# Selecting and Augmenting a FOSS Development and Deployment Environment for Personalized Video-Oriented Services in a Telco Context

Submitted in fulfilment
of the requirements of the degree of

Doctor of Philosophy

of Rhodes University

Zelalem Sintayehu Shibeshi

*Grahamstown, South Africa*
March 2016

**Abstract**

The great demand for video services on the Internet is one contributing factor that led telecom companies to search for solutions to deliver innovative video services, using the different access technologies managed by them and leveraging the capacity of enforcing Quality of Service (QoS). One part of the solution was an infrastructure that guarantees QoS for these services, in the form of the IP Multimedia Subsystem (IMS) framework. The IMS framework was developed for delivering innovative multimedia services, but IMS alone does not provide the required services. This has led to further work in the area of multimedia service architectures. One noteworthy architecture is IPTV. IPTV is more than what its name implies, as it allows the development of various innovative video-oriented services and not just tv.

When IPTV was introduced, many thought that it would bring back the revenue loss that telecom companies experienced to over-the-top (OTT) service providers. However, despite all its promises, the IPTV implementation has not shown as wide an uptake as one would expect. Although there could be various reasons for the slow penetration of IPTV, one reason could be the technical challenge that IPTV poses to service developers. One of the main reasons for the embarking of the research reported in this thesis was to identify and select free and open source software (FOSS) based platforms and augment them for easy development and deployment of video-oriented services.

The thesis motivated how the IPTV architecture, with some modification, can be a good architecture to develop innovative video-oriented services. For a better understanding and investigate the issues of video-oriented service development on different platforms, we followed an incremental and iterative prototyping method. As a result, various video-oriented services were first developed and implementation-related issues were analyzed. This has helped us to identify problems that service developers face, including the requirement to utilize a number of protocols to develop an IPTV-based video-oriented service and the lack of a platform that provides a consistent programming interface to implement them all. The process also helped us to identify new uses cases through the process.

As part of our selection process, we found that the Mobicents service development platform can be used as the basis for a good service development and deployment environment for video-oriented services. Mobicents is a Java-based service delivery platform for quick development, deployment and management of next generation network applications. Mobicents is a good choice because it provides a consistent programming interface and

supports the various protocols needed in a consistent manner or an easy way to include the support for them. We used Mobicents to compose the environment that developers can use to build video-oriented services. Specifically we developed components and service building blocks that service developer can use to develop various innovative video-oriented services.

During our research, we also identified various issues with regard to support from streaming servers in general and open source streaming servers in particular and also with the protocol they use. Specifically, we identified issues with Real Time Streaming Protocol (RTSP), a protocol specified as the media control protocol in the IPTV specification, and made proposals for solving them. We developed an RSTP proxy to augment the features lacking in the current streaming servers and implemented some of the features we proposed in it.

**Acknowledgements**

First and foremost I thank the Almighty God for helping me throughout my life. I am also deeply indebted to my supervisor, Prof. Alfredo Terzoli and my once co-supervisor Dr. Karen Bradshaw for inspiring me to do this research and for their timely and valuable advice throughout my study period. I am very grateful to you. In addition to his academic support, Alfredo has tirelessly worked to support me in many other aspects of my life, and I am very grateful. The thesis and my papers are witnesses of his contributions.

I would also like to thank the staff members and students of the Department of Computer Science at Rhodes University, especially Dr Mosiuoa Tsietsi and my colleagues in the Convergence Research Group for all the support I received throughout my study period. Dr Tsietsi spent many hours helping me setup the environment and compile my components.

This thesis is dedicated to my father, Mr Sintayehu Shibeshi, who always wanted to see me excel in my career and tried his best to provide me with all the support I ever needed, but passed away four years ago. Abaye, you have done all you can to put me where I am now. I am very grateful for the support you gave me all my life. I love you very much and will remember you always. May your soul rest in peace.

A very special thank you to my wife, and my best friend, Yanet Daniel, for being with me and supporting me in the most difficult part of this research. Her support and encouragement was in the end what gave me the endurance to finish this thesis. Thank you Yani. It is about time to celebrate! I also would like to thank Elilita Zelalem Sintayehu (Elu), and Haset Zelalem Sintayehu (Bibiyo), my beautiful angels who endured the missing of their dad at their young age. A big thank you is also due to my mother, Mulu Adane, and my brothers, Yiheyis, Abiy, Bemnet, Tibeb, Tamrat, Eyob, and Yeabsira, who always want me to succeed and excel. Thank you my family, God has heard your prayers.

I want to take this opportunity to thank my South African family, Prof. Philip Court and Mrs Kathryn Court, for their unlimited love and support throughout my study here in Grahamstown. The Courts, you are such a wonderful family that have a special place in my heart. Thank you and God bless you.

I also want to say thank you to all my close friends who have been encouraging me to move ahead in times of great challenge. Finally, a big thank you to my sponsors: Telkom SA, Comverse, Stortech, Tellabs, Easttel, Bright Ideas 39, THRIP and CTIT/ETC who generously provided the financial support that allowed me to complete this work.

