Abstract

Within residential networks digital devices may share information between themselves through several discovery standards such as Bluetooth, Bonjour, or Universal Plug & Play. Similarly, all residential services are discovered easily so that they can be interconnected and, consequently, cooperate. However, all of these discovery methods are only possible as far as local networks are concerned. Accessing services offered by remote devices is a functionality that is not supported by these standards, resulting in the fact that consumers are requesting this functionality to an ever increasing degree. Likewise, IP Multimedia Subsystem is the wired network which includes the internet and the mobile phone networks, providing features such as integrated delivery of multimedia services, or mobility and roaming; hence causing benefits for consumers. This is why companies are working on the discovery of remote services using such technologies. So far, this functionality has been embedded inside the operation of service discovery standards. However, since the stack architecture of these mechanisms presents many underlying protocols and other built-in functionalities, the practical implementation of the required functionality is fairly complex. In addition, consumers would have to care about the service discovery standards which are serving them. Therefore, implementing this remote functionality outside the scope of operation of the service discovery standards using IP Multimedia Subsystem is the current aim of the companies and the purpose of this project. In this way, consumers would be able to use services offered by remote devices as if they appeared to be similar to services offered by local devices.

This project demonstrates that remote services are virtualized to the extent that they appear, so that they appear to be local services. Thus, the concept of Service Virtualization has been explained in the context of residential networks, as well as a certain number of use cases of Service Virtualization have been presented in order to reinforce its importance for consumers compared to current remote access methods. A solution architecture for Service Virtualization for a particular use case has been designed, explaining the parts that comprise Service Virtualization functionality and how a personal video recorder is brought into a local network using Service Virtualization. As a final step, a prototype implementation has been carried out as a real application; both the framework and how the prototype has been implemented have been explained, hence verifying the concept of Service Virtualization using real devices.

For the realization of the present project Universal Plug & Play and IP Multimedia Subsystem are the technologies used as service discovery mechanism and wired core network, respectively. However, since this remote functionality is decoupled from the service discovery mechanism, the possibility of using another service discovery standard as another alternative is feasible. Service Virtualization provides good experiences to consumers, since they can consume remote services with great ease of use without caring which service discovery mechanism and which operator are serving them; that is, using any remote device from anywhere through completing a few easy steps. Finally, since IMS is used as wired network, SV shall be used in the context of the third generation networks, letting operators offer new services with added values for customers.

Preface

This project has been realized in cooperation between the Faculty of Engineering and Science of University of Agder (Norway) and the School of Engineering of San Sebastian of University of Navarra (Spain). It has been carried out from September 2008 to March 2009, within the master thesis course IKT590.

My thesis project has been carried out in Agder Mobility Lab. It is based on previous work in a joint UiA-Ericsson project with focus on both the third generation networks and service discovery standards applied for residential networks.

I am grateful to Pål Grandal and Tor Erik Christensen, members of the UiA International Office, for the hearty welcome to UiA and Grimstad. I also thank my technical supervisors PhD students Andreas Häber and Selo Sulistyo, as this project would not have been realized without their excellent guidance and advice. Finally, I want to thank my main supervisor Stein Bergsmark for his support throughout this project.

Grimstad, March 2009

Jesús Gómez Ruiz de Mier

Contents

CHAPTER 1: INTRODUCTION 1

1.1 Background....1
1.1.1 Need of accessing numerous of personal devices....1
1.1.2 Service discovery for local devices....2
1.1.3 Towards convergent networks....2
1.1.4 Easy access to remote devices....3
1.1.5 Service Virtualization as a solution....4
1.2 Project definition....4
1.3 Project solution....5
1.3.1 Project description....5
1.3.2 Solution approach....6
1.4 Report outline....6

CHAPTER 2: THEORETICAL FRAMEWORK 7

2.1 Service discovery....7
2.1.1 Service discovery in local networks....7
2.1.2 UPnP Concepts....8
2.1.3 Scenario example....9
2.2 IP Multimedia Subsystem....12
2.2.1 IMS Concepts....12
2.2.2 Scenario example....14
2.3 Remote service access....15
2.3.1 Definition....15
2.3.2 Remote service awareness....16
2.3.3 Remote service usage....17

CHAPTER 3: USE CASES FOR SERVICE VIRTUALIZATION 19

3.1 Service Virtualization in residential networks....19
3.2 Remote multimedia access use case....20
3.2.1 Streaming media....20
3.2.2 Placeshifting of TV delivery....22
3.3 Remote upload use case....24
3.3.1 Live photo album....24
3.3.2 Live video recording....26
3.4 Remote control use case....28
3.4.1 Building management....28
3.4.2 Home security....29
3.5 Features of Service Virtualization....31
3.5.1 Easy remote access....31
3.5.2 Mobility....32
3.5.3 Services instead of devices....32
3.5.4 Considerations....33

CHAPTER 4: SOLUTION DESIGN FOR SERVICE VIRTUALIZATION 34

4.1 Solution goal....34

4.2 Requirements **36**
4.3 Solution architectural design **36**
4.4 Service Virtualizer design **37**
4.5 Solution: Principles of operation **38**
4.5.1 Phase 1: Available devices discovery 38
4.5.2 Phase 2: Access video source 39
4.5.3 Phase 3: Access video destination 40
4.5.4 Phase 4: Video recording 41

CHAPTER 5: PROTOTYPE IMPLEMENTATION FOR SERVICE VIRTUALIZATION **42**

5.1 Implementation goal **42**
5.2 Implementation architecture **43**
5.3 Service Virtualizer implementation **44**
5.4 Implementation testing **46**
5.5 Implementation verification **46**

CHAPTER 6: RESULTS **48**

6.1 Use cases results **48**
6.2 Solution architecture results **48**
6.3 Prototype implementation results **49**

CHAPTER 7: DISCUSSION **51**

7.1 Realization of use cases **51**
7.2 Solution design for Service Virtualization **52**
7.3 Considerations adopted for prototype implementation **52**
7.3.1 Incomplete Service Virtualizer implementation 52
7.3.2 Real application instead of emulation technology 52
7.3.3 Devices used for the implementation 53
7.4 The prototype implementation process **53**
7.4.1 Incomplete methods embedded in HIGA 53
7.4.2 SIP package size 54

CHAPTER 8: CONCLUSION **55**

8.1 Main conclusions **55**
8.2 Further work **56**

REFERENCES **57**

APPENDICES **59**

Appendix A: Glossary and Abbreviations **59**
Appendix B: Prototype implementation **61**
Appendix C: SIP messages **63**
Appendix D: SSDP messages **68**

Figures

Figure 1.1: Solution architecture.....5
Figure 2.1: UPnP architecture protocol stack.....9
Figure 2.2: Watching a movie in a local network through UPnP.....10
Figure 2.3: Flow diagram between a media renderer and a media server.....11
Figure 2.4: IMS architecture overview.....13
Figure 2.5: Signalling flow diagram between two different locations using IMS.....15
Figure 2.6: Logic diagram for remote service awareness.....16
Figure 2.7: Flow diagram for remote service awareness.....17
Figure 2.8: Signalling flow diagram for remote service usage.....18
Figure 3.1: General logic diagram for SV.....20
Figure 3.2: Logic diagram for streaming media to a car.....21
Figure 3.3: High-level flow diagram between remote media server and local screens.....22
Figure 3.4: Logic diagram for TV placeshifting to job office.....23
Figure 3.5: High-level flow diagram between remote digital satellite system and local TV.....24
Figure 3.6: Logic diagram for uploading photos to the digital album.....25
Figure 3.7: High-level flow diagram between remote photo album and digital camera.....26
Figure 3.8: Logic diagram for recording a live documentary to the PVR.....27
Figure 3.9: High-level flow diagram between remote PVR and video camera.....27
Figure 3.10: Logic diagram for scheduling a washing machine from a bus.....28
Figure 3.11: High-level flow diagram between remote washing machine and PDA-phone.....29
Figure 3.12: Logic diagram for remote monitoring of house security.....30
Figure 3.13: High-level flow diagram between security camera and remote monitoring screen.....31
Figure 4.1: Logic diagram solution using service virtualizaton.....35
Figure 4.2: SV phases for the designed solution.....35
Figure 4.3: Solution architectural design.....37
Figure 4.4: Stack architecture of Service Virtualizer.....37
Figure 4.5: Searching for available devices in the vicinity.....39
Figure 4.6: Getting the video from a video camera.....40
Figure 4.7: Setting up PVR.....41
Figure 4.8: Importing video from the video camera to the PVR.....41
Figure 5.1: Logic diagram showing the solution using service virtualizaton.....43
Figure 5.2: Architectural design for the implementation prototype goal.....44
Figure 5.3: Stack architecture implementation.....44
Figure 5.4: Class diagram for prototype implementation.....45
Figure 5.5: Implementation testing set and process.....46
Figure 6.1: Implementation testing results.....49
Figure 6.2: Addresses verification for prototype implementation.....50
Figure 6.3: Message body verification for prototype implementation.....50

Chapter 1

Introduction

In this chapter, a project overview is provided according to the following sections: in section 1.1 the background details the big picture where Service Virtualization is involved, such as how many digital devices are increasingly becoming important items to users, which technologies are used for solving the consequences that stem from this importance, and how these technologies fit in Service Virtualization; in section 1.2 the project definition is formulated according to three assignments; section 1.3 covers the methods used for the project definitions; and in section 1.4 the report outline is given.

1.1 Background

Over time, digital devices have become even more useful items for people in their daily tasks. People use them in several activities such as work, travel, or entertainment. However, both having many digital devices at homes for such activities, and the fact that digital devices are nowadays essential items for people may entail some complications for end-users in order to enjoy the facilities these items provide.

1.1.1 Need of accessing numerous personal devices

On the one hand, since most devices require manual configuration steps, users spend some time configuring them in order to enjoy their best use. However, considering that many digital devices are filling up homes, this configuration step can sometimes be very cumbersome as it requires the user to access the device and set up its configuration manually. It would be great to enable all devices located in a local network to recognize each other with simple steps.

Just consider a printer scenario as an example: a user wants to access a printer connected to a computer from another while both connected to the same local network. The aim is to be able to automatically join the printer to the local network so that the rest of devices connected to the same local network can recognize it and subsequently control it.

Taking this printer scenario one step further, since many devices are currently used at home, it would be feasible to be able to join all the devices to the home local network and make them cooperate with themselves afterwards. This is why consumers are requesting more and more systems capable of controlling personal devices through one interface. It is like a central point interface capable of recognizing and controlling both the different devices and their services within a local network is to be defined.

On the other hand, personal digital devices have become so useful items that consumers sometimes need to access them even when they are not carrying them. For example, imagine if a user needs a file located at his computer and he tries to get it from his office [1]. Or consider if that the customer is a textile company that wants to manage remotely its patterns such as garment development regarding quantities, sizes and amount of material to be made [2]. There is some kind of remote access need from customers that is required more and more due to this elementary use. This being, a central point device which enables local devices to be available in the wired network to be developed.

In this way, the fact that digital devices are filling up homes entails researchers to focus on the available technologies in order to solve these two issues: handle devices in a local network through an entity and access remotely these devices from an external network.

1.1.2 Service discovery for local devices

Nowadays devices located in local networks may be aware of each other through several methods: Bluetooth, USB, Firewire, IrDa, IEEE, Ethernet, Universal Plug and Play, Bonjour, etc. However, among all these methods, some are much easier than others to deal with, which is the case regarding service discovery mechanisms.

Service discovery mechanisms through an unique interface let end-users discover and control all the devices located in the same local network, such as offices, homes, or residences. This interface embeds a client entity which enables end-users to search for available devices, view the services these devices provide, and even make them interact with each other.

In this way, service discovery appears to be the proper technology to be used by end-users who want to easily access their personal digital devices through a single interface as far as local networks are concerned. Thus, the problems end-users had to solve in order to manually set up each device seems to be solved already.

1.1.3 Towards convergent networks

Nowadays, different networks exist, such as fixed networks, wireless networks, cellular networks, or telephony networks. If a service is supposed to be created for all these networks, it will generally be developed in different ways; or, in other words, one specific service development for each network just to provide the same type of service for all the abovementioned networks. Since this requires a lot of resources, enterprises conduct research in a global network which includes all these ones, so that only one development for the mentioned service is necessary. One particular service is developed for a global network as several service developments have been carried out according to several networks in order to provide the same particular service. Now, since this global network unifies several networks, one global developed service is suficient. This is one reason why these networks are converging to one network, a convergent technology called IP Multimedia Subsystem (IMS).

IMS merges data, voice and video services so that the customer will have to pay only one bill for all services offered by his contracted provider. And here is where the operators will have to offer much better services than they do now just to play a more important role than 'bit pipe' managers

[3]. Since the abovementioned services are merged in a convergent network, end-users will be able to use his mobile phone in any network such as local wireless network, fixed network, or cellular network; depending on his convenience. It is like IMS provides a kind of agnostic access network entailing great mobility benefits for end-users besides integrated delivery of multimedia services.

Like this, with IMS consumers will not care about carrying their devices to everywhere they are. They just access remote devices without being aware of the access network they should use. Therefore, IMS is the core network that is to act as the proper basis for remote access methods from anywhere.

1.1.4 Easy access to remote devices

Seizing up the features from both service discovery mechanisms in local networks, and IMS as core wired network; enterprises are nowadays making focused research into such technologies in order to achieve easy access to remote devices.

At first glance, embedding such remote functionality in the client used in these service discovery mechanisms is the first task to handle. In fact, P. Belimpasakis and V. Stirbu used the Atom Publishing Protocol [4] for allowing devices, located in different networks, to offer their services and consume them through a specific service discovery mechanism using current internet network as wired network. As a solution proposal, some modifications were carried out in the principle of operation of the concerned mechanism. However, since the the stack architecture of a central point entity is really complex due to the amount of underlying protocols and functionalities built-in, making such modifications can jeopardize the ability of that mechanism.

Since this procedure seems to be very cumbersome to carry out, it seems that this remote functionality should be decoupled from the client entity. Moreover, end-users are not required to be aware of which service discovery environment is serving them, consequently allowing them to be free for controlling remote devices from anywhere. For instance; if the remote functionality has been embedded inside a UPnP control point, end-users would be able to access remote devices only from networks serving UPnP as service discovery mechanism.

Therefore, this remote functionality implementation in particular service discovery devices is not a proper step. It seems to be implemented outside such entities, letting end-users not care about from where they want easy remote access. In this way, the remote functionality should inform the control points about the available remote devices' services.

A first step forward in this field was done by A. Häber et al [5], regarding awareness of a remote service. Devices in any external local network, through a service discovery mechanism, are aware of the devices' services presence located in a home local network using IMS as wired core network. In this way, users within external networks are aware of the available remote home devices they want to access.

A further advance was developed again by A. Häber et al [1], so that remote services within a residential network are consumed by users in an external local network. This latter step permits a end-user to access personal content stored in his home network from an external network using a service discovery mechanism.

This advance seems to achieve the ease of use for accessing a remote service through a service discovery mechanism and IMS. Still, this procedure did not decouple the remote functionality from the client entity yet. And this is the origin of the research question in this project: the need of implementing this remote functionality outside such entity.

Decoupling this remote functionality to another entity implies that the service discovery mechanism client is informed about the remote services it offers, without recognizing this remote

service as a remote issue. It believes that such a service is in the same external local network within which it is operating. In other words, it seems a remote device can be hosted into a local network so that this remote device is recognized as another local device, including the services it offers.

Thus, the present project demonstrates that services offered by remote devices are virtualized in a local network making them appear to be part of this local network, allowing users to be unconcerned about which service discovery is serving them.

1.1.5 Service Virtualization as a solution

If the aforementioned aims are considered, the concept of Service Virtualization (SV) comes in: a service within a remote home local network is virtualized so that it seems to be part of an external local network; concerning the ease of use for the end-user in terms of the capability of being aware of, accessing, and controlling local and remote devices' services without caring which service discovery mechanism is serving them.

In this way, with the benefits stemming from both easy remote access devices' services and IMS, such as mobility and roaming, or integrated multimedia services; SV becomes a real easy useful item for end-users to operate, for example, their home devices while they are away from their homes, such as the washing machine, security cameras, stored files, etc. from wherever they might be. Users staying in a visited-external network can remotely consume the services located in their home network as if these services were in the same local network where users are staying.

In this project, these terms already mentioned such as service discovery mechanisms, IMS, remote accessing, when SV is convenient for the user, SV's principle of operation, and a demonstration of SV have been explained.

1.2 Project definition

The project definition is formulated as:

> *The concept of Service Virtualization shall be explained, with focus on residential networks. Use cases for Service Virtualization in the context of residential networks shall be developed and analyzed. Part of the analysis shall show advantages and disadvantages of using Service Virtualization compared to normal remote service access.*
>
> *A solution architecture for Service Virtualization shall be designed and explained. This includes infrastructures, services, protocols, devices and software requirements comprising the solution.*
>
> *A prototype implementation of the solution architecture shall be developed. The prototype shall virtualize external services within a local network for verification of our goal. The prototype shall be implemented with emulation technology, and if time permits, as a final step be carried out as a real application.*

1.3 Project execution

The architectural framework where SV is involved is described in subsection 1.3.1 followed by the methods used in the present project in order to realize the project definition in section 1.3.2 according to the described framework.

1.3.1 Project description

Since the goal is to bring a remote service from one place to another, at first two local networks are considered. One local network belongs to the network where the device or device's service is located; this local network is called the 'home network'. The other local network belongs to the network which the remote device or the remote device's service is brought into; this local network is named the 'visited network' or 'external network'. In this way, a service from the home network appears to be a local service in the visited network; in other words, a service is virtualized into the visited network as if it would be in this network.

In order to bring a remote service from the home network to the visited network a core network is used for connecting both local area networks (LANs). It is here internet comes in as the wired area network (WAN) used. Both LANs are connected to each other through the WAN. Since WAN signalling has nothing to do with LAN, a gateway between them is considered to handle this signalling.

Moreover, in order to realize SV, an extra device is required to be used within the visited network just to find and control the remote service. Since the project consists of bringing a remote service from anywhere into a local network, a mobile device is the one used for handling all the necessary operations for SV.

This situation is depicted in Figure 1.1:

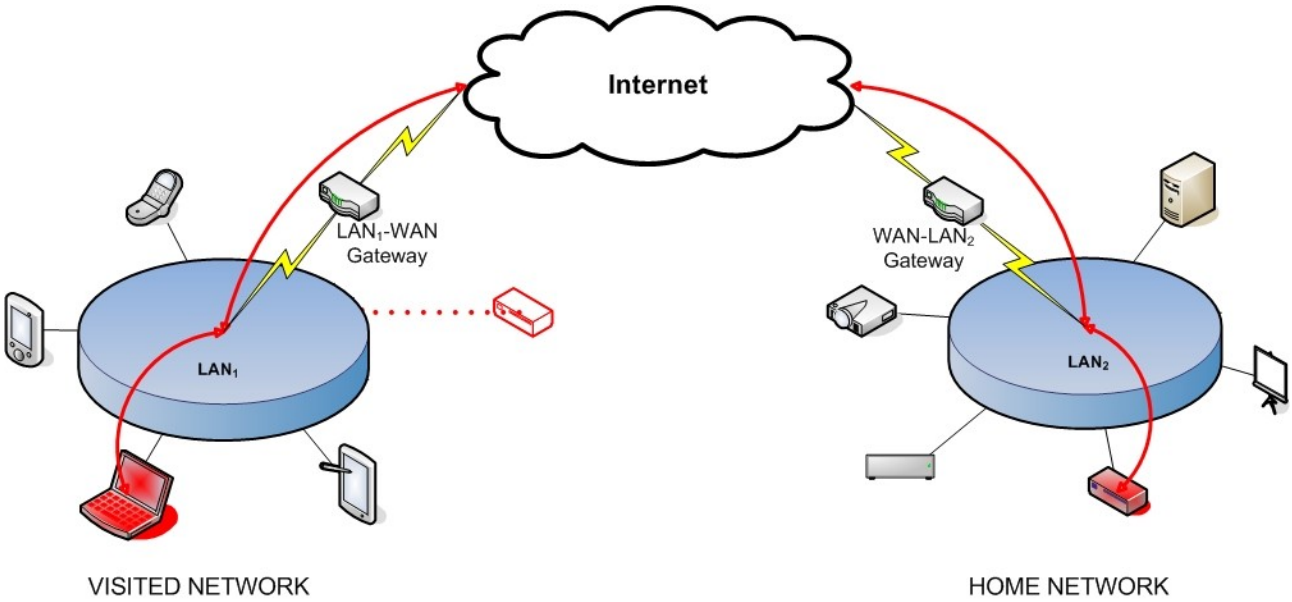

Figure 1.1: Solution architecture

As we can observe, a device is virtualized in the visited network by a mobile device so that users can enjoy its services as if these would be located within the visited network. As an example,

the picture shows a laptop as a mobile device and a storage device as a virtualized device.

1.3.2 Solution approach

First of all, a set of uses cases for SV will be analyzed and developed, all of them based on the project description. Two scenarios will be explained for each use case; this includes a logical diagram showing the scenario framework, a description of the abovementioned diagram and a main flow diagram that graphically shows the necessary steps which makes possible the scenario described before. After showing these use cases, one of them will be chosen as the basis for the main solution for SV.

In order to complete the adopted solution based on the description of the previous section, chosing a proper service discovery protocol within the LANs will be the first task. Then, once devices' services are aware of each other within the respective LANs, the gateway will connect the remote device that is to be virtualized (home network) to the IMS core network in order to enable access to it from the wired network. Both gateways will establishe a connection between themselves using Session Initiation Protocol (SIP) as signalling protocol afterwards. Therefore, through the service discovery mechanism, the visited network will be aware of the service provided by the virtualized device which is located in the home network. As a final step, the virtualized device will be handled and used according to the goal of the solution adopted.

To conclude, a demo-prototype will be implemented in the solution developed before. This prototype meets the specifications of the solution architecture fulfilling the requirements and corroborating the meaning of SV. It will be based on a language programming and both its technical requirements and how it handles the solution will be described and explained as well. As a final step, a real application has been carried out using SV.

1.4 Report outline

The next chapter deals with the basic theory for the whole project has been executed, such as service discovery mechanisms, IMS and remote access including one scenario example for each issue.

The third chapter develops and analyzes several use cases related to SV that are convenient for the user. As explanations, both a logic and a flow diagrams have been provided for the development and analysis, respectively.

The fourth chapter explains the solution designed for the SV while the solution is based on the previous use cases, it explores further details such as architecture design, protocols, services and software used.

The fifth chapter describes the tools used to carry out the SV implementation and how it has been implemented.

The sixth chapter explains the results obtained from the realization of the aims of project definition.

The seventh chapter covers the other different ways that could have been chosen for the executed project according to the results obtained in the previous chapter.

The last chapter presents the conclusions, such as SV benefits for customers, or its role in the next years for service providers and end-users.

Chapter 2

Theoretical framework

The basic theory behind the concept of Service Virtualization is given in this chapter. It begins with how devices can interact with each other in a local network through service discovery mechanisms in section 2.1. The wired core network used for connecting the home network and the visited network is explained in Section 2.1. Finally, how to access a remote device from a visited network is described in Section 2.3.

2.1 Service discovery

This section defines service discovery and gives a list of the different current service discovery mechanisms. The basic concepts required for understanding the chosen service discovery mechanism, such as its definition and its principle of operation, are given as a second step, followed by a scenario example.

2.1.1 Service discovery in local networks

Service discovery is the capability of automatically detecting devices and their services in a network [6]. Service discovery mechanism enables devices and services to interact with each other in terms of discovery, configuring and communication [7]. This capability is handled by a network protocol called a Service Discovery Protocol (SDP).

In the last years, a lot of SDPs have been developed by a lot of companies and organizations in order to offer customers an easy way to enable interaction between customer's networking devices. Some of these protocols have been developed in universities and institutions, such as Intentional Naming System [8], Ninja Service Discovery Service [9] and DEAPspace [10]; others have been developed by software vendors such as Jini [11], Universal Plug and Play [12] and Rendezvous [13]; and others by other industries such as Salutation [14], Service Location Protocol [15] and Bluetooth SDP [16].

However, since this project focuses in Universal Plug and Play as the service discovery mechanism used for residential networks, only its concepts are described below.

2.1.2 Universal Plug and Play

All the theoretical background regarding Universal Plug & Play has been based on [24].

2.1.2.1 Definition

As [24] defines, “The Universal Plug and Play (UPnP) [17] architecture, an initiative by the UPnP Forum, is designed to connect networked devices such as PCs, entertainment devices, and intelligent appliances. The UPnP architecture leverages existing standards like TCP/IP, HTTP and XML. The architecture consists of a set of standardized protocols that each UPnP device implements to provide for discovery, control and data transfer between UPnP devices. The UPnP technology is supported on any common operating system and it works with almost any type of physical networking media providing maximum user and developer choice.”

