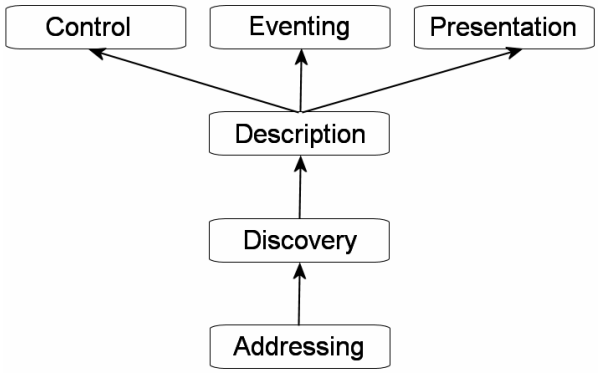


Figure 6: UPnP operation phases

The following sections will give an in-depth overview of each operation phase.

4.1.1 Addressing

The foundation for UPnP networking is *addressing*, the process by which a device automatically acquires an IP address. As the first step of UPnP, addressing allows devices to join the network and to communicate with other UPnP devices. The addressing protocols built in to UPnP devices allow them to join an IP network dynamically and to acquire an address without user configuration.

Addressing takes into account whether a device is operating in an unmanaged or managed network. An unmanaged, or *ad-hoc*, network is a network where there are no pre-existing infrastructure devices (or they are currently inoperable) and the network nodes themselves make up the network. A managed, or infrastructure network, allows devices to acquire an IP address from a DHCP server on the network.

After the addressing phase all the devices and control points available in the network have an IP address and can participate in the network.

4.1.2 Discovery

The discovery process enables control points to find devices and services and retrieve information about them. Also, once a device has acquired an IP-address, the device may advertise itself and its services within the network. Devices include a URL for their device description document in their advertisements and discovery responses. The URL provides control points with the information they need to retrieve the device and service descriptions. This enables the control points to learn all about the device and the services it offers.

In UPnP the devices will not monitor each other's descriptions, i.e. devices are not aware of each other. Control points, however, can detect the presence of devices, and control these devices, subscribe to events sourced by the device's services, or retrieve the device's presentation page.

4.1.3 Description

The next step in UPnP networking is description. After a control point has discovered a device, the control point still knows very little about the device. For the control point to learn more about the device and its capabilities, or to interact with the device, the control point must retrieve the device's description from the URL provided by the device in the discovery message. This URL is typically hosted on the physical device itself, but need not be.

Devices may contain other, logical devices and services. The UPnP description for a device is expressed in XML [9] and includes vendor-specific, manufacturer information including the model name and number, serial number, manufacturer name, etc. The description also includes a list of any embedded devices or services, as well as URLs for control, eventing and presentation. After the description phase, the control point can start one of the three following phases: control, eventing or presentation.

4.1.4 Control

After a control point has retrieved a description of the device, the control point has the essentials for device control. To learn more about the service, a control point must retrieve a detailed UPnP description for each service. The description for a service is also expressed in XML and includes a list of the commands, or actions, the service responds to, and parameters or arguments, for each action. The description for a service also includes a list of variables; these variables model the state of the service at run time, and are described in terms of their data type, range, and event characteristics.

To control a device, a control point sends an action request to a device's service. To do this, a control point sends a suitable control message to the control URL for the service (provided in the device description). Control messages are also expressed in XML using SOAP. A control message can be compared to a remote procedure call. The requested action is sent to the device; the device performs the action, and informs the control point of its result. Typically the actions are initiated by the user from a user interface.

4.1.5 Eventing

A UPnP description for a service includes a list of actions the service responds to and a list of variables that model the state of the service at run time. The service publishes updates when these variables change, and a control point may subscribe to receive this information.

The service publishes updates by sending event messages. Event messages contain the names of more state variables and the current value of those variables.

A special initial event message is sent when a control point first subscribes; this event message contains the names and values for all evented variables and allows the subscriber to initialize its model of the state of the service.

To support multiple control points, all subscribers are sent all event messages, subscribers receive event messages for all evented variables, and event messages are sent no matter why the state variable changed (in response to an action request or due to a state change).

4.1.6 Presentation

Devices can be controlled by implementing a graphical user interface in the control point. In this case different implementations need to be made for different devices. Alternatively a device can present an HTML page with information and controls of the device. In this way any control point can control any device that presents its controls using an HTML page.

4.2 UPnP devices and services

In the previous sections it was already mentioned that UPnP distinguishes between devices and control points. Control points are used to control devices and are always in control, devices cannot directly communicate to each other without an action from a control point. The following sections describe in more detail how devices can be further subdivided and which services these devices can offer. In UPnP, devices such as printers, scanners and AV devices are standardized. In the remainder of this section the focus will be on the AV devices.

The AV devices are divided into media servers and media renderers. The servers supply content over the network to renderers such as TVs, CD players etc. It is important to note that a UPnP device does not need to match exactly with a physical device. Often physical devices are described as multiple UPnP devices. A TV, for example, could be a pure media renderer device. The TV might have a digital input, such as a network connection or IEEE1394 that is connected to the PC. The PC in that case should implement the media server. When the TV has storage (e.g. a hard disk), it can implement a media renderer device as well as a media server device, to also allow other devices in the network to make use of the content stored (on the hard disk) in the TV. The control point can be implemented on the PC, the TV, or a third device (for example a handheld). Multiple control points are possible as well. When a control point is implemented on the PC, it allows the user to select a video file to play on the TV, change the volume, the brightness etc. When a control point is implemented in the TV, it allows the user to select e.g. a video file from the PC using the TVs normal remote control selecting from a list of titles that are displayed via an on screen display.

Each UPnP device can offer an arbitrary number of services. These services offer one or more functions. Functions are associated to so-called state variables, and most functions are intended to set or get the value of a state variable. To structure the amount of services a UPnP device offers, a device can contain embedded devices with their services. The device containing the embedded devices is called the root device.

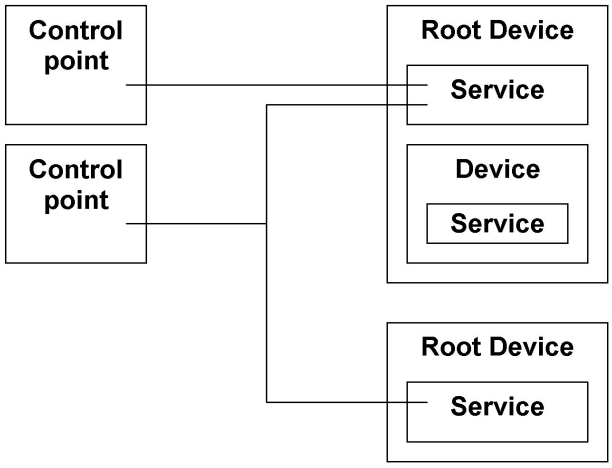


Figure 7: Structure of UPnP

For control points it is possible to search for services on the network, allowing them to search for a specific service. Additionally, control points can subscribe to receive event notifications when a certain state-variable changes. Figure 7 shows the possibilities of structuring UPnP functions in the network.

4.2.1 Media Renderer Device

The Media renderer device is a standardized UPnP device. It supports exchange of information of file formats and protocols to negotiate with a server how content should be transferred. It offers functions to alter rendering characteristics and it offers functions to control the flow of content. Three services with accompanying functions and state-variables have been standardized to support this functionality.

These services are called:

- Rendering Control
- Connection Manager
- AVTransport Service (optional)

4.2.2 Media Rendering Control

The rendering control offers functions to control how content is rendered. This includes functions such as changing the volume, checking the current volume setting, muting the audio, adapting contrast and brightness etc. In total a list of 35 functions have been described in the media renderer standard. These functions are associated with state variables, and in general the functions can be divided into functions to set values, and functions to get values.

4.2.3 Connection Manager

The Connection manager service is responsible for setting up and destroying connections between devices. When a control point wants to setup a connection between a media server and a media renderer, it searches for the connection managers of the server and renderer. One of the functions of the connection manager is to supply a list of supported protocols and formats. The control point needs to match the protocols and formats of the server and the renderer with each other and select an appropriate one. Then it can inform each device, and from thereon the server and the renderer can send and receive data without any intervention from a control point. Additionally the connection manager service offers an action to allow a control point to terminate a connection and free the used resources.

Evented variables of the connection manager allow control points to be notified when the list of supported protocols and when the current number of active connections changes.

AV Transport Service
The AV-Transport Service enables control over the transport of audio and video streams. It is used to control streams from CD players, mp3 players, DVD recorders etc. It includes controls such as starting, pausing, and stopping playback, and setting the playback speed. Additionally, it allows for requesting information about the medium type, the number of tracks, current track, etc.

Media Server device
The media server standardizes how devices like CD players, DVD players, mp3 servers, Personal video recorders etc. acting as media servers in the network can be addressed. Additionally, the PC can also implement this service allowing the PC to act as a service that can supply content to, for example, a TV or an mp3 player.

The media server can contain three services:

- Content Directory Service
- Connection Manager
- AVTransport Service (optional)

Content Directory Service
The connection manager and the AV Transport Service have been described in the previous section. The content directory service makes it possible to search for content. The content directory service is basically a service, which provides various views of the stored content. Using detailed descriptions of the content a search query can return a set of content items. Additionally, organization of content is supported via containers, which can be used to cluster content similar to directories or folders.

5. JXTA Basics

Project JXTA was originally conceived by Sun Microsystems and designed with the participation of a small number of experts from academic institutions and industry. The project defined a set of objectives based on what was perceived as shortcomings in many peer-to-peer systems, existing or under development.

Interoperability
Many peer-to-peer systems are built for delivering a single type of services. For example, Napster once provided music file sharing, Gnutella provides generic file sharing, and AIM provides instant messaging. Given the diverse characteristics of these services and the lack of a common underlying P2P infrastructure, each P2P software vendor tends to create incompatible systems. This means each vendor creates its own P2P user community, duplicating efforts in creating software and system primitives commonly used by all P2P systems. Moreover, for a peer to participate in multiple communities organized by different P2P implementations, the peer must support multiple implementations, each for a distinct P2P system or community, and serve as the aggregation point.

Project JXTA aims to bring to the P2P world what HTTP and the browser brought to the Internet.

Platform Independence
Many P2P systems today offer their features or services through a set of Application Program Interfaces (APIs) that are delivered on a particular operating system using a specific networking protocol. For example, one system might offer a set of C++ APIs, with the system initially running only on Windows, over TCP/IP, while another system offers a combination of C and Java APIs, running on a variety of UNIX systems, over TCP/IP but also requiring HTTP. A P2P developer is then forced to choose which set of APIs to program to, and consequently, which set of P2P customers to target.

Because there is little hope that the two systems will interoperate, and if developers want to offer the same service to both communities, the same service needs to be developed twice for two P2P platforms or develop a bridge system between them. Both approaches are inefficient and impractical considering the dozens of P2P platforms in existence. JXTA technology is designed to be independent of preferred programming languages, development environments, or deployment platforms.

Ubiquity
JXTA technology is designed to be implementable on every device, including sensors, consumer electronics, PDAs, appliances, network routers, desktop computers, data-center servers, and storage systems.

5.1 JXTA Architecture

P2P is the solution to a straightforward question: “How to connect a set of devices in such a way that it is possible to share information, resources, and services?” On the surface, it seems a simple question, but to answer it properly requires some basic knowledge about P2P networking.

The following sections define the basic terminology of P2P networking. These sections provide definitions that use the JXTA terminology but omit JXTA-specific implementation details.

5.1.1 Peers

A peer is a node on a P2P network that forms the fundamental processing unit of any P2P solution. This limited definition discounts the possibility that a peer might be an application distributed over several machines or that a peer might be a smaller device, such as a PDA, that connects to a network indirectly, such as via a synching cradle. A single machine might even be responsible for running multiple peer instances.

There exits three possible types of peers in any P2P network:

- Simple peers;
- Rendezvous peers;
- Router peers.

Each peer on the network can act as one or more types of a peer, with each type defining a different set of responsibilities for the peer to the P2P network as a whole.

Simple Peers
A simple peer is designed to serve a single end user, allowing that user to provide services from his device and consuming services provided by other peers on the network.

Rendezvous Peers
Taken literally, a rendezvous is a gathering or meeting place; in P2P, a rendezvous peer provides peers with a network location to use to discover other peers and peer resources. A rendezvous peer can augment its capabilities by caching information on peers for future use or by forwarding discovery requests to other rendezvous peers.

Router Peers
A router peer provides a mechanism for peers to communicate with other peers separated from the network by firewall or Network Address Translation (NAT) equipment. A router peer provides a go-between that peers outside the firewall can use to communicate with a peer behind the firewall, and vice versa.

5.1.2 Peer Groups

Before JXTA, the proprietary and specialized nature of P2P solutions and their associated protocols divided the usage of the network space according to the application. Consider a P2P system in which all clients can speak the same set of protocols, as is possible in JXTA, the concept of a peer group is necessary to subdivide the network space. A peer group is defined as a set of peers formed to serve a common interest or goal dictated by the peers involved. Peer groups can provide services to their member peers that are not accessible by other peers in the P2P network.

Peer groups divide the P2P network into groups of peers with common goals based on the following:

- The application they want to collaborate on as a group;
- The security requirements of the peers involved;
- The need for status information on members of the group.

Peer group members can provide redundant access to a service, ensuring that a service is always available to a peer group as long as at least one member is providing the service.

5.1.3 Network Transport

To exchange data, peers must employ some type of mechanism to handle the transmission of data over the network. This layer, called the *network transport layer*, is responsible for all aspects of data transmission, including breaking the data into manageable packets, adding appropriate headers to a packet, etc. The concept of a network transport in P2P can be broken into three constitute parts:

- **Endpoints:** The initial source or final destination of any piece of data being transmitted over the network;
- **Pipes:** Unidirectional, asynchronous, virtual communication channels connecting two or more endpoints;
- **Messages:** Containers for data being transmitted over a pipe from one endpoint to another.

Notice that a pipe provides communication in only one direction, thus requiring two pipes to achieve two-way communication between two peers.

5.1.4 Services

Services provide functionality that peers can engage to perform useful actions on a remote peer. The work can be transferring a file, providing status information, etc. Services are the motivation for gathering devices into a P2P network; without services, there is no P2P network. Services can be divided into two categories:

- **Peer services:** functionality offered by a particular peer on the network to other peers;
- **Peer group services:** functionality offered by a peer group to members of the peer group.

Most of the functionality required to create and maintain a P2P network, such as the underlying protocols required to find peers and resources, could also be considered services.

5.1.5 Advertisements

An advertisement can be defined as a structured representation of an entity, service, or resource made available by a peer or peer group as part of a P2P network. All the building blocks discussed so far can be described by advertisements, including peers, peer groups, pipes, endpoints, etc. Advertisements can be used to simplify the task of organizing P2P networks.

5.2 JXTA Protocols

JXTA defines a series of message formats, or protocols, for communication between peers. Peers use these protocols to discover each other, advertise and discover network resources, and to communicate and route messages. Figure 8 gives an overview of the JXTA protocol stack.