# Selecting and Augmenting a FOSS Development and Deployment Environment for Personalized Video-Oriented Services in a Telco Context

Submitted in fulfilment
of the requirements of the degree of

Doctor of Philosophy

of Rhodes University

Zelalem Sintayehu Shibeshi

*Grahamstown, South Africa*
March 2016

# Contents

# List of Figures

# List of Tables

# List of Accronyms

3GPP 3rd Generation Partnership Project

AS Application Server

B2BUA Back-to-Back User Agent

BC Broadcasting

DVNS Dynamic Video Notification Server

ETSI European Telecommunication Standards Institute

FSSS Festival Speech Synthesis System

GOP Group of Pictures

GPRS General Packet Radio Service

GSM Global System for Mobile Communications

IETF Internet Engineering Task Force

ifc Initial Filter Criteria

IMPI IP Multimedia Private Identity

IMPU IP Multimedia Public Identity

IMS IP Multimedia Subsystem

ISC IMS Service Control

ITU International Telecommunications Union

JAIN Java API for Integrated Networks

JCA Java Connector Architecture

MCF Media Control Function

MDF Media Delivery Function

MRF Media Resource Function

N-PVR Network-based Personal Video Recorder

NGN Next Generation Network

OIPF Open IPTV Forum

PBX Private Branch Exchange

PC Parental Control

PCh Personalized Channel

PSIs Public Service Identities

QoS Quality of Service

RTCP Real Time Control Protocol

RTP The Real Time Protocol

RTSP Realtime Streaming Protocol

SBB Service Building Block

SCF Service Control Function

SDOs Standard Development Organizations

SDP Session Description Protocol

SIMPLE SIP for Instant Messaging and Presence Leveraging

SIP Session Initiation Protocol

SLEE Service Logic Execution Environment

SMTP Simple Mail Transfer Protocol

SPT Service Point Triggers

SSC Streaming Server Controller

STP Service trigger point

TISPAN Telecommunications and Internet converged Services and Protocols for Advanced Networks

UA User Agent

UE User Equipment

VoIP VoIPVoice over IP

XCAP XML Configuration Access Protocol

# Chapter 1

# Introduction

## 1.1 Developments in Telecommunications and Video-Oriented Services

Since the introduction of an electrical telecommunications system in the mid-18th century, the telecommunication world has evolved to the current stage through many innovations. Both the infrastructure and services have developed a lot. It has become easy now to connect with our family while we are on the move. Wireless communication has become the order of the day. We are at a time where video conferencing is not only a technology for big companies but is a service easily available for ordinary people.

The development of the telecommunication infrastructure has been mostly influenced by the need for the inclusion of video, which requires large bandwidth.

Since video is key for experience delivery, most services on the Internet, including the mushrooming social networks, are trying to include video. In fact, literally every service on the Internet seems to be moving towards including video. For example, a video-based online journal, called *Jove* (that publishes biological research in video format)[1] emerged to show where the web is going with respect to the use of video. The online data traffic has grown very much because of the inclusion of video. According to an article published on the BBC website [2], YouTube had reached 2 billion hits per day in 2010 and because of its huge user base, it became the second-largest search engine on the Internet in just six years after its establishment. This trend is expected to continue in the years to come. A recent study by Cisco estimated that 80% of Internet traffic will be video by 2019 and 2 Zettabytes of data would be transferred online [3]. As mentioned in [4], the growth of on-line video spending also surpassed $2.12 billion in 2008, up 36% from 2007, and

forecasted to continue double-digit increases through the years to come.

The rate of growth of video data is also expected to rise in mobile networks. Mobile video is projected to account for more than 75% of total mobile data traffic by 2020, compared to data, web browsing Mobile Voice over IP (VoIP) combined, which accounts for 17% of the traffic [5]. All these show the huge demand for video-oriented services. Figure 1.1 shows how video traffic increased over the last few decades in the global Internet traffic.