2.1.2.2 Principle of operation

A UPnP device contains at least one embedded device, and each one implements zero or more units of functionalities called *services*. Each service has a set of *actions,* each one with input, output, and return parameters; and a set of *state variables*, each one with a name, type, and value. The formers are the methods that comprise the provided device's service and the latters are the properties of such device's service grouped in a *state table*.

The operation point in a UPnP device is to be able to invoke the actions belonging to its services. This invocation issue can be done using a *control point*, an entity that works with the functionality provided by a network device. Besides being capable of invoking actions, control points can monitor state changes occurred in the UPnP device. When a state variable changes, the UPnP device may send an *event* to let the control points know.

In this way, control points can discover devices, invoke actions on a device's services and subscribe to events. Concurrently, devices respond to such invoked actions and send events when state variables change. In order to carry out these operations, a UPnP device must follow these *phases of operation*:

- **Phase 0: Addressing**

  It is the phase in which the device obtains an Internet Protocol (IP) address through Dynamic Host Configuration Protocol (DHCP), or using Auto-IP. In this way, the device is able to connect to the local network and consequently be able to interact with other devices connected to the same network.

- **Phase 1: Discovery**

  It is the phase in which UPnP devices and their services are discovered by a control point. In this way, the control point multicasts along the local network the available devices' services through Simple Service Discovery Protocol (SSDP), based on HyperText Transfer Protocol (HTTP).

- **Phase 2: Description**

  It is the phase in which UPnP devices shows descriptive information about themselves and their services. Once a control point has discovered a device's services, it obtains through HTTP two XML description documents from it: one regarding device information such as manufacturer, or serial number; and another regarding its services such as actions, or

parameters. SSDP is the protocol still used in this phase.

- **Phase 3: Control**

  It is the phase in which a control point controls the available devices' services. Once the control point is aware of its availability and its offered services, it invokes the actions these services provide through Simple Object Access Protocol (SOAP).

- **Phase 4: Eventing**

  It is the phase in which the control point is notified through General Event Notification Architecture (GENA) about status changes occurred in the available devices' services such as invoked actions.

- **Phase 5: Presentation**

  It is the phase in which an available device presents an interface programmed in HyperText Markup Language (HTML), letting the user control the device, change parameters, or invoke actions.

To conclude, a UPnP protocol stack including the phases is shown in Figure 2.1.

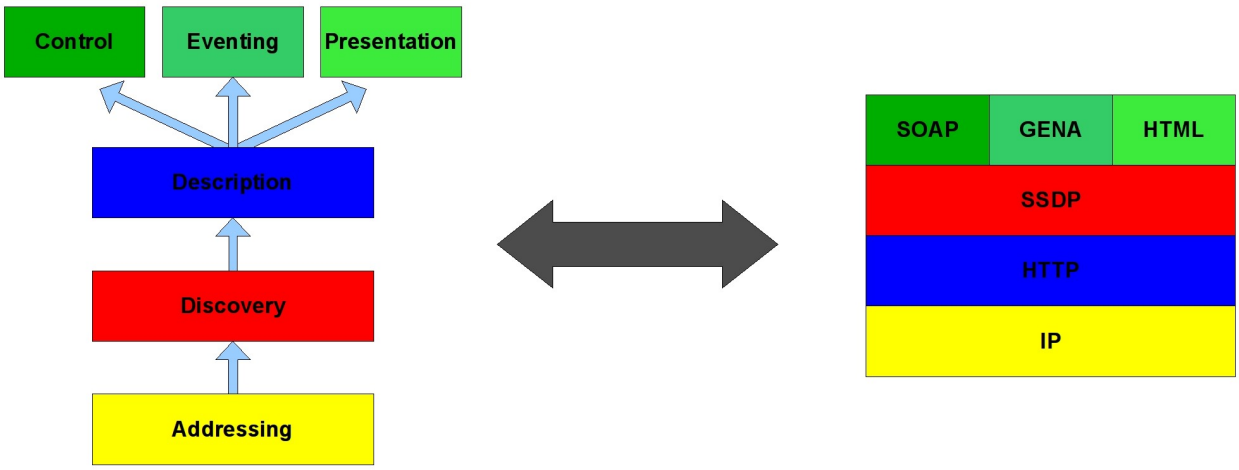

Figure 2.1: UPnP architecture protocol stack

2.1.3 Scenario example

This example has been based on a flow diagram developed in [18]. In addition, all the actions names, arguments names, and services names has been extracted from documentation regarding a media server and a media renderer found in [17].

Alejandro has plenty of digital devices that can interact with each other through UPnP as service discovery mechanism. Since he is getting bored due to the rainy day that does not let him go out with his friends, he decides to watch a movie stored in his room computer on the TV located in his living room. In this way, he takes his mobile phone as control point entity and searches for the available devices in the home local network. Among all the digital devices that show up in the mobile phone screen, he gets access to both the media renderer connected to the TV and the media server installed in his computer. After a couple of minutes, he transfers the movie from his personal computer located in his room to the TV located in the living room. Now Alejandro enjoys the movie with some popcorn on his hands. Figure 2.2 shows how Alejandro transfers the movie to

the TV.

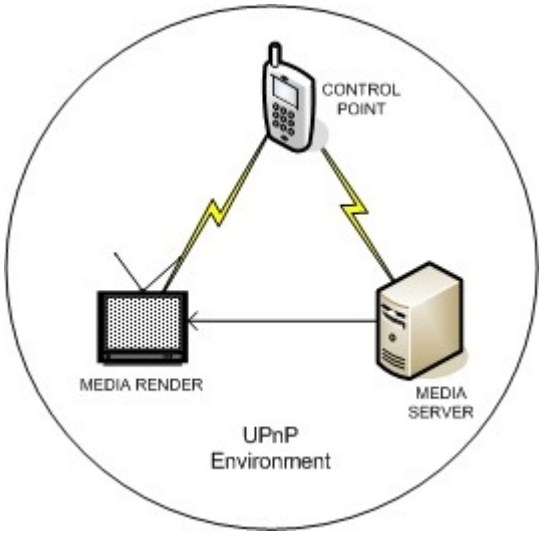

Figure 2.2: Watching a movie in a local network through UPnP

In order to realize the abovementioned scenario, the steps regarding the phases of operation of UPnP have been described according to Figure 2.3:

- **Discovery**

In this phase Alejandro with his mobile phone searches for the available devices in his home. After a few seconds, a media renderer and a media server show up on the mobile phone screen.

Since it is the discovery phase, SSDP is the protocol used for all the transactions between the devices and the control point. The control point sends {1} an M-SEARCH multicasting in the home local network. While the media renderer and the media server detect that they are being searched for and they send {2} a response informing that they have detected such a request. Then, they notify {3} the control point about their presence in the local network so that they are available for being used afterwards.

- **Description**

In this phase Alejandro looks for the proper service in order to choose the movie he wants to watch afterwards.

The description phase uses HTTP messages included in SSDP protocol in order to pass device description and service description of UPnP devices. Alejandro selects the media server device and the control point sends {4} a request to it in order to see its services. The media server validate the request sending a response {5} including the services it offers. Then, Alejandro looks at the different services available, in order to find the proper action which lets him to play a movie. Every time Alejandro looks at a service, the control point sends a request {6} to the media server and this responds the control point by sending {7} information about the actions and state variables of each service. After a few seconds, Alejandro decides to enter a service called “ContentDirectoryService”.

- **Eventing**

In this phase the media server notifies the control point about possible changes concerning the status of the state variables and actions related to the selected service using GENA protocol. Thus, the control point subscribes to the media server and the latter notifies the former about such changes.

So, the control point sends a subscription request {8} to the media server, and this validates it sending a response {9}. Then, the media renderer notifies {10} the control about the actual status of the state variables and actions, and the control point confirms the reception of such a notification by sending a response {11}.

- **Control**

It is the phase where Alejandro choses the action that lets him select a movie and then play it in the media renderer. SOAP is the protocol used in this phase.

Among all the actions available, Alejandro selects 'Browse' action, have a look at several movies he has stored on the media server and selects one. Thus, The control point sends a request {12} to the media server, and the latter responds {13} the former including the URI address of the selected movie.

Then, Alejandro access the media renderer with his mobile phone and, after following the same steps from discovery phase to control phase, Alejandro informs the media renderer about the movie he selected in the media server and plays it afterwards. In this way, the control point sends a

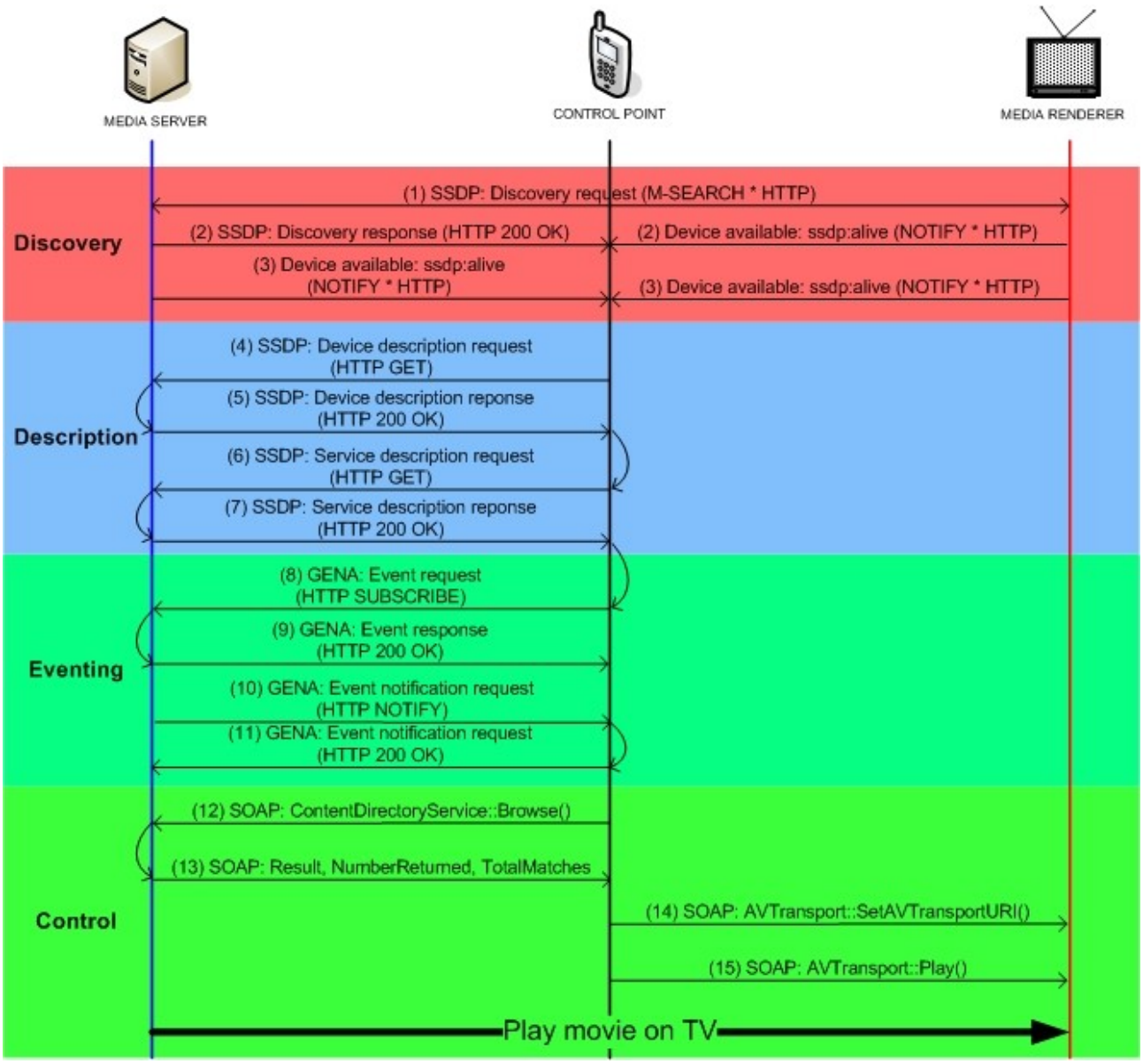

Figure 2.3: Flow diagram between a media renderer and a media server

request {14} to the media renderer in order to invoke 'AVTransportURI' action, including the address that was received from the media server. Finally, the control point sends {15} a last request to the media renderer, and the movie is automatically played on the TV screen.

2.2 IP Multimedia Subsystem

This section explains IMS from a basic point of view: only the most relevant IMS concepts for Service Virtualization understanding, such as IMS definition and the main parts of its architecture with a figure depicting it, and a scenario example as a final step are explained.

2.2.1 IMS Concepts

All theoretical background regarding IP Multimedia Subsystem has been based on [19] and [25].

2.2.1.1 Definition

As [19] defines, “IP Multimedia Subsystem (IMS) is a set of specifications that defines a complete framework for converging voice, video, and data communications over a Session Initiation Protocol (SIP) infrastructure.” IMS is part of the Next Generation Networks (NGN) architecture core which provides fixed and mobile multimedia services through both circuit-switched and packet-switched domains.

2.2.1.2 Architecture

The aim of IMS is to establish and control the connection between two IMS terminals so that they can directly transfer to each other multimedia data afterwards. This connection establishment is carried out by SIP and the multimedia transfer can be handled by Real Transfer Protocol (RTP) or HTTP. As depicted in Figure 2.4, a IMS terminal can be a laptop, or a conventional telephone, or a cell phone.

In order to carry out the establishment and control signals, IMS architecture has been broken down into three layers: transport layer, which allows IMS terminals to connect to the IMS network; control layer, which handles all SIP signalling between peers through several functional proxies; and session layer, which let providers offer multimedia services to IMS terminals leveraging the functionalities of control and transport layers. Figure 2.4 depicts the IMS architecture including its most relevant functionalities of each layer.

Before describing IMS architecture, the concepts of home network and visited network must firstly be explained. The *home network* is the network that is supported by our country operator. The *visited network* is the network supported by another operator, in other words, the network IMS terminals use when they are visiting another country.

- **Transport Layer**

Since it is the network-access layer for IMS terminals, its aim is to provide an IP-address to them in wireless, mobile, or fixed networks. For Public Switched Telephone Network (PSTN), this layer uses a media gateway (MGW) in order to enable interactions between IP devices and conventional telephones.

- **Control Layer**

The principle of operation of the control layer is carried out by the Call Service Control Function (CSCF) that is composed of three SIP signalling functionality servers: the proxy CSCF (P-CSCF), the interrogating CSCF (I-CSCF), and the serving (S-CSCF).

The P-CSCF is the first SIP proxy which contacts the IMS terminal. Its aim is to pass this IMS terminal's SIP registration to its belonged home network's S-CSCF. In this way, P-CSCF acts as the SIP signalling gateway between the visited network and the home network. P-CSCF is located in both visited and home networks.

The I-CSCF is a SIP proxy located at the edge of an administrative IMS domain. Its aim is to act as a forwarding point for SIP messages coming from SIP remote servers to its domain. It is located in the home network.

The S-CSCF is the signalling domain's central node. Its aim is to register users and control SIP requests, as well as providing services to registered users. It retrieves information regarding registered users from HSS database. It is located in the home network.

The Home Subscriber Server (HSS) is the database which contains users information such as profiles, authentication, authorization or location. A network may contain more than one HSS due to the possibility of a high number of subscribers. If this is the case, networks require an Subscription Locator Function (SLF).

The Media Resource Function (MRF) provides media functions in the IMS architecture. It controls the resources and implements all the media functions and is located in the home network.

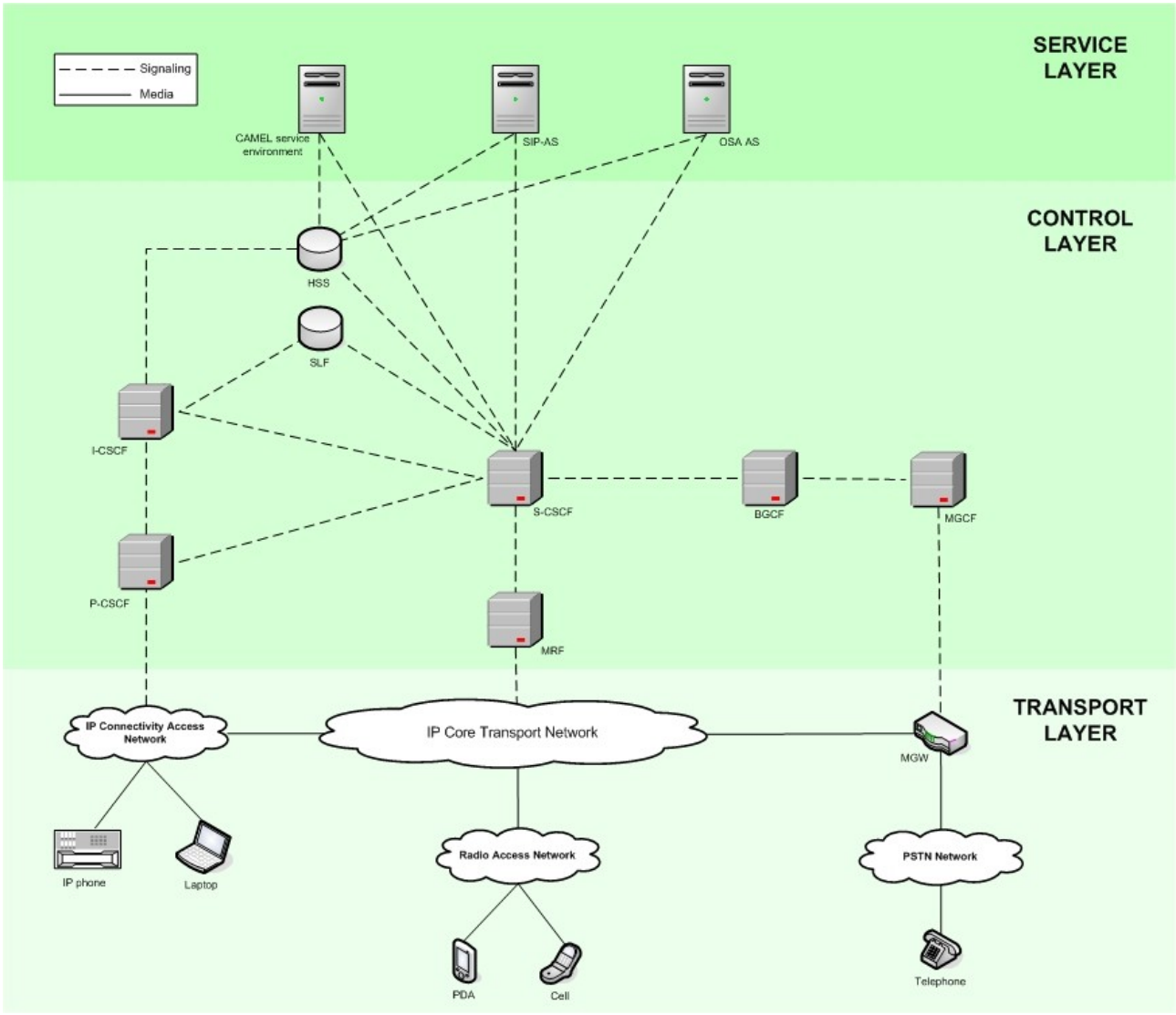

Figure 2.4: IMS architecture overview

The Breakout Gateway Control Function (BGCF) determines the next hop for routing SIP messages to circuit-switched networks. It selects a Media Gateway Control Function (MGCF) that will route the call to the PSTN through a MGW.

- **Service Layer**

The service layer lets service providers offer multimedia services leveraging the transport and control layers functionalities. Three types of application server (AS) exist: SIP AS, OSA AS, and the CAMEL service environment.

SIP AS is the native AS that hosts and executes IP Multimedia Services based on SIP. Its aim is to watch the calls' signalling flow, this is, it translates an IMS service's execution in the SIP application server into a sequence of SIP procedures.

OSA AS (Open Service Access AS) implements external services that could be located in a visited network or a third-party platform.

The CAMEL (Customized Applications for Mobile network Enhanced Logic) service environment let GSM mobile network operators control an IMS session.

2.2.2 Scenario example

This example has been based on another given in [25].

Miguel, from Spain, is staying in Grimstad (Norway) due to a summer job at the University of Adger. His girlfriend Teresa, from France, is staying in Berlin (Germany) for a month in order to visit a friend. One day, they decide to establish a video conference just to exchange information and share experiences. In this way, Miguel takes his laptop and, with his SIP account provided by Telefónica, his Spanish home operator, logs on to Telenor, the Norwegian operator which is serving him, and calls Teresa typing her SIP account provided by France Telekom, her French home operator, as destination call. Teresa, thanks to the roaming facilities provided by IMS, receives the call from Miguel through a redirectioning from her home operator and Deutsche Telekom. Now, the couple can keep a quite lovely conversation about their respectives daily tasks in a foreign country.

Since Miguel is Spanish and Teresa is French, Telefónica and France Telekom are considered as home networks, whereas Telenor and Deutsche Telekom are considered as visited network. In addition, since the origin of the call starts in Telenor using a Spanish SIP account, Telenor is considered as the *originating visited network* and Telefónica as the *originating home network*. On the other hand, since the call is destinated to French SIP account in Germany, France Telekom is considered as the *terminating home network* and Deutsche Telekom as the *terminating visited network*.

Following the aim of the P-CSCF explained before, when Miguel starts calling, its SIP account is registered in Telenor. In this way, the originating P-CSCF will pass Miguel's SIP registration to Telefónica's S-CSCF, which will register Miguel SIP account and will provide him an audio session service with the destination call. In order to serve this service to the terminating IMS terminal, Telefónica will send the SIP request to the domain which the destination address (Teresa's SIP account) belongs to, this is, France Telekom's I-CSCF. Then, I-CSCF will retrieve the destination SIP account from HSS and will pass the SIP request to S-CSCF of the terminating home network. Since S-CSCF is aware of the terminating visited network of the destination SIP account, S-CSCF will forward SIP requests to Deutsche Telekom's P-SCSCF and this to Teresa's IMS terminal, hence completing the connection establishment between two IMS terminals.

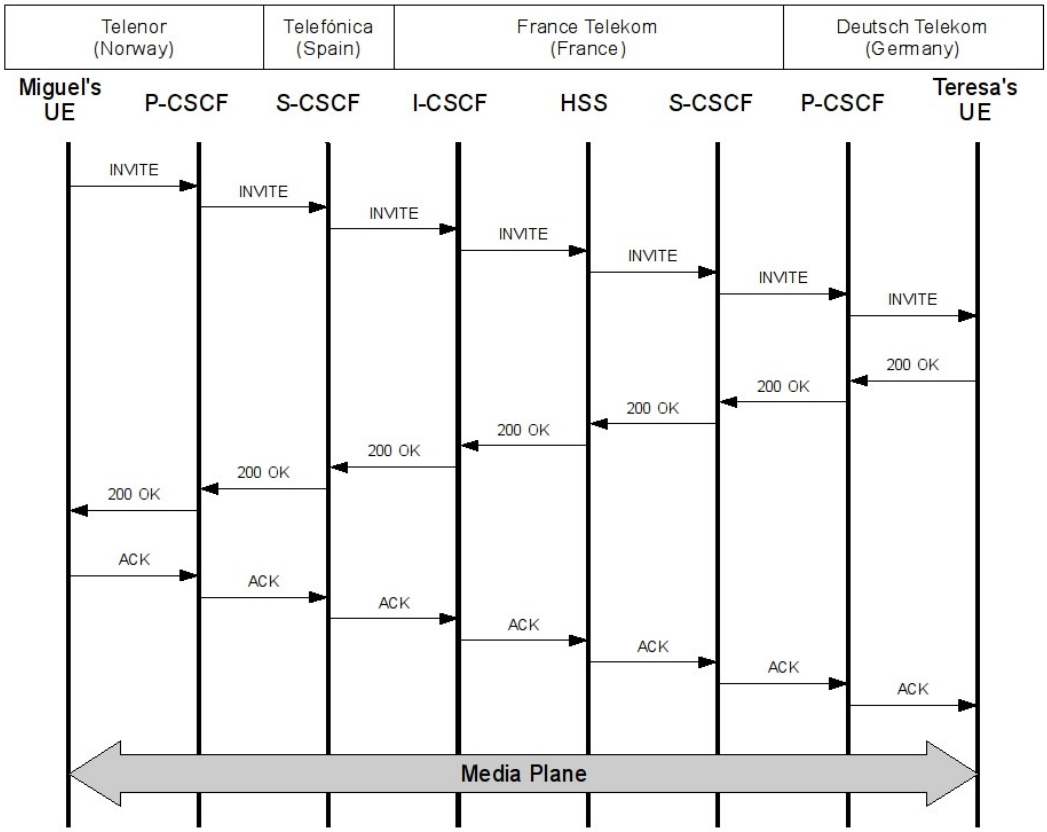

Figure 2.5: Signalling flow diagram between two different locations using IMS

2.3 Remote service access

The basis of accessing a remote service from a visited network is covered in this section. The definition of remote service access is given as first step. Afterwards, two operations have been explained with the help of figures that depict both the framework the remote service access is involved and the signalling space between visited and home networks in order to enable remote service usage.