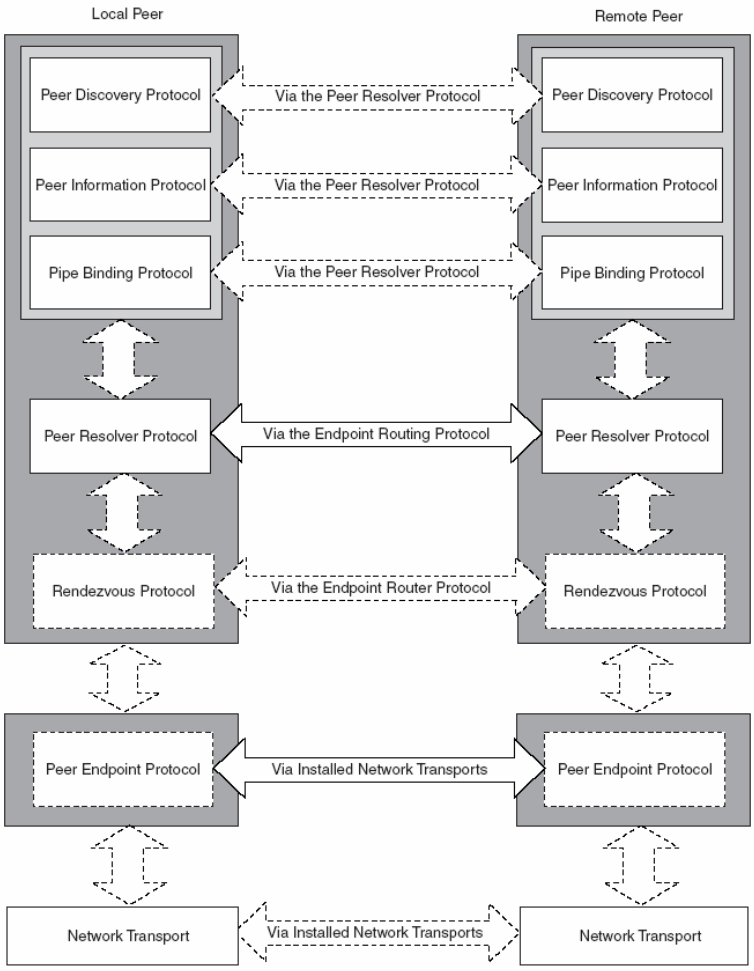


Figure 8: JXTA Protocol Stack

A set of six protocols were designed. Each of these protocols addresses exactly one fundamental aspect of P2P networking. Each protocol conversation is divided into a portion conducted by the local peer and another portion conducted by the remote peer. The local peer's half of the protocol is responsible for generating messages and sending them to the remote peer. The remote peer's half of the protocol is responsible for handling the incoming message and processing the message to perform a task.

Each protocol is semi-independent of the others. A peer can select to implement only a subset of the protocols to provide functionality, while relying on pre-specified behavior to eliminate the need for a protocol. For example, a peer could rely on a preconfigured set of router peers and, therefore, would not require an implementation of the Endpoint Routing Protocol (ERP). However, the protocols are not entirely independent of each other because each layer in the JXTA protocol stack (figure 8) depends on the layer below to provide connectivity to other peers.

5.3 Logical Layers of JXTA

The JXTA platform can be broken into three layers, as shown in figure 9.

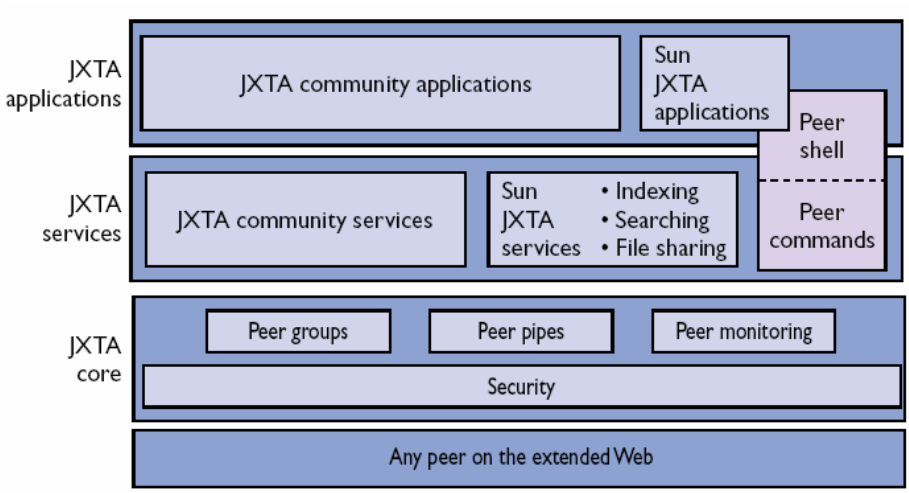


Figure 9: JXTA Software Architecture

Each layer builds on the capabilities of the layer below, adding functionality and behavioral complexity.

5.3.1 Core Layer

The core layer provides the elements that are absolutely essential to every P2P solution. Ideally, the elements of this layer are shared by all P2P solutions. The elements of the core layer are:

- Peers
- Peer groups
- Network transport (pipes, endpoints, messages)
- Advertisements, etc.

The core layer includes the six main protocols provided by JXTA. Although, these protocols are implemented as services, they are located in the platform layer and are designated as core services to distinguish them from the service solutions of the services layer.

The core layer, as its name suggests, is the fundamental core of the JXTA solution. All other aspects of a JXTA P2P solution are in the services or application layers build on this layer to provide functionality.

5.3.2 Services Layer

The services layer provides network services that are desirable but not necessarily a part of every P2P solution. These services implement functionality that might be incorporated into several different P2P applications, such as the following:

- Searching for resources on a peer;
- Sharing documents from a peer;
- Performing peer authentication.

The services layer encompasses additional functionality that is being built by the JXTA community. Services built on top of the JXTA platform provide specific capabilities that are required by a variety of P2P applications and can be combined to form a complete P2P solution.

5.3.3 Applications Layer

The applications layer builds on the capabilities of the services layer to provide the common P2P applications that are known nowadays, such as instant messaging. No application might encompass only a single service or aggregate several services. Therefore it is difficult sometimes to determine what constitutes an application and what constitutes a service.

6. Network Setup

According to the specification, there need to be two (or more) UPnP networks that must be connected using the JXTA protocol. This setup requires some specific hardware. Networks in general may be interconnected in various ways, through repeaters, bridges, routers and gateways (or a combination of these).

6.1 Interconnection

A summary of these configurations is given and also how to interpret their function.

- ***Repeaters***
  Repeaters operate at the physical layer. Conventionally, they operate bit-by-bit receiving the input signal, regenerating the signal and emitting it. In the context of wired to wireless (or vice versa) repeaters, this may imply changing the encoding scheme. However, repeaters operate transparently to the protocols above the physical layer.

  In the last decade, repeaters with multiple ports (called hubs) have become available. Simple hubs are examples of repeaters. There exist however more sophisticated hubs that act as bridges.

- ***Bridges***
  A bridge is a data link layer (DLL) relay. A bridge receives a complete media access control (MAC) or logical link control (LLC) frame, checks it and possibly forwards it on the other side(s). Contrary to repeaters, bridges filter the information that is relayed from one LAN to another. The different types of interconnected LANs lead to various categories of bridges:
  - *Pass-through bridges* can be used when the LANs on both sides offer identical data link layer functions and addressing. The frames can then be passed unchanged. Switches are an example of this category of bridges;
  - *Translation bridges* are used when both LANs have data link layer function and addressing that are sufficiently similar to allow a direct translation of protocol data units (PDUs);
  - *Encapsulation (or tunneling) bridges* may be used when the translation is not possible. An incoming frame is encapsulated in the data link layer format on the other link before being forwarded. This is the kind of bridge used when two, or more, LANs of identical technology needs to be interconnected through another one of a different kind.

  A bridge participates as a node on each of the LANs it interconnects. It receives a copy of all frames transmitted on each one. Obviously, not all frames need to be relayed. For instance, a frame received on side A, whose destination is also on side A, does not need to be relayed on side B. The bridge has thus to learn which frames should be relayed on side B, and which ones should not. In particular, special care should be taken to avoid frame looping when several paths exist between two nodes that communicate.

- ***Routers***
  Routers operate at the network level. The main difference between bridges and routers is that the latter are not transparent and use path oriented forwarding instead of broadcasting. Routers interpret the payload of the packets they forward, in particular their address fields. Routers exchange information between themselves in order to find a route on which the packet is conveyed. They can thus find a better path between two nodes than bridges that use only a loop-free subset of the available topology.

- ***Gateways***
  Gateways relay the information at the application layer. When a gateway node receives some application service request, it converts it into a service request on the other LAN to which it is connected. When the corresponding confirmation is received, it transforms it into a reply on the other side. Depending on the available services, an invocation may correspond to more than one request. Similarly, the reply may be constructed from a number of confirmations.

  In order to overcome the additional delay in the reply to an indication on the LAN on one side, some gateway may make requests on the other side in advance. This is especially true in industrial networks where the gateway may keep an image of all inputs on one side so that, when a read is received on the other side, the cached value is returned in the response. These are called proxy gateways.

After this summarization the first conclusion is that repeaters are out of scope or not an issue with respect to the needed setup. In this context, repeaters cannot be used to link two different protocols. They may be used to directly interconnect two UPnP networks to form one single UPnP network. In such a case, they are invisible for UPnP. In fact, UPnP sees only devices reachable through a broadcast.

When using UPnP and JXTA, two different protocol layers will be used. Interoperability between UPnP and JXTA concerns the co-operation of devices or layers that need to interoperate. This results in two notions of interoperability that can be distinguished:

- **Protocol interoperability:** concerning the **interoperability** between two devices speaking / understanding the same protocol;

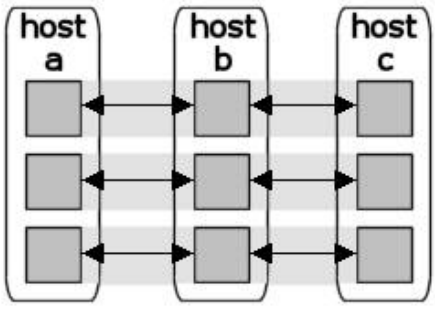


Figure 10: Protocol Interoperability

- **API interoperability:** concerning the interoperability between a service and a service client/user communicating via the agreed interface. Such type of an interface within a device is called Application Programming Interface (API). This type of interoperability could also be called service or platform interoperability.

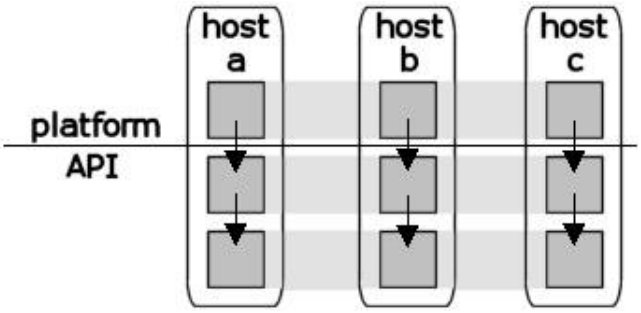


Figure 11: API Interoperability

6.1.1 Protocol interoperability

Protocol interoperability concerns the interoperability between different devices that will end up in a network. For the devices to interoperate, it is of major importance that the devices communicate in the same language, i.e. that the same protocols are used. Connecting two or more UPnP networks can be done by using two or more encapsulation bridges. A tunnel between these bridges is responsible for the communication between the UPnP networks.

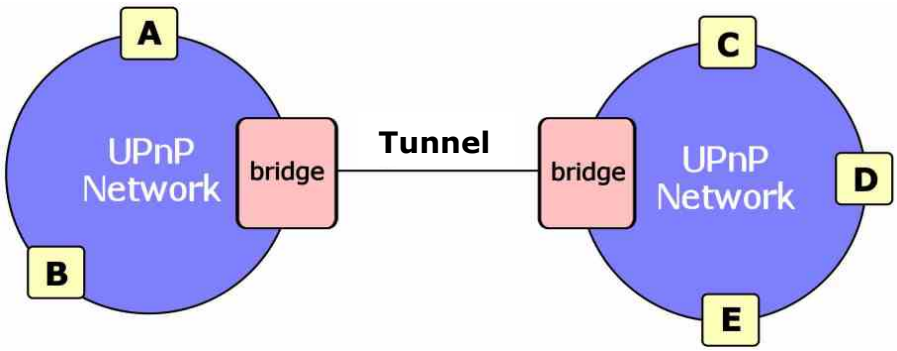


Figure 12: Interconnection through bridges

In the setup depicted in figure 12, two UPnP networks may communicate using tunneling bridges. The letters A till E are UPnP devices within the UPnP network. Communication between the two bridges can be handled directly by using the pipe-mechanism, which is part of the JXTA protocol. To communicate using a pipe, a peer first needs to find the endpoints, one for the source of the message and one for each destination of the message, and connect them by binding a pipe to each of the endpoints. The bridges can serve as an endpoint. JXTA peers are most likely needed between the two bridges to ensure communication between the UPnP networks.

Note also that using tunneling bridges, there is a danger to flood the other networks with unnecessary traffic. This may be easily solved with learning bridges that learn within a certain amount of time which device is on which side and tunnel only the traffic that needs to be sent to the other side. Traffic generated with a destination at the other UPnP network will be packed and sent over the tunnel.

Learning bridges can be used but will introduce some drawbacks. One of the problems that can rise with this configuration is when more UPnP networks are combined. If the number of connected peers is relatively big, the traffic that will be generated by the broadcasting is significant. Therefore, UPnP is limited with respect to scalability. Using tunneling bridges is thus feasible within a certain range of connected UPnP networks.

Other minor drawbacks with respect to learning bridges:

- If not well configured it will lead to unnecessary network traffic;
- No possibility to control the tunnels from inside the UPnP network, i.e. blocking the visibility of some UPnP device for the outside world.

A second option is to use routers instead of bridges. Routers interpret the payload of the packets and determine where to send the packet to (based on the information within the payload). There is however a potential problem with the handover of messages between the networks. Each UPnP network will have its own subnet. Connecting multiple UPnP networks together will result in at least two (or more) subnets. UPnP needs to operate within the same subnet, so binding two different UPnP networks together is not possible without changing IP's (NAT), unless both networks use the same subnet. Changing IP's will exclude the router option, as the router interprets the payload of the packets and determines where the packet needs to be sent to. If a remote UPnP device has an IP address within the subnet of the local UPnP network, the router will not route the messages pointed at the remote UPnP device to the remote UPnP network. Another potential risk is when the subnets are identical. Two UPnP devices, both available within a different network, could have the same IP address. Packets sent to a remote UPnP device with the same IP address of another local UPnP device will also not be routed to the remote UPnP network.

6.1.2 API interoperability

The second method is to use a gateway between the UPnP networks. Such a gateway has to translate or forward messages from UPnP to JXTA and vice-versa. The API interoperability concerns the interoperability between applications in a distributed system at one side and the underlying services at the other. As gateways operate at the application layer this solves most of the problems caused by the other technique:

- There is no additional network traffic;
- Possibility to control from inside the UPnP network;

Different techniques can be introduced to discover devices on other UPnP networks, without using the broadcast mechanism. Therefore, the range of connected UPnP networks will increase, also dependent on the discovery technique that will be used.