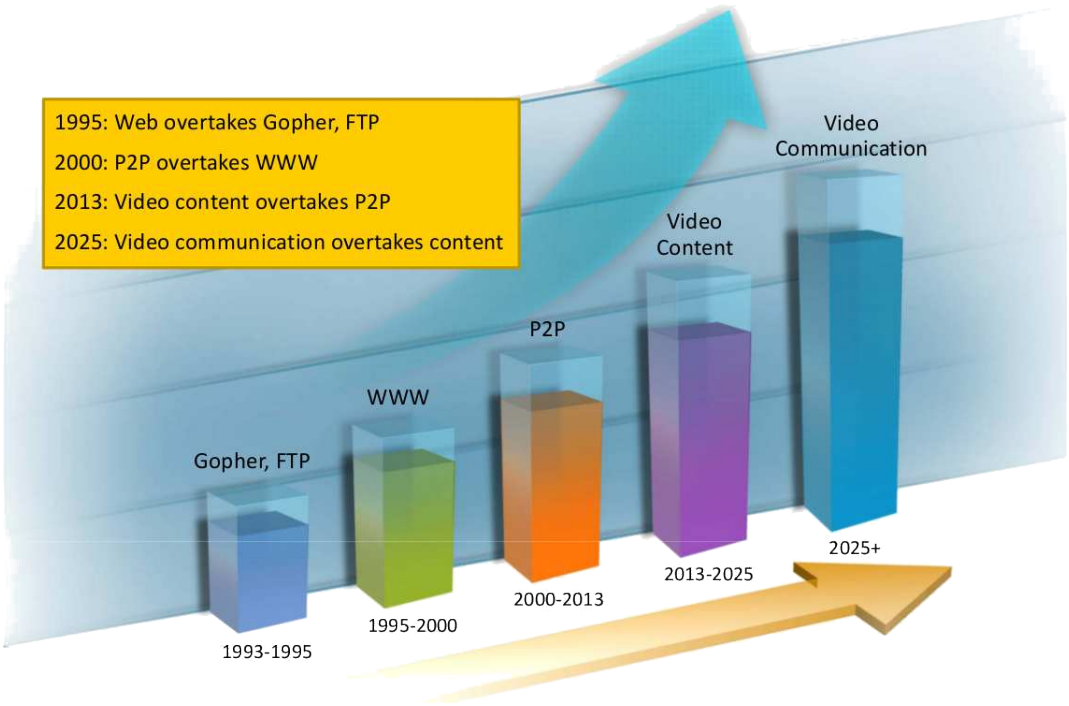



Figure 1.1: Role of video in global IP traffic. Source: [6]

In light of these developments, user devices have also developed drastically, getting ready to provide high quality video to users. However, traditional telecommunication companies could not tap this huge demand for video service adequately. Surprisingly, the basic video service, which is video call, is not widespread. One of the objectives of this thesis is to investigate why video-oriented services did not flourishing in the Telecom industry and provide a solution by identifying a proper infrastructure and developing a service development and deployment environment that will enhance the expansion of these services. The infrastructure should be a service deployment platform that allows users to access all their services using any of the available technologies.

There are many reasons that attributed to the migration of users to over-the-top service providers. One major issue that was a problem to the delivery of any service in the Telecommunication (abbreviated in this thesis as Telecom) industry relate to how Telecom

services are accessed by users.

From the beginning, services in the telecommunication world were tied to a specific network and customers were not able to access their service using a different access technology than the one they registered to. As part of finding solution to this problem, Telcos[1] came up with an infrastructure called the Next Generation Network (NGN) where users can access their service using any of the access mechanisms they deem necessary. The interesting aspect of this solution is that, NGN was not only developed to solve the access problem, but also to facilitate the delivery of multimedia services in the Telecom industry. As a result, on top of NGN, Telcos also introduced other architectures that facilitate the development and delivery of video-oriented services. However, the solutions still could not attract service developers and video-oriented services are not widespread in the Telecom world. This thesis tries to investigate the issues related to the development of video-oriented services in the Telco context and provides solution as a form of service development and delivery environment. The following section briefly introduces the technologies introduced by Telcos for the delivery of video-oriented services in the Telcom world.

## 1.2 Preliminary Solution for the Support for Multimedia Video-Oriented Services in the Telecom World

As mentioned above Telcos took various steps to solve the problem of the traditional telecommunication infrastructure in order to bring back the revenue loss they experienced by supporting the delivery of video-oriented services in a better way than the over-the-top service providers. These steps include the introduction of NGN and other architectures that facilitate multimedia service development and provisioning. The following sub-sections briefly introduce these technologies.

### 1.2.1 The Next Generation Network (NGN)

As mentioned above, telecommunication services were developed in a *stove pipe* fashion, specifically attached with a specific access network. A user who has a voice mail service from his[2] mobile network provider could only access his voicemail using his cell phone.

[1] Telco refers to Telecommunication Companies in this thesis

[2] he is used to refer to users and represents both genders in this thesis

The Internet, however, had shown us long ago how one can access the same service using different technologies. A user can access his email using his Personal Digital Assistant (PDA), mobile phone, or a personal computer. As a result, as the demand from users to get a converged service became increasingly high, Telcos also followed the trend and started to move towards an environment that supports convergence. This demand for a converged service is the major reason for the introduction of NGN, an infrastructure geared towards bringing the different telecommunication networks together for the provisioning of telecommunication services including multimedia services.

The main difference between traditional telecommunications services and services in NGN is the shift from separate vertically integrated application-specific networks to a single network capable of carrying all services, including voice, data and video [7]. The interesting part of NGN is that, unlike the Internet, NGN service delivery is beyond *best effort* and ensures the Quality of Service (QoS) required by different services and here lies the advantage for Telcos to deliver video-related services.

The International Telecommunications Union (ITU), the United Nations specialized agency responsible for information and communication technologies, has defined the NGN as *"a packet-based network able to provide telecommunication services to users and able to make use of multi-broadband, QoS-enabled transport technologies and in which service related functions are independent from underlying transport-related technologies. NGN offers unfettered access by users to different service providers and it supports generalized mobility which will allow consistent and ubiquitous provision of services to users"* [8]. As its main objective is building a global consensus on new technologies, the ITU through its standardization sector, the ITU-T, has released the first specification of the NGN (Release 1) at the beginning of 2006. Since then the ITU has released different specifications to include new services and features.