2.3.1 Definition

Remote service access is a procedure which enables end-users to access a service offered by a remote device located in a local network from an external local network. Like this, people are able to find the remote services they want to access and consume them afterwards from any local network environment such as stores, offices, hotels.

In order to establish remote service access, the first step to proceed is to be aware of the remote service end-users want to access to. Afterwards, a usage for this remote service is the final

step to proceed to. Next two sections have been based on [5] and [1], respectively.

2.3.2 Remote service awareness

The aim is to be aware of a remote service offered by a remote device located in 'Residential Network B' from 'Residential Network A'. As depicted in Figure 2.6, both local networks are interconnected through IMS, and each network has it own service discovery method; this is, SDP_B for the former and SDP_A for the latter. Each local network also includes a functional entity which discovers the devices in its vicinity as well as the services they offer called Service Discovery Gateway (SDG).

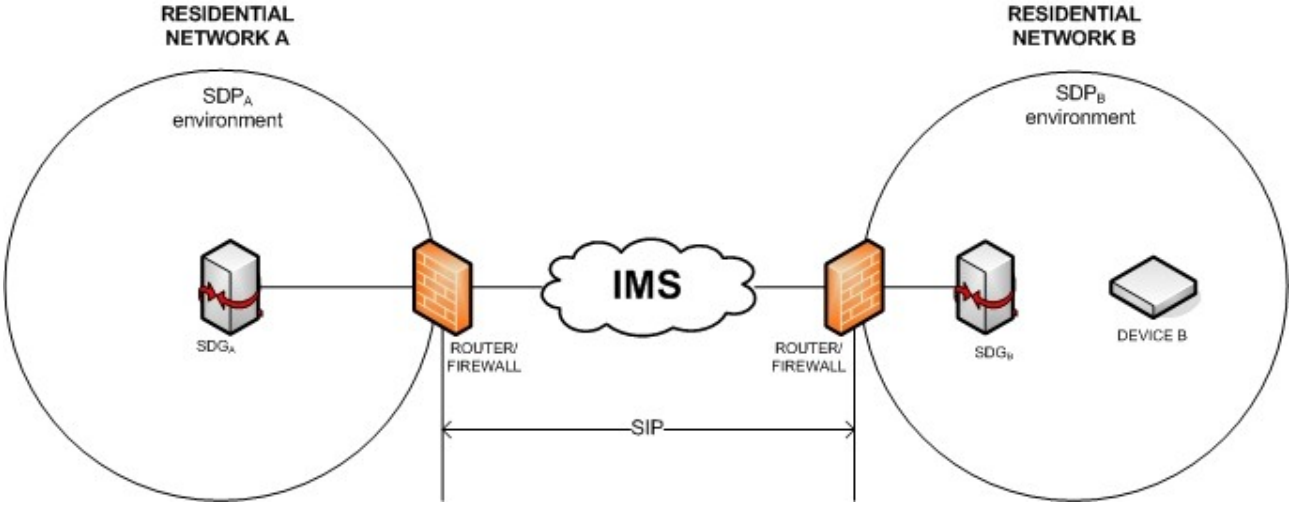

Figure 2.6: Logic diagram for remote service awareness

The role of SDGs in remote service awareness is to make devices' services' presence available for remote access. Like this, the Piranha protocol used for a remote service usage has been used here too. Piranha protocol manages to connect both SDGs over IMS using SIP. In this way, Piranha contains information such as SDG_B, the remote device (Device B), or IP addresses.

Since the aim is that SDG_A wants to be aware of the services offered by Device B, SDG_A will have to subscribe to SDG_B about the available services in 'Residential Network B'. Once the subscription is done, SDG_B will notify SDG_A about the services offered by Device B, in 'Residential Network B'. Moreover, if the status of the services have changed, SDG_B will notify SDG_A about such status update as well.

From the point of view of the signalling domain, SDG_A subscribes to SDG_B {1-2} in order to get the information concerning the services offered by Device B, and SDG_B relays {3-4} such information to SDG_A, hence achieving the remote service awareness issue. If for some reasonervice offered by 'Device B' changes its status, SDG_B will notify {5-6} subscriber SDG_A. As shown in Figure 2.7, the remote service awareness steps have been carried out using SIP between both SDGs.

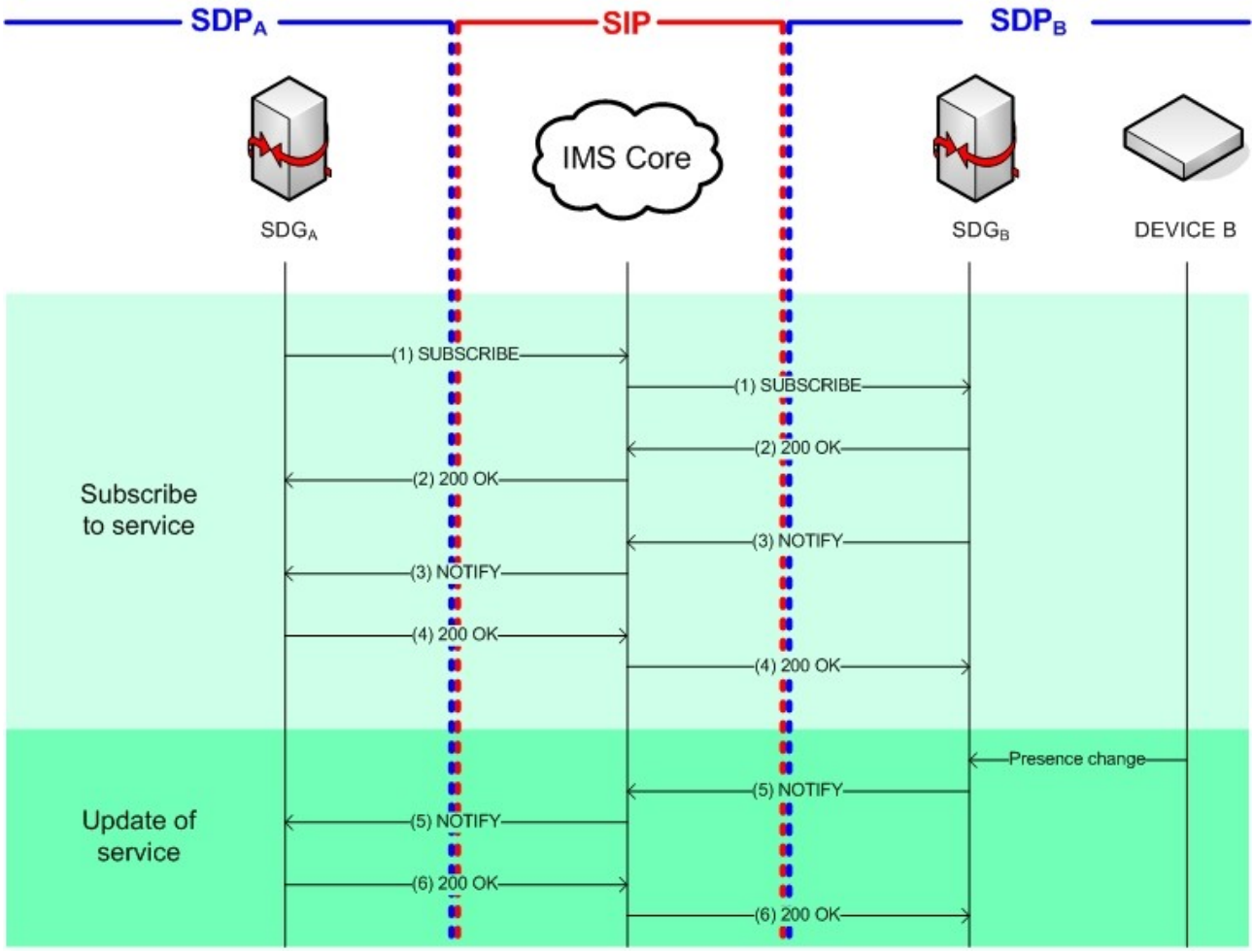

Figure 2.7: Flow diagram for remote service awareness

2.3.3 Remote service usage

Once end-users are aware of remote services and also want to consume such services, a connection must be established between both local networks. The framework for remote service usage is the the same as it is shown in Figure 2.6. As in remote service awareness case, Piranha also handles all the transactions between both SDGs.

To do so, once SDG_A is aware of the service presence of Device B, both SDGs must establish a connection between themselves in order to consume such a service. This connection is realized in three steps: initiating a service session, updating the service session, and closing the service session.

- **Service initiation**

It is the step in which SDG_A initiates a connection establishment with SDG_B. To do so, SDG_A sends an invitation request {1} to SDG_B, and the latter responses the former such an invitation sending an afirmative message {2}. Finally, once SDG_A receives the positive response from SDG_B, it sends a confirmation message {3} in order to finish the connection establishment. SDG_B receives it, hence completing the service initiation.

- **Service update**

Once the connection is already established, both SDGs can maintain audio sessions, or video sessions, for instance. However, during the session some parameters of the connection may alter the description of the session, just like changing the configuration of the streaming video, or changing parameters related to the audio session. Thus, the session must be updated to the new changes. To do so, SDG_A sends an update message {4} to SDG_B, and this accepts such a message responding an afirmative message {5}. Now the session can be re-used according to the new parameters, so that there is no need to finish the session and re-initiate it.

- **Service ending**

Once both SDGs do not want to keep the session, SDG_A asks SDG_B for ending the session. Thus, SDG_A sends a closing session message {6} to SDG_B, and this accepts the closing requests by sending an afirmative message {7} to SDG_A. Figure 2.8 shows all the signalling transaction between both SDGs in order to enable a remote service usage.

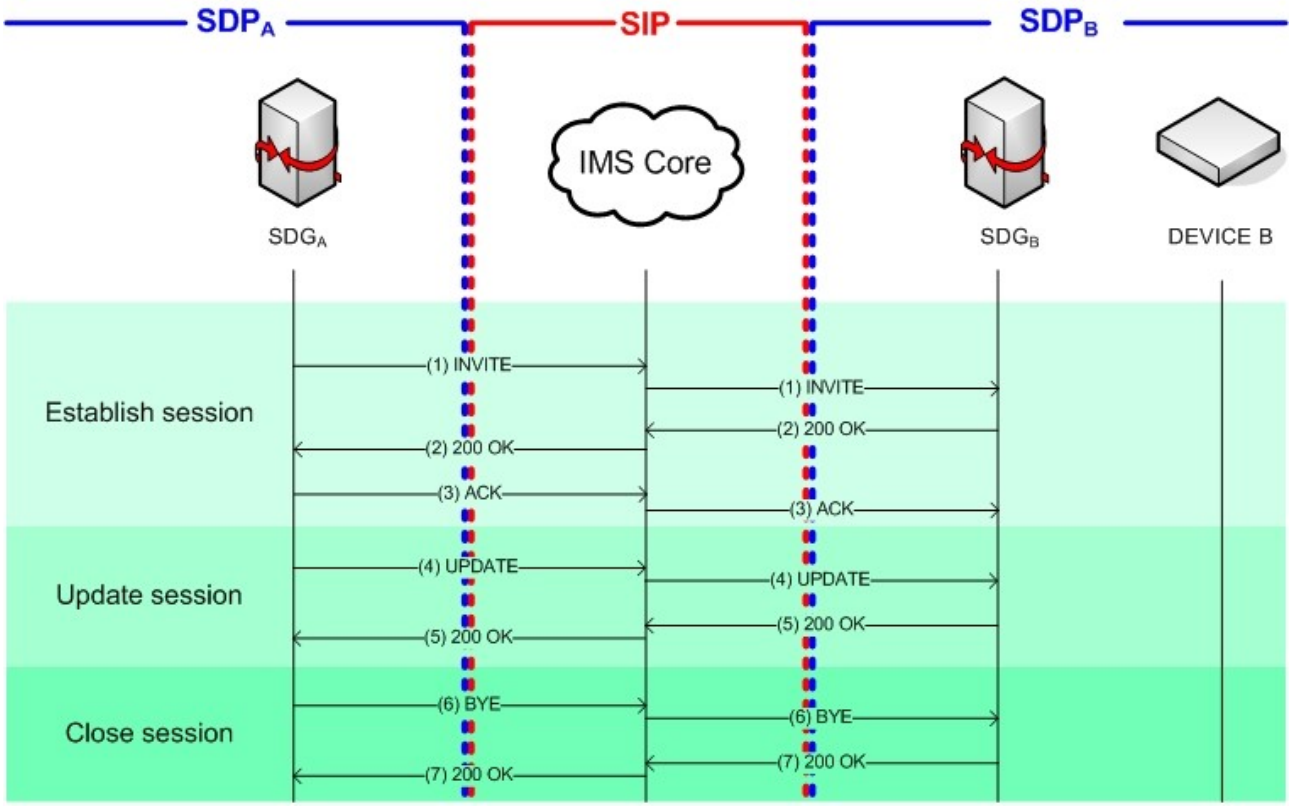

Figure 2.8: Signalling flow diagram for remote service usage

Chapter 3

Use cases for Service Virtualization

This chapter shows the use cases that have been developed and analyzed for Service Virtualization. It begins with the definition of Service Virtualization in the context of residential networks in section 3.1, followed by three use cases with two scenarios for each of them. These use cases are broken down into three sections: remote multimedia access in section 3.2, remote multimedia upload in section 3.3 and remote control in section 3.4. At the end of the chapter, both the importance and advantages of Service Virtualization compared to existing remote access methods have been explained in section 3.5.

3.1 Service Virtualization in residential networks

Service Virtualization (SV) is the process which virtualizes both a remote device and its services located in a local network, into an external network. Like this, SV makes this remote device appear to be a local device within the external network, so that end-users can enjoy remote services as if they were local services. As a service is linked to a device, saying that a remote device is virtualized is the same as saying that a remote service is virtualized too. The concept of SV is only enabled as far as residential networks are concerned. A residential network is defined [5] as a Local Area Network (LAN) where all the local devices are interconnected and can interact with each other through a service discovery mechanism.

For the realization of SV, the framework where SV is involved must be explained. Like this, two residential networks are considered, home and visited networks. On the one hand, the home network is defined as the LAN where end-users keep their devices, such as media renderer, media server, video recorder, camera, TV, or digital photo album, and want to access them from anywhere. In other words, it is the LAN where the virtualized device is located. On the other hand, the visited network is the residential network to which the remote device is brought, this is, where the virtualized device (or the virtualized device's service) is invoked.

Devices within a residential network can interact with each other through a service discovery mechanism, such as Bluetooth, UPnP, or Rendezvous .One service discovery protocol (SDP) is

used in each local network and they need not be the same SDP, that is why SDP_1 is the protocol used in the visited network and SDP_2 is used in the home network. To make local devices interact and cooperate, a mobile device is used as a control point. Moreover, this mobile device is the device that enables the virtualization of the remote device and its usage must take place within the the visited network, and not in the home network.

Concerning the wired network, since enterprises are looking to the next generation networks due to its benefits [19] in terms integrated delivery of multimedia services, mobility and roaming, and QoS efficiency; IMS is the chosen technology in order to connetc the home network with the visited network.

Finally, as the IMS architecture has nothing to do with the local environment in terms of working methodology, signalling, invokations, etc., a gateway which supports translation of messaging and protocols from the IMS core to the LAN environment is also taken into account.

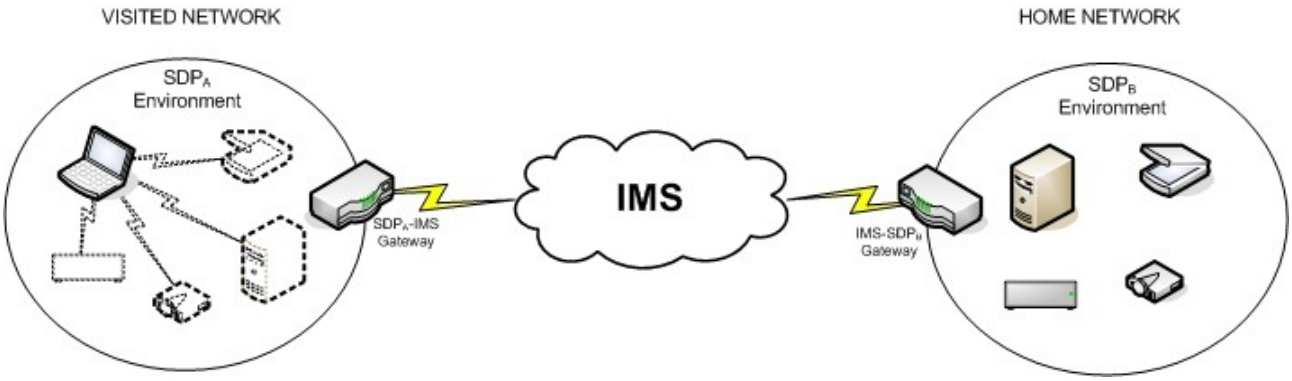

Figure 3.1: General logic diagram for SV

As depicted in Figure 3.1, any device located in the home network can be virtualized in the visited network through a mobile device as if it would be part of the latter network. Since several devices can be virtualized, a set of use cases has been developed and analyzed.

3.2 Remote multimedia access use case

The general case where users are away from their homes and want to access their multimedia content located at their home local network is considered. Specifically, this section is divided in two scenarios: first, playing remotely a multimedia file stored in our media server; and second, accessing the Pay-TV content delivered by your digital satellite TV tuner.

3.2.1 Streaming media

Fernando's family uses a media server at home as storage device for all multimedia files, such as documentaries, movies, music and photos. This media server offers several services, one of them called 'Multimedia Entertainment' which allows the user to play a selected file from a media portal menu. The following story has been explained according to the steps depicted in Figure 3.3.

One day, Fernando plans to make a weekend trip to Madrid with his family. Since they live in Málaga, Fernando must drive five hours by car to Madrid. During the journey, the children placed in the back seats startet complaining about the boring long distance from Málaga to Madrid causing Fernando and his wife Amanda terrible headaches. In order to calm the children, Fernando's wife

decides to play a movie on the screens fixed in front of the back seats using her mobile phone. To do so, she uses her cell phone {0} as remote controller to search for services available {1}. Two services shows up on her mobile screen {2}, one belongs to the media server located at home called 'Multimedia Entertainment' and another belongs to the screens within the car called 'Screen'. She selects {3} the latter and a source location is requested {4}. In this way, she selects {5} the 'Multimedia Entertainment' service as a source so that a media portal menu lets {6} her choose a media file to be played. She chooses {7} an animated cartoon movie and automatically {8} it is played on the screens. Now Fernando and Amanda are satisfied as they stopped their children from complaining anymore during the journey.

The logic diagram is shown in Figure 3.2:

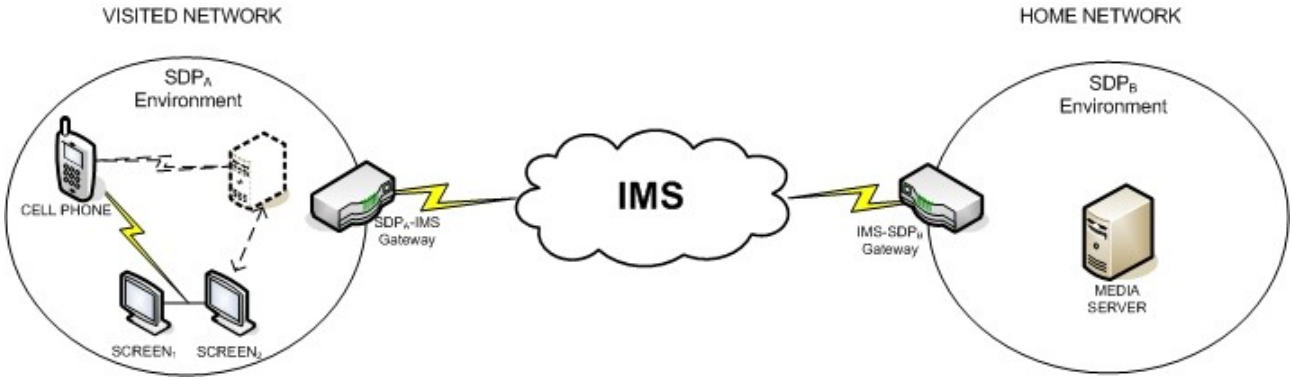

Figure 3.2: Logic diagram for streaming media to a car

Amanda's cell phone acts as the control point for the environment within the car so that, once it discovers all the services available such as those offered by the media server and the screen, she can transfer the selected movie to the abovementioned screens.

As it is shown, SV's procedure brings the remote media server into the car local network, making the services of the media server available within the abovementioned network as if it would be in the car.

In order to show how the media server is virtualized, a flow diagram is shown in Figure 3.3:

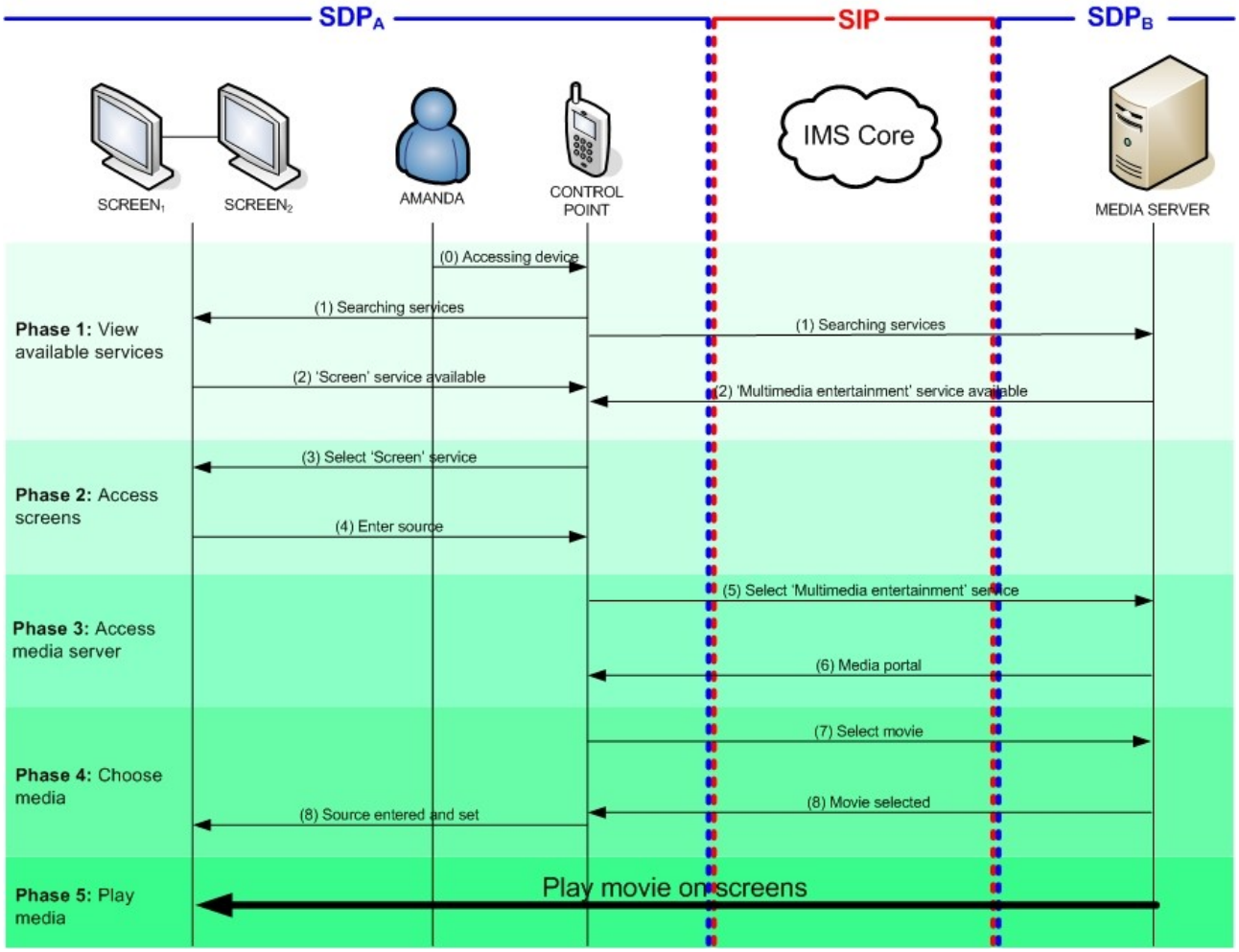

Figure 3.3: High-level flow diagram between remote media server and local screens

This case shows how important SV is for users who do not care about how the connections are set up in order to remotely access a device located in another local network. Here, Amanda wants to use the services offered by the media server in order to play a movie on a screen, not to access services remotely by entering the device address and watch its shared folders and so on.

3.2.2 Placeshifting of TV delivery

Manuel has got a Digital Satellite TV (DST) system he contracted for a year. The DST uses a decoder which forwards the services offered via the satellite link, such as 'Pay-TV'. If this service is selected, a list of channels are offered for the user, so that he can choose the channel he is interested in. The following story has been explained according to the steps shown in Figure 3.5.