Identical IP addresses of UPnP devices within identical subnets, as discussed earlier, are also a potential risk with respect to gateways. Multiple UPnP networks connected through gateways imply that there will also be at least two (or more) subnets. These networks can have identical subnets, so different UPnP devices can have equal IP addresses. This potential risk can be solved in two ways:

- Change the IP's of all remote UPnP devices to the range of IP-addresses used within the local UPnP network at gateway level;
- Use one device, which takes care of leasing IP addresses on all UPnP networks.

The second option can only be used when the UPnP networks are directly coupled with the help of just one gateway device. But to fulfill the graduation assignment two gateways are needed with JXTA as transport layer in between. Therefore the IP addresses of remote UPnP devices need to be changed to IP addresses within the local subnet (first option). As gateways operate at the application level, this can be done directly by the gateway itself.

The fact that gateways operate at the application layer creates other possibilities. Gateways can be used as a proxy for example. A built-in proxy feature gives some extra freedom to control and manage UPnP networks separately.

The technical definition of a proxy that will be used throughout this document is: a proxy is a service which acts on behalf of another service. This includes for example sending requests on behalf of a local UPnP device. Using a proxy can also be used to overcome the additional delay in the reply to an indication on the LAN on one side, thereby making requests on the other side in advance. This can be considered as bringing the devices and services of a remote UPnP network within the local UPnP network. Figure 13 depicts the gateway setup (the precise JXTA implementation is abstracted) for the interconnection of two UPnP networks.

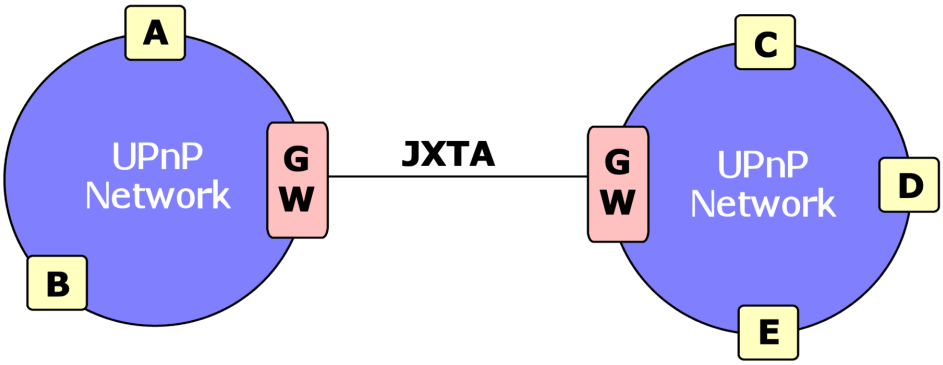


Figure 13: Interconnection through gateways

Table 2 provides an overview of how the different hardware components are related to the different aspects where the graduation assignment needs to focus on (also discussed in chapter 2 – Specification).

	Repeaters	Bridges	Routers	Gateways
Protocol Layer	Physical	Datalink	Network	Application
Transparent	Yes	Yes	No	No
Scalability	Limited	Limited	Improved	Optimal
Main purpose	Reinforce the strength of an electrical signal sent	Split a network into segments	Connect two networks together and route packets of data in the networks	Connect networks that are running different protocols
Interoperability	Protocol	Protocol	Protocol	API
Usability	No, cannot be used to link two different protocols	No, limited scalability and no control from within the network	Yes, but recognition problem and addressing problem needs to be solved	Yes, as an option extra software can be used to extend possibilities (e.g. a proxy)

Table 2: Interconnection comparison

Therefore, as the gateway can be used to solve the addressing problem and the fact that some extra freedom is gained into account (e.g. a proxy), gateways will be used to interconnect two (or more) different UPnP networks. Also the recognition problem, i.e. sending a packet to a remote UPnP device which has an IP address within the range of the local subnet, can be solved by using a gateway.

6.2 Gateway architecture

As discussed in the previous section, the gateway approach is most applicable for the desired setup. As can be seen in figure 13 the gateway needs to take care of UPnP devices available within remote UPnP networks to become available within the local UPnP network. This applies for both sides (networks) of the figure.

The left part of figure 13 consists of two UPnP devices, named A and B. The right part consists of three UPnP devices, named C, D and E. To make devices A and B available within the right UPnP network and the devices C, D and E within the left UPnP network a mapping mechanism of the gateway is needed. This mapping can be done in two ways:

- Using a virtual mapping;
- Using a gateway mapping.

Which mapping is most applicable for the desired setup will be shown in the next sections.

6.2.1 Virtual mapping

Virtual mapping will display all devices within remote UPnP networks as if they actual join one big UPnP network. As can be seen in figure 14, all UPnP devices which where first spread over two or more UPnP networks (as in figure 13) are available within one UPnP network. For simplicity the gateway device is left out. Hence, mapping remote UPnP devices onto the local UPnP network is done at the gateway level. The gateway is responsible for the correct mapping and translation of remote UPnP devices and bringing them in or taking them out of the local UPnP network.

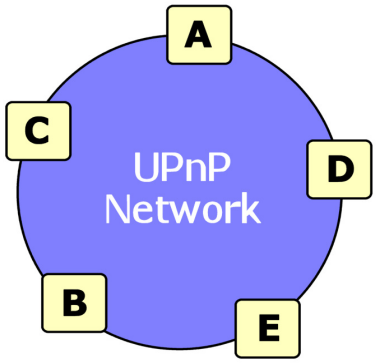


Figure 14: Virtual Mapping

This architecture has some drawbacks. First drawback is when for example two identical UPnP devices are available, one at the local UPnP network and one at a remote UPnP network. This will lead to a name-collision as the remote UPnP devices are virtually mapped within the local UPnP network. Originally the UPnP Device Architecture is not equipped to differentiate between two identical UPnP devices, other than by a unique ID. For a user, which is in control of the UPnP network, it will be very difficult to distinguish the two different UPnP devices. This also applies to the place where the UPnP devices are situated, because it is not possible to see where a UPnP device is situated (without changing the configuration of one of the UPnP devices).

Second drawback is that when virtual mapping is used the number of UPnP devices available on the local UPnP network will grow rapidly with the number of connected UPnP networks. Every time a UPnP network connects, all UPnP devices available on that network (those that are allowed to be visible to the outside world), will also become available within other connected UPnP networks.

6.2.2 Gateway mapping

Another solution is to map all remote devices directly on the gateway device. The left part of figure 13 is depicted in figure 15, but with the remote UPnP devices (right side of figure 13) directly mapped on the gateway device. This will result in a list consisting of different joining UPnP networks, listing all their devices and services (depending on the security and authorization level, which will be discussed later). The remote devices are available within the local UPnP network but are separated at the gateway device.

This architecture has some advantages in contradiction to the virtual mapping:

- Only one IP needed (Gateway IP) for every remote UPnP device;
- Separation of local and remote UPnP devices by using a prefix (for every remote UPnP network);

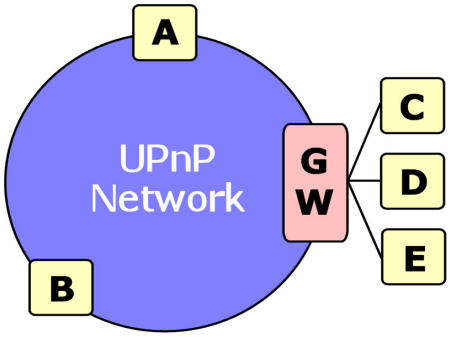


Figure 15: Gateway Mapping

Every device mapped on the gateway will have the same IP address as the gateway itself. For this kind of mapping there must be a list, which takes care of changing the IP address of the remote UPnP devices into the gateway IP, and vice-versa. Messages pointing at some UPnP device outside the local UPnP network needs to be routed to the corresponding remote UPnP network. The remote UPnP devices are added directly to the gateway, as if the remote UPnP devices are part of the gateway. This separates the remote UPnP networks from the local UPnP network.

6.2.3 Conclusion

A number of issues are involved making the remote UPnP devices available within other UPnP networks. Both solutions presented above only take two UPnP networks into account. But what happens if a third and even a fourth network joins? This depends also on the JXTA configuration, which applies between the UPnP networks.

Both mapping techniques allow simply adding of extra functionality. The proxy mechanism can be considered as an application acting on behalf of another application or system in responding to protocol requests. Every device for example, within the local network is known by the gateway, even all remote UPnP devices and networks. Configuring for example some kind of proxy which is capable of responding on behave of a remote UPnP device is easy when all information is available within the local UPnP network, or at least within the gateway device. Security rules can also easily be applied. When for example a new device is added to the local UPnP network and it may not be visible for the outside world, this can be configured at the gateway device.

Note that when some kind of proxy is used, certain commands/messages cannot be handled locally and need to be sent to the other UPnP network, containing the actual remote UPnP device. Therefore, all messages send within the local network and pointed at the gateway, need to be read. The information contained within the message determines if the message needs to be forwarded or if the message can be dealt with locally by the proxy.

Fact is that the gateway mapping provides a better visual separation of remote UPnP networks to the user. Every connected remote UPnP network will be grouped and published within the local UPnP network. Therefore, this technique will be used to carry out an implementation approach. From now on the gateway device will be called the UPnP Proxy device, which is a proxy between the UPnP networks.

6.3 JXTA architecture

So far, not much has been said about the JXTA architecture, except for the fact that it is used to bind multiple UPnP networks together. But there is more to deal with then just connecting multiple UPnP networks.

6.3.1 Peer groups

The JXTA part is built up out of simple peers, which can connect to a particular peer group. This peer group is a virtual entity that speaks the set of peer group protocols. Typically, a peer group is a collection of cooperating peers providing the same services. Connecting the gateways as peers in peer groups provides a service where only connected UPnP networks are available. In this way, different peer groups can be set up to define multiple collections of UPnP networks.

But how to establish a connection to these peer groups from behind a private local network (i.e. the home-network)? For most peers existing on such a network, finding rendezvous peers and router peers is critical to participate. Rendezvous peers provide these simple peers with the capability to broadcast messages to other members of a peer group outside the private network. This functionality is independent of the underlying network transport, allowing message propagation over transport layers that do not have multicast or broadcast capabilities.

The easiest way to ensure that a simple peer can find rendezvous and router peers is to seed the peer with a hard-coded set of rendezvous peers. These rendezvous peers usually exist at static, resolvable IP addresses and are used by a peer as an entrance point to the network. A gateway peer located behind a firewall can use these static rendezvous peers as a starting point for discovering other peers and services.

As most of the existing rendezvous peers are IP based, it will suffice to start with a JXTA configuration that is also IP based. In this way some difficulties can be avoided with respect to the independency of the network platform, as already mentioned in chapter 5. Therefore, all communication between JXTA gateway peers is done via TCP/IP. This gives rise to the following network setup as depicted in figure 16.

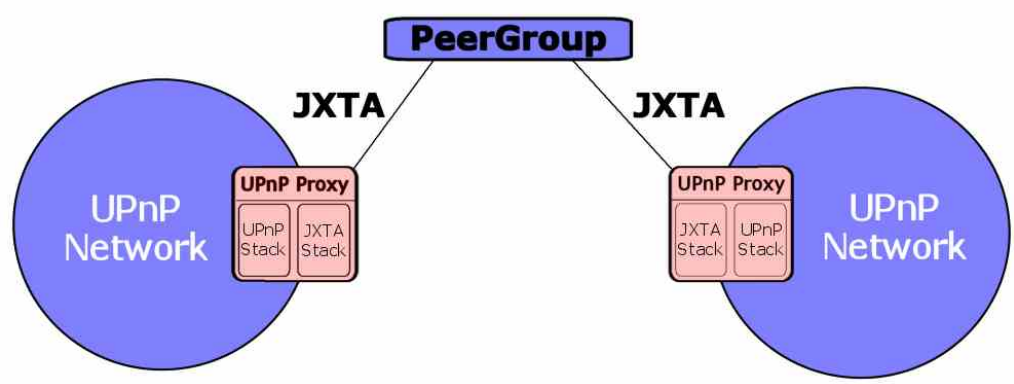


Figure 16: JXTA gateway peers

As shown in figure 16, different UPnP networks can be connected via JXTA peer groups. Lines between the UPnP proxy and the peer groups can contain a number of rendezvous peers to propagate messages between the gateway peers.

6.3.2 Peer pipes

As already mentioned in the JXTA basics (chapter 5), simple peers are able to send and receive messages, and are able to cache the advertisements of joining peers. The actual exchange of messages among peers is done normally with peer pipes. Peer pipes are a message transfer mechanism used for service communication between peers.

For this architecture two kinds of pipes are interesting. Pipes consist of two different endpoints: an input pipe, which receives messages and an output pipe, which can send messages. JXTA pipes can have endpoints that are connected to different peers at different times, or may not be connected at all.

Point-to-point Pipe
To connect only two UPnP networks it is sufficient to use the *point-to-point* pipe, which exactly connects two peer endpoints together (figure 17). This means that an input pipe on one peer receives messages sent from the output pipe of another peer. To get a bi-directional transport of messages, two point-to-point pipes need to be created. Thus every gateway peer at least needs one input pipe and one output pipe.

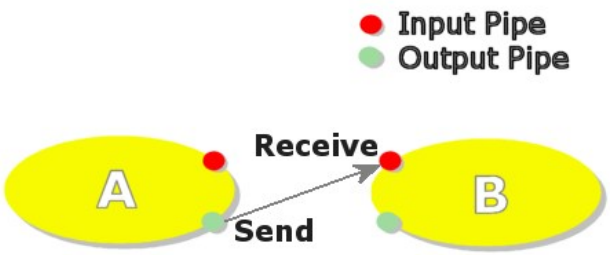


Figure 17: Point-to-point pipe

Propagation Pipe
The so-called propagation pipe is a collection of one output pipe and multiple input pipes. Looking at the future this propagation pipe is more interesting (figure 18). One output pipe will be connected to multiple input pipes. Messages flow from the output pipe (the propagation source) into the input pipes. All propagation is done within the scope of a peer group. That is, the output pipe and all input pipes must belong to the same peer group. When for example more then two UPnP networks need to be connected, this propagation pipe can be applied (broadcasting mechanism). In this way there is no need to set up different point-to-point pipes between every connected UPnP network.

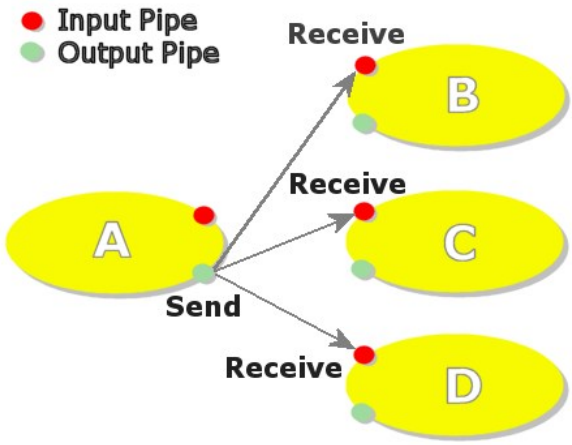


Figure 18: Propagation pipe

On the other hand, point-to-point communication between different UPnP networks is still needed as sometimes messages need to be sent directly to a UPnP network. Sending this message with the propagation pipe will seed all other networks with superfluous information.