Although there are many competing visions for how service delivery can be achieved in NGN, there is one framework that has considerable momentum, which is the IP Multimedia Subsystem (IMS) [9]. The following section presents a brief discussion of IMS, an infrastructure that creates a platform to deliver multimedia services by allowing users to get their service using any of the access technologies in the Telecom environment. We chose IMS as a service deployment platform and it is central in our own work.

### 1.2.2 The IP Multimedia Subsystem (IMS)

The realization of IMS as a service deployment platform for NGN is discussed by many authors [10, 11, 12]. Specifically, Camarillo and Garcia-Martin [13] discussed how IMS,

specially the one defined by Telecommunications and Internet converged Services and Protocols for Advanced Networks (TISPAN), is applicable to NGN.

IMS was designed to fill the gap between the existing traditional telecommunications technology and Internet technology, because increased bandwidth alone could not fulfill the promises of NGN. IMS enables new services and applications by providing an infrastructure that gives more robust control and management than the Internet does today [14]. As technology is evolving and more and more IP-enabled devices emerge, the IMS framework found to be the way forward. However, it had problem with regard to support for some of the popular video-oriented services.

The standard IMS architecture through its component called MRF (Media Resource Function) was able to provide video-oriented multimedia services like video announcement and recording services. However, it was not able to provide the most popular services like Streaming and Broadcasting (BC) services. So, Telcos started to think how they could seize the opportunity currently available and decided to merge the traditional tv technology with NGN. This is what different stakeholders have also been suggesting all along. For example, Cisco Systems [6] mentioned that Telcos need to evolve into being "experience providers". A report on "The Mobile TV Market" from the ABI Research Group also revealed that the mobile TV market has tremendous long-term promise as a next-generation infotainment experience [15]. Because of this, TISPAN developed another specification different from the standard IMS specification with an architecture that can be used to provide various types of video services, like BC and streaming services, and they called it the IMS-based IPTV architecture. We also base our solution on IPTV. The following section presents how IPTV support the delivery of video-oriented services in NGN/IMS.

### 1.2.3 IPTV as a Video Service Development and Delivery platform for NGN/IMS

As mentioned above, video services are becoming very popular on the Internet. At the same time, users are also changing their behavior moving beyond viewing short, low-quality clips of user-generated content on YouTube and increasingly seeking TV shows, films, and other professionally created, high-quality video content. It is obvious that because of the inherent characteristics of the Internet, quality of service cannot be guaranteed by Internet based video-oriented services. This brings the opportunity for Telcos to engage themselves in the delivery of video-oriented services. It is a critical time and Telcos has to seize the opportunities.

IPTV differs from typical NGN-based voice and data services by the fact that it combines three conceptually unfamiliar (until now) components: streamed video, broadcasting services, and personalized video recorder. In fact, IPTV represents a broad term used to reference the delivery of a wide variety of video content using the IP platform as a mechanism for transporting content.

The services that IPTV provides could not have been considered with the previous technologies of broadcast TV, especially personalization and interactivity. IPTV can also be used to provide converged services. For example, a user who is watching an advert on his TV can now check out the item directly from his TV. As mentioned in [16], IPTV is one of the key services for future media distribution. All these claims about IPTV are made because IPTV is a very open architecture and helps service developers to come up with innovative services. Because of this, we also considered this architecture as a base architecture for the delivery of personalized video-oriented service and show how this can be done in this thesis. The architecture of IPTV as a proper architecture for video service delivery, including the different services, is discussed in detail in Chapter 4.

But despite all its promises, the IPTV deployments have not shown as wide an uptake as would have been expected. Although there could be various reasons for the slow penetration of IPTV, one reason is the technical challenge that IPTV poses for service developers.

IPTV runs on top of IMS and even though the IMS environment is good for the delivery of multimedia services, its architecture, which is explained in detail in Chapter 4, is claimed to be complex. As explained in [17], IMS is complex from the service developer point of view because it has so many components and interfaces that utilize various protocols, which service developers have to implement.

The complexity of IMS increase even more when it comes to using it to deliver video service like IPTV, because video service development and delivery by itself brings its own complexity. The following section presents a brief overview of the challenges of video-oriented service development and delivery in NGN/IMS that we want to provide solution in this thesis.

## 1.3 Challenges of Video Service Development and Delivery in NGN/IMS

If a service developer wants to develop a service for the IMS environment he is required to implement an Application Server (AS) that will contain the service logic, which, in the

case of IPTV, is called the Service Control Function (SCF).