After several hours spent in his job office, Manuel ends working and has the intention of going home. However, before leaving he listens to the radio for a while and he sees that there is a two-hour rush hour drive between job and his home. He gets really sad as today is the Champions League final which is only broadcasted in Pay-TV mode. In order to not miss the football match, he decides to watch remotely his Pay-TV content delivered by his digital satellite TV system located at his home through his Tablet PC {0} as a remote controller. He searches {1} for the services available and two services shows up {2} on his Tablet PC screen. One belongs to the DST located at home named 'Pay-TV' and another belongs to the TV within the office named 'HD-TV'. He

selects {3} the latter and a source location is then requested {4}. In this way, he selects {5} the 'Pay-TV' service as a source so that a list of channels are offered {6}, one of them broadcasting the Champions League final. He chooses {7} this channel and automatically {8} it is played on the TV. Now Manuel gets very happy to be able to watch what he expects to be the match of the year.

The logic diagram is shown in Figure 3.4:

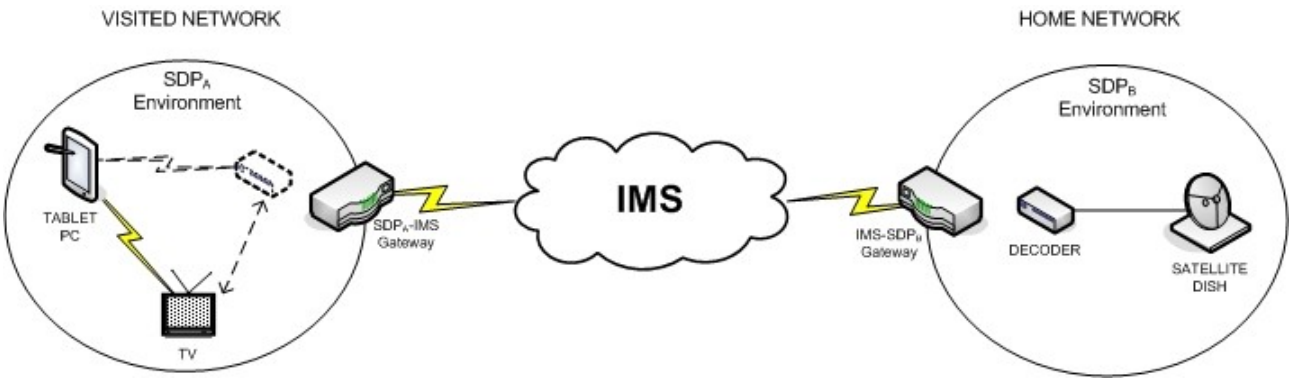

Figure 3.4: Logic diagram for TV placeshifting to job office

Manuel's Tablet PC is the control point in his office. Manuel uses the 'Searching for Services' option of his Tablet PC. Once a list of services are shown on the Tablet PC screen, he selects the service called Pay-TV. As a final step, he transfers the signal broadcasted by this service to the TV located in the living room of the office.

In this case, SV brings a remote decoder into the office network making its services available as if the device would be in the visited network.

A flow diagram which details the SV procedure is shown in Figure 3.5:

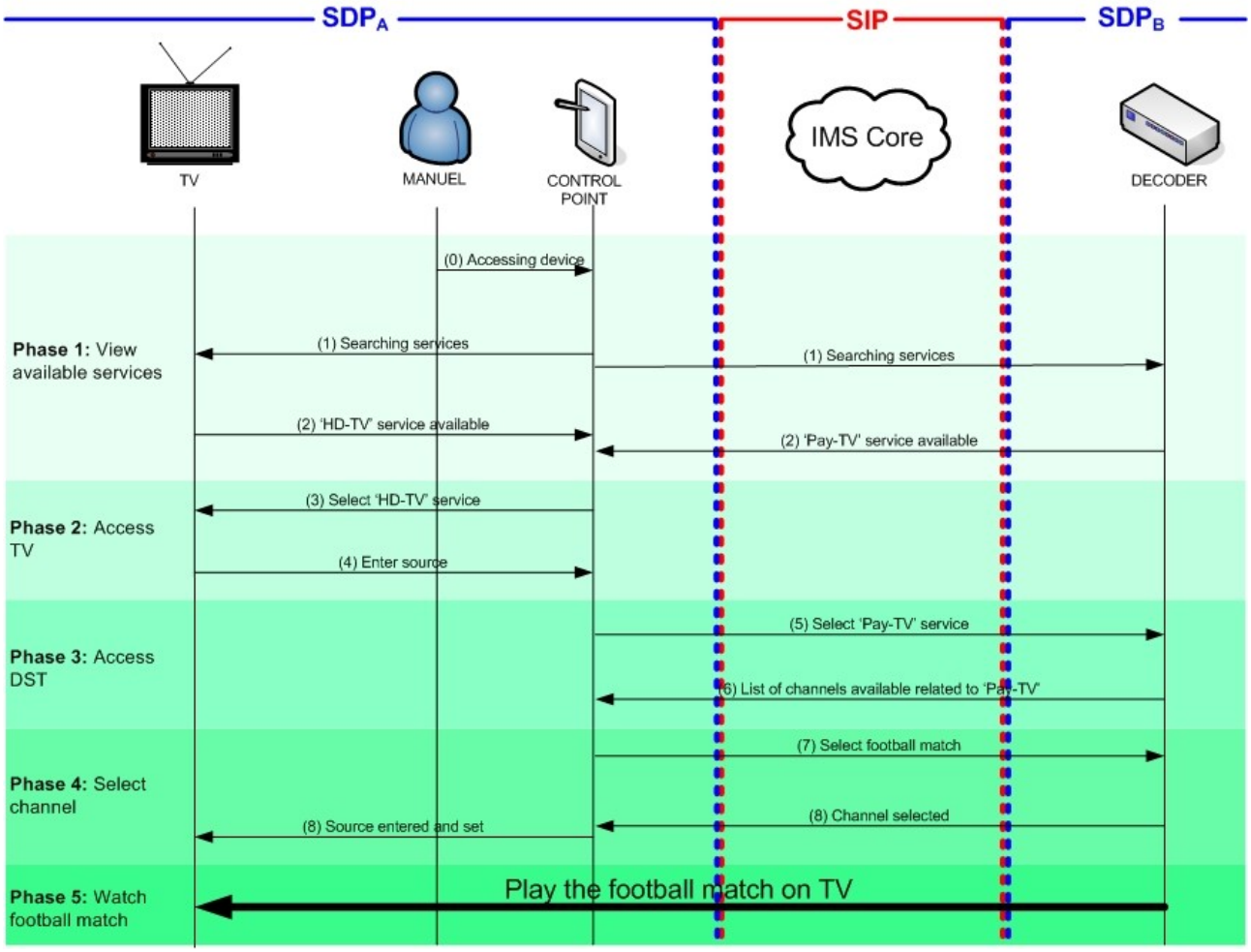

Figure 3.5: High-level flow diagram between remote digital satellite system and local TV

Now Manuel wants to use one of the services offered by his DST system. That is the reason why SV is important here. Antonio does not want to access the decoder, he wants to see the services offered and select one of them in order to enjoy its benefits.

3.3 Remote upload use case

The general case case where users are away from their homes and want to store their multimedia contents on their home storage devices is considered. Two scenarios are developed: first, live-uploading the pictures taken from a camera to a remote digital photo album; and second, live-recording a video from a digital video camera to a remote personal video recorder.

3.3.1 Live photo album

Antonio has got a digital photo album (DPA) where he stores all his photos. The old paper fromat photo album has been replaced by a digital device composed by a HD-screen and a hard disk as the storage medium for the stored photos. The service providing storage is called 'DPA Folder'. The following story has been explained according to the steps shown in Figure 3.7.

Antonio is celebrating a party with his friends in a downtown disco on his thirtieth birthday. They are having so much fun that Antonio takes a lot of pictures with his camera. After taking all those pictures, he realizes that he cannot shoot anymore because the camera flash memory is full. Since he is celebrating his birthday and he considers thirty to be a special, important age, Antonio gets a bit annoyed. In order to solve this problem, he decides to shoot more pictures and save them into the digital photo album he has got at home. Therefore, with his cell-phone {0} he searches {1} for the services available within his environment and two services show up {2} on the mobile phone screen: one belongs to the DPA located at home called 'DPA Folder' and another belongs to the camera called 'Camera Saving Location'. He selects {3} the latter and a source location is then requested {4}. In this way, he selects {5-6} the 'DPA Folder' service as the source so that all the pictures taken are directly stored on the DPA. Now, Antonio keeps taking pictures with his friends and keeps having fun for the next night hours as he will be able to remember that his thirtieth birthday was a nice celebration for a nice number.

The logic diagram is shown in Figure 3.6:

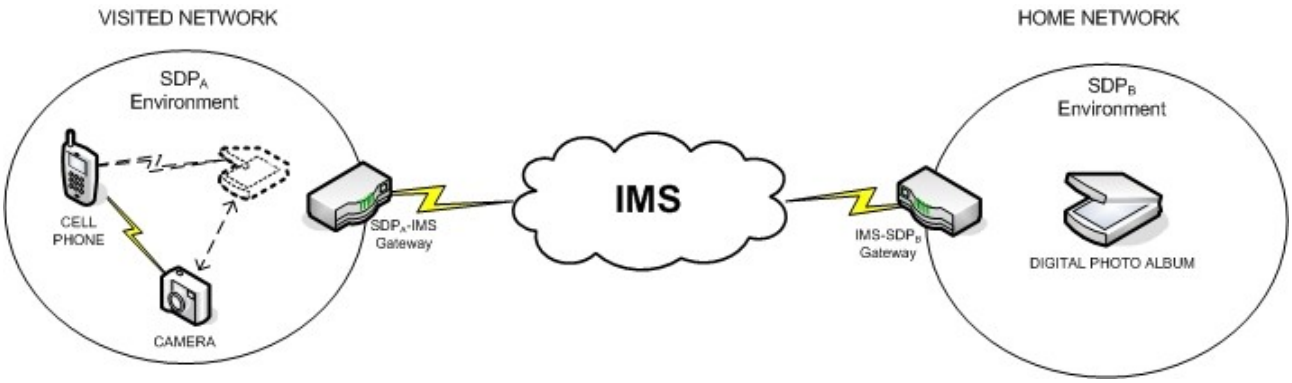

Figure 3.6: Logic diagram for uploading photos to the digital album

The cell phone is the control point within the disco. Before shooting, with his mobile phone selects a service from the camera which lets the user choose the destination of the pictures taken. This destination is set automatically by choosing the service belonged to the DPA.

SV's procedure brings a remote digital photo album into the disco making its services available as if the it would be in the visited network, and is described in Figure 3.7:

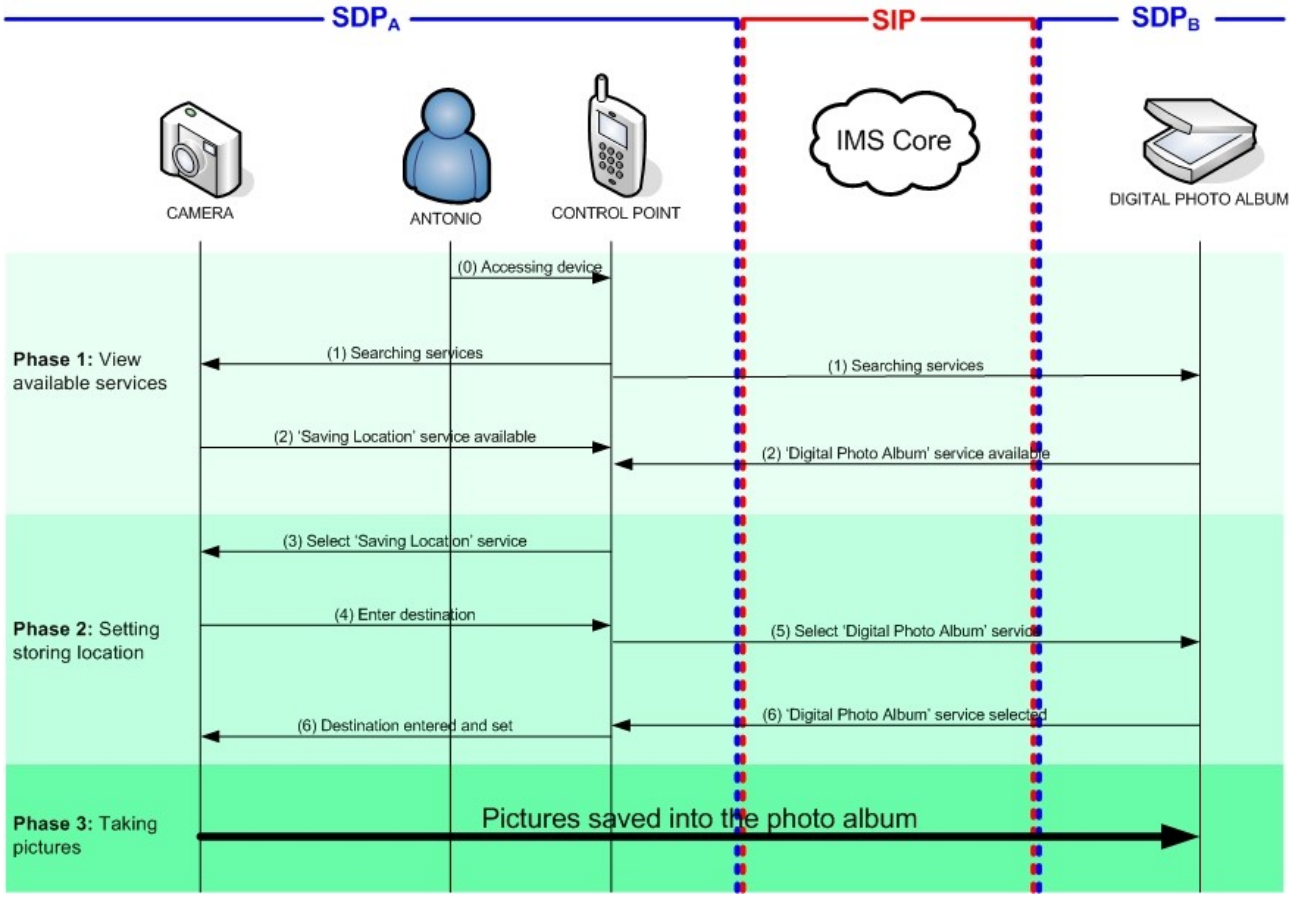

Figure 3.7: High-level flow diagram between remote photo album and digital camera

Once more the advantages of having a service which lets the user to set up automatically the 'download location' of a picture taken from a digital camera are convenient for the customer.

3.3.2 Live video recording

Pedro, natural from Spain, is staying in Norway in order to complete his master studies. He has bought a Personal Video Recorder (PVR) which is used for recording all kind of videos. This recording is provided by a service called 'PVR Videos'. The following story has been explained according to the steps shown in Figure 3.9.

During his stay in the Nordic country, he gets so amazed of the beautiful Norwegian snowed sights that he wants to make a video about the city and the landscapes surrounding his place. Since it is the first time he has ever seen snow, he wants to record a video with the best quality available. Unluckily, he realizes that the storage device within his video camera is not large enough to record a long time video with the highest resolution. As a solution, with his mobile phone {0} he searches {1} for the services available and two services show up {2} on the cell phone screen: one belongs to the PVR located at his Norwegian home called 'PVR Videos' and another belongs to the video camera called 'Video Camera Saving Location'. He selects {3} the latter and a source location is then requested {4}. In this way, he selects {5-6} 'PVR Videos' service as source so that all the live-recorded videos are saved in his PVR located at his Norwegian home. As his PVR has a lot of capacity, Pedro can record a long video so that, once he is back in Spain, he can show his family a great recorded documentary about the beautiful snowy Norwegian life.

The logic diagram representing the frameworj Pedro is involved is shown in Figure 3.8:

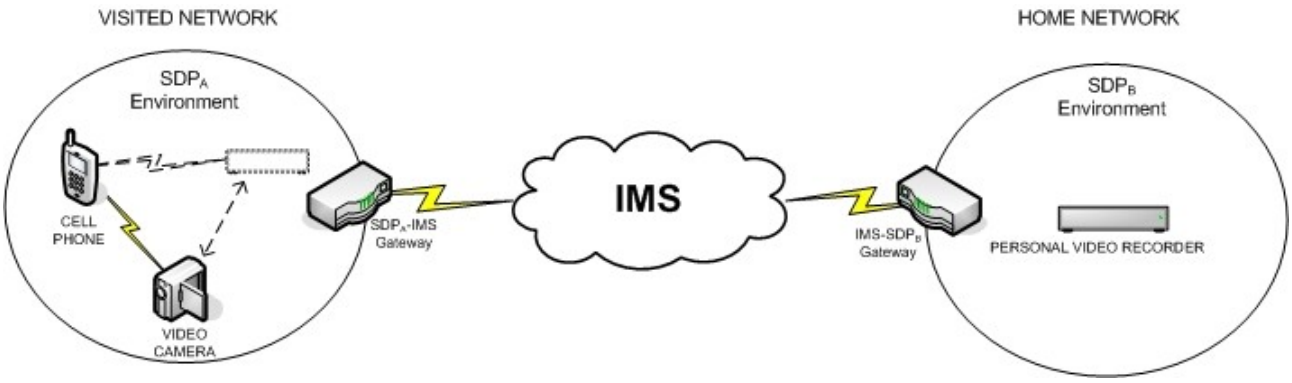

Figure 3.8: Logic diagram for recording a live documentary to the PVR

Summarizing this use case, it is shown that it is the same as the one for the live photo album: in the previous case a digital camera as a source and a digital photo album as storing destination were considered: here, a video camera as source and a PVR are taken instead.

The cell phone is the control point used for the documentary. With the mobile phone a service from the video camera is selected and therefore lets the user choose the destination of the videos recorded. This destination is set automatically by the service offered by the PVR.

SV brings a remote PVR into the disco making its services available as if it would be in the visited network.

The flow diagram related to the SV is shown in Figure 3.9:

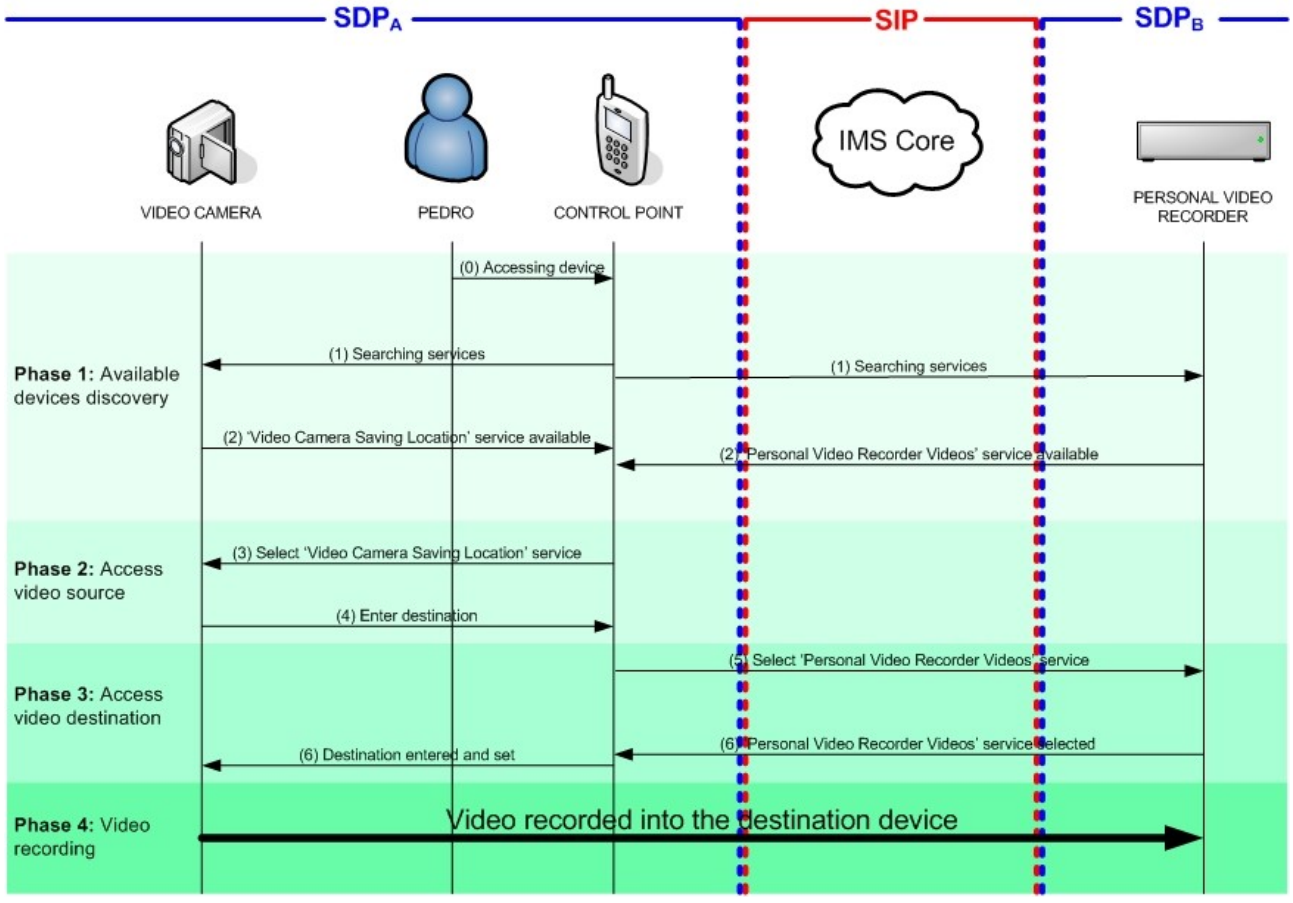

Figure 3.9: High-level flow diagram between remote PVR and video camera

A great advantage of such a service is clearly seen, since it lets the user set up automatically the recording destination of a live-video being recorded from a video camera inside a city or even in the mountains.

3.4 Remote control use case

The general case where users are away from their homes and want to control their home appliances is considered. Two scenarios are developed: first, controlling a remote washing machine in order to have end-user's clothes cleaned before arriving at home; and second, monitoring a remote home security camera from a car.

3.4.1 Building management

The following story has been explained according to the steps shown in Figure 3.11.

Ana has got a washing machine which is intelligent like the rest of her home appliances. Ana has just got up in the morning and is preparing to go to job. Before leaving home, she takes a shower, has breakfast and loads the washing machine in order to hang her clothes once she is back home. However, since she got up very late because of having spent another sleepless night, she is in a hurry and she forgets to push the start button of the washing machine before leaving home. After thirty minutes, while in a traffic jam on the way to her office by bus, she remembers her lack of concentration and decides to start the washing task through {0} her PDA-phone mobile device. In this way, with her mobile device she looks for {1} the washing services offered {2} by the washing machine and she selects {3} a service called 'Start Washing'. Several hours later, Ana reaches home and she can see that the washing machine had finished its task. Now, Ana can hang and dry her clothes as planned.

The logic diagram is shown in Figure 3.10:

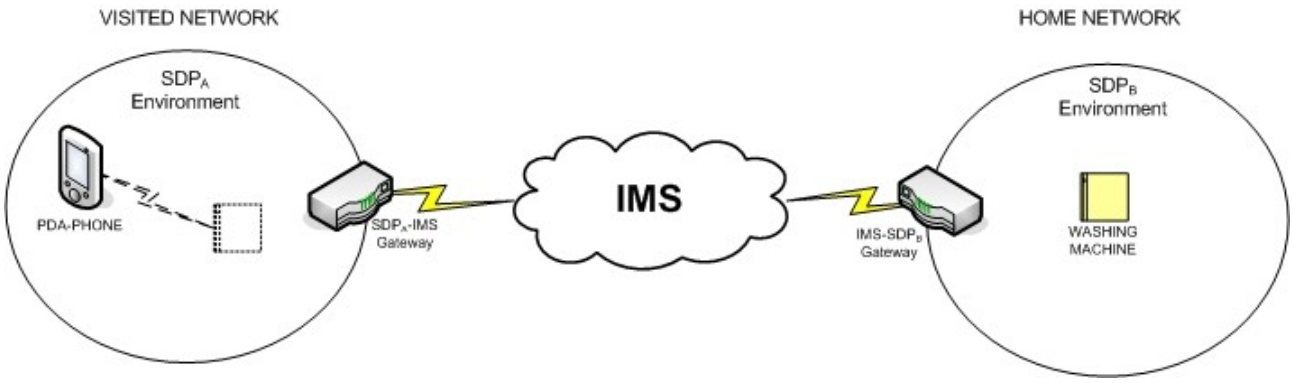

Figure 3.10: Logic diagram for scheduling a washing machine from a bus

Ana's PDA-phone is the control point. Through the PDA interface she looks for the services offered by her different home appliances and she selects the service of the washing machine called 'Start Washing'.

As shown in Figure in 3.10, SV brings the remote washing machine into the bus local network, making its services become available within the abovementioned network as if it would be in the bus.