Secure Unicast Pipe
As a variant on these two pipes a third one can be used, the secure unicast pipe. A secure unicast pipe is a type of point-to-point pipe that provides a secure communication channel. Security issues enter at this point. Communication for example over the Internet must be secure. There are three dominant requirements with respect to security:

- Confidentiality;
- Integrity;
- Availability.

These requirements translate automatically into specific functionality requirements that include authentication, access control, audit, encryption, secure communication, etc. Secure communication can for example be addressed by use of the secure unicast pipe. Details like security are not part of the graduation assignment and will therefore not be taken into account. This does not imply that security is not important but the main goal of the assignment is to set up an architecture to bind two (or more) UPnP networks with JXTA as communication technology.

7. UPnP – JXTA Gateway Architecture

The UPnP – JXTA Gateway architecture can be treated as if it consists of two different parts. Therefore, the first part of this chapter contains the architecture of a UPnP Proxy, which ensures the communication between the devices available within the local UPnP network. On the other hand JXTA will be used to establish a connection between UPnP networks. This is the other part of the UPnP – JXTA Gateway architecture and will follow directly after the UPnP Proxy part.

The UPnP architecture defines devices and control points. A physical device can contain one or more logical UPnP devices and zero, one or more UPnP control points. The UPnP Proxy works the same as a normal UPnP control point / device. The control point can discover other UPnP devices on the local network. On the other hand it is possible for other UPnP control points to control the UPnP Proxy device, i.e. configuring JXTA peers, peer groups, etc (figure 19).

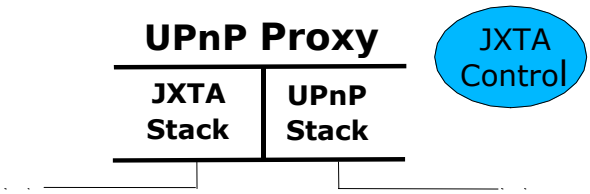


Figure 19: UPnP Proxy Implementation Stack

As can be seen in figure 19, the UPnP Proxy consists of the JXTA Protocol Stack combined with the UPnP Protocol Stack. The UPnP Proxy needs to traverse messages traveling from one stack to the other. Extra functionality is added to the UPnP Proxy by means of the JXTA Control part. For now, it is enough to have just a basic set of actions to control the UPnP Proxy from within a UPnP network. In terms of action and state variables the UPnP Proxy must at least be able to:

- change between JXTA peer groups;
- create new JXTA peer groups;
- generate a list of available peer groups;
- generate a list of connected peers within a peer group;
- returns the current connected peer group.

Every action contains also one or more state variables. The precise device and service descriptions can be found in Appendices B and C.

At startup, the UPnP Proxy will load all necessary configuration files and all device and service descriptions. Basic proxy feature is to act on behave of local and remote UPnP devices. To achieve this behavior, the UPnP proxy needs to be up-to-date at any time. Continuous communication between the UPnP networks is needed to make sure that every proxy keeps up-to-date. How this can be achieved is explained in the following sections.

7.1 UPnP – Proxy Architecture

The UPnP Proxy Protocol Architecture contained herein defines a protocol for communication between UPnP proxies and devices. Because of the mapping between UPnP and the UPnP Proxy this part has the same structure (chapters and numbering) as the UPnP Device Architecture document.

This architecture defines the protocols for communication between controllers, or control points, and devices. For addressing, discovery, description, control, eventing, and presentation, UPnP uses the protocol stack as mentioned in chapter 4.1 – UPnP Architecture.

The remaining sections of this chapter describe the content and format of these protocol layers. Every section contains a UPnP Proxy Extension, which describes how this protocol layer can be used within the UPnP Proxy, eventually with the changes needed to achieve the behavior desired.

7.1.1 Addressing

Addressing is step 0 in UPnP networking. Each UPnP device which does not itself implement a DHCP server must have a Dynamic Host Configuration Protocol (DHCP) client and search for a DHCP server when the device is first connected to the network (if the device itself implements a DHCP server, it may allocate itself an address from the pool that it controls). If a DHCP server is available, i.e., the network is managed; the device must use the IP address assigned to it. If no DHCP server is available, i.e., the network is unmanaged, the device must use automatic IP addressing (auto-IP) to obtain an address.

Auto-IP defines how a device:

- Determines if DHCP is unavailable;
- Intelligently chooses an IP address from a set of link-local IP addresses.

This method of address assignment enables a device to easily move between managed and unmanaged networks.

UPnP Proxy Extension
For the UPnP Proxy this addressing architecture does not need to be changed. Every UPnP Proxy device will first search for a DHCP server and, if not available, assign an Auto-IP address.

To use this addressing architecture to bind two or more different UPnP networks together, a mapping mechanism between these networks is needed. This means that:

- Devices available on the remote network will become available on the local network;
- Devices on the remote network will have their own IP. Within the local network this IP address needs to be mapped to an IP address within the subnet of the local network (actually, this will be the gateway IP as proposed in chapter 6: Gateway Mapping);

In this way all local UPnP devices can be extended over the network, independent of the network transport layer between two or more different UPnP networks. The UPnP Proxy will handle messages pointed at devices not available on the local network. The UPnP Proxy knows to which network the message needs to be sent and will exchange the IP address to the IP address of the remote UPnP device (which also is available because the IP addresses of remote UPnP devices are mapped to the gateway IP). The message will then be packed and send (via JXTA) to the corresponding UPnP Proxy which will unpack it and publish it on its local network. Messages received by the UPnP Proxy which contain an IP address not known will be dropped automatically, as the UPnP Proxy knows all local UPnP devices.

7.1.2 Discovery

Discovery is step 1 in UPnP networking. When a device is added to the network, the UPnP Discovery Protocol allows that device to advertise its services to control points on the network. Similarly, when a control point is added to the network, the UPnP discovery protocol allows that control point to search for devices of interest on the network. When a new device is added to the network, it multicasts a number of discovery messages advertising itself, its embedded devices, and its services. Any interested control point can listen to the standard multicast address for notifications that new capabilities are available.

Similarly, when a new control point is added to the network, it multicasts a discovery message searching for interesting devices, services or both. All devices must listen to the standard multicast address for these messages and must respond if any of their embedded devices or services matches the search criteria in the discovery message.

When a device is removed from the network, it should, if possible, multicast a number of discovery messages revoking its earlier announcements, effectively declaring that it's embedded devices will no longer be available. Also when the configuration of a device changes, it should revoke any earlier announcements and re-advertise using the new configuration. When, for example, the IP address of a device changes, the outstanding advertisement needs to be revoked and needs to be republished using the new IP address.

For devices and control points that have multiple network interfaces, UPnP advertisements and searches should be sent on all local network interfaces enabled for UPnP networking. Each advertisement or search must specify an address that is reachable on that interface. UPnP advertisements and searches must not be sent on remote Internet interfaces.

UPnP Proxy Extension

For the UPnP Proxy, it is enough to act as a control point. In this way all local UPnP devices can be discovered using the discovery mechanism mentioned earlier. This means that local UPnP devices can be discovered and be stored within the UPnP Proxy (just like a normal control point). The local UPnP devices can be sent as a list by some push or pull mechanism to other listening UPnP Proxies, which are interested in the UPnP devices available on that particular network.

When a device list is received from a remote UPnP proxy, it needs to be published within the local network. This will be done as described above, thereby revoking the outstanding UPnP Proxy advertisements and republishing with the new device list included.

Looking more thoroughly at the discovery messages that will be sent when the devices of a remote UPnP network are included at the UPnP Proxy, will reveal that every root device will be published, but also the embedded devices (thus also the remote UPnP devices) and any services. Revoking and republishing the advertisements means that every control point on the local network will be updated with the new configuration of the UPnP Proxy.

When a control point is added to the network, the UPnP discovery protocol allows that control point to search for devices of interest on the network. This search protocol can also be used by the UPnP Proxy itself. When a UPnP Proxy receives a search message from another control point, the same discovery messages needs to be published as when a new control point is added to the network. Only difference is that in this case the response messages are sent using unicast instead of multicast.

Building a complete list of remote UPnP devices is needed to respond to queries and to announce the presence of remote UPnP devices. The UPnP Proxy is responsible for maintaining this device list. It needs to be synchronized every so often to make sure the list keeps up-to-date. For example: every time a UPnP device appears or disappears from a UPnP network, this network change needs to be known (after some time) by all other joining UPnP networks. As can be seen later on, discovery is closely related to the next step, description. Therefore the next section deals with the different synchronization methods that can be used to keep the UPnP Proxy up-to-date.

7.1.3 Description

Description is step 2 in UPnP networking. After a control point has discovered a device, the control point still knows very little about the device – only the information that was in the discovery message, i.e. the device's (or service's) UPnP type, the device's universally-unique identifier, and a URL to the device's UPnP description. For the control point to learn more about the device and its capabilities, or to interact with the device, the control point must retrieve a description of the device and its capabilities from the URL provided by the device in the discovery message.

The UPnP description for a device is partitioned into two, logical parts: a *device description* describing the physical and logical containers, and *service descriptions* describing the capabilities exposed by the device. A device description also includes a description of all embedded devices and a URL for presentation of the aggregate.

A single physical device may include multiple logical devices. Multiple logical devices can be modeled as a single root device with embedded devices and services or as multiple root devices. In the former case, there is one UPnP device description for the root device, and that device description contains a description for all embedded devices. In the latter case, there are multiple UPnP device descriptions, one for each root device.

A service description includes a list of commands, or *actions*, the service responds to, and parameters, or *arguments*, for each action. A service description also includes a list of variables. These variables model the state of the service at run time, and are described in terms of their data type, range, and event characteristics (state variables).

Retrieving a UPnP device description within the local network is simple: the control point issues an HTTP GET request on the URL in the discovery message, and the device returns the device description. Retrieving a UPnP service description is a similar process that issues a URL within the local device description (see figure 20).

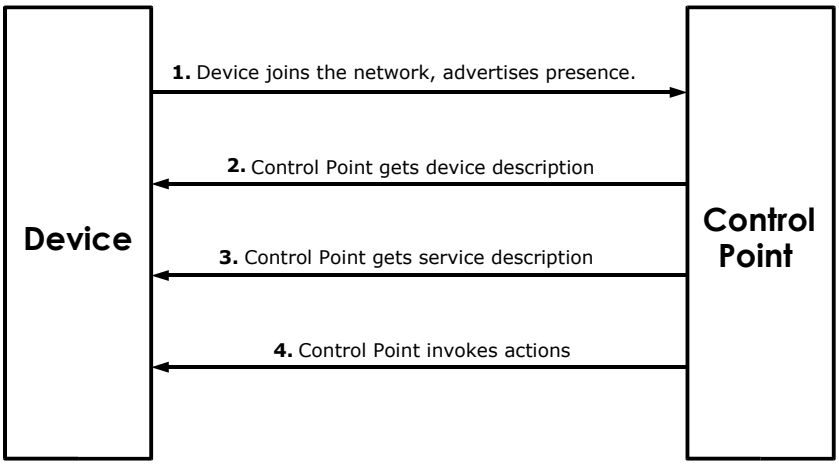


Figure 20: Retrieving Device and Service Descriptions

UPnP Proxy Extension

Retrieving a UPnP device description from a device outside the network is more complex. There is no possibility to do a HTTP GET request directly to the device. The HTTP GET request must be pointed at the local UPnP Proxy. The UPnP Proxy can only respond directly to this request if it has the corresponding device description cached. All device descriptions of all available joining UPnP Proxies have to be known and must be cached. Only in this way it is possible to issue directly a response message when a request for a remote UPnP device is received.

Before actually publishing the presence of a new remote UPnP device within the local network, gathering the device description and the service description belonging to the remote UPnP device is needed. This means that after the local announcement of a new remote UPnP device, interested control points can request for the device description and the service description locally.

Figure 21 contains a message sequence chart for the case that a control point invokes an action on a remote UPnP device. The chart only covers the message sequences for the communication between a control point and a device. It starts with the announcement of a UPnP device on network 2. When the UPnP Proxy on network 2 notices that the local network configuration has changed, it sends the changes to network 1. After the changes have been received by the UPnP Proxy on network 1 it revokes any earlier local announcements and re-advertise the new status of the UPnP Proxy, including the presence of the remote UPnP device. After the control point has received all descriptions, it is capable of invoking actions on the new remote UPnP device.

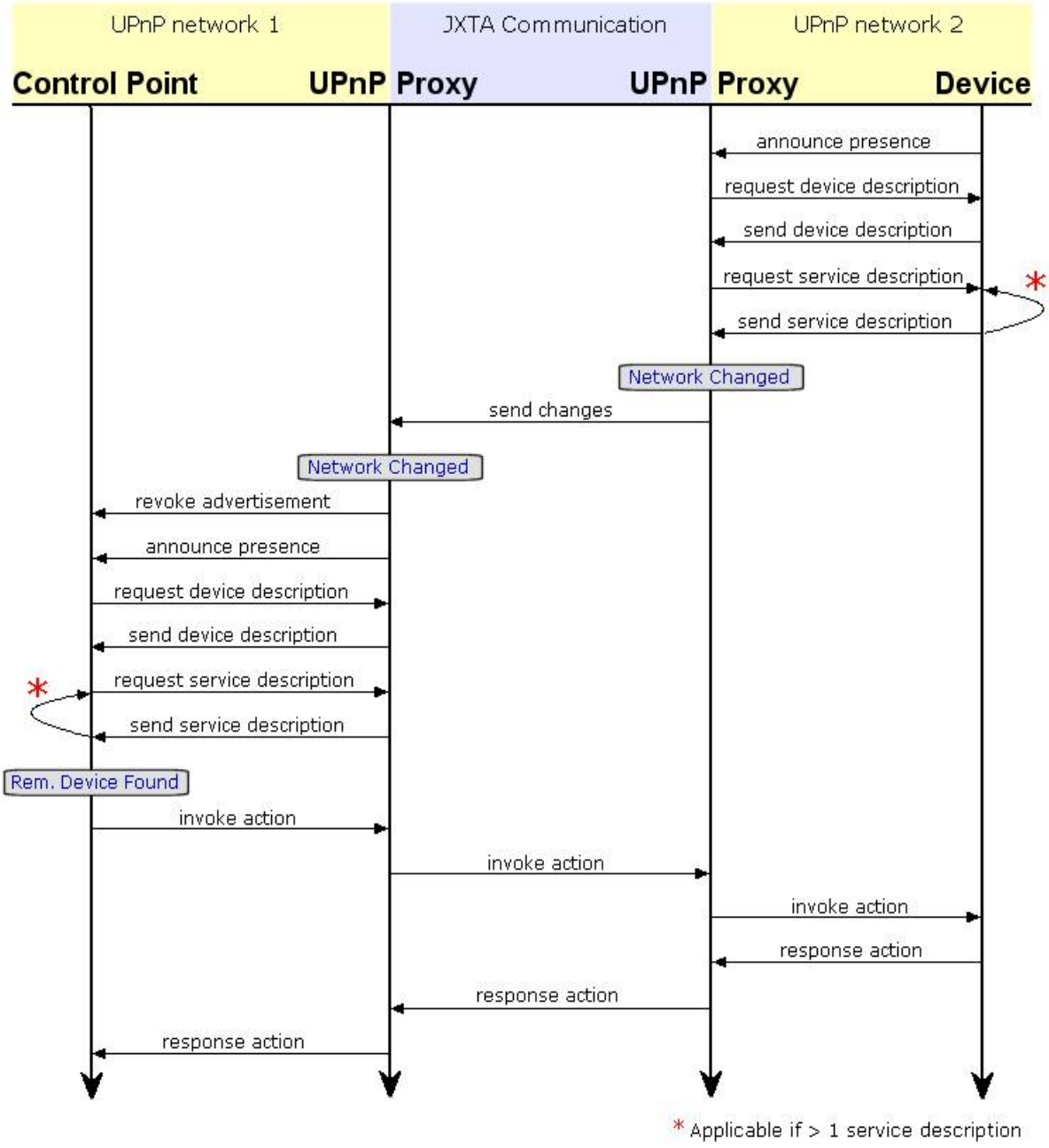


Figure 21: Message Sequence Chart

Consequence of this technique is that the UPnP Proxy cache needs to be up-to-date all the time, for example when some action is pointed at a remote UPnP device not available any more. It will take a maximum of 30 seconds (the time a device has to react to some request) before the requesting control point knows the remote UPnP device does not respond. When the UPnP Proxy does not know that a remote UPnP network has changed, it consequently contains incorrect information.