In NGN/IMS, SIP is the signaling protocol used for session setup and management and ASs are implemented as a SIP AS. If the IPTV AS had to utilize only the SIP protocol, it would have been easier to develop it as a standalone AS based on one of the available open source SIP stacks. However, the IPTV specification defines several components that need to be connected to the AS with different interfaces in order for the AS to provide the required services. This means the different interfaces need to be implemented to provide a basic IPTV service and the most challenging part of all is that each of these interfaces use different protocols. As a result, to create a full-fledged IPTV service, one needs to create clients for these protocols, some of which can be found as standalone services or enablers, and embed them in the AS. The XCAP and Diameter protocols are the two most important protocols, in addition to SIP, that the IPTV AS has to utilize. This also means that service developers require a service development platform that supports most of the protocols together with enablers. This is one of the challenges for IPTV service developers.

The most challenging part is that there is no fully-fledged open source testbed that contains all the standard components specified in the architecture of IPTV, in which service developers can put their AS and test a new service logic. As a result, service developers have no choice but to develop these components as well, far from an ideal situation. In fact, we also identified missing components from the standard IPTV architecture specified by TISPAN that are important to develop innovative video-oriented services. In addition to this, there are also implementation challenges for some of the components.

One of the components defined specifically for the control of media in IPTV is the Media Control Function (MCF). The MCF is responsible for controlling the media delivery unit (media server), also called Media Delivery Function (MDF). The MDF is the component that provides content to the user equipment (UE). As a result, in addition to the AS, one has to implement the MCF in order to implement an IPTV service. In fact, the specification does not clearly specify what type of protocol the MCF should use to communicate with the MDF. However, for streaming related services, the interface between these two units seems to be the Realtime Streaming Protocol (RTSP). Nevertheless, as explained in the thesis, RTSP in general, and open source streaming servers in particular, have issues when it comes to meeting the requirements of IPTV service delivery.

The challenge with regards to streaming service development is that open source streaming servers (which are used as the IPTV media delivery function by the open source research community) do not behave and work as required by the IPTV specification. As mentioned above, MCF (and not the UE) is responsible for initiating and controlling media, but most

open source streaming servers do not allow media request to come from a different source other than the client that consumes it. In other words, they expect the media to be delivered to the same place where the request comes from. So this is one problem that service developers have to deal with if they want to use open source streaming servers to serve as MDF. In fact, we have also identified various issues with the RTSP protocol. The thesis discusses these issues and presents the solutions we came up with.

One of the most interesting aspects of IPTV is the delivery of personalized video services, which depends on user profiles. Based on our assessment, the user preference schema provided by the IPTV specification lacks some attributes of the user profile that ASs may require in order to provide some of the innovative services. We included these attributes into the user profile document and also developed an open source component to manage the user profile.

As mentioned above, an IPTV service development environment has to support a number of protocols. For example, a basic IPTV service may require at least six protocols: SIP, SDP, Diameter, XCAP, DNS, HTTP and one media control protocol (it could be RTSP). The more advanced an IPTV service is, the more complex it becomes in relation to the use of protocols. The service provisioning also involves more and more parties – Telcos, content/service providers, third party networks and service providers, and even end-users – increasing the complexity of the environment in which services must live. This means the development environment needs to support at least the major protocols, and as mentioned above building an environment from scratch is not a practical approach. As a result, as part of our selection process, we investigated candidate platforms that can be used as a basis to develop the envisaged environment. We present our analysis of the available open source service development platforms in this thesis.

We believe that the demand for video-oriented services can be met by providing a service development environment with basic components and reusable modules that service developers can utilize to develop different services.

Because the environment needs to be extensible and should also enable developers to develop converged services, we tried to first develop an exhaustive requirements list that the environment should meet. In general, this thesis presents our effort to design and develop an environment that service developers can use to easily implement innovative video-oriented services.

## 1.4 Thesis Objectives

The major objective of this thesis is to investigate a proper video service development architecture and select and augment an environment that service developers can use to easily implement video-oriented services for NGN/IMS. This includes identifying the problems and challenges of video service development activity by looking at implementation issues of the various video services expected to be delivered in such an environment. Basically, the research aims to select appropriate service development environment and augment it by developing missing components that will help video-oriented service developers, reducing the programming effort, time to develop applications, and the cost of application development.

The objectives of this thesis can be broken down into the following goals:

- Identify issues of video-oriented service development and delivery in NGN/IMS
- Critical review of available video-oriented service development and delivery infrastructure and technologies in the Telco context
- Develop a comprehensive video-oriented service requirements
- Identify high-level video service functionalities and components that service developers can use
- Analyze and present design and implementation decisions
- Select an architecture that would support all the different video-oriented service scenarios
- Select a service development platform that supports most of the protocols involved in the provisioning of video-oriented services, and augment it by building core components and various databases
- Do an initial validation of the architecture through implementation of important functions of innovative video-oriented services
- Provide recommendations for protocol and standard amendments

## 1.5 Methodology

This section describes the methodology followed when pursuing the aims and objectives of the research described above.

### 1.5.1 Literature Review

To get an understanding of video-oriented services in the Telecom environment and do critical review of IMS and IPTV, relevant published specifications, standard documents, materials on the Internet and journal articles were reviewed. Literature was also reviewed in order to establish the background and current development situation of IMS and video service development frameworks and how they can be extended to allow an IPTV service implementation.