The main flow diagram is shown above in Figure 3.11:

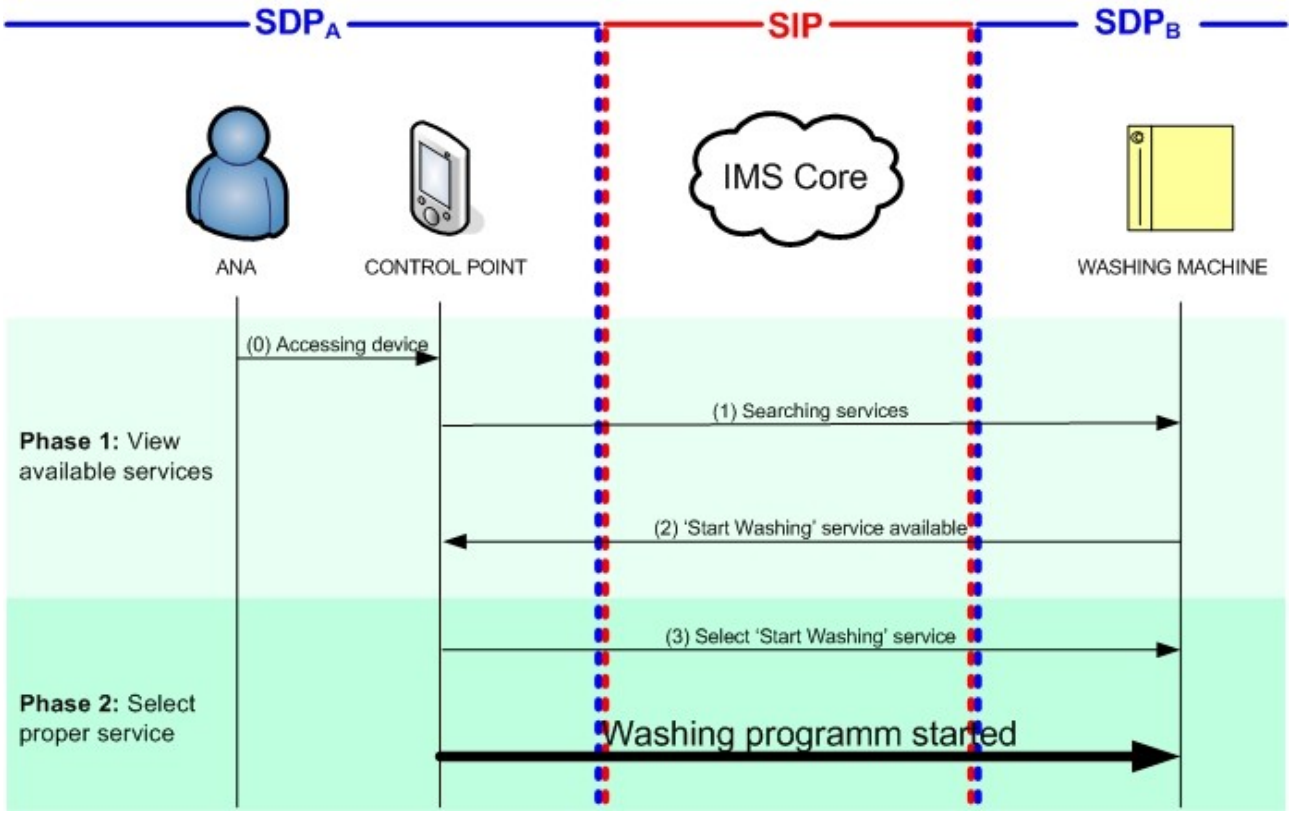

Figure 3.11: High-level flow diagram between remote washing machine and PDA-phone

The washing machine was an scenario example involving home appliances. Another home appliance could have been considered too, such as the oven, TV, dishwasher, fridge, or freezer.

3.4.2 Home security

Pepe has got an Advanced Security System (ASS) installed at his place for suvelliance of house while away. All main entrances to the house, such as windows and doors, are monitored so that if someone tries to open them, the ASS sends a warning signal to his mobile phone. Additionally, several hidden security cameras are distributed in all the rooms of the house and controlled by the ASS as well. The ASS offers several services such as 'Security Cameras'. The following story has been explained according to the steps shown in Figure 3.13.

One lovely night, Pepe and his girlfriend are driving to the best restaurant in the city in order to celebrate their relationship with a great dinner. Suddenly, he receives a message sent by the ASS informing him that one window of the house has been opened. Just to check, he wants to transfer the video from the home security cameras to the browser screen installed inside the car through his cell phone. Thereby, through {0} his cell phone he searches {1} for the services available and

two services are discovered {2}: one belongs to the ASS called 'Security Cameras' and another belongs to the browser screen installed inside his car called 'Screen'. He selects {3} the latter and a source location is requested {4} immediately. In this way, he selects {5-6} the 'Security Cameras' service as source, so that he can watch what is going on his home. Unluckily, he realizes that two masked men are inside the house stealing his multimedia home cinema. He calls the police and goes then home quickly.

The logic diagram takes as follows in Figure 3.12:

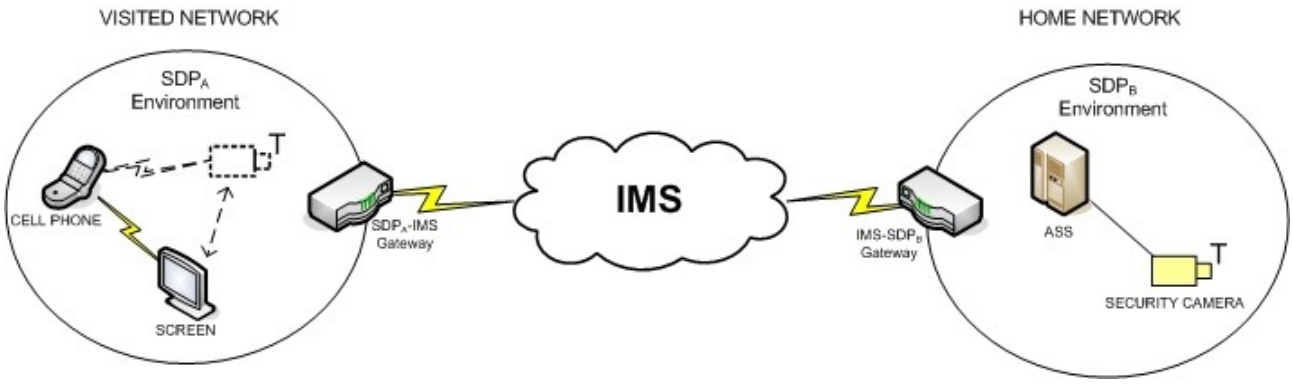

Figure 3.12: Logic diagram for remote monitoring of house security

Pepe's cell phone acts as the control point for the UPnP environment within the car. It searches for the services available such as those offered by the ASS and the browser screen. He selects the service related to ASS called 'Security Cameras' and he transfers the video to the browser screen.

As it is shown, SV's procedure brings the home security camera into the car local network, making its services available within the abovementioned network as if it would be in the car.

The flow diagram is shown in Figure 3.13:

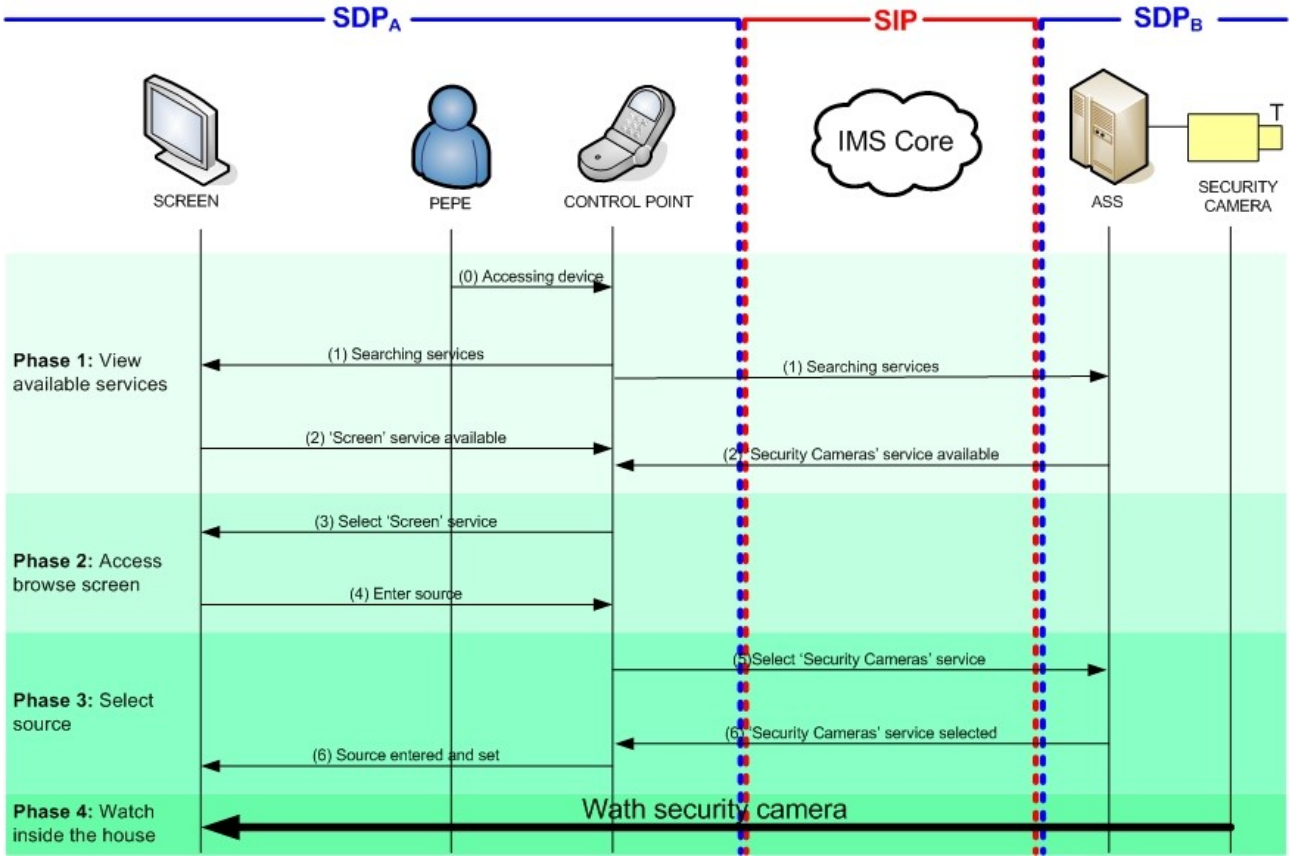

Figure 3.13: High-level flow diagram between security camera and remote monitoring screen

Again, the user can easily access his security system in case the home is being threatened by third parties by choosing one of the services listed on the mobile phone permitting the end-user to pay attention to what is happening within his home.

3.5 Features of Service Virtualization for end-users

The use cases previously analyzed has shown the importance of the services offered by SV for people in an ordinary day: they can enjoy remote services from everywhere such as at the job office, inside a car, in the natural field, or even at a disco. On the other hand, nowadays some remote access methods let end-users gain access to remote devices as well. However, these remote access methods limits them to achieve remote access, such as getting movies from a remote location, or just store pictures to a remote device.

In this section, the advantages of SV compared to normal remote access are explained based on the previous use cases, reinforcing in this way the importance role that SV plays for end-users.

3.5.1 Easy remote access

When normal remote access is concerned, the way end-users remotely access their home devices is not as easy as the way SV uses. For remote access, end-users must first set up both

the remote device they want to access, and the device from which they want to access it.

Let's take an example with File Transfer Protocol (FTP) [20] and 'Streaming media' scenario. First of all, end-users should configure the FTP-Server in terms of ports, shared directories, nicknames,or passwords; in order to access the multimedia files they want. Then, with a laptop, for example, they could get such media content through a FTP-client entering the FTP-host-address, port, nickname and password; and then pass it to the screens with a memory stick.

What SV does is to facilitate for end-users such steps with both the advantages that service discovery mechanisms provide through control points, such as find screen's and remote media server's services, or make them to interact with each other; and the advantages that the remote access functionality provides the control points in order to find and control the remote device.

Definitely, SV becomes really useful for end-users in terms of easy remote access, making the 'dirty job' end-users might have to do in normal remote access case.

3.5.2 Mobility

Using a mobile phone with a control point built-in, end-users need not use other devices for such remote issues. `However, normal remote access requires end-users to use specific interfaces.

For example, if users wanted to remotely access a device via FTP, they would need an FTP-client interface to achieve this. Or if users used the Virtual Private Network (VPN) [21], they would need to use an VPN-client as well. Normal remote access limits end-users to use specific client-interfaces in order to gain access to remote devices. Regarding the 'Live photo album' scenario, if end-users wanted to upload their pictures through a normal remote access mechanism, they would have to deal with that specific mechanism, meaning they would not to be able to use their mobile phone in case this remote mechanism would not be implemented inside it. Then they would have to go to a terminal which provides such mechanisms in order to upload pictures, consequently making end-users leave the tasks they are dealing with at the moment such as celebrating your important thirtieth birthday as it is shown in a previous scenario.

On the other hand, since IMS is the used wired core network, it has mobility benefits [19] for end-users, the concept of home operator and visitor operator [25] disappears, letting end-users to use SV with whatever operator is serving them.

In this way, end-users need not move from one place to another just to achieve this remote necessities. SV fixes this remote issue with just the use of the mobile phone as a terminal, hence giving mobility benefits to end-users, and consequently giving the possibility of remote access from anywhere, without caring the operator serving them.

3.5.3 Services instead of devices

End-users sometimes want to access a remote device in order to use a particular functionality of it instead of making use of the whole entity. Regarding normal actual remote access, it is very common to connect two devices peer-to-peer, server and client, and once they are connected the client gets something particular from the server. On the other hand, when SV is concerned, this access to a remote device is done in terms of the services the remote device provides.

Let us take the 'Home security' scenario case where end-users use, for example, Remote Desktop Protocol (RDP) [22] as remote access mechanism. They gain remote access to the computer who handles the ASS, but they have then to find the service regarding the security

cameras. They cannot access directly the remote device's service they want to access to. However, with SV benefits as far as devices' services are concerned through service discovery mechanisms, end-users enjoy the remote services they directly want to access to.

Definitely, SV becomes easier for end-users since SV cares about the services provided by remote devices instead of just caring about remote devices, issue that is cared by normal actual remote access mechanisms.

3.5.4 Considerations

If the features about SV aforementioned are considered, it is assumed that normal remote access cares more about how to gain access to remote devices in terms of technical issues such as nicknames, passwords, which server-interface is used, or devices' URL. This means that normal remote access is not so easy forend-users who wants to use remote specific functionalities of these devices. However, SV offers several advantages compared to normal remote access in terms of easy remote access, mobility, and services; implying an ease of use of remote services from anywhere for end-users who wants to use remote devices' specific functionalities.

Chapter 4

Solution design for Service Virtualization

This chapter explains the solution design, based on a previous use case, for Service Virtualization. First of all, the solution goal of our project is explained in a high level overview through a logic diagram and a flow diagram in section 4.1, followed by requirements in section 4.2. Further details are given in section 4.3 explaining both how the solution architecture has been deployed and its functionalities with a figure depicting them. Afterwards, SV functionality has been designed and described in section 4.4. Finally, the detailed operation steps of our solution are described in section 4.5 .

4.1 Solution goal

The solution goal is to realize a video stream transfer from a video camera located in 'Residential Network A' to a PVR within 'Residential Network B' with the advantages that SV provides.

A control point within 'Residential Network A' is the device used for discovering the services provided by both the video camera and the remote PVR as if this would be within the abovementioned local network. Once these services are discovered, the control point handles the video camera in order to record such a transfer into the remote PVR.

The scenario where this solution is involved is shown in Figure 4.1:

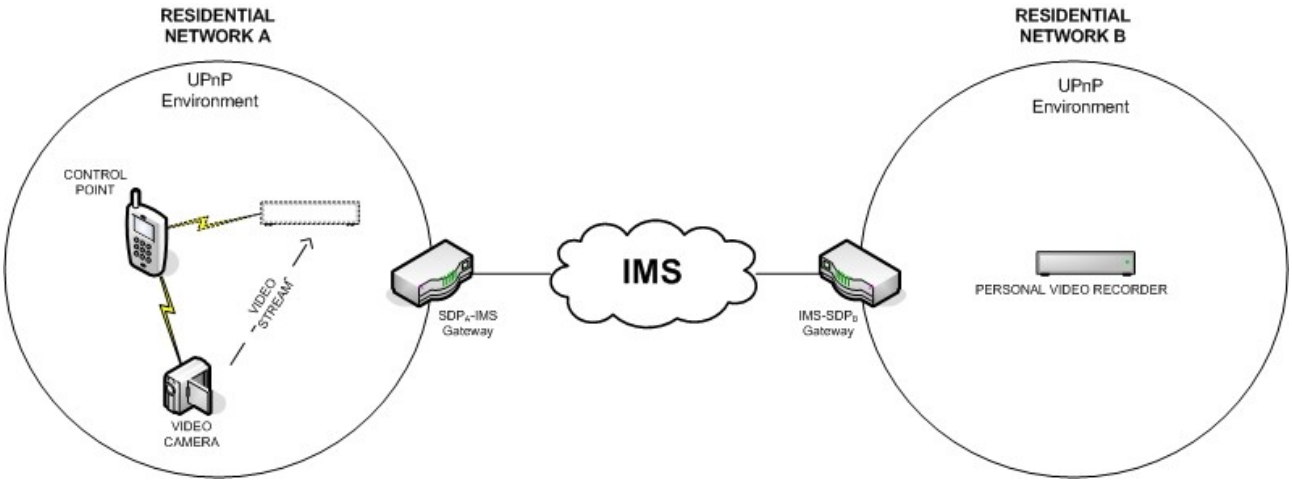

Figure 4.1: Logic diagram solution using service virtualizaton

The phases of operation for the video transfer are based on the use case scenario called 'Live video recording'. In this way, the high-level phases of the flow diagram is shown in Figure 4.2

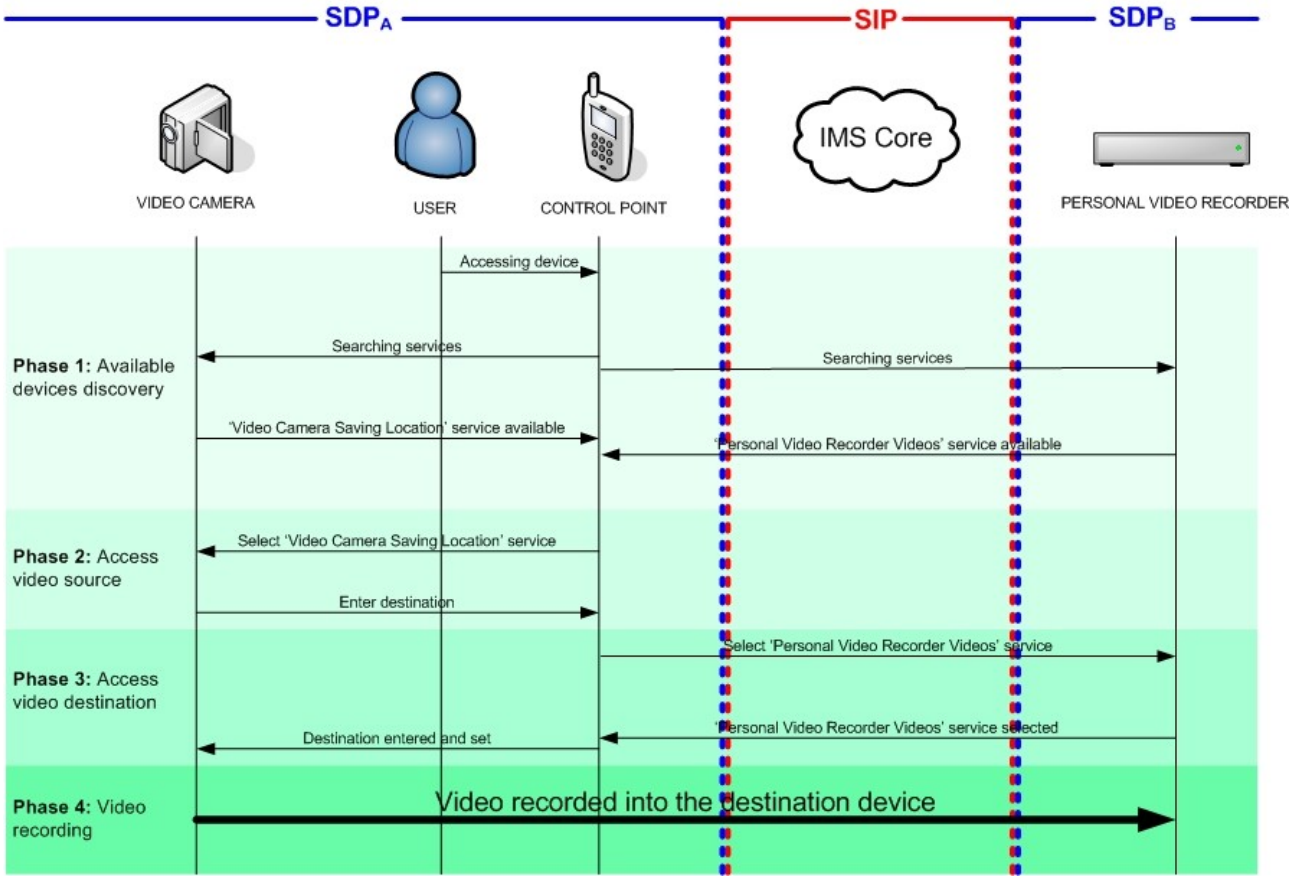

Figure 4.2: SV phases for the designed solution

4.2 Requirements

- A control point must be provided in order to discover the services available along the local network. This entity can be implemented in a mobile phone, laptop, PDA-phone, etc.
- A SIP account provided by the contracted operator.
- A broadband connection.
- A video camera from which the video stream is obtained, such as mobile phone, video camera, camera, or TV, compatible with UPnP.
- A PVR in which the video stream is stored, such as storage hard-disk, PVR, or media server, compatible with UPnP.

4.3 Solution architectural design

The solution infrastructure is composed of two local networks, 'Residential Network A' and 'Residential Network B', which are connected to each other through IP Multimedia Subsystem (IMS) as the core network. The video camera from which the data is transferred is located in the former local network and the PVR -also called remote device- to which the data is transferred is located in the latter -also referred to as the remote local network-.

Concerning both residential networks, among all of the SDPs listed in section 2.1.1, the most relevant are UPnP, Rendezvous and Bluetooth SDP. However, due to both the great acceptance that UPnP has experienced in the market about residential audio/video services and the advantages UPnP provides [7] against Rendezvous and Bluetooth SDP in features like service usage, service status inquiry, authentication, confidentiality and integrity; UPnP is the chosen SDP for these residential networks. Indeed, UPnP is an easy protocol to treat in terms of service development and configuring, which implies a fast non-cumbersome development for our prototype implementation. Therefore, in SV UPnP adopts the role of discovering the devices located in the external network as well as the virtualized device located in the remote network.

In order to transfer signalling between IMS and the UPnP environment a residential gateway has been deployed at a Service Discovery Gateway (SDG) and a Service Virtualizer.

On the one hand, the SDG establishes the communication between both networks in order to discover the presence of the remote device. SDG_A initiates the establishment connection signalling between both local networks and SDG_B accepts the establishment connection requests from SDG_A and sends presence information to it, so that SDG_A is aware of the presence of the PVR.

On the other hand, Service Virtualizer enables the function of SV, hence virtualizing the PVR into 'Residential Network A'. Its principle of operation is described in the next section.

To conclude, a logic diagram is shown below in Figure 4.3:

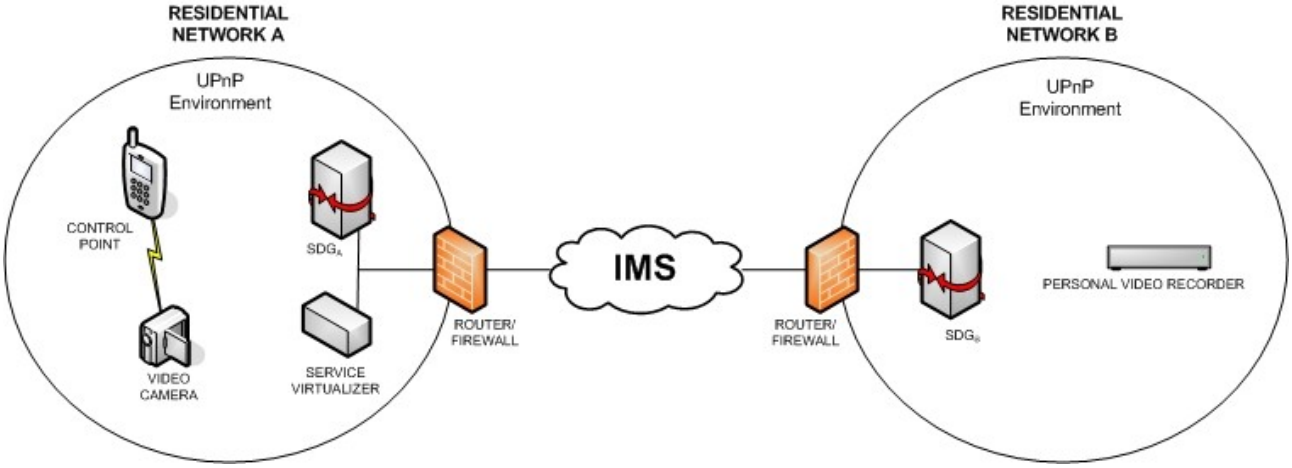

Figure 4.3: Solution architectural design

4.4 Service Virtualizer design

The functionality introduced in the architectural design in order to enable SV is called Service Virtualizer (SVR). Since SV is based on remote service access as it was explained in chapter two, SVR is designed on top of SDG.