Updates need to take place whenever a UPnP network changes. When a device joins or leaves a UPnP network, a change notification message will be send to all joining UPnP Proxies. In this way, the cached remote UPnP device list will always be up-to-date, except for the time that it takes to inform every joined UPnP Proxy to update the list. This can be the case in the following example: if a device leaves a UPnP network (e.g. network 2) and at the same time, on another UPnP network (network 1), a control message is sent to that particular remote device (e.g. an action invocation). The UPnP Proxy in network one checks the availability of the remote UPnP device in its remote UPnP device list, and will forward the message to the UPnP Proxy on network two (it does not know that the remote UPnP device is not available any more). If the UPnP Proxy on network two knows the device has just left the network, the message will not be forwarded to the device. Within this time (the time it takes before the UPnP Proxy cache of network one knows that the remote device is not available any more) the cache of the UPnP Proxy on network one is incorrect. Any outstanding action invocation will result in a time-out, notifying the control point that the external UPnP device is not available anymore or not reachable within the predefined time. This is a consequence of connecting two or more UPnP networks together and can be dealt with also by the UPnP Proxy. If the UPnP Proxy on network two already knows that the specific device is not available anymore it can respond on behalf of the disappeared device with an error response, avoiding the time-out.

Therefore, as long as the discovery advertisements from a local device have not expired, a control point may assume that the device and its services are still available. The device and service descriptions may be retrieved at any time since the device and service descriptions are static as long as the device and its services are available.

7.1.3.1 UPnP Device descriptions

The UPnP description for a device contains several pieces of vendor-specific information, definitions of all embedded devices, URLs for presentation of the device, and listings for all services, including URLs for control and eventing. In addition to defining non-standard devices, UPnP vendors may add embedded devices and services to standard devices.

To implement a UPnP Proxy all device descriptions (locally and remote) will be parsed and the most important elements will be stored in a table. Figure 22 shows the structure of the UPnP device description document. Not all sub elements are shown, but only the essential elements for identifying a device, a service, a state variable, an action or an argument. There are also type identifiers for devices and services.

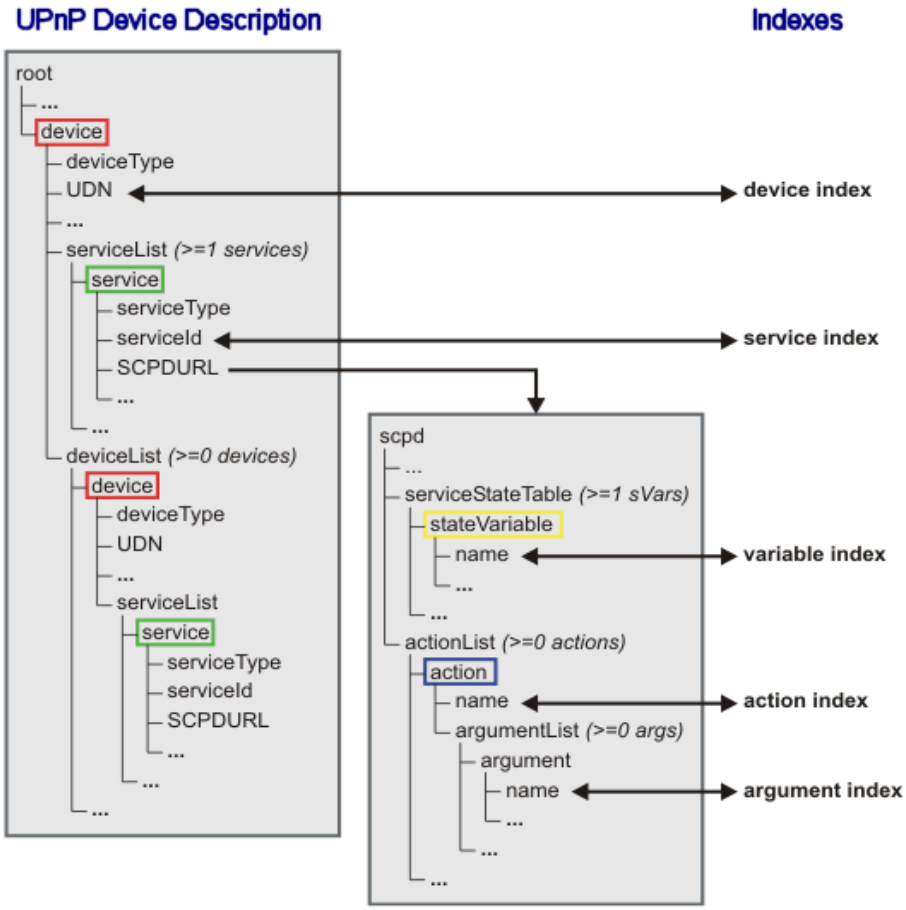


Figure 22: Structure of a UPnP device description

The device description of the UPnP Proxy consists of two parts. One part is the description of the UPnP Proxy device itself; the other is the device list with a mapping of all joining UPnP Proxies and their connected devices and services. The former describes the UPnP Proxy; the latter contains all joining UPnP Proxies with all remote UPnP devices in a hierarchical way (as discussed in chapter 6.2.2 – Gateway mapping).

The right side of figure 22 shows all indexes with their relationship to the identifiers of the basic UPnP elements. These indexes can be used to define for example the visibility of a certain device (or of its services) within remote UPnP networks.

Defining the indexes with the basic UPnP elements as described in the device and service descriptions can be done best directly on the UPnP Proxy. The UPnP Proxy must provide the possibility to directly insert the index with a certain value into a device or service description or to define a list which will store these values. Note that, if applicable, a user needs to manually configure the values of every device separately, because this part is not included in the definition of the UPnP Device Architecture. To configure each description more easily a user interface can be provided to do so.

The device and service descriptions can be physically divided into three main parts, each identifying a specific part of the description. The following summary contains the main indexes and will describe its use:

1. device-index : this index can be used to completely hide a certain device from the outside world;
2. service-index: this index can be used to hide a certain service from the outside world;
3. action-index: this index can be used to hide a certain action within a service from the outside world;

These indexes are the most important for now. Later on, not within this graduation assignment, the argument-index and the state variable-index can be used to hide other parts of the description or to define other configuration settings for the outside world.

Appendix D contains two examples of UPnP device descriptions. The first one contains a device description that is not visible for the outside world, just by setting the index value in the device-tag. The latter one contains an example of a service description where one of its actions is not visible for the outside world, by setting the action-index to the correct value. When a list of available devices needs to be sent to another interested UPnP Proxy, the local UPnP Proxy reads the configuration settings and will disable certain parts of the descriptions before sending, according to these settings. These examples are based on the index-value 0 or 1, which means invisible or visible. Other values can be used to define other configuration settings.

These settings can also be stored in a XML file, which can be read every time a description needs to be sent to another UPnP Proxy. In this case, the indexes can be used to identify a specific device, service or action, combined with a value indicating the visibility-status.

The option to enable or disable certain devices, services or actions for the outside world is desired whenever the user wants to be in control of the UPnP network (as discussed in chapter 3 – Usage Scenarios). As a standard, control of the UPnP Proxy for example will be disabled. It can be argued that a user needs to be able to configure the UPnP Proxy from outside the local network, but for now the assumption is made that only from within the local UPnP network the UPnP Proxy can be controlled and configured.

7.1.3.2 UPnP Proxy synchronization

The different UPnP phases of publishing a device on a network are depicted in figure 6. As already mentioned in step 0 (addressing) and step 1 (discovery), the description phase is required for UPnP communication. Together with addressing and discovery they form the basic building blocks of publishing a UPnP device within a network.

Publishing the UPnP devices within a remote network, the device descriptions and service descriptions available within the local network need to be published in the remote network as if they are in the local network itself. Therefore some mechanism is needed which keep track of the local and remote devices. There must be at least a local and a remote device list, which keeps track of the device descriptions of the local and the remote devices (this also applies for the service descriptions).

Synchronizing UPnP Proxies with remote device and service descriptions can be done in two ways:

- ***Static* way:**
  Keep track of all local device and service descriptions into two lists (the local device description list and the local service description list). Changes with respect to the local network will be changed immediately into these lists. When for example a new UPnP network announces it presence, it can be sent directly the two lists (push) or the new UPnP Proxy itself can ask for these lists (pull).

  Changes with respect to the local network can be published to other UPnP Proxies by sending the complete device and service lists again. This will produce some overhead, because sending the complete descriptions over and over again introduces redundancy, as most of the time only a small part of a description has been changed. Therefore, sending the complete device and service descriptions will only be done at startup of a UPnP Proxy. At this time it sends its own descriptions to other UPnP Proxies, which on their turn will send their complete descriptions in response. After this initialization process, only the changes with respect to the local UPnP network need to be sent to the other remote UPnP networks. For only sending the local changes, the dynamic synchronization way is more appropriate.

- ***Dynamic* way:**
  For every change on the local network this change will be sent immediately to all other joining UPnP networks using broadcast. This will be done with a message containing the ID of the corresponding UPnP Proxy, the index of the device and/or service description and what kind of change that must be applied. If necessary, a device or service description can be included within this message.

  The receiving UPnP Proxy can decide how to deal with this message. When it knows the ID of the corresponding UPnP Proxy, it can include, change or exclude the changed device and/or service descriptions from its own lists. Deletion of a remote UPnP device can for example be done by sending a change message (actually a deletion message) which contains the UPnP Proxy ID but also the ID of the remote UPnP device that must be deleted. When the deletion message has been received, the specific device can be searched for in the lists and when found, being removed.

7.1.4 Control

Control is step 3 in UPnP networking. Given knowledge of a device and its services, a control point can ask those services to invoke actions and receive responses indicating the result of the action. Invoking actions is a kind of remote procedure call; a control point sends the action to the device's service, and when the action has completed (succeeded or with an error) the service returns any results or errors.

To control a device, a control point invokes an action on the device's service. To do this, a control point sends a SOAP message to the control URL for the service (provided in the <*controlURL*> sub element of the service element of the device description). In response, the service returns any results or errors from the action. The effects of the action, if any, may also be modeled by changes in the variables that describe the run-time state of the service. Changes of these state variables lead to events which are published to all subscribed control points (see section 7.1.5 – Eventing).

A SOAP message is sent as an HTTP POST with the content type set to text/xml to indicate that the content of the message is an XML document. SOAP request messages include a new HTTP header, SOAPAction, whose value is a URI that indicates the intention of the SOAP request. This is for example to let the receiver of the SOAP message know that the message is meant for some particular service.

The other HTTP message is the SOAP HTTP Response. This message, like the request, is an XML document contained in a standard HTTP message whose content type is set to text/xml. SOAP over HTTP follows the semantics of the HTTP status codes. For example, 2xx status code indicates success which means that the client's request is processed correctly (see Appendix E for a list of http status codes).

UPnP Proxy Extension

In a UPnP device description document, each service element has a control element. This is an element that contains an URL where all control messages for that service need to be sent to. Control points send SOAP based control messages to this control URL and, in response, the service returns any results or errors from the action performed.

Note that for the graduation assignment it is sufficient to just change the control URL as meant in the HTTP headers. The SOAP message, included within the body of the HTTP message does not contain any information about how to send the message to the correct receiver. Therefore, changing the control URL of the HTTP header and point it directly to the local UPnP Proxy, nothing has to be changed to the actual content of the message itself. More specifically, a UPnP Proxy is used to route the SOAP message to the correct receiver, which is within another UPnP network. Therefore, from now on only the headers of HTTP messages will be inspected, without changing the actual content of the SOAP message itself.

Action Request

To invoke an action using the POST method, a control point must send a request in the following format to the service's control URL.

```
POST controlURL HTTP/1.1
Host: controlURL host:port
Content-Length: length of body in bytes
Content-Type: text/xml; charset="utf-8"
SOAPAction: "urn:schemas-upnp-org:service:serviceType:v#actionName"
```

For this message to be correctly routed to the corresponding UPnP device on a remote UPnP network, the *host* element needs to be changed. A control point will not know that it is sending some SOAP message to a remote UPnP device. Messages need to be pointed directly to the UPnP Proxy which implies that the *host* element will contain the control URL of the UPnP Proxy instead of the control URL of the remote device. This control URL is also contained in the remote UPnP device description which is available on the UPnP Proxy. Changing this control URL prior to publishing, it will ensure that action requests are directly pointed at the UPnP Proxy.

Consequence is that when an action request is received by the UPnP Proxy, pointed at a remote UPnP device, it needs to be sent to the other UPnP network containing the remote UPnP device. Therefore, the UPnP Proxy needs to change the *host* element of the action request itself. It needs to be changed to the control URL of the actual remote UPnP device description. When this request is sent to the corresponding UPnP Proxy, it only has to send the message to the correct UPnP device, acting as if the proxy itself issued the action request.

Action Response

A service has thirty seconds to complete the action and to respond to the control point, including the time needed to send messages from the sender to the receiver. According to the UPnP Device Architecture, actions that are expected to take longer than this should return a message to keep the request alive. When the action completes, the service will return the respond message.

This respond message will be pointed at the local UPnP Proxy. As the UPnP Proxy has actually issued the action request it will wait for a response message. When the response is received in time it can directly be sent to the corresponding UPnP Proxy on the other network. This UPnP Proxy is also still waiting for a response message, just like the control point which has initially sent the request. This ensures that the response can be forwarded directly to the waiting control point.

According to the UPnP Device Architecture, if the service can not respond in time, it should return early and send an event when complete. This does actually not guarantee that the early response will be received in time by the sender. Therefore, the UPnP Proxy, which traversed the request from the sender to the other UPnP Proxy, needs to generate an early response message within the thirty seconds to extend the waiting time with an additional thirty seconds. This will ensure that every response will be received in time.

When a device disappears from the local network, the UPnP Proxy will notice this (see section 7.1.3.2 – UPnP Proxy synchronization). Action requests already pointed at the device can not be handled any more by the device itself. Using the build-in time-out can cover this problem. Another option is to respond to remote action requests by the UPnP Proxy. The HTTP headers can be used to show that an error has occurred during transmission, which is the case when a device is no longer available.