### 1.5.2 Incremental and Iterative Prototyping

Within the software development scene, prototyping is a common software developmental methodology. As mentioned in [18], the rapid prototyping process belongs to the generative (or additive) production process. The developer develops a small part of the software and then after getting feedback, he modifies it to create a bigger part (or component).

There are various types of prototyping techniques, but incremental and iterative prototyping are the most common ones. The authors in [19], mention that evolutionary iterative development implies that the requirements, plan, estimates, and solution evolve or are refined over the course of the iterations, rather than fully defined and *frozen* in a major up-front specification effort before the development iterations begin. The basic principles in this type of prototyping are refining requirements for the building of a more robust system through various iterations.

We normally use prototypes when the requirements are unclear and changing. In the case of this research, we have theoretical specifications for services explained in TISPAN specification, and we need to find out whether they are doable with the existing technology and recommended protocols as specified in the specification. Iterative prototyping help us to achieve this.

In fact, most of the time, specifications do not dictate implementation options and in the same manner the IPTV specification also presents high-level requirements and does not include implementation aspects. As a result, various implementation options need to be investigated. Because most of the technologies (protocols) proposed for use for IPTV were developed by considering a basic client server model, they need to be checked to see if they work with the case at hand (the IMS/IPTV architecture). Figure 1.2 summarizes the methodology followed in this thesis.

As can be seen from Figure 1.2, the research started with ideas (various specifications, practical experiences, other recommendations) and based on that we developed a pro-

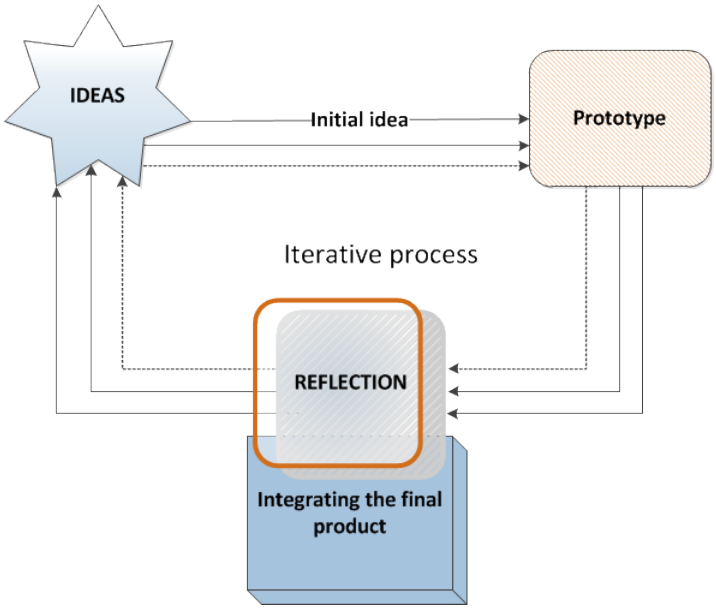



Figure 1.2: Research process followed

totype to see if the system fulfilled the promised requirements. When the developed prototype lacked some features or needed modification, based on the assessment, new ideas were gathered and the prototype was refined. This was done iteratively.

A good example for this is our development effort for MDF using open source streaming servers. Issues related to the *destination* parameter and *end of stream* signal became apparent when making an exhaustive test of the developed system. Open source streaming servers that do not support the *destination* parameter seemed to work fine when the the MCF and the UE were running on the same machine, but when an exhaustive testing was done, issues surfaced and solutions had to be found.

In general, to get a better understanding and investigate the issues of video-oriented service development on different platforms, various video-oriented services were first developed and implementation-related issues were reported. Based on the identified problems, solutions were also proposed.

We also reviewed best practices from other service development platforms and included these ideas when designing the architecture to come up with a robust environment.

In general, the following was done to achieve the objectives of the thesis:

- Review of various specifications and available multimedia frameworks
- Development of various services to identify particular issues of video-oriented service development technologies

- Identification and specification of current and future requirements
- Analysis of best practices of service development platforms
- Designing an extensible system architecture
- Identification of software code to be reused

Once these factors were determined, a service development platform with a proper programming model was selected and the necessary components of the extensible video-oriented architecture were implemented and integrated to create the service development and deployment platform.

As we used different frameworks to develop the different services, we have made extensive use of codes from various open source research communities.

## 1.6 Scope

Video-oriented service development is a large area and we only focused on a limited part of it. Specifically we considered streaming related services together with the user profile management aspects of IPTV and ignored issues like charging and accounting. The research also did not consider services related to broadcasting (BC). The IMS is a converged infrastructure that integrates the Internet. However, non-IMS compliant web-based video services, are not addressed in this research.

On the other hand, from a service development and deployment perspective, what makes the NGN an interesting platform is the fact that it allows developers to focus on the development of the service without worrying too much about the transport infrastructure used to deliver the service or how users will access the service. This relates to one of the most important principles of Computer Science - the separation of concern. As a result, the thesis also will not discuss issues related to the transport layer of IMS/NGN.