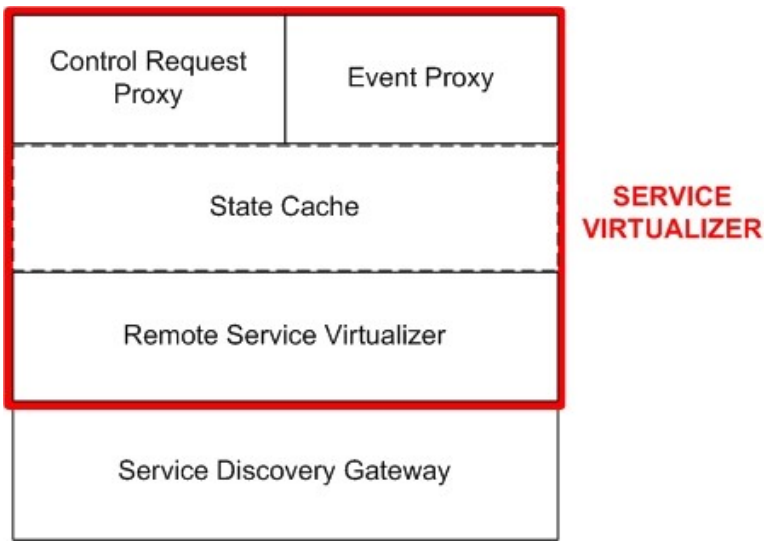

Figure 4.4: Stack architecture of Service Virtualizer

As the remote device is to be virtualized as a local one, it will be discovered, evented, and controlled by the control point as if it would be a local UPnP device. In this way, as depicted in Figure 4.4, SVR is composed of four functionalities: Remote Service Virtualizer, State Cache, Control Request Proxy and Event Proxy.

Remote Service Virtualizer (RSV) is the main functionality of SV, since it is responsible of bringing remote devices into 'Residential Network A'. Its aim is to get the information regarding the remote device that is to be virtualized, this is, the device description and service description, then hosts the remote device into 'Residential Network A'. To do so, SDG_A receives such an information through a notification message stemming from SDG_B and then RSV generates a new description device document and a new service description document afterwards. Once these documents are generated, RSV hosts the remote device using these two documents, so that the control point and local devices can recognize it as a local device. Every time the control point searches for local devices, RSV will realize these procedure in order to inform the control point through these documents about the presence of such a remote device as a local device.

The size of the information field regarding the device description and the service description sometimes is very large, causing some problems during the transmission of the notification message which includes this information. To solve this issue, the notification message includes an HTTP address regarding the device description and service description of the remote device. This HTTP address points to the address of SDG_B since it is the functionality that holds the information concerning the device description and service description of the remote device to be virtualized. Thus, once SDG_A receives the HTTP address pointing to this information, it will fetch it automatically and will pass it to the RSV afterwards.

Control Request Proxy (CRP) is the functionality which handles all the operations concerning SOAP protocol, so that the virtualized device can be controlled from 'Residential Network A'. Once the control point is aware of the virtualized device and tries to consume one of its services, the control point will send SOAP requests to the SVR, so that CRP will forward such requests to SDG_B, and from it to the remote PVR. For the way back, the procedure is exactly the same, but from SDG_B to CRP.

Event Proxy (EP) is the functionality which handles the operations concerning GENA protocol, so that the virtualized device can be evented from 'Residential Network A'. If actions change their state in the remote device, SVR will be aware of such a modification thanks to EP. If the remote PVR is evented by the control point through GENA requests, EP will forward the event signalling to SDG_B, and from it to the PVR. For the way back, the procedure is exactly the same, but from SDG_B to EP.

State Cache is the functionality which optimizes the performance of SV, updating the properties that have been modified in the remote device. This functionality is optional to be implemented, since SV does not require this extra functionality in order to completely enable the virtualization of remote devices.

4.5 Solution: Principles of operation

This section goes into further details about signalling steps, such as services, protocols, methods and software according to the four phases described in the first section of the present chapter. All the steps described in each section have been realized according to the theoretical background regarding UPnP and remote service access explained in chapter 2, as well as information extracted from documents regarding digital security cameras from [17].

4.5.1 Phase 1: Available devices discovery

The aim of this phase is to search for the available devices in its vicinity in order to make the available devices show up on the control point screen so that the user is free to choose one of both devices. SSDP is the protocol used for local networks signalling as it was explained in point

2.1.2.2.

The first task consists of bringing both the device and service descriptions from the remote PVR located in 'Residential Network B' to 'Residential Network A'. To do so, SDG_A subscribes to SDG_B in order to be notified about the events occurred in the remote local network regarding steps {2-5}. Once SDG_B figures out the remote device's information {6-10}, it sends both PVR's descriptions to SDG_A {11}.

The second task consists of virtualizing the remote device from the information that stems from the SDG_A. As it was explained before, SVR (specifically RSV) generates two new description files, one for the device and another for the service in order to virtualize the PVR. However, this is the step in which only the availability of the virtualized PVR is sent to the control point, as it is shown in {12}. The information about both the device and service descriptions is delivered afterwards once the control point tries to access the remote device (phase 3).

Finally, the last task consists of receiving the information concerning the availability of the video camera {14-15}.

The signalling flow diagram for the first solution work operation step is depicted in Figure 4.5:

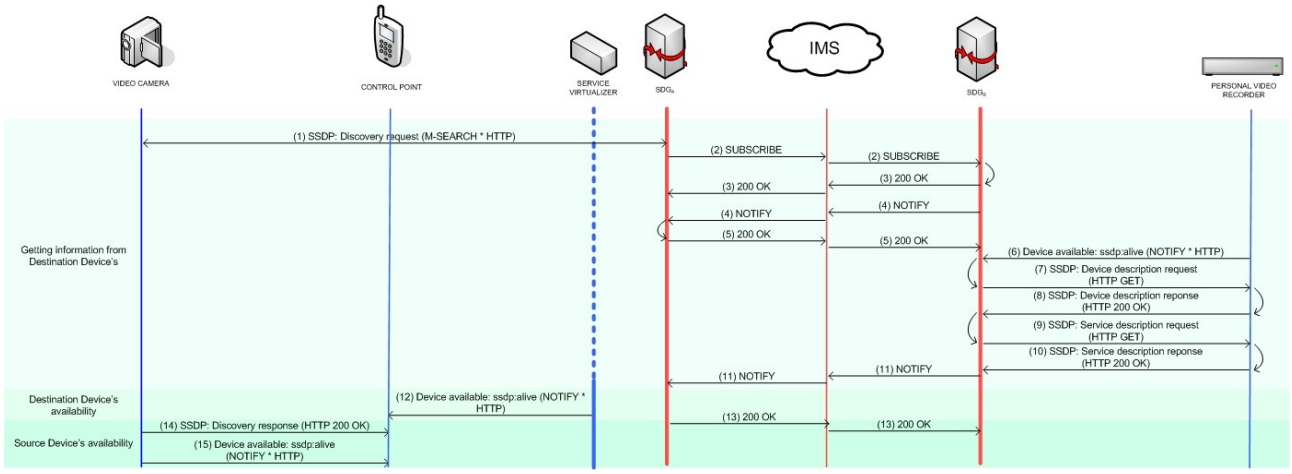

Figure 4.5: Searching for available devices in the vicinity

4.5.2 Phase 2: Access video source

The aim of this step is to get the video location that is within the video camera in order to pass it to the PVR afterwards. Here SSDP and GENA protocols are used for the signalling concerning 'Residential Network A'; the former for getting device and service informations, and the latter for getting information about state variables related to actions.

Once the available devices have shown up on the control point screen, the video camera is selected. Like this, the control point gets the information about both the device {16-17} and its services {18-19}.

The following step consists of subscribing to the video camera in order to get the information about its state table changes {20-23}. Depending on the state of the variables which comprise the abovementioned table, the user is ready to invoke a proper action according to the status variables.

This phase is summarized in Figure 4.6:

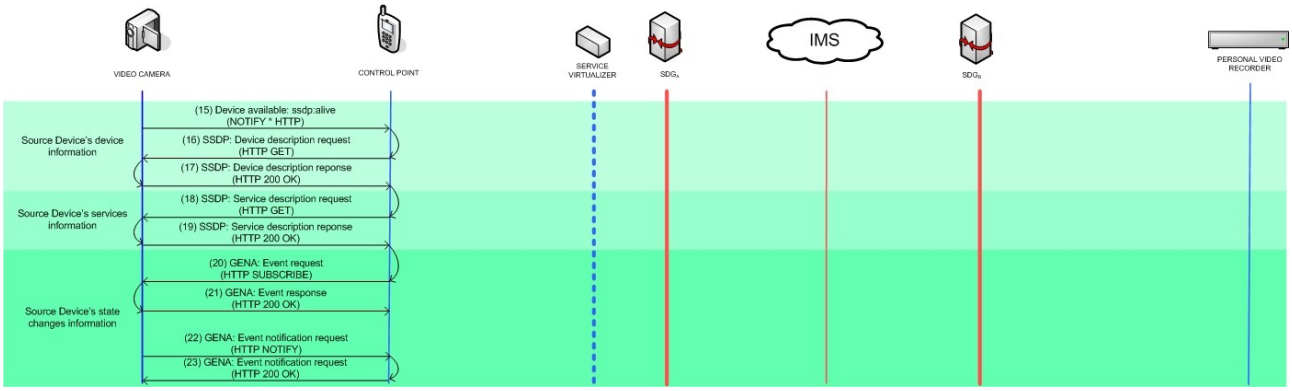

Figure 4.6: Getting the video from a video camera

4.5.3 Phase 3: Access video destination

The aim of this phase is to set up the remote device so that it becomes ready to start recording from the video source. SSDP, GENA and SOAP protocols are used in this phase in order to get device and service descriptions, then get information related to state variables of the remote PVR, and invoke its actions afterwards, respectively.

The first task consists of getting both the device and the service descriptions belonging to the remote device from SDG_A. Since SVR got this information in the first phase, RSV only has to pass it to the control point regarding steps {24-27}.

As a second task, the control point subscribes {28} to the remote device in order to get its state variable information in the same way as in the previous phase. However, since the remote device is located in 'Residential Network B', the control point subscribes to SVR (specifically EP). In order to relay GENA signalling over the core network, both residential networks must initiate a new session establishment between each other so that a 'tunnel' connection is set up and ready to be used {29-31}. Once this tunnel is set, EP meets its aim sending the proper signals to SDG_B, whose role is to forward these signals {32-41} to the remote device in order to achieve this event step.

Finally, an action is invoked {42-47} in the PVR just to start relaying the video source {48-49}. In this case, there is no need to set a new tunnel connection for these invocation requests as the previous one is seized up for this issue. Here is where CPR operates in order to translate signalling from 'Residential Network A' to SDG_B.

A signalling flow diagram concerning this step is depicted below in Figure 4.7:

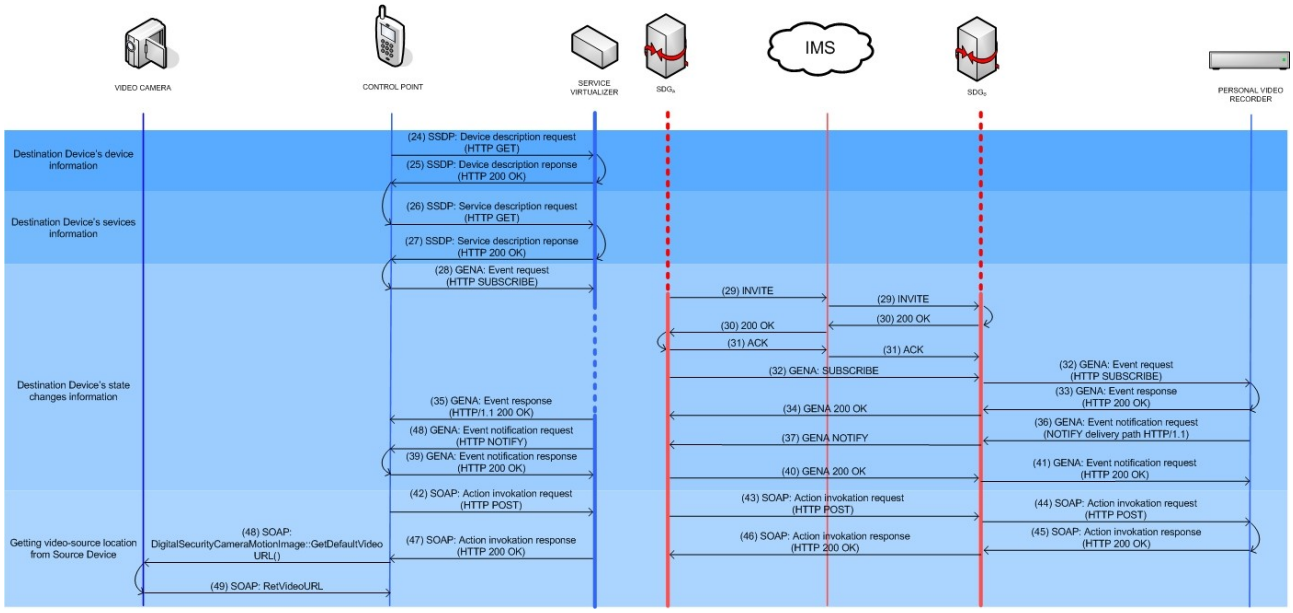

Figure 4.7: Setting up PVR

4.5.4 Phase 4: Video recording

The aim in this phase consists of importing the video from the video camera to the PVR in order to achieve the solution goal. All the operations have been carried out by CRP through SOAP signalling.

Since the control point got the video source location in the previous phase, it must import it to the remote device seizing up the tunnel connection previously set. To do so, the first task for CPR consists of creating an object inside the remote device to which the video is sinked {50-52}. Once the creation of the object is confirmed {53-55}, the video is imported according to {56-58} messages. Finally, an output is returned to the control point {59-61} so that a transfer bar {62-67} shows up on the control point screen in order to follow such importing issue, achieving like this the solution goal. Figure 4.8 shows this last phase.

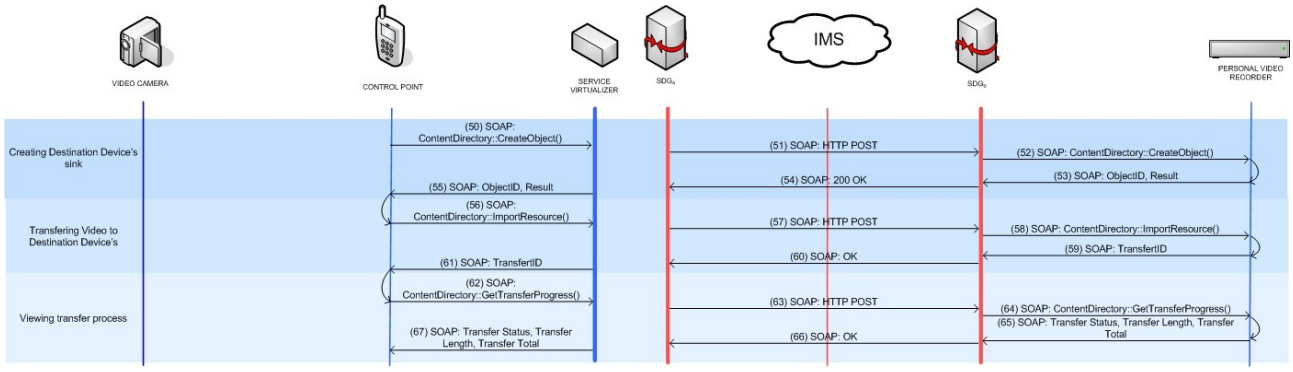

Figure 4.8: Importing video from the video camera to the PVR

Chapter 5

Prototype implementation for Service Virtualization

This chapter describes the prototype implementation developed for the verification of the concept of Service Virtualization. It begins with the aim of the implementation, including a figure depicting the framework in section 5.1. Afterwards, the implementation architecture is explained in section 5.2, which shows and describes all the need requirements in order to achieve the aim of the implementation. How the prototype has been implemented has been explained in section 5.3, followed by the description of the prototype implementation testing in section 5.4. Finally, the description of the verification that shows the prototype implementation works properly has been described in section 5.5.

5.1 Implementation goal

The implementation goal is to realize a video stream transfer from a real media renderer located in 'Residential Network A' to a real media server within 'Residential Network B' with the advantages that SV provides.

A laptop including a control point within 'Residential Network A' is the device used for discovering the services provided by both the media renderer and the remote media server. Once these services are discovered, the control point controls the media renderer in order to import such a transfer into the remote media server.

A scenario where this solution is involved is shown in Figure 5.1 below:

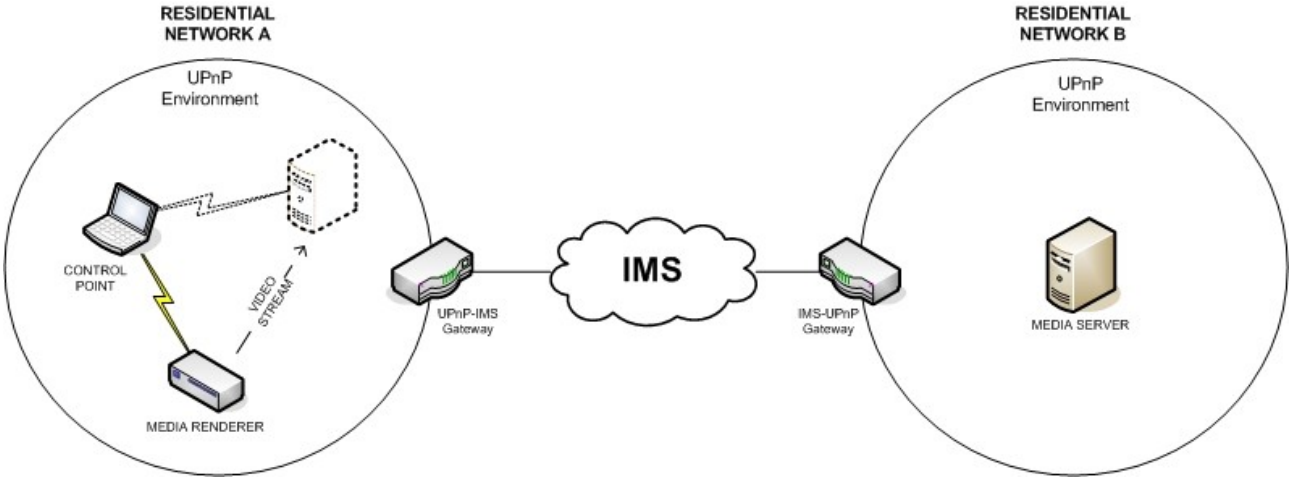

Figure 5.1: Logic diagram showing the solution using service virtualizaton

5.2 Implementation architecture

Like in the solution design of the previous chapter, the implementation infrastructure is composed of two local networks, 'Residential Network A' and 'Residential Network B'. Within the former all devices acquire 10.1.0.X IP address, and within the latter devices acquire 10.0.0.X IP address. Both uses UPnP as a service discovery standard and they are connected to each other through IMS as the core network, using 192.168.1.X as IP address.

A D-Link DSM 320RD is the media renderer used for streaming the video source and a TwonkeyMedia Server embedded in a Mini-MAC A1176 is the media server used for storing such a transfer.

The prototype implementation involves of embedding SVR inside Home IMS Gateway (HIGA) [26], a software package developed by Ericsson [27] which acts as a gateway between a local network with UPnP environment and IMS. Like this, the prototype introduces SV functionality in HIGA, hence virtualizing external services in a UPnP local network. Since HIGA includes SDG functionality, $HIGA_A$ is the one within 'Residential Network A' and $HIGA_B$ is the one within 'Residential Network B', each of them with internal and external IP addresses as depicted in Figure 5.2.

Finally, since a SIP account is necessary in order to operate over IMS, these two accounts have been used:

- alice@ims.ict-fiesta.test for 'Residential Network A', also known as the sip destination address of the remote device to be virtualized.
- bob@ims.ict-fiesta.test for 'Residential Network B', also known as the sip origin address of the remote device to be virtualized.

Figure 5.2 depicts the architecture and the framework where the prototype implementation is involved:

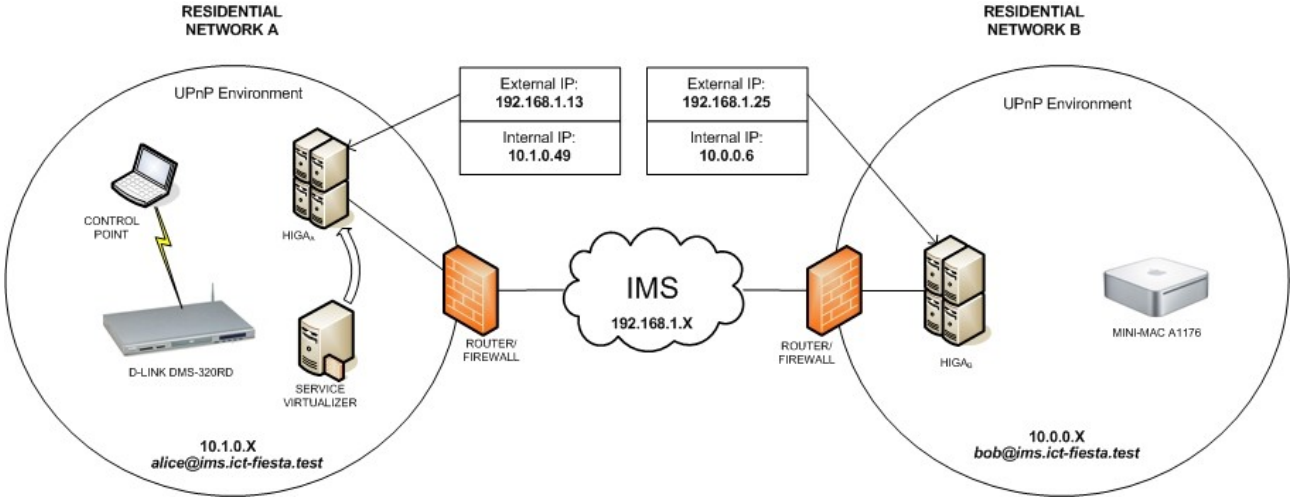

Figure 5.2: Architectural design for the implementation prototype goal

5.3 Service Virtualizer implementation

Since HIGA includes SDG functionality, the SVR has been built on top of it according to the stack architecture explained in section 4.5, hence leveraging the functionalities previously built-in. In turn, SDG is composed of several functionalities, notably Piranha and Hosting as depicted in Figure 5.3:

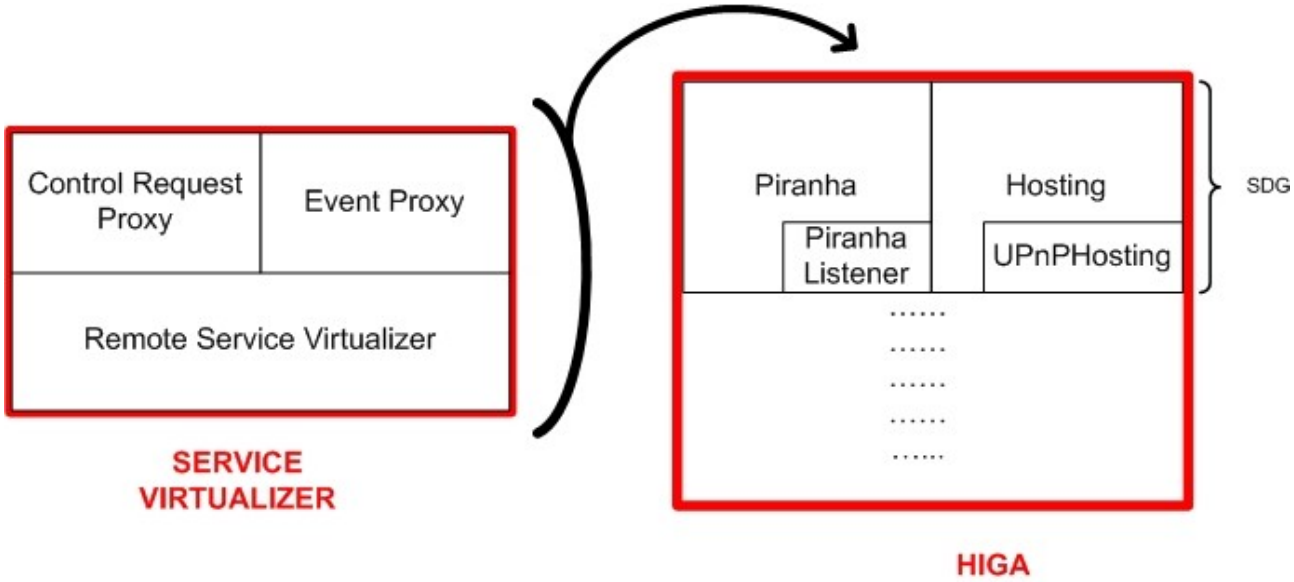

Figure 5.3: Stack architecture implementation

Piranha [1] is the protocol used handle all the operations between the both SDGs, these are, SDG_A in 'Residential Network A', and SDG_B in 'Residential Network B'. Since it passes all

transactions between both SDGs, Piranha has information content related to the SDGs, devices in the vicinity of SDG_B, IPs, ports, etc. .