Actually, every HTTP header contains the status of the response message. This means that the HTTP header can indicate that some action was successfully received, understood and accepted. When something goes wrong in the communication between the receiver and the sender, the status code will return an error. This technique can also be used for the UPnP Proxy architecture. Reading the headers of the message and eventually change the status code the UPnP Proxy can indicate that an error has occurred or that the message included is not the same as initially send. If, for example the corresponding length of some message does not correspond with the initial length and the status code was initially released as 2xx, the UPnP Proxy notices this and can adjust the status code to 5xx (see Appendix E for a list of http status codes).

7.1.5 Eventing

Eventing is step 4 in UPnP networking. After a control point has discovered a device and retrieved a description of the device and its services, the control point has all the ingredients to start eventing. As section 7.1.2 - Descriptions explains, a UPnP service description includes a list of actions the service responds to and a list of variables that model the state of the service at run time. If one or more of these state variables are evented, the service publishes updates when these variables change, and a control point may subscribe to receive this information. Throughout this section, *publishers* refer to the source of the events (typically a device's service), and *subscriber* refer to the destination of events (typically a control point).

A publisher/subscriber model is typically used to implement event notification. Within this model, the publisher is the source of events and grants a client a subscription when the client registers interest in receiving events provided by the publisher. The publisher delivers an event notification to the client, which is the subscriber. The subscription, which is issued by the subscriber, may be granted for a particular duration and requires a periodic renewal by the client to maintain the subscription.

The publisher/subscriber model is relatively simple. It is asynchronous in nature and is good at modeling systems where asynchronous events need to be sent from one source to another. One particular protocol for event notification is the General Event Notification Architecture (GENA).

UPnP Proxy Extension
The GENA Architecture is a publisher/subscriber system whereby a subscriber may request, renew or cancel a subscription. Every message issued by the subscriber or by the publisher is based on HTTP. Every GENA message will always have some host element, which contains the domain name or IP address and eventually the port where the message is pointed at. Therefore, messages can be routed directly to other remote UPnP devices by changing this host element. More precisely, by changing the IP address and sending the message to the corresponding UPnP Proxy, this proxy is capable of sending the message to the correct remote UPnP control point. The contents of an HTTP message can be ignored for now as if these messages will not (and may not) be changed during transfer. This technique follows the guidelines as mentioned before with the control protocol.

7.1.6 Presentation

Presentation is step 5 in UPnP networking. After a control point has discovered a device and retrieved a description of the device, the control point is ready to begin presentation. If a device has a URL for presentation, then the control point can retrieve a page from this URL, load the page into a browser and depending on the capabilities of the page, allow a user to control the device and/or view the status of the device. The degree to which each of these can be accomplished depends on the specific capabilities of the presentation page and device.

UPnP devices have embedded web servers because most of the communication protocols used run over HTTP. Like the Control and Eventing phases of the UPnP architecture, a control point can view a device's presentation page only after the control point has discovered the device. If the device is supplying a presentation page, its device description document contains some specific element.

UPnP Proxy Extension
To retrieve local presentation pages, the control point simply issues an HTTP GET request to the address specified in the device description. The device will return a presentation page in the body of the HTTP response.

Communication, with respect to the presentation phase, is based on HTTP. Therefore, the same techniques as used with Control and Eventing can be applied, thus configuring the presentation URL to be pointed at the local UPnP Proxy.

7.2 JXTA Protocols

As described in chapter 5.1.5 - Advertisements, advertisements are the basic unit of data exchange between peers to provide information on available services, peers, peer groups, pipes and endpoints. With advertisements, finding peers and all their different types of resources can be reduced to finding advertisements describing these resources.

For the graduation assignment it is sufficient to describe how the different JXTA protocols operate and how they can corporate to establish the connection needed. Chapter 6 contains the basics how to connect two or more UPnP networks together via JXTA using the pipe mechanism. This section also offers a solution to connect UPnP networks together without using the pipe mechanism. How this interconnection can be established without using the pipe mechanism will be shown in section 7.2.5 - Peer Resolver Protocol.

First of all there must be some peer group which is built up out of simple peers. This peer group is a collection of all joining UPnP networks (the peers). Within this peer group all joining peers need to discover each other to get knowledge about their resources. Only peers with some specific characteristics, i.e. UPnP Proxies, can participate within this peer group. The Peer Discovery Protocol, which defines a protocol for requesting the advertisements from other peers, can be used to accomplish this.

7.2.1 Peer Discovery Protocol

Peers can discover resources by sending a request to another peer, usually a rendezvous peer, and receive responses which contain advertisements describing the available resources on the P2P network.

The Peer Discovery Protocol (PDP) consists of two messages:

- the *Discovery Query Message*, which defines a request format to discover advertisements;
- the *Discovery Response Message*, which defines a response format for responding to a discovery request.

These message formats can be used to provide a mechanism for:

- Retrieving local JXTA advertisements;
- Retrieving remote JXTA advertisements;
- Publishing JXTA advertisements locally;
- Publishing JXTA advertisements remotely;
- Flushing local JXTA advertisements.

These JXTA advertisements are actually advertisements describing the UPnP Proxy. For most peers existing on a private local network, finding rendezvous and router peers is a critical factor to participate in a P2P network. Most of these private local networks are restricted by use of a firewall. A peer on such a local network has no capability to use direct discovery to perform discovery outside the local network. However, a peer might still be capable of performing indirect discovery using rendezvous and router peers on the local network.

Instead of taking care of finding a specific peer the advertisements available on the P2P network need to be found. A peer can discover an advertisement in three ways:

- without any discovery, using a cache;
- via Direct Discovery;
- via Indirect Discovery.

7.2.1.1 No Discovery

The easiest way to find advertisements for a peer is to eliminate discovery entirely. Instead of searching for advertisements, a cache can be used to provide information on peer resources. This method can be used to reduce network traffic (as there is no search process) and it will allow a peer to obtain nearly instantaneous results, unlike other active searching methods. Therefore, the cache must rely at least on some set of previously discovered advertisements or must contain some list with the IP addresses and ports hard-coded into the application itself.

Biggest drawback of using a cache is the potential that advertisements describe resources that are no longer available in the network, i.e. the cache is not up-to-date. This presents a problem when a peer attempts to engage a resource described by a faulty advertisement and fails to engage the service. This problem can be overcome by using timestamps, so every advertisement can expire, thereby removing them from the cache.

7.2.1.2 Direct Discovery

Peers existing on the same Local Area Network (LAN) might be capable of discovering each other directly without relying on a rendezvous peer to aid the discovery process. Direct discovery requires peers to use the broadcast or multicasting mechanisms. Direct discovery can be used for large networks, which may contain sub-LANs within this large LAN network.

When other peers have been discovered using the direct discovery mechanism, other advertisements can be discovered by communicating directly with the peers discovered (without using broadcast or multicast). Unfortunately, this technique can only be used to discover peers within the same LAN segments and cannot be used to discover peers outside the local network. Discovering peers outside the local network requires the indirect discovery mechanism.

7.2.1.3 Indirect Discovery

The indirect discovery mechanism relies on rendezvous peers to act as a source of known peers and advertisements, and to perform discovery on behalf of a simple peer. This technique can be applied by peers who are available on a local network to find peers without using the broadcast and multicast principles (which can of course not be used outside local networks).

There are two possible ways to discover peers outside the local network:

- **Propagation**: a rendezvous peer is used to traverse the discovery request to other peers on the P2P network that it knows about, including other rendezvous peers that also can propagate the request to other peers.
- **Cached Advertisements**: rendezvous peers can use cached advertisements to respond to discovery queries.

Using propagation and cached advertisements together with the timestamp technique, it forms an effective solution for rendezvous peers to cache a large number of advertisements. As each simple peer or rendezvous peer responds to some discovery request, the rendezvous peer can cache the responses for future use. This technique reduces network traffic further and increase network performance on the other hand.

UPnP Proxy Extension
For the UPnP-JXTA architecture, the indirect discovery mechanism is most applicable. It can be used to find resources available within some peer group, even from behind a firewall using the rendezvous protocol. As can be seen from the usage scenarios (chapter 3 – Usage Scenarios) the situation can occur that UPnP networks are physically separated and that a firewall is in between. Therefore, rendezvous peers need to be used to communicate between the different UPnP networks.

7.2.2 The Rendezvous Protocol

The rendezvous service is responsible not only for allowing a user to propagate messages to other peers but also for providing rendezvous peer services to other peers on the network. This section describes in short how a peer can use the Rendezvous Protocol (RVP) to connect to rendezvous peers and how this can be applied to the graduation assignment.

Before a peer can actually use a rendezvous peer to discover other peers, it must first discover the rendezvous peer itself by finding the Rendezvous Advertisement belonging to the rendezvous peer. After a rendezvous peer has been discovered, the peer sends a request to the rendezvous peer using the Endpoint service, which will be discussed later.

UPnP Proxy Extension
As already mentioned, the rendezvous mechanism can and must be used to discover other peers, i.e. UPnP networks. The UPnP Proxy must act as a peer, trying to find rendezvous peers by sending Discovery Query Messages for Rendezvous Advertisements. In this way the P2P network is not limited to a very small set of peers, but can be extended to a very large network. Within this network it is easy to discover other peers with the same services, providing one or more peer groups.

7.2.3 The Pipe Binding Protocol

Pipes are a construction within the JXTA specification that can send data to or receive data from a specific remote peer. Before a pipe can be used to communicate, it must be bound to a peer endpoint.

The endpoint is the lowest element in the network transport abstraction as defined by the JXTA specification (see also figure 8 – JXTA Protocol Stack). Endpoints are encapsulations of the native network interfaces provided by a peer. Endpoints are responsible for producing, sending, receiving, and consuming messages across the network.

Pipes are an abstraction that describes the connection between a sending endpoint and one or more receiving endpoints. Although pipes might appear as the providing service to access a network, the endpoint abstraction is responsible for the actual task of sending and receiving data over the network.

UPnP Proxy Extension
Pipes are specified by the JXTA architecture as unidirectional, meaning that data travels in one direction only. Pipes are also asynchronous which means that data can be sent or received at any time. This allows peers to act independently of other peers without any sort of state synchronization. A variant, which is more applicable for communication between UPnP networks, are bi-directional pipes. Bi-directional pipes can be easily created by defining two unidirectional pipes: one for sending data and one for receiving data.

Last variant, which can be used to send messages from one UPnP network to all other listening UPnP networks at once, is the so called propagation pipe. This one connects one output pipe to multiple input pipes. These propagation pipes provide the peer with a mechanism to broadcast data to multiple endpoints, see also chapter 6.3.2 – Peer Pipes.

Note that all communication with pipes is not secure. Secure pipes can be created by changing the Pipe Advertisement used when creating the input and output pipes. These secure pipes will use the Transport Security Layer protocol, which is a variant of the Secure Socket Layer (SSL) 3.0 standard, to secure the communication channel.

7.2.4 The Endpoint Routing Protocol

To determine how a message should be sent between two endpoints, a mechanism is required to allow a peer to discover route information. The Endpoint Routing Protocol (ERP) provides peers with a mechanism capable of determining a route between endpoints, allowing the peer to send or receive data.

As mentioned before, pipes are an abstraction built on top of endpoints. Pipes themselves are not responsible for the actual transmission and reception of data. Endpoints actually encapsulate a set of networking interfaces, allowing peers to send and receive data independently of the type of network transport layer used.

An endpoint is an interface to a set of network transports that allow data to be sent across the network. Network transport is assumed to be unreliable, even though actual endpoint protocol implementations might use reliable transport mechanisms, such as TCP/IP. This endpoint functionality does not have a protocol definition. Details about how data must be formatted before transport across the network are the responsibility of the endpoint protocol implementation itself.

There are multiple Endpoint Transport Implementations possible. Think of TCP/IP, HTTP, Transport Layer Security (TLS), etc. Propagating messages to a number of remote peers works in a similar fashion as using propagation pipes. However, unlike propagation pipes, the Endpoint service cannot propagate message across a firewall. This feature is offered by the Rendezvous protocol.

UPnP Proxy Extension
Communication between different UPnP networks is based on TCP/IP as transport mechanism. Note that using this particular endpoint protocol implementation directly makes the architecture for the UPnP – JXTA gateway less flexible because the architecture will be dependent on a particular network transport. Nevertheless, TCP/IP is used as a basic building block to set up an architecture to connect two or more UPnP networks. Eventually, other Endpoint Transport Implementations can be used to communicate between UPnP networks.

7.2.5 The Peer Resolver Protocol

The peer discovery protocol showed how to discover peers and advertisements using the Discovery service, but it did not address how the discovery service handles sending and receiving messages. The Discovery service is not responsible for sending its Discovery Query and Response Messages. Instead, the Discovery service is built on top of another service, the Resolver service, which handles sending and receiving of messages.

The Peer Resolver Protocol (PRP) defines a protocol for sending a generic query to a named handler located on another peer and processing a generic response to a query.

The Resolver service is responsible for wrapping a query string in a more generic message format and sending it to a specific handler on a remote peer. In the case of the Discovery service, the query string is the Discovery Query Message and the handler is the remote peer's Discovery service. On the remote peer, a Resolver service instance is responsible for passing an incoming message to the appropriate handler and sending any response generated by the handler.

In the general case, the Resolver service needs only two types of messages:

- *Resolver Query Message* – A generic message format for sending queries;
- *Resolver Response Message* – A generic message format for sending responses to queries.

These two message formats define generic messages to send queries and responses between peers, as can be seen in figure 23.

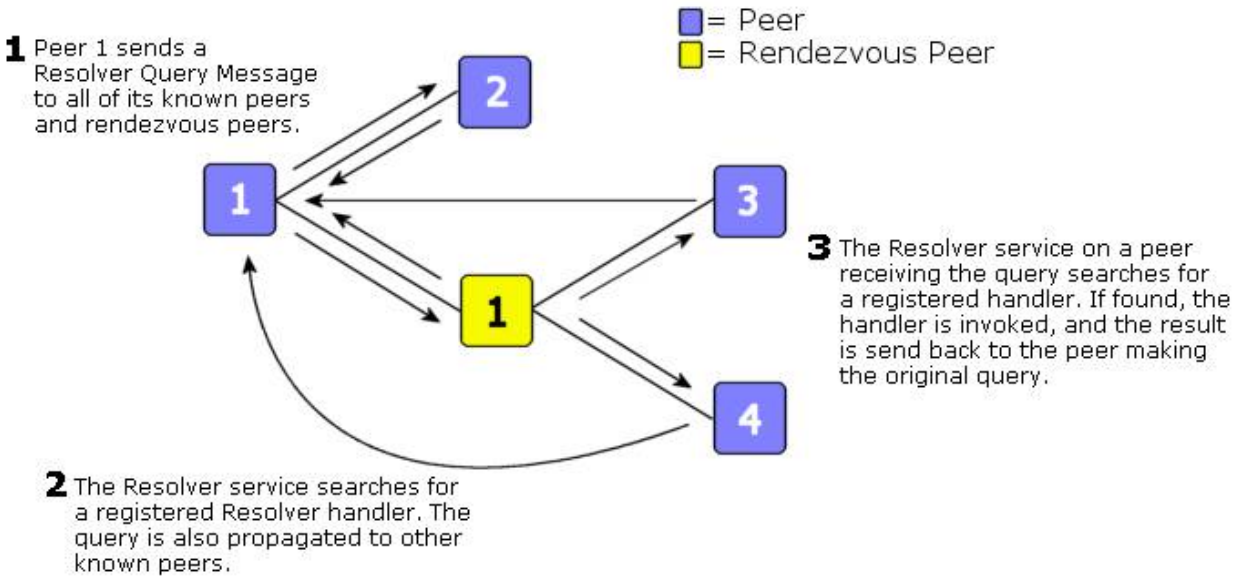


Figure 23: Exchange of Resolver messages

At each end, a handler registered with a peer group's Resolver service instance processes query strings and generates response strings (if necessary). One of the nice features is that the Resolver service does not require a response to a given query. Any peer's Resolver service that receives a Resolver Query Message attempt to find a suitable handler for it. If a matching handler is found, the Resolver passes it the message.