## 1.7 Contributions

The major contribution of this thesis is the critical review of issues surrounding the development of video-oriented services in the Telco context and selecting and augmenting the extended IPTV architecture and environment for easy development and deployment of capability-based video-oriented services. In this regard, the thesis exposed many issues

that need to be tackled and also identified new use cases that the environment must support if it need to be used to develop innovative video-oriented services. The requirements were compiled and the environment designed and implemented.

One contribution that relates to the design and implementation of the environment is the decision of how the environment should be implemented, which includes the choice of appropriate design patterns and programming style for the implementation of the environment. With regard to the actual implementation environment, we selected a development platform, called Mobicents, which was found to be a very appropriate platform to base the implementation of the environment. Mobicents is appropriate because it allows easy extension of the environment in all aspects. For example, new protocol support can easily be integrated in Mobicents. In fact, Mobicents, by and large, is a good implementation of IMS in the context of IPTV.

For the purpose of easing video service development, we identified high-level video service functionalities and based on that we developed components that can be used for developing video-oriented services. This is another major achievement of this research. We based our solution using the IPTV architecture and incorporated the new components that we identified into it to come up with the modified IPTV architecture. Various communication messages (events) that need to be used between the different components were identified and implementation strategies presented. In this regard, the thesis identified new interfaces for the AS, other than the ones specified in the standard documents.

The thesis focused on all aspects of streaming services and presented issues related to the development and deployment of these services and solutions to the various problems and challenges identified in the research. One major hindrance to the development of IPTV-based streaming services, for example, is lack of an open source media delivery function (streaming server) that works in accordance with the specifications of IPTV. For this purpose, an extensible and easy to use RTSP Proxy, that can be used with the available open source streaming servers, was developed and integrated into our testbed. The Proxy exposes various functionalities that one can use to implement new features of advanced streaming services in IPTV. In fact, from the point of view of delivering new streaming service functionalities and implementing new commands, we defined a new way of representing streaming sessions and demonstrated its implementation in the Proxy. In general, new functions were defined and we explained how they are integrated in the Proxy. Some of the new functions were considered as enhancement areas for the RTSP protocol itself for the purpose of supporting innovative video-oriented services.

There are also contributions related to the user profile data structure which is used for the development of personalized services. We have designed database schema that are

needed to deliver services like *Recommendation* and *Notification*. We also extended the data structure defined by the TISPAN so that it supports the new use cases described in this thesis.

Finally, video-oriented service development issues that need further investigation were also identified.

The outputs of this research have been presented at various international conferences and published in journals. Appendix A presents the list of publications which originated from this research.

## 1.8 Thesis Organization

The same methodology as the one we used to develop the environment is reflected in the style of writing of the thesis, where we narrate the different experiments from the conception to execution and results. We feel that this style of writing would better convey our work and the results we found. In general, the thesis is organized into nine chapters, as follows:

**Chapter** 2: This chapter gives background information and critical analysis on digital video and protocols for video-oriented service development. The chapter provides a brief discussion about the fundamentals of digital video, including video codecs and compression, multimedia and container formats. Discussion is also made about the characteristics of video-oriented services in general and network-based streaming services in particular. In this regard, the other major part of the chapter is discussion about the various protocols used to deliver video-oriented services over the network.

**Chapter** 3: This chapter presents part of our experimental work related to the development of video-oriented services that we did to better understand the issues related to video service development and delivery in different service development platforms. In this chapter the work we have done related to the development of basic video services and streaming services is presented. The chapter explains how these experiments provided major inputs for the selection and development of the envisaged service development and delivery environment that is described in Chapters 6 and 7.

**Chapter** 4: This chapter is dedicated to providing background information and a critical analysis of the main architectures used as basis for the envisaged environment. More

specifically discussion was made on the architecture and components of IMS and IPTV and also how services are deployed in IMS. Since personalization is what differentiates IMS from traditional telecommunications service delivery environments, discussion of how the IPTV architecture can be used for delivering personalized and interactive video-oriented services is the other discussion point of this chapter.

**Chapter** 5: This chapter continues the presentation of our experimental work towards the development of video-oriented services to better understand the available service development environments. The chapter specifically presents the work done in relation to IPTV services. Discussion of user profiles and the event management aspects of IPTV services are some of the focus areas of the chapter. IPTV service implementation issues and a discussion to show how the Proxy solves the problems of open source streaming servers was also made in this chapter.

**Chapter** 6: This chapter summarizes the findings of our experimental work, by presenting the summary of the requirements of the environment. The requirements include those specified in the standard documents but not detail implementation aspect given and the new requirements that we found through our experiments and investigations. Based on the requirements, the chapter also presents the design of the envisaged environment by presenting the modified architecture of IPTV that we came up in this research. The design considerations that we used to develop the architecture of our environment are also included in the discussion.

**Chapter** 7: This chapter presents the process we followed for selecting and augmenting the service development and delivery environment. We discuss how we implemented missing components. We based our implementation on Mobicents and the chapter shows how Mobicents is a good candidate to base our implementation of the environment on. In general, implementation considerations and prototypical implementation of the major components are the main focus of this chapter. Issues related to implementation are also discussed in this chapter.