Hosting is the functionality which hosts all the UPnP devices making them available in the local network. In this case, the media server is the device that is hosted. Since Piranha manages all transactions between both SDGs, it keeps information about the media server. Thus, RSV is the functionality that passes this media server information obtained from the Piranha protocol to the Hosting functionality in order to host the device.

In this way, since RSV must be aware of the operations Piranha is managing, the interface 'PiranhaListener' is implemented in RSV. Since RSV must be aware of events regarding Piranha protocol and must host a remote device, RSV must also include some attributes regarding Piranha and Hosting functionalities, respectively. The class referring to Piranha is called 'Piranha', and the class referring to Hosting is called 'UPnPHosting'. Finally, since some remote devices can be virtualized, RSV uses a list to keep track of virtualized devices in order to unregister these devices through 'ArrayList' class. Moreover, since SVR is to be embedded in HIGA, its class will be part of the class of HIGA, called 'RaHiga'.

Like this, when Piranha is aware of the remote device, RSV will be automatically notified due to the 'processSubscribeNotification' method embedded in this interface implementation. Afterwards, RSV hosts this device calling 'registerDevice' function from 'processPiranhaSdgDeviceAdded' function. If for some reaons a device is unvirtualized, this device must be unregistered when 'processPiranhaSdgDeviceRemoved' or 'processPiranhaSdgUnsubscribed' are called.

Figure 5.4 depicts a class diagram in UML format, which shows the main properties of SVR, as well as how it has been implemented.

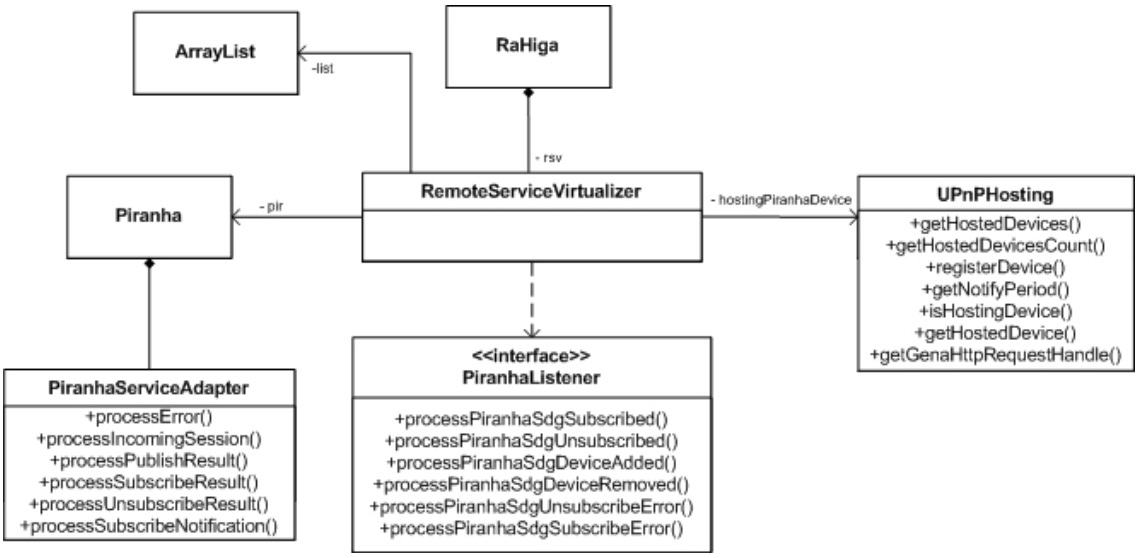

Figure 5.4: Class diagram for prototype implementation

Concerning CRP and EP functionalities, they have not been implemented yet due to the reasons given in section 7.3. These two functionalities shall also be included in HIGA and developed using Piranha protocol too. Thus, these entities should be part of the 'RaHiga' class and refer to the 'Piranha' class. Event Proxy should be related to the UPnP classes concerning GENA, such as 'GenaHttpRequestHandler'; and Control Request Proxy should be related to classes

concerning SOAP and UPnP control classes, such as 'SoapInvoke', and 'MediaServer' and 'MediaRenderer', respectively.

Moreover, the 'List' class has not been used. To use it, importing 'ArrayList' class into 'RemoteServiceVirtualizer' would be the solution to proceed to. Once the device has been registered, 'RemoteServiceVirtualizer' checks if the device has been registered properly through 'isHostingDevice' method belonging to the 'UPnPHosting' class.

5.4 Implementation testing

In order to check that our implementation prototype worked properly, the media renderer should be able to recognize Twonkey Media Server as a UPnP device. To do so, the media renderer was connected to a TV, so that with the remote media renderer controller the user gets into the configuration system of the D-Link media player and then searches for the media server. If the media server is recognized by the media renderer, a message should appear on the TV screen showing the media server name. In case the media server is not found, a message like “no media servers found” should appear instead.

The testing process consists of entering a 'Setup' option of the D-Link DSM 320RD. A setup menu shows up and 'System' option is selected afterwards. Among all the parameters and options available in 'System' menu, one called 'View All Servers' is selected. Finally, a screen shows up informing whether the TwonkeyMedia Server has been found or not. Figure 5.5 depicts the testing set and process:

Figure 5.5: Implementation testing set and process

5.5 Implementation verification

Once SV has been tested, the fact that the media server is being virtualized must be verified. As section 4.4 explains, RSV virtualizes the remote device with the generation of a device description document and a service description document stemming from SDG_A. Since this information

regarding these documents is included in a notification message stemming from SDG_A, watching the content of the message body of such a notification message will prove that the media server is being virtualized properly.

To view the content of this notification message, a network monitoring software is used. This software monitors all information concerning the transactions that occur between both SDGs in the signalling domain, hence showing information like protocols, IPs, packet length, message body, origin address, destination address, etc.

In the message body, the most relevant information concerning SV implementation relates the origin address and the destination address. In addition, the information regarding the device description and the service description is very important as well. However, as it was explained in section 4.4, due to the large size of these information fields, an HTTP address pointing to the location of both descriptions is the information included in the notification message body. As it was also explained in section 4.4, SDG_B is the functionality that holds these descriptions. So, since $HIGA_B$ includes SDG_B functionality, this HTTP address included in the notification message will point $HIGA_B$.

Thus, once the destination address, origin address, and the HTTP address included in the notification message matches the ones chosen in this implementation, the fact that the media server is being virtualized will be definitely verified.

Chapter 6

Results

This chapter gives the results obtained from the realization of the assignments for Service Virtualization regarding the project definition. This realization has been done for the concept explanation of Service Virtualization, the set of use cases for Service Virtualization, importance and advantages of Service Virtualization, the solution design for a particular use case, and the prototype implementation. So, the chapter has been broken down into three section: section 6.1 regarding results obtained from use cases chapter, section 6.2 regarding results obtained from solution design chapter, and section 6.3 regarding results obtained from the prototype implementation chapter.

6.1 Use cases results

The concept of SV has been described in the context of residential networks in section 3.1. This description includes the definition of residential networks, and the framework where SV is involved as depicted in Figure 3.1, such as the wired network, local networks and technologies that can be used within them, and possible devices that can be used for the SV issue.

The end-user benefits, such as enjoying remote services from anywhere, with great ease of use, have been shown through a set of use cases that have been developed and analyzed. These use cases concern users that are away from their homes. One use case for users who want to remotely access their multimedia content located at their homes is described in section 3.2. A second use case for users who want to remotely store multimedia content on their home storage devices is described in section 3.3. Finally, another use case for users who want to remotely access them based on three use cases previously explained.

6.2 Solution architecture results

A PVR can be virtualized into a local network in order to store a video stream transfer from a video camera. The PVR has been brought into the local network where the video camera stays.

For such a realization, a solution architecture has been developed in order to explain how a PVR can be virtualized into a local network. This solution includes the logic diagram in Figure 4.1 and a high-level flow diagram in Figure 4.2 which describe the aim of the solution design in section 4.1; requirements to achieve the solution has been listed in section 4.2; an explanation of the framework where the solution is involved is given in section 4.3, such as the wired network, local networks and the technology used within them as shown in Figure 4.3; and the technical functionality that enables SV has been described in section 4.4, using Figure 4.4 for showing its stack architecture.

Moreover, the principle of operation of the solution design has been described for the devices used for SV in section 4.5. This description is broken down into four phases of operation, each of them described through a figure showing the steps carried out in terms of signalling messages, protocols, and software used.

6.3 Prototype implementation results

A prototype implementation for the verification of SV functionality has been partly developed for a real application. As a first step, a logic diagram in Figure 5.1 describes the aim of the application in section 5.1. Next, in section 5.2 the framework where the application has been carried out has been described as depicted in Figure 5.2, such as wired network, local networks, real devices, the control point used, and the functionalities that enable SV. Finally, in section 5.3 the prototype depicted in Figure 5.3 has been explained according to the class diagram depicted in Figure 5.4.

The prototype demonstrates that a real remote device can be brought into a local network so that a local device recognizes it as a local one. Since SV functionality has only been partly developed, only RSV functionality has been built on top of HIGA, so that a media renderer is aware of a remote media server as it would be a local device. Figure 6.1 shows that the media renderer did not find anything before virtualizing it at the left side; besides, at the right side it is shown that the virtualized media server has been recognized as a local one by the media renderer.

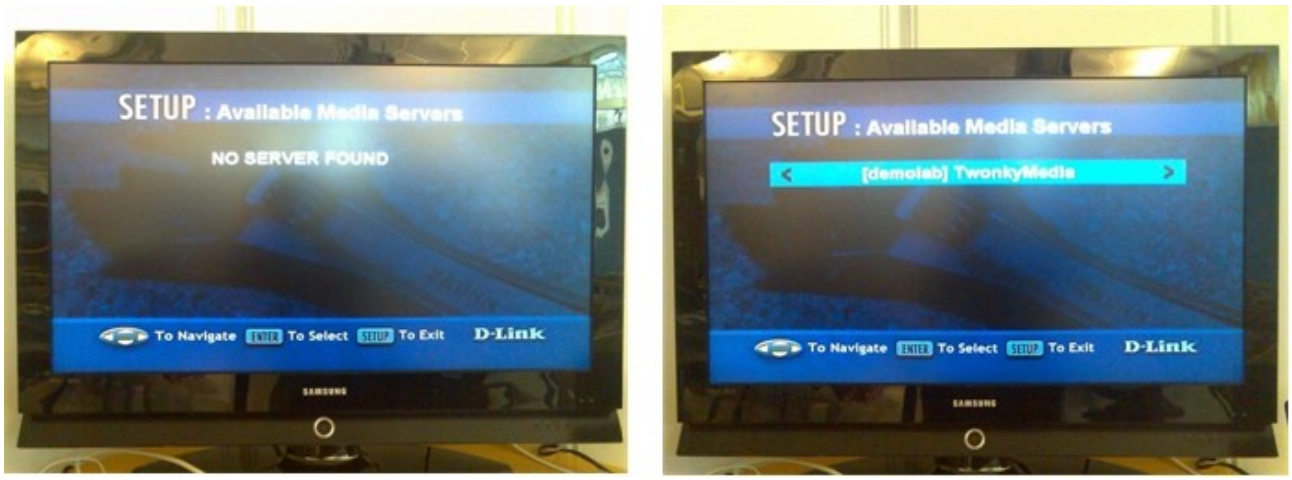

Figure 6.1: Implementation testing results

On the other hand, the following figures verifies that the media server has been brought into the local network properly. Figure 6.2 depicts the network monitoring software showing part of the notification message content. The destination address and the origin address chosen for the prototype implementation match the ones included in the notification message. The destination address of the notification message is marked with a red square, and the origin address is marked with a green square.

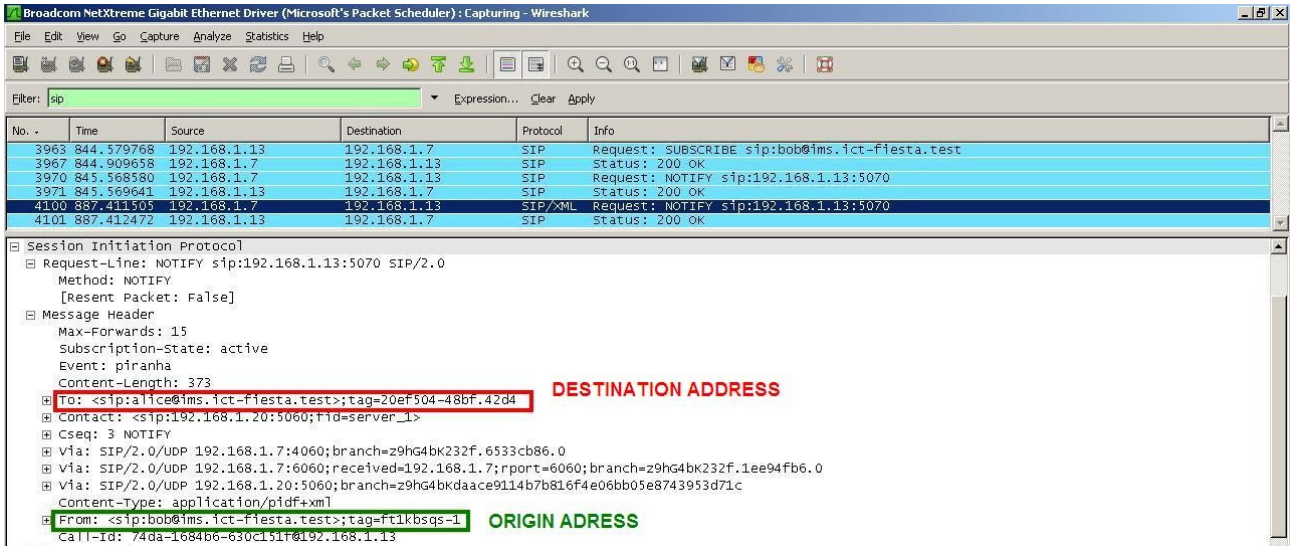

Figure 6.2: Addresses verification for prototype implementation

Moreover, Figure 5.3 depicts the notification message body, in which the address of the device description and service description is included. It is shown that the address of HIGA in 'Residential Network B' matches the one marked with a blue square included in the message body.

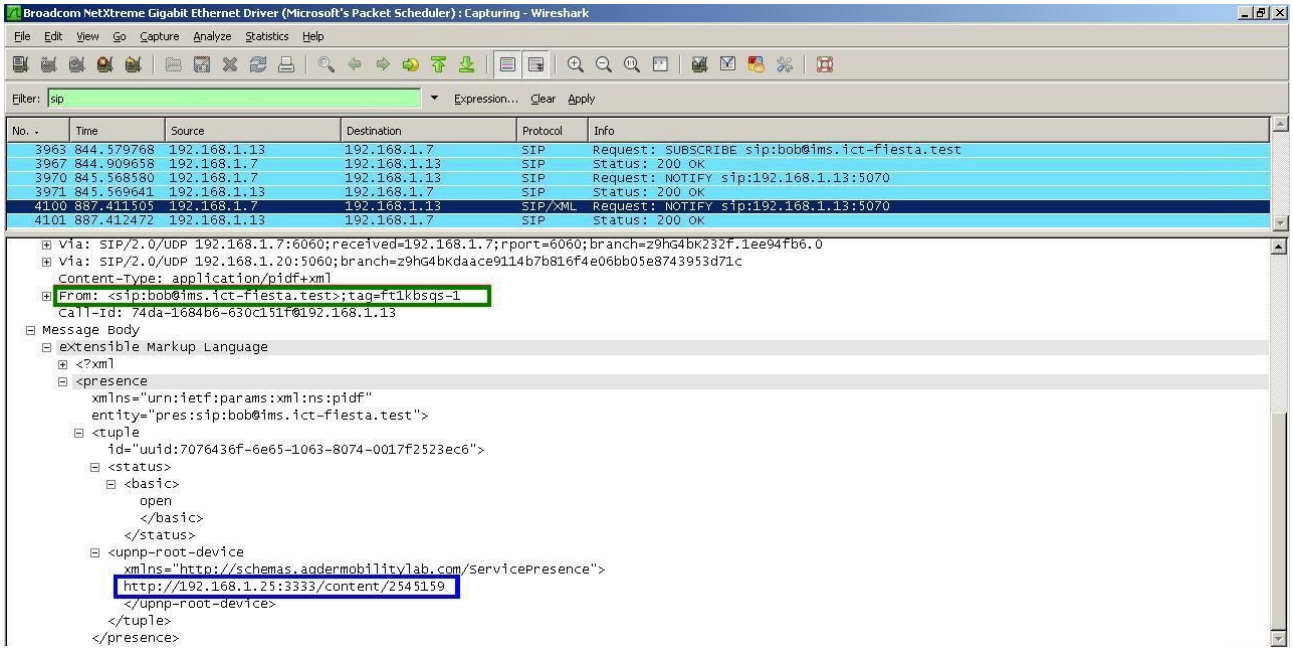

Figure 6.3: Message body verification for prototype implementation

What was not developed was functionalities like CRP and EP, implying that the prototype implementation cannot completely verify that a virtualized can be controlled and evented into a local network.

Chapter 7

Discussion

In the previous chapter, some results were obtained from the realization of a set of use cases, a solution design, and a prototype implementation, all of them concerning the Service Virtualization concept. This chapter starts with a brief overview of other options that might have been adopted in section 7.1. Then, there is an analysis concerning the steps carried out for the designed solution architecture in section 7.2. Finally, decisions adopted for the implementation issue have been discussed in section 7.3, as well as the problems encountered during the prototype testing in section 7.4.

7.1 Realization of use cases

For the realization of use cases, scenarios have been taken into account from a high-level view. On the one hand, a more detailed logic diagram depicting the framework where each scenario is involved might have been produced. For instance, functionalities that comprise the gateway located between each local network and the wired network could have been included in the diagram, as well as the service discovery mechanism used in each local network. However, the aim of the logic diagrams provided in section 3.2, section 3.3, and section 3.4, is to show the concept of SV in the context of residential networks, showing that SV can be enabled inside any LAN serving any service discovery standard.

On the other hand, a similar case with the flow diagrams might have been adopted as well. These diagrams might have shown more technical information in every step, such as signalling messages, or protocols for each flow. However, the aim of the flow diagram is to show the ease of use that SV provides consumers in order to access remote devices, hence letting end-users to consume remote services through completing a few easy steps.

Therefore, this high-level realization of use cases through these types of diagrams entails the explanation of the advantages of SV compared to current remote access methods, reinforcing the advantages that SV offers for end-users in terms of using remote services from anywhere with great ease of use.

7.2 Solution design for Service Virtualization

In the solution design, several steps have been described in order to virtualize a remote device. However, one of these steps becomes very important as it concerns some advantages of Service Virtualizer.

As shown in Figure 4.7, a 'tunnel connection' is set up between both SDGs in order to carry out all eventing and controlling operations. This tunnel permits all the signalling concerning eventing and controlling within a local network to be relayed to another local network only using one connection. For instance, if in one local network there are five control points, and these control points want to access a remote device, only one tunnel connection for these five clients is enough, since the Service Virtualizer manages to fetch the local signals from the control point in order to send them through the tunnel afterwards.

In previous chapters, the necessity of decoupling the remote access functionality has been described. If the case where this functionality is not decoupled from control points is considered, then five tunnels would be necessary for the aforementioned five control points, causing possible traffic congestions over the core network.

For this project, this issue may not be so relevant. However, in the case where in a country several control points within each of the many local networks of this country want to access a remote device, the probability of getting connection traffic congestions will increase severely.

Thus, SV functionality not only uses this tunnel in order to complete virtualization processes, but also takes advantages of it in terms of reducing possible traffic congestions over a core network.

7.3 Considerations adopted for prototype implementation

Some functionalities of the implementation has not been developed, as well as some classes have not been used for the developed prototype. In addition, a couple of issues have been considered for the implementation prototype according to the project definition. As it was formulated, emulation technology was to be used as a first step and the implementation was to be developed according to the solution architecture. Nevertheless, real application was finally used and other devices were used for the implementation instead.

Thus, this section explains the undeveloped and unused parts of the prototype and explains why emulation technology was rejected and why a video camera and a PVR were not used.

7.3.1 Incomplete Service Virtualizer implementation

SVR includes three functionalities: RSV, CRP and EP. However, only a part of it was developed and implemented. RSV is the functionality that was developed, including all its classes except the 'ArrayList' one, which should store in a list all the remote devices to be virtualized.

The reason why the 'ArrayList' class, CRP and EP have not been developed is because of the lack of time for the complete realization of this project. This has several causes. First, time spent in vain on emulation technology; and second, time spent in solving problems not related to SV, such as incomplete methods inside HIGA and the transmission of large SIP messages over IMS.

7.3.2 Real application instead of emulation technology

Emulation technology includes building both the video camera and the PVR in the programming code domain as if they would be real UPnP devices for a control point. In addition, their services, actions, state variables, and even the transfer of the video stream were to be emulated too as if they would be handled by the SDP.

This emulation technology was supposed to be based on Cyberlink [23], a package that allows developers to build UPnP devices very quickly. Although a video camera was built using Cyberlink, HIGA was finally the chosen technology because of the facilities it provided, such as handling SIP, Piranha protocol, or SDGs; in order to use SV as a real application, as well as the fact of having the challenge to show end-users what SV is capable of.

Real UPnP devices, including their services, actions and state variables can be handled with HIGA. Like this, the chance of using real application increased quickly, so that the idea of using emulation technology was skipped. Afterwards, a real media server and a real media renderer were used with UPnP serving them, showing that end-users can have good experiences with SV.

7.3.3 Devices used for the implementation

Although a video camera and a PVR were not used, the aim of the implementation prototype is to verify that SV is feasible with real devices, no matter which ones. In this way, since a solution architecture was designed for SV functionality using a video camera and a PVR, other devices were finally chosen in order to check that SV can be used for numerous devices: a video camera and a PVR as the solution design shows, a media server and a media renderer as the prototype implementation shows, and other personal devices as shown in chapter 3.

7.4 The prototype implementation process

Part of the prototype implementation verified that control points are aware of the presence of remote devices as if they would be local devices. This awareness involves information about the device description, such as manufacturer, friendly name, list of services, actions, or state variables. However, since HIGA is still in development, and since the 'Service Virtualizer' prototype was implemented on the top of the HIGA stack architecture, some unforeseen problems have been encountered for the device description transmission, making the implementation testing virtually impossible in the HIGA environment.

In this section, the problems encountered during the implementation testing are described, as well as the adopted ways to solve them. Bugs before the transmission of the content of device information between both local networks are discussed as a first step, followed by the bugs that occurred during the transmission of such an information.

7.4.1 Incomplete methods embedded in HIGA

The device description was partly transferred over SIP messaging due to the lack of HIGA in sending the whole description information. Since HIGA is not fully developed, some attributes regarding the description of the remote device could not be parsed, such as list of services, actions belonging to these services, and state variables related to these actions.

Such problems caused some time in adding some algorithms in order to parse the whole information properly. Once this parsing issue was solved, the prototype was tested afterwards,

encountering new problems concerning the transmission of such a description.

7.4.2 SIP package size

The device description size can sometimes be very large to transport over IMS, causing consequently some bugs inside the architecture of HIGA during the reception of such a content.

During the implementation testing, the transport of large description files caused some bugs known as 'overrun buffer', entailing illegible data regarding these descriptions at the receiver. This bugs occurred due to the length of the packet containing this description, around 20KB, relayed over a SIP message, while the maximum default length for the SIP packets is set to 8KB. There are some alternatives to solve this issue, such as packet compression, changing the default packet length in the whole framework, or transmitting this information over HTTP.

Although the latest alternative is the easiest, it does not really solve the root problem. However, since the two first ones would involve considerable development time, and since the purpose of the prototype implementation is just to verify SV issue, the HTTP solution was the chosen alternative. After patching this HIGA bug, RSV worked properly.