Queries send to other peers will be wrapped inside a Resolver Query Message using the following format:

```
<?xml version="1.0" encoding "UTF-8"?>
<jxta:ResolverQuery xmlns: jxta=http://jxta.org>
   <HandlerName> . . . </HandlerName>
   <Credential> . . . </Credential>
   <QueryID> . . . </QueryID>
   <SrcPeerID> . . . </SrcPeerID>
   <Query> . . . </Query>
</jxta:ResolverQuery>
```

Short explanation of the elements:

- **HandlerName**: A required element which contains a specific name identifying the handler that the Resolver service on the destination peer needs to invoke to process the query.
- **Credential**: An optional element that contains an authentication token to identify the peer within a peer group.
- **QueryID**: An optional element that defines an identifier for the query.
- **SrcPeerID**: A required element containing the ID of the peer.

- **Query**: A required element containing the query string that is sent to the remote peer's handler. This string could be anything. It is the responsibility of the handler to understand the query string, to process it and possibly generate a response message.

UPnP Proxy Extension
The Peer Resolver Protocol can be used to send UPnP related messages as a query to other peers. To call the correct handler when a UPnP message is wrapped inside a Resolver Query Message, there must be some definition of the different handlers needed. For every UPnP operation phase there must be one handler.

The previous sections (section 7.1.1 Addressing till 7.1.6 - Presentation) showed that the UPnP architecture is built up out of 6 different steps. Some of these UPnP messages involved needs to be sent to remote network(s). These messages can be put directly into the query element of the Resolver Query Message. Responses to these messages can also be put into the query element, if needed. To handle these messages at the remote side, it is required to define some remote handler for it. This handler needs to translate the query and transform it into a UPnP message corresponding to the message sent on the local network. As can be concluded from the previous sections, the UPnP messages that directly need to traverse to remote UPnP devices are within the steps:

- Step 3: control;
- Step 4: eventing;
- Step 5: presentation.

Addressing, discovery and description only apply for the local UPnP network as the UPnP Proxy acts on behalf of the remote UPnP devices. Communication between UPnP Proxies, such as retrieving the different UPnP descriptions, can also be provided by the Peer Resolver Protocol. Defining an appropriate handler and query to send and receive UPnP descriptions is sufficient to fulfill this task.

Using the Peer Resolver Protocol is another approach of UPnP – JXTA communication. It can be used instead of pipes to communicate between UPnP networks. As this approach has not been a research topic throughout the graduation assignment, more research is required to determine if this technique has some specific benefits or disadvantages with respect to the pipe mechanism.

7.3 Architecture Design Proposal

This part of the document defines an architecture design proposal based on chapter 6 – Network setup and chapter 7 – UPnP-JXTA gateway architecture. These chapters contain the fundamental parts about the techniques used and solutions proposed to design a UPnP – JXTA architecture. A number of design decisions have been made and are discussed in detail within these chapters.

This section can be considered as a summarization of all conclusions and abbreviations and will present (in short) an approach to setup a UPnP – JXTA architecture. For an in-depth analysis the corresponding sections or chapters needs to be consulted.

7.3.1. Network architecture

According to the specification as discussed in chapter 2 – Specification, there need to be two (or more) UPnP networks that must be connected using the JXTA protocol. To accomplish this kind of setup, some specific hardware is required. Gateways are the most appropriate hardware element that will be used because gateways operate at the application layer. Information received will be interpreted prior before sending to the corresponding receiver. Messages that need to be sent to remote UPnP devices initiated from inside the local UPnP network will therefore be interpreted before forwarding (see section 6.1 – Interconnection).

UPnP networks can be considered as different subnets. The addressing structure of these subnets may be identical, therefore containing duplicate IP addresses. When these networks are coupled together a network addressing collision can occur. To overcome this problem all IP addresses of all remote devices need to be changed to the gateway IP (see section 6.2.2 Gateway mapping). This gateway mapping keeps a list of all remote UPnP devices. This also applies to the device and service descriptions belonging to all remote UPnP devices. To avoid name collisions, that is when there exist duplicate UPnP devices containing identical names; separation on the gateway ensures that this cannot happen between different UPnP networks. An architecture problem that still exists is when there are multiple identical UPnP devices available within the same local UPnP network. This is, as already stated, a problem of the UPnP Device Architecture and will therefore not be taken into account for this graduation assignment.

To connect the different UPnP networks together the JXTA architecture will be used. Every UPnP network can be considered as a simple peer. This peer will connect to other peers, forming a particular peer group. Connecting the gateways as peers in peer groups provides a service where only connected UPnP networks will be available. Security measurements can be applied if necessary. With the help of different peer groups multiple collections of UPnP networks can be distinguished.

To communicate between the gateways so-called propagation pipes need to be used. This kind of pipes propagates messages from one gateway to all other gateways within the scope of the peer group. Most of the communication will be based on this principle (as can be seen later on), but sometimes single-point communication is necessary. In this case the point-to-point pipe needs to be used (see section 6.3 – JXTA Architecture). This situation can occur when for example a local UPnP device performs an action on a specific remote UPnP device.

7.3.2 UPnP – JXTA Gateway Architecture

The network setup proposed uses the proxy mechanism to establish a connection between UPnP networks. This proxy is included within the gateway. The proxy service can act on behalf of another service (the original service). The proxy service can be used for passing requests to a remote UPnP device on behalf of a local UPnP device, but also for discovery, that is when remote UPnP devices are advertised within the local UPnP network. When a control point within the local UPnP network wants to know the device and service descriptions from a remote UPnP device, the gateway can act as if the remote UPnP device is directly available within the local UPnP network. From now on the gateway will also be called the UPnP Proxy.

The UPnP Proxy will also contain functionality to control the JXTA configuration from within the local UPnP network. This introduces the possibility for example to retrieve from within the local UPnP network a list of available peer groups or to change from one peer group to another. Therefore extra device and service descriptions need to be introduced (these descriptions can be found in Appendix B and C). As the UPnP Proxy also needs to discover local UPnP devices it is both a control point and a device.

The following extensions with respect to the UPnP Device Architecture contain the UPnP Proxy extensions that are needed to achieve the behavior desired. It is actually a summarization of the extensions as proposed in chapter 7 – UPnP – JXTA Gateway Architecture.

UPnP Proxy Addressing Extension
To use the original addressing architecture, a mapping mechanism needs to be introduced. This mapping mechanism keeps a list of all remote UPnP devices. When all necessary information has been gathered by the UPnP Proxy, this list of remote UPnP devices can be published within the local UPnP network. Within the local UPnP network, every IP address of a remote UPnP device needs to be mapped to an IP address within the subnet of the local UPnP network. Using the gateway as proposed in section 6.2.2 – Gateway mapping, the IP address where to map to will be the IP address of the UPnP Proxy.

UPnP Proxy Discovery Extension
Discovery of local UPnP devices can be achieved to use the UPnP Proxy as a control point. In this way all local UPnP devices can be discovered using the original discovery mechanism. To publish the list of local UPnP devices within other remote UPnP networks, the UPnP Proxy must always have a list of all local UPnP devices available. To reduce latency and network traffic also the corresponding device and service descriptions can be sent to the remote UPnP Proxy. When a new list of devices is received by the remote UPnP Proxy, any outstanding UPnP Proxy advertisement needs to be revoked and republished with the new device list included.

UPnP Proxy Description Extension
This phase requires the availability of all remote UPnP device and service descriptions within the local UPnP network. There is no possibility for a control point to issue a description outside the local UPnP network. Therefore, all descriptions need to be know (and cached) at time of publishing the advertisements within the local UPnP network. To achieve a cache which is always up-to-date, all changes need to be communicated directly to all other joining remote UPnP networks. This implies that at startup the complete descriptions will be sent but when a remote UPnP network changes, this change will be know with a delay by all other UPnP networks. Note that at startup the complete descriptions need to be sent but that when a description changes or needs to be removed only a change message needs to be sent. Within this delay the cache contains incorrect information and for example any request issued for a remote UPnP device not available anymore will result automatically in a time-out. Therefore, as long as the discovery advertisements from a remote UPnP device have not expired, a control point on the local UPnP network may assume that a remote UPnP device and its services are still available.

UPnP Proxy Control Extension
Within a UPnP device description, each service element has a control element where all control messages needs to be sent to. Changing this element within the device descriptions prior to publishing ensures that any control message initiated from within the local UPnP network will be pointed to the UPnP Proxy. When a control message is received by the UPnP Proxy, this message can be routed to the corresponding remote UPnP Proxy containing the remote UPnP device, if necessary. Changing the control element implies that the headers of the message also needs to be changed. Therefore, before the control message is sent to the corresponding remote UPnP Proxy, the header needs to be changed and set to point to the remote UPnP device. When the remote UPnP Proxy receives this message it only needs to forward the message, initiating a connection with the remote UPnP device. As long as no response has been returned by the remote UPnP device, the connection is kept alive till a timeout has occurred. According to the UPnP Device Architecture, if the service can not respond in time, it should return early and send an event when complete. This will actually not guarantee that the early response will be received in time by the sender itself. Therefore, the local UPnP Proxy needs to generate an early response message on behalf of the remote UPnP device within thirty seconds (only if the early response message is not received within time).

UPnP Proxy Eventing Extension
Every message issued by this mechanism is based on HTTP. Messages that need to be sent to a remote UPnP device can be sent using the UPnP Proxy. Therefore, the same techniques must be used as with the UPnP Proxy Control Extension. More precisely, by changing the event element of the remote UPnP device description, event messages will be pointed to the local UPnP Proxy. When such a message is received by the UPnP Proxy, only the headers of the message need to be set before sending the message to the corresponding remote UPnP Proxy.

UPnP Proxy Presentation Extension
To retrieve remote presentation pages, the control point needs to issue an HTTP GET request directly to the local UPnP Proxy. As all communication is also HTTP based, the same techniques as mentioned in the UPnP Proxy Discovery Extension can be applied, thus configuring the presentation URL to be pointed to the local UPnP Proxy and changing the headers of any presentation message pointed to a remote UPnP device.

UPnP Proxy JXTA Extension
Chapter 6 – Network setup contains the basics how to connect two or more UPnP networks together using the pipe mechanism of JXTA. To use this mechanism all UPnP Proxies need to be configured as simple peers. Peer groups can be formed to differentiate collections of UPnP networks (actually simple peers). To find other peers JXTA contains different options. The indirect discovery mechanism is most appropriate to be used for this kind of architecture as rendezvous peers can be used to find peers, even from behind a firewall. When the indirect discovery mechanism is build up out of propagation and cached advertisements, together with a timestamp, it forms an effective solution for rendezvous peers to cache a large number of advertisements. Using propagation pipes in combination with point-to-point pipes as mentioned earlier, provides the UPnP Proxy with a mechanism to broadcast messages to multiple other UPnP Proxies in once and to sent specific actions to only one UPnP Proxy. The Endpoint Routing protocol determines how a message should be sent between two peers (endpoints), allowing peers to send and receive data independently of the type of network transport layer used. Multiple endpoint implementations are possible, think of TCP/IP, HTTP, TLS, etc. For this graduation assignment the assumption has been made that the classic TCP/IP endpoint implementation will be used as the UPnP communication is also TCP/IP based.

Another approach has been introduced (see section 7.2.5 – The Peer Resolver Protocol) in stead of using JXTA pipes as transport mechanism. The Peer Resolver Protocol can be used to send UPnP related messages as a query to other peers. This protocol defines a generic query to a named handler which can be used to wrap the content of a UPnP message. As this approach has not been a research topic throughout the graduation assignment, it is just meant to indicate that there is more than one approach to set up a UPnP-JXTA gateway architecture. It can be considered as an extra feature, offered by the JXTA architecture, which can be used for sending UPnP messages or just for extending the possibilities of the UPnP Proxies.

More research is required to determine if this approach has some specific benefits or disadvantages with respect to the pipe mechanism. On the other hand, this approach can most likely be applied by the architecture as presented, as using the Peer Resolver Protocol is only for the transportation of the messages between the different UPnP networks.

8. Conclusions

There are a multitude of connectivity methods to connect all sorts of devices, PCs, mobile phones, cameras, handheld computers and so on. All these methods increase the need for self-configuring networks to allow devices to join and leave networks easily. Home networks can be extended with technologies that automate device and service discovery and control. First step towards this automotive control of devices has been made by UPnP. This technology defines a base set of standards for all sorts of devices within the home network. These standards define how to describe the devices and services provided.

JXTA is another technology, based on a set of open, generalized peer-to-peer protocols that allow any connected device on the network to communicate and collaborate as peers. Primary goal of this technology is to provide a platform with the basic functions necessary for a P2P network. Just like UPnP, this technology defines services to discover other peers and to communicate with these peers.

The graduation assignment started to find a solution to bridge two (or more) UPnP home networks using JXTA. To bring more focus in the graduation assignment important aspects were investigated, like bridging, interoperability and scalability. These aspects helped with addressing the boundary conditions of the graduation assignment.

To gain more insight in the functionality needed by connecting UPnP networks together, some usage scenarios were introduced. These scenarios address the aspects which need to be taken care of, like transparency, security and interconnection.

In general, networks may be interconnected in various ways, through repeaters, bridges, routers and gateways. Investigated is what kind of hardware component is most applicable for the desired setup. There were a number of options, all with their own advantages and disadvantages. The gateway architecture approaches the best solution to connect multiple home networks together. Especially the fact that a gateway operate at the application layer, which offers the possibility to gain more control then other options, could be used very efficiently to gain more user control within the home network.

Mapping the remote UPnP devices within the local UPnP network by the gateway can be done in two ways. Virtual mapping will display all remote UPnP devices within the local UPnP network as if they actually join one big network. The other approach is gateway mapping, which maps all remote UPnP devices directly on the gateway. Gateway mapping is preferred as just one IP address is needed for every remote UPnP device and there is a clear separation between local and remote UPnP networks and devices.

The gateway mapping introduces the need for a proxy. Every remote UPnP device is mapped directly on the gateway. Communication with other remote UPnP networks will be done by the gateway. The gateway will act on behalf of local UPnP control points and devices. This built-in proxy can also be used to bring the remote UPnP devices and services within the local UPnP network. This will reduce network traffic and leverages more control as a whole.

JXTA plays the central key with respect to communication between different UPnP networks. Peer groups need to be used to define a service where only connected UPnP networks are available. Different peer groups can be set up to define multiple collections of UPnP networks. These groups can become easily secured, just like the communication between the different UPnP networks. JXTA offers a number of techniques, like point-to-point pipes and propagation pipes, to communicate between these UPnP networks. Which technique must be used at which time depends on what type of message needs to be sent and where the message needs to be sent to.

This approach leads to an architecture, capable of detecting and controlling UPnP devices within different networks on a UPnP based manner, without the scalability problems that come with UPnP. The architecture is based on the UPnP Device Architecture and is extended where necessary. The gateway has become both a device and a control point. The control point functionality is needed to discover local UPnP devices, the device functionality to advertise remote UPnP devices. The gateway (actually the proxy) is used to keep the list of remote UPnP devices up-to-date at all times. Inconsistency can only occur within the time needed to communicate network changes to all remote UPnP networks. This inconsistency can be overcome by using the original time-out mechanism as defined by the UPnP Device Architecture. Most of the techniques used to bring all remote UPnP devices within the local UPnP network are based on changing the descriptions belonging to devices and to adapt the payload of messages received by the gateway, dependent on the action that needs to be performed.

9. Future Work

The research done for this graduation assignment is already a step in the direction of connecting multiple UPnP networks with JXTA as communication technology. Research would not be research if it would not result in some new research questions that could be investigated in the future. After this graduation assignment a number of questions are open and left untouched. The rest of this section provides possible future research topics that could be investigated.

This graduation assignment is done in the area of UPnP and JXTA. There was no question if this combination is the best to use. Maybe another technology, like UPnP-Jini [10] is more appropriate for extending UPnP networks.

The approach to use a proxy mechanism to control UPnP networks could be investigated more. Applying proxy-filters, for example to limit the publication of UPnP devices or to search for particular devices within the network, is one of the great benefits that could be reached with this technique.

One part of chapter 4 deals with the UPnP A/V architecture. This architecture uses an out of band communication mechanism. This means that communication between devices is done without the help of a control point. But what does this mean if out of band communication is needed between UPnP devices available in different networks? How could it be applied on the proxy mechanism? More questions can arise using this architecture which need to be investigated.

Using a UPnP-JXTA gateway architecture introduces extended scalability with respect to UPnP. Chapter 7 introduces a communication mechanism to keep all connected UPnP networks up-to-date, which introduces extra network traffic. This graduation assignment did not point at creating the optimal solution to connect as many as possible UPnP networks. Investigated could be for example how many networks can be connected before network collisions occur and what factors are involved in different scenarios.

After choosing the best implementation approaches it needs to be really tested. After implementing the chosen implementation approach a question could be answered like, what is the maximum time to wait for the response of some action performed on a device outside the local UPnP network? When, for example, this response time is too long, the next step could be to investigate if it is possible to add some additional caching, and if so, what kind of caching level is needed.

References

[1] UPnP Forum: http://www.upnp.org

[2] Project JXTA: http://www.jxta.org

[3] Sun Microsystems, http://www.sun.com

[4] Institute of Electrical and Electronics Engineers (IEEE)
http://www.ieee.com

[5] Dynamic Host Configuration Protocol, RFC2131
http://www.ietf.org/rfc/rfc2131.txt

[6] Ritchie J. and Kuehnel T., UPnP AV architecture 0.83, June 2002

[7] UPnP Forum, UPnP Device Architecture 1.0, December 2003

[8] Auto-IP: Automatically Choosing an IP Address in an Ad-Hoc Ipv4 Network.
http://www.upnp.org/download/draft-ietf-zeroconf-ipv4-linklocal-01-Apr.txt

[9] eXtensible Markup Language (XML): www.w3.org/XML/

[10] Jini Network Technology
http://www.jini.org

[11] Computernetwerken
4th edition 2003, Prentice-Hall PTR Publishing
Andrew S. Tanenbaum

[12] UPnP Design by Example
1st edition 2003, Intel Press
Michael Jeronimo, Jack Weast

[13] JXTA
1st edition, 2002, New Riders Publishing
Brendon J. Wilson

List of figures and tables

List of figures:

List of tables:

Abbreviations and Acronyms

API	Application Programming Interface
AV	Audio Video
CD	Compact Disc
CE	Consumer Electronics
DHCP	Dynamic Host Configuration Protocol
DLL	Data Link Layer
DVD	Digital Versatile Disc
ERP	Endpoint Routing Protocol
FTP	File Transfer Protocol
GENA	General Event Notification Architecture
IC	Integrated Circuit
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IR	Infrared
I/O	Input Output
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
LAN	Local Area Network
LLC	Logical Link Control
MAC	Media Access Control
NAT	Network Address Translation
PC	Personal Computer
P2P	Peer to Peer
PDA	Personal Digital Assistant
PDU	Protocol Data Unit
PDP	Peer Discovery Protocol
PRP	Peer Resolver Protocol
RC	Remote Control
RVP	Rendezvous Protocol
SAN	Systems Architecture and Networks
SSDP	Simple Service Discovery Protocol
SOAP	Simple Object Access Protocol

SPATION	Services, Platforms and Applications for Transparent Information Management in an In-Home Network
SSA	Storage Systems and Applications
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TV	Television
UDP	User Datagram Protocol
UPnP	Universal Plug and Play
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VCR	Video Cassette Recorder
XML	eXtensible Markup Language

Appendix A

Graduation Assignment

UPnP – JXTA Bridging

Background
The group Storage Systems and Applications of Philips Research the Netherlands (located in Eindhoven) offers a graduation assignment on the topic of *UPNP-JXTA bridging*.

We assume that in the future all Consumer Electronics (CE) devices in the home will become interconnected. Users will be faced with the issues of storing and retrieving content in a distributed system. It is also interesting to share content on a peer-to-peer basis between homes. For in the home the UPnP middle ware standard is used, the JXTA standard is used between homes.

UPnP is a networking middleware standard. This standard provides ways to dynamically discover other UPnP enabled devices and their capabilities. This discovery mechanism uses UDP broadcast for the discovery and therefore is limited to Local Area Networks. UPnP is a mature and well-specified standard and more and more CE manufacturers are investigating UPnP to enhance their products.

JXTA is a standard for peer-to-peer networking. It is a transport layer platform agnostic protocol based on XML. It can use UDP broadcast and HTTP for discovery of other JXTA enabled devices. Within JXTA different mechanisms are defined to traverse firewalls and improve scalability, e.g. by use of rendezvous and relay peer. JXTA is less mature than UPnP, but it provides a technology for scalable home-to-home networking.

For Philips both in-home and home-to-home networking is important. UPnP gets more and more momentum in the CE market and it looks like UPnP will be the de-facto standard in in-home networking for networked enabled CE-devices. JXTA provides a solution for home-to-home networking where UPnP is not applicable because of its scalability.

Goals and task
Both technologies have their own advantages in their specific areas, UPnP for in-home and JXTA for home-to-home. The challenge of the assignment is to use the best of both technologies and make the technologies able to communicate with each other. Therefore a so-called bridge between UPnP and JXTA is needed. This bridge understands both protocols and translates UPnP messages into JXTA message and vice versa.

The student assignment is to investigate both the JXTA and UPnP protocols and come up with a solution to bridge both technologies. The assignment will include the implementation of a UPnP-JXTA bridge in a small demonstration set-up (preferably written in Java). For example the demonstration set-up can be a UPnP home network where content (e.g. photos) is available; this content needs to be shared between different UPnP home networks using JXTA.

References
Websites of the standards.

Pre-requisites
Knowledge of networking, software architecture, distributed systems, middleware.

Appendix B

UPnP Proxy Device Description

```
<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
   <specVersion>
      <major>1</major>
      <minor>0</minor>
   </specVersion>
   <device>
      <deviceType>urn:schemas-upnp-org:device:upnproxy:1</deviceType>
      <friendlyName>Gertjan's UPnPProxy</friendlyName>
      <manufacturer>Gertjan</manufacturer>
      <manufacturerURL>http://www.philips.com</manufacturerURL>
      <modelDescription>UPnP Proxy Service</modelDescription>
      <modelName>UPnPProxy</modelName>
      <modelNumber>1.0</modelNumber>
      <modelURL>http://www.philips.com</modelURL>
      <serialNumber>135790</serialNumber>
      <UDN>uuid:dac8-a0fc-cfc9-e0c9</UDN>
      <UPC>1234567890</UPC>
      <iconList>
         <icon>
            <mimetype>image/gif</mimetype>
            <width>48</width>
            <height>32</height>
            <depth>8</depth>
            <url>icon.gif</url>
         </icon>
      </iconList>
      <serviceList>
         <service>
            <serviceType>urn:schemas-upnp-org:service:proxy:1</serviceType>
            <serviceId>urn:schemas-upnp-org:serviceId:proxy:1</serviceId>
            <SCPDURL>/service/description.xml</SCPDURL>
            <controlURL>/service/control</controlURL>
            <eventSubURL>/service/eventSub</eventSubURL>
         </service>
      </serviceList>
      <presentationURL>http://www.philips.com</presentationURL>
   </device>
</root>
```

Appendix C

UPnP Proxy Service Description

```
<?xml version="1.0"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0" >
   <specVersion>
      <major>1</major>
      <minor>0</minor>
   </specVersion>
   <actionList>
      <action>
         <name>changeJXTAGroup</name>
         <argumentList>
            <argument>
               <name>Group</name>
               <relatedStateVariable>Group</relatedStateVariable>
               <direction>in</direction>
            </argument>
            <argument>
               <name>Result</name>
               <relatedStateVariable>Result</relatedStateVariable>
               <direction>out</direction>
            </argument>
         </argumentList>
      </action>
      <action>
         <name>createNewJXTAGroup</name>
         <argumentList>
            <argument>
               <name>Group</name>
               <relatedStateVariable>Group</relatedStateVariable>
               <direction>in</direction>
            </argument>
            <argument>
               <name>Result</name>
               <relatedStateVariable>Result</relatedStateVariable>
               <direction>out</direction>
            </argument>
         </argumentList>
      </action>
      <action>
         <name>reload</name>
         <argumentList>
            <argument>
               <name>JXTAGroups</name>
               <relatedStateVariable>JXTAGroups</relatedStateVariable>
               <direction>out</direction>
            </argument>
         </argumentList>
      </action>
      <action>
         <name>CurrentConnection</name>
         <argumentList>
            <argument>
               <name>ConnectedTo</name>
               <relatedStateVariable>ConnGroup</relatedStateVariable>
               <direction>out</direction>
            </argument>
         </argumentList>
      </action>
      <action>
         <name>getJXTAGroups</name>
         <argumentList>
            <argument>
               <name>JXTAGroups</name>
               <relatedStateVariable>JXTAGroups</relatedStateVariable>
               <direction>out</direction>
            </argument>
         </argumentList>
      </action>
```

```
      <action>
         <name>getJXTAPeers</name>
         <argumentList>
            <argument>
               <name>JXTAPeers</name>
               <relatedStateVariable>JXTAPeers</relatedStateVariable>
               <direction>out</direction>
            </argument>
         </argumentList>
      </action>
   </actionList>
   <serviceStateTable>
      <stateVariable sendEvents="yes">
         <name>ConnectedTo</name>
         <dataType>String</dataType>
      </stateVariable>
      <stateVariable sendEvents="no">
         <name>JXTAGroups</name>
         <dataType>String</dataType>
      </stateVariable>
      <stateVariable sendEvents="yes">
         <name>JXTAPeers</name>
         <dataType>String</dataType>
      </stateVariable>
      <stateVariable sendEvents="no">
         <name>Group</name>
         <dataType>String</dataType>
      </stateVariable>
      <stateVariable sendEvents="no">
         <name>Result</name>
         <dataType>String</dataType>
      </stateVariable>
      <stateVariable sendEvents="no">
         <name>ConnGroup</name>
         <dataType>String</dataType>
      </stateVariable>
   </serviceStateTable>
</scpd>
```

Appendix D

UPnP Proxy Extended Device Description

```
<?xml version="1.0" ?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
   <specVersion>
      <major>1</major>
      <minor>0</minor>
   </specVersion>
   <device proxy:Index="0">
      <deviceType>urn:schemas-upnp-org:device:light:1</deviceType>
      <friendlyName>CyberGarage Light Device</friendlyName>
      <manufacturer>CyberGarage</manufacturer>
      <manufacturerURL>http://www.cybergarage.org</manufacturerURL>
      <modelDescription>CyberUPnP Light Device</modelDescription>
      <modelName>Light</modelName>
      <modelNumber>1.0</modelNumber>
      <modelURL>http://www.cybergarage.org</modelURL>
      <serialNumber>1234567890</serialNumber>
      <UDN>uuid:dac8-a0fc-cfc9-e0c9</UDN>
      <UPC>1234567890</UPC>
      <iconList>
         <icon>
            <mimetype>image/gif</mimetype>
            <width>48</width>
            <height>32</height>
            <depth>8</depth>
            <url>icon.gif</url>
         </icon>
      </iconList>
      <serviceList>
         <service>
            <serviceType>urn:schemas-upnp-org:service:power:1</serviceType>
            <serviceId>urn:schemas-upnp-org:serviceId:power:1</serviceId>
            <SCPDURL>/service/power/description.xml</SCPDURL>
            <controlURL>/service/power/control</controlURL>
            <eventSubURL>/service/power/eventSub</eventSubURL>
         </service>
      </serviceList>
      <presentationURL>http://www.cybergarage.org</presentationURL>
   </device>
</root>
```

UPnP Proxy Extended Service Description

```
<?xml version="1.0"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0" >
   <specVersion>
      <major>1</major>
      <minor>0</minor>
   </specVersion>
   <actionList>
      <action proxy:Index=”0”>
         <name>SetPower</name>
         <argumentList>
            <argument>
               <name>Power</name>
               <relatedStateVariable>Power</relatedStateVariable>
               <direction>in</direction>
            </argument>
            <argument>
               <name>Result</name>
               <relatedStateVariable>Result</relatedStateVariable>
               <direction>out</direction>
            </argument>
         </argumentList>
      </action>
      <action>
         <name>GetPower</name>
         <argumentList>
            <argument>
               <name>Power</name>
               <relatedStateVariable>Power</relatedStateVariable>
               <direction>out</direction>
            </argument>
         </argumentList>
      </action>
   </actionList>
   <serviceStateTable>
      <stateVariable sendEvents="yes">
         <name>Power</name>
         <dataType>boolean</dataType>
         <allowedValueList>
            <allowedValue>0</allowedValue>
            <allowedValue>1</allowedValue>
         </allowedValueList>
         <allowedValueRange>
            <maximum>123</maximum>
            <minimum>19</minimum>
            <step>1</step>
         </allowedValueRange>
      </stateVariable>
      <stateVariable sendEvents="no">
         <name>Result</name>
         <dataType>boolean</dataType>
      </stateVariable>
   </serviceStateTable>
</scpd>
```

Appendix E

HTTP Status Codes

Number	Meaning	Description
2xx	Success	The action was successfully received, understood and accepted.
3xx	Redirection	Further action must be taken to complete the request.
4xx	Client Error	The syntax is invalid or cannot be fulfilled.
5xx	Server Error	The server failed to fulfill an apparently valid request.