Chapter 8

Conclusion

The conclusions based on results and discussion in the previous chapters are given in section 8.1, followed by suggestions for further steps in section 8.2, to realize uncompleted work and introduce improvements regarding SV functionality.

8.1 Main conclusions

Users are requesting more and more that systems are capable of controlling their personal devices through service discovery standards when they are away from their homes. A remote functionality was embedded inside these mechanisms; however, the complexity of this embedding process entails the need of decoupling it from service discovery clients. In order to solve this decoupling issue, the concept of SV has been introduced in residential networks to discover and control remote devices.

The concept of SV has been introduced and discussed in great detailed, showing the benefits for end-users who want to access remote devices from anywhere with great ease of use. So, SV can definitely virtualize remote devices including their services, so that they appear as local devices.

As shown by the set of use cases, SV has advantages over current remote access mechanisms in terms of easy remote access, mobility and services. SV lets end-users enjoy services offered by remote devices from anywhere, without caring which service discovery standard and which operator are serving them. Furthermore, SV allows consumers not to carry their personal devices with them, since they can now access their remote devices.

The SV functionality design demonstrates that a remote PVR can be virtualized so that a user can record a video and stream it to the PVR as if it would be a local device. Moreover, SV reduces the chance of causing signalling traffic congestions, letting operators have better control and management of the core network.

As verified by the prototype, a remote media server has been virtualized so that a media

renderer recognizes it as local one. Thus, the prototype shows that SV can be used not only with particular devices, but with many devices that are nowadays filling up the homes, letting end-users use SV with whatever personal device they might have.

Consequently, with SV end-users shall be able to easily access and/or control their home appliances remotely like ovens, lamps, air-conditioned, washing machines, Pay-TV, media servers, personal video recorders, dishwashers, home security, etc. In addition to end-users, companies shall be able to manage remotely its development processes and devices, or do networked research in different cities.

Finally, since SV uses IMS as wired core network and it is decoupled from the operation of the service discovery mechanisms, operators will be able to use SV in the context of the third generation networks with any service discovery standard, offering a lot of new future services with a wide scope of added values for customers. Thus, there will be a lot of business cases for service providers and operators.

8.2 Further work

Two more functionalities must be developed in order to complete the Service Virtualizer software architecture. A Control Request Proxy for controlling actions invoked by control points must be realized so that SOAP protocol is handled between the home network and visited network. An Event Proxy also must be realized in order to enable GENA protocol handling between both local networks.

These two functionalities shall also be embedded in HIGA and developed based on Piranha protocol too. In addition, Event Proxy should be related to the UPnP classes concerning GENA; and Control Request Proxy should be related to classes concerning SOAP and UPnP control.

As an extended solution, State Cache could be embedded between the Remote Service Virtualizer and the layer composed of Control Request Proxy and Event Proxy. Since it is related to the service descriptions of the remote device, State Cache shall be based on Piranha and referred to Hosting functionality too.

References

[1] A. Häber, M. Gerdes, F. Reichert, R. Kumar, and A. Fasbender, *"Remote Service Usage through SIP with Multimedia Access as an Use Case,"* in *18th IEEE Annual International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC 2007)*, Athens, Greece, 2007.

[2] Armas, J.; Leon, C.; Miranda, G.; Segura, C. , *"Remote Service to Solve the Two-Dimensional Cutting Stock Problem: An Application to the Canary Islands Costume"* Complex, Intelligent and Software Intensive Systems, 2008. CISIS 2008. International Conference on Volume , Issue , 4-7 March 2008 Page(s):971 - 976

[3] Antonio Cuevas, Jose Ignacio Moreno, Pablo Vidales, Hans Einsiedler, *"The IMS Service Platform: A Solution for Next-Generation Network Operators to Be More than Bit Pipes", Communications Magazine, IEEE Volume 44, Issue 8, Aug. 2006 Page(s):75 - 81*

[4] Belimpasakis, P.; Stirbu, V.; *"Remote Access to Universal Plug and Play (UPnP) Devices Utilizing the Atom Publishing Protocol" , Networking and Services, 2007. ICNS. Third International Conference on 19-25 June 2007 Page(s):59 - 59*

[5] Andreas Häber, Martin Gerdes, Frank Reichert, Andreas Fasbender, Ram Kumar, *"Using SIP Presence for Remote Service Awareness "*

[6] The Free Dictionary by Farlex http://encyclopedia2.thefreedictionary.com/Service+discovery

[7] Zhu, F.; Mutka, M.W.; Ni, L.M.; *"Service Discovery in Pervasive Computing Environments"*; Pervasive Computing, IEEE Volume 4, Issue 4, Oct.-Dec. 2005 Page(s):81 - 90

[8] W. Adjie-Winoto et al., "*The Design and Implementation of an Intentional Naming System*," *Proc. 17th ACM Symp. Operating System Principles* (SOSP 99), ACM Press, 1999, pp. 186–201

[9] T. Hodes et al., *"An Architecture for Secure Wide-Area Service Discovery," ACM Wireless*

Networks J., vol. 8, nos. 2/3, 2002, pp. 213–230.

[10] M. Nidd, *"Service Discovery in DEAPspace," IEEE Personal Comm.*, Aug. 2001, pp. 39–45.

[11] *Jini Technology Core Platform Specification*, v. 2.0, Sun Microsystems, June 2003; www.sun.com/software/jini/specs/core2_0.pdf.

[12] *UPnP Device Architecture 1.0*, UPnP Forum, Dec. 2003; www.upnp.org/resources/documents/CleanUPnPDA10120031202s.pdf.

[13] S. Cheshire and M. Krochmal, "DNS-Based Service Discovery," IETF Internet draft, work in progress, June 2005.

[14] *Salutation Architecture Specification*, Salutation Consortium, 1999.

[15] E. Guttman et al., *Service Location Protocol*, v. 2, IETF RFC 2608, June 1999; www.ietf.org/rfc/rfc2608.txt.

[16] *Specification of the Bluetooth System*, Bluetooth SIG, Feb. 2003.

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

[18] Li Zhu *"Handling of IP-Address in the Context of Remote Access";* Master's thesis in Information- and Communication Technology, University of Adger http://student.grm.hia.no/master/ikt08/ikt590/g21/master%20thesis%20of%20Li%20Zhu.pdf

[19] Hechmi Khlifi, Jean-Charles Grégoire, *"IMS Application Servers: Roles, Requirements, and Implementation Technologies";* Internet Computing, IEEE Volume 12, Issue 3, May-June 2008 Page(s):40 - 51

[20] FTP http://en.wikipedia.org/wiki/Ftp

[21] VPN http://en.wikipedia.org/wiki/Virtual_private_network

[22] RDP http://en.wikipedia.org/wiki/Remote_Desktop_Protocol

[23] Cyberlink http://www.cybergarage.org/

[24] Michael Jeronimo and Jack Weast, *"UPnP Design by Example: A Software Developer's Guide to Universal Plug and Play"* ; Intel Press.

[25] Gonzalo Amarillo, Miguel A. García-Martín, *"The IP Multimedia Subsystem (IMS): Merging the Internet and the Cellular Worlds";* Wiley, Second Edition.

[26] Andreas Fasbender, Martin Gerdes, Johan Hjelm, Bo Kvarström, Justus Petersson, Robert Skog; *"Virtually at home: High-performance access to personal media"*; edited by: Ericsson *Ericsson Review*, No. 2. (2008), pp. 58-63.

[27] www.ericsson.com

Appendices

Appendix A: Glossary and Abbreviations

USB	Universal Serial Bus
IrDa	Infrared Data Association
IEEE	Institute of Electrical and Electronics Engineers
IMS	IP Multimedia Subsystem
SDP	Service Discovery Protocol
UPnP	Universal Plug and Play
TCP	Transmission Control Protocol
HTTP	HyperText Transfer Protocol
XML	Extensible Markup Language
IP	Internet Protocol
DHCP	Dynamic Host Configuration Protocol
SSDP	Simple Service Discovery Protocol
SOAP	Simple Object Access Protocol
GENA	General Event Notification Architecture
HTML	HyperText Markup Language
URI	Uniform Resource Identifier
SIP	Session Initiation Protocol
RTP	Real-time Transport Protocol
PSTN	Public Switched Telephone Network

CSCF	Call Session Control Function
P-CSCF	Proxy-CSCF
I-CSCF	Interrogating-CSCF
S-CSCF	Serving-CSCF
HSS	Home Subscriber Server
SLF	Subscriber Location Function
MRF	Media Resource Function
BGCF	Breakout Gateway Control Function
MGCF	Media Gateway Control Function
AS	Application Server
OSA	Open Service Access
CAMEL	Customized Applications for Mobile network Enhanced Logic
GSM	Global System for Mobile communications
SDG	Service Discovery Gateway
FTP	File Transfer Protocol
VPN	Virtual Private Network
RDP	Remote Desktop Protocol
SVR	Service Virtualizer
RSV	Remote Service Virtualizer
CRP	Control Request Proxy
EP	Event Proxy
SV	Service Virtualization

Appendix B: Prototype implementation

RemoteServiceVirtualizer.java

```
################################################################################
package no.uia.no.xiga.net.virtualization;

import org.apache.log4j.Logger;

import no.uia.one.xiga.net.hosting.UPnPAction;
import no.uia.one.xiga.net.hosting.UPnPActionArgument;
import no.uia.one.xiga.net.hosting.UPnPHosting;
import no.uia.one.xiga.net.hosting.UPnPHostingDeviceBase;
import no.uia.one.xiga.net.hosting.UPnPHostingServiceBase;
import no.uia.one.xiga.net.hosting.UPnPStateVariable;
import no.uia.one.xiga.net.piranha.Piranha;
import no.uia.one.xiga.net.piranha.PiranhaDevice;
import no.uia.one.xiga.net.piranha.PiranhaListener;
import no.uia.one.xiga.net.piranha.ServiceDiscoveryGateway;

public class RemoteServiceVirtualizer implements PiranhaListener {
   private static Logger LOG = Logger.getLogger(RemoteServiceVirtualizer.class);

   private UPnPHosting hostingPiranhaDevice = null;
   private Piranha pir = null;
   private UPnPHostingDeviceBase hpdev = null;
   private PiranhaDevice pdev = null;


   public RemoteServiceVirtualizer(Piranha p, UPnPHosting hosting)
   {
      LOG.debug("Remote Service Virtualizer constructor");

      pir = p;
      pir.addPiranhaEventListener(this);
      hostingPiranhaDevice = hosting;
      LOG.debug("Remote Service Virtualizer constructor passed");
   }

   /*
    * (non-Javadoc)
    * @see
no.uia.one.xiga.net.piranha.PiranhaListener#processPiranhaSdgDeviceAdded(no.uia.one.xiga.net.
piranha.ServiceDiscoveryGateway, no.uia.one.xiga.net.piranha.PiranhaDevice)
    * Devices added in the external SDG handled by piranha
    */


   public void processPiranhaSdgDeviceAdded(ServiceDiscoveryGateway sdg,
            PiranhaDevice dev)
   {
```

```
        LOG.debug(
            "Piranha device (" + dev.getFriendlyName() +
            ") added to SDG (" + sdg.getEntity());

        hostingPiranhaDevice.registerDevice(dev);
    }

    public void processPiranhaSdgDeviceRemoved(ServiceDiscoveryGateway sdg,
                PiranhaDevice dev)
    {
        // TODO Auto-generated method stub
    }

    public void processPiranhaSdgSubscribeError(String sdgAddress, long retryAfter)
    {
        // TODO Auto-generated method stub
    }

    public void processPiranhaSdgSubscribed(ServiceDiscoveryGateway sdg)
    {
        // TODO Auto-generated method stub
    }

    public void processPiranhaSdgUnsubscribeError(ServiceDiscoveryGateway sdg, long retryAfter)
    {
        // TODO Auto-generated method stub
    }

    public void processPiranhaSdgUnsubscribed(ServiceDiscoveryGateway sdg)
    {
        // TODO Auto-generated method stub
    }

}
################################################################################
```

RaHiga.java

```
################################################################################
Piranha piranha = xiga.getPiranha();
RemoteServiceVirtualizer rsv = new RemoteServiceVirtualizer(
            piranha,
            xiga.getHosting());
piranha.addPiranhaEventListener(rsv);

################################################################################
```

Appendix C: SIP messages

In this appendix all the SIP messages between both HIGAs has been monitored in order to verify that $HIGA_A$ has subscribed to $HIGA_B$ properly, as well as $HIGA_B$ has succesfully notified $HIGA_A$. These messages concerns those that were described in Figure 4.5, specifically steps {2-5}. All relevant information has been bolded, such as destination address, origin address, and methods.

(2) SUBSCRIBE----------

Session Initiation Protocol

Session Initiation Protocol

Request-Line: SUBSCRIBE sip:bob@ims.ict-fiesta.test SIP/2.0

Method: SUBSCRIBE

[Resent Packet: False]

Message Header

Via: SIP/2.0/UDP 192.168.1.13:5070;branch=z9hG4bK15050.4100.14c

Transport: UDP

Sent-by Address: 192.168.1.13

Sent-by port: 5070

Branch: z9hG4bK15050.4100.14c

To: <sip:bob@ims.ict-fiesta.test>

SIP to address: sip:bob@ims.ict-fiesta.test

From: <sip:alice@ims.ict-fiesta.test>;tag=20a6644-612.445d

SIP from address: sip:alice@ims.ict-fiesta.test

SIP tag: 20a6644-612.445d

Call-ID: 85e-4c1b6-263628f@192.168.1.13

CSeq: 13086 SUBSCRIBE

Sequence Number: 13086

Method: SUBSCRIBE

Max-Forwards: 70

Route: <sip:orig@scscf.ims.ict-fiesta.test:6060;lr>

Contact: <sip:192.168.1.13:5070>;+g.piranha.uia.no

Contact Binding: <sip:192.168.1.13:5070>;+g.piranha.uia.no

URI: <sip:192.168.1.13:5070>

SIP contact address: sip:192.168.1.13:5070

Content-Length: 0

[truncated] Proxy-Authorization: Digest username="alice@ims.ict-fiesta.test",realm="ims.ict-fiesta.test",nonce="fda7cd77f3a2105e626d6c8800ddfe98",uri="sip:bob@ims.ict-fiesta.test",response="c6b4be764f4f49ecf5ddef49064ddbf5",algorithm=MD5,c

Authentication Scheme: Digest

Username: "alice@ims.ict-fiesta.test"

Realm: "ims.ict-fiesta.test"

Nonce Value: "fda7cd77f3a2105e626d6c8800ddfe98"

Authentication URI: "sip:bob@ims.ict-fiesta.test"

Digest Authentication Response: "c6b4be764f4f49ecf5ddef49064ddbf5"

Algorithm: MD5

CNonce Value: "1fdae9"

QOP: auth-int

Nonce Count: 00000003

Expires: 3600

Event: piranha

P-Preferred-Identity: sip:alice@ims.ict-fiesta.test

Accept-Contact: *;+g.piranha.uia.no

---

(3) 200 OK----------

Session Initiation Protocol

Status-Line: SIP/2.0 200 OK

Status-Code: 200

[Resent Packet: False]

Message Header

Subscription-State: active

Allow-Events: piranha

Record-Route: <sip:mo@scscf.ims.ict-fiesta.test:6060;lr>

Record-Route: <sip:mo@pcscf.ims.ict-fiesta.test:4060;lr>

Event: piranha

Content-Length: 0

Expires: 3600

Contact: <sip:192.168.1.20:5060;fid=server_1>

Contact Binding: <sip:192.168.1.20:5060;fid=server_1>

URI: <sip:192.168.1.20:5060;fid=server_1>

SIP contact address: sip:192.168.1.20:5060

To: <sip:bob@ims.ict-fiesta.test>;tag=ft2x8due-1

SIP to address: sip:bob@ims.ict-fiesta.test

SIP tag: ft2x8due-1

Cseq: 13086 SUBSCRIBE

Sequence Number: 13086

Method: SUBSCRIBE

Server: Glassfish_SIP_1.0.0

Via: SIP/2.0/UDP 192.168.1.13:5070;rport=5070;branch=z9hG4bK15050.4100.14c

Transport: UDP

Sent-by Address: 192.168.1.13

Sent-by port: 5070

RPort: 5070

Branch: z9hG4bK15050.4100.14c

Call-Id: 85e-4c1b6-263628f@192.168.1.13

From: <sip:alice@ims.ict-fiesta.test>;tag=20a6644-612.445d

SIP from address: sip:alice@ims.ict-fiesta.test

SIP tag: 20a6644-612.445d

---

(4) NOTIFY----------

Session Initiation Protocol

Request-Line: NOTIFY sip:192.168.1.13:5070 SIP/2.0

Method: NOTIFY

[Resent Packet: False]

Message Header

Max-Forwards: 15

Subscription-State: active

Event: piranha

Content-Length: 0

To: <sip:alice@ims.ict-fiesta.test>;tag=20a6644-612.445d

SIP to address: sip:alice@ims.ict-fiesta.test

SIP tag: 20a6644-612.445d

Contact: <sip:192.168.1.20:5060;fid=server_1>

Contact Binding: <sip:192.168.1.20:5060;fid=server_1>

URI: <sip:192.168.1.20:5060;fid=server_1>

SIP contact address: sip:192.168.1.20:5060

Cseq: 2 NOTIFY

Sequence Number: 2

Method: NOTIFY

Via: SIP/2.0/UDP 192.168.1.7:4060;branch=z9hG4bKb3b8.8ebe44a5.0

Transport: UDP

Sent-by Address: 192.168.1.7

Sent-by port: 4060

Branch: z9hG4bKb3b8.8ebe44a5.0

Via: SIP/2.0/UDP 192.168.1.7:6060;received=192.168.1.7;rport=6060;branch=z9hG4bKb3b8.e963c432.0

Transport: UDP

Sent-by Address: 192.168.1.7

Sent-by port: 6060

Received: 192.168.1.7

RPort: 6060

Branch: z9hG4bKb3b8.e963c432.0

Via: SIP/2.0/UDP 192.168.1.20:5060;branch=z9hG4bKdaacb5c7417f386e4eb99ba9ed1f4c2740f2

Transport: UDP

Sent-by Address: 192.168.1.20

Sent-by port: 5060

Branch: z9hG4bKdaacb5c7417f386e4eb99ba9ed1f4c2740f2

Content-Type: application/pidf+xml

From: <sip:bob@ims.ict-fiesta.test>;tag=ft2x8due-1

SIP from address: sip:bob@ims.ict-fiesta.test

SIP tag: ft2x8due-1

Call-Id: 85e-4c1b6-263628f@192.168.1.13

---

(5) 200 OK----------

Session Initiation Protocol

Status-Line: SIP/2.0 200 OK

Status-Code: 200

[Resent Packet: False]

Message Header

Via: SIP/2.0/UDP 192.168.1.7:4060;branch=z9hG4bKb3b8.8ebe44a5.0

Transport: UDP

Sent-by Address: 192.168.1.7

Sent-by port: 4060

Branch: z9hG4bKb3b8.8ebe44a5.0

Via: SIP/2.0/UDP 192.168.1.7:6060;received=192.168.1.7;rport=6060;branch=z9hG4bKb3b8.e963c432.0

Transport: UDP

Sent-by Address: 192.168.1.7

Sent-by port: 6060

Received: 192.168.1.7

RPort: 6060

Branch: z9hG4bKb3b8.e963c432.0

Via: SIP/2.0/UDP 192.168.1.20:5060;branch=z9hG4bKdaacb5c7417f386e4eb99ba9ed1f4c2740f2

Transport: UDP

Sent-by Address: 192.168.1.20

Sent-by port: 5060

Branch: z9hG4bKdaacb5c7417f386e4eb99ba9ed1f4c2740f2

To: <sip:alice@ims.ict-fiesta.test>;tag=20a6644-612.445d

SIP to address: sip:alice@ims.ict-fiesta.test

SIP tag: 20a6644-612.445d

From: <sip:bob@ims.ict-fiesta.test>;tag=ft2x8due-1

SIP from address: sip:bob@ims.ict-fiesta.test

SIP tag: ft2x8due-1

Call-ID: 85e-4c1b6-263628f@192.168.1.13

CSeq: 2 NOTIFY

Sequence Number: 2

Method: NOTIFY

Contact: <sip:192.168.1.13:5070>

Contact Binding: <sip:192.168.1.13:5070>

URI: <sip:192.168.1.13:5070>

SIP contact address: sip:192.168.1.13:5070

Content-Length: 0

Allow: NOTIFY, SUBSCRIBE

Appendix D: SSDP messages

This appendix verifies that the prototype implementation is informing the devices in its vicinity about its presence. This messages concerns step {12} of Figure 4.5.

On the one hand, information regarding the location of the device description has been bolded. As it is shown, the IP address belongs to the internal address of $HIGA_A$ (10.1.0.49). This is logic as it has virtualized the remote device, this is, hosted the remote device into the local network.

On the other hand, information regarding what is being notified (NT) in $HIGA_A$'s neighborhood as also been bolded: (A) notification shows the 'id' number of the device, (B) the name of the device, (C) that there is only one device embedded in the UpnP Media Server, (D) shows that the media server contains a service called 'ConnectionManager', and (E) shows that the media server has another service called 'ContentDirectory'.

(A)----------

Hypertext Transfer Protocol

NOTIFY * HTTP/1.1\r\n

Request Method: NOTIFY

Request URI: *

Request Version: HTTP/1.1

HOST: 239.255.255.250:1900\r\n

CACHE-CONTROL: max-age = 35\r\n

LOCATION: http://10.1.0.49:3333/hosting/uuid_7076436f-6e65-1063-8074-0017f2523ec6/devicedescription.xml\r\n

NT: uuid:7076436f-6e65-1063-8074-0017f2523ec6\r\n

NTS: ssdp:alive\r\n

SERVER: MIDP/2.0 UPnP/1.0 ONEPP/1.2\r\n

USN: uuid:7076436f-6e65-1063-8074-0017f2523ec6\r\n

\r\n

(B)----------

Hypertext Transfer Protocol

NOTIFY * HTTP/1.1\r\n

Request Method: NOTIFY

Request URI: *

Request Version: HTTP/1.1

HOST: 239.255.255.250:1900\r\n

CACHE-CONTROL: max-age = 35\r\n

LOCATION: http://10.1.0.49:3333/hosting/uuid_7076436f-6e65-1063-8074-0017f2523ec6/devicedescription.xml\r\n

NT: urn:schemas-upnp-org:device:MediaServer:1\r\n

NTS: ssdp:alive\r\n

SERVER: MIDP/2.0 UPnP/1.0 ONEPP/1.2\r\n

USN: uuid:7076436f-6e65-1063-8074-0017f2523ec6::urn:schemas-upnp-org:device:MediaServer:1\r\n

\r\n

(C)----------

Hypertext Transfer Protocol

NOTIFY * HTTP/1.1\r\n

Request Method: NOTIFY

Request URI: *

Request Version: HTTP/1.1

HOST: 239.255.255.250:1900\r\n

CACHE-CONTROL: max-age = 35\r\n

LOCATION: http://10.1.0.49:3333/hosting/uuid_7076436f-6e65-1063-8074-0017f2523ec6/devicedescription.xml\r\n

NT: upnp:rootdevice\r\n

NTS: ssdp:alive\r\n

SERVER: MIDP/2.0 UPnP/1.0 ONEPP/1.2\r\n

USN: uuid:7076436f-6e65-1063-8074-0017f2523ec6::upnp:rootdevice\r\n

\r\n

(D)----------

Hypertext Transfer Protocol

NOTIFY * HTTP/1.1\r\n

Request Method: NOTIFY

Request URI: *

Request Version: HTTP/1.1

HOST: 239.255.255.250:1900\r\n

CACHE-CONTROL: max-age = 35\r\n

LOCATION: http://10.1.0.49:3333/hosting/uuid_7076436f-6e65-1063-8074-0017f2523ec6/devicedescription.xml\r\n

NT: urn:schemas-upnp-org:service:ConnectionManager:1\r\n

NTS: ssdp:alive\r\n

SERVER: MIDP/2.0 UPnP/1.0 ONEPP/1.2\r\n

USN: uuid:7076436f-6e65-1063-8074-0017f2523ec6::urn:schemas-upnp-org:service:ConnectionManager:1\r\n

\r\n

(E)----------

Hypertext Transfer Protocol

NOTIFY * HTTP/1.1\r\n

Request Method: NOTIFY

Request URI: *

Request Version: HTTP/1.1

HOST: 239.255.255.250:1900\r\n

CACHE-CONTROL: max-age = 35\r\n

LOCATION: http://10.1.0.49:3333/hosting/uuid_7076436f-6e65-1063-8074-0017f2523ec6/devicedescription.xml\r\n

NT: urn:schemas-upnp-org:service:ContentDirectory:1\r\n

NTS: ssdp:alive\r\n

SERVER: MIDP/2.0 UPnP/1.0 ONEPP/1.2\r\n

USN: uuid:7076436f-6e65-1063-8074-0017f2523ec6::urn:schemas-upnp-org:service:ContentDirectory:1\r\n

\r\n