**Chapter** 8: This chapter presents the results of our testing of the environment by showing how the environment supports the development of innovative services and also by showing the interaction of the different components of the environment with other units like the IMS core and streaming server that are necessary to provide a service to the user.

**Chapter** 9: The conclusions drawn from this research and the recommendations are presented in this chapter. The chapter summarizes and presents the major contri-

butions of the thesis and also proposes the work that needs to be carried out in the future.

# Chapter 2

# Background on Digital Video, Protocols Used and Issues with Video-Oriented Service Development

In order to understand video services, it is important to understand the characteristics and features of video. In this chapter we present background information about digital video and video-oriented services. The delivery of video services and especially network-based video-oriented services involve several protocols and this is another discussion point of the chapter. We also discuss the issues we found with the available protocols and present proposals for solutions to the problems. We start the chapter with the discussion on digital video.

## 2.1 Creation, Storage and Transmission of Digital Video

Because of bandwidth limitations and storage capacity constraints, video has to be compressed before it can be transmitted to meet the real-time nature of communication in the limited bandwidth. Video is not always created for streaming over a network. Sometimes it needs to be stored in a file. Because a video stream or file contains different types of media elements compressed differently but somehow related, we need special form of file storage. In this section we discuss these aspects of video in brief.

### 2.1.1 Video Compression

Digital video is a representation of real-world scenes sampled spatially and temporally. If we observe successive pictures of a video carefully, we see spatial and temporal redundancies. This is the motivation for video compression.

There are different types of compression techniques used by different codecs. Recent compression techniques use an approach called the *Group of Pictures (GOP)*, where a compressed video can be considered as a group (also called Group of Pictures). GOP codecs compress a group of images together. With these type of codecs the different frames are encoded differently by considering the group characteristics. Basically we have four types of frames, the significant ones being the I and P Frames. The other two are the B and D frames, which allow more efficient encoding.

The idea is to have an I-Frame, which is like a normal fully defined frame (image) and then have P-Frames which are not full images on their own but contain information which is different from the previous (thus the name "P") or the first frame. The B-frames use both previous and future frames for data reference to get the highest amount of data compression. We also have a special type of I-frame, which is called the *Instantaneous Decoder Refresh (IDR)* frame, where frames after it will not refer any frame behind it. IDR is especially good for seeking to a specific time frame in a video stream, because playback will be smooth if it starts from an IDR frame.

One of the most popular codec that uses the GOP encoding technique is the H264 codec. In the current IPTV standard document, the H.264 compression codec is specified as one of the main video codecs for IPTV.

### 2.1.2 Video Container Formats

Because video consists different types of data elements or streams, if we want to store a video and play the streams together, then it is better that these data elements (most of the time audio and visual) be stored in a way that the visual and the audio part for a specific time must be close to each other so as to make sequential read possible. As a result, for the purpose of storing and playing back audio-visual data together, container formats are developed.

Container formats are wrapper formats that specify ways to store multimedia data together while avoiding dependencies between them. A container generally stores one video stream and sometimes multiple audio streams and maybe even other content like subtitles

together. There are open source libraries, like libvlc [20] that can be used to read and decode what is contained in container files.

### 2.1.3 Playing and Delivering Video Over a Network

The two popular methods of playing and delivering video over a network are Broadcasting and Streaming. Video broadcasting normally implies the delivery of live video stream, whereas streaming refers to the delivery of Video on Demand (VoD). Another form of network based video delivery technique is real-time video communication, such as video call or video conferencing. In this thesis because of our scope we focus on streaming related services.

Streaming servers require bitrate information of an encoded video to stream it over a network, which in some case is stored in the container file. The process of storing bitrate information in a container file is called *Hinting*. Not all container formats support hinting and also not all streaming servers require a hinted file.

Streaming video also requires the use of an appropriate transmission protocol. There are different data transmission techniques for streaming video over a network. The two most popular techniques are using the MPEG Transport Stream (MPEG TS) and the RTP protocol.

Apart from the transmission protocol, there are also other protocols required for the development and delivery of different types of video-oriented services that are delivered over a network. The next section presents the different protocols required for streaming video and for the development of other value-added services for basic and advanced video-oriented services.

## 2.2 Protocols for Video-Oriented Service Development and Delivery

In the development and delivery of video-oriented services, different types of protocols are required for the purpose of session management, to transport the media, handle user preferences, etc. For this purpose we organize them into different categories and present them as such.

### 2.2.1 Session Control Protocols

In any telecommunications system, besides the actual media data, a great deal of information needs to be passed back and forth between users devices and also between the different network elements for the completion of a call [21]. Signaling is the term used to refer to these information. Signaling allows call information to be carried across network boundaries and it is signaling that also is used to develop value-added services [22]. So any discussion of a telecommunications service needs to start with a discussion about the signaling protocol involved.

The popular signaling protocol for communication within the IP sphere is the Session Initiation Protocol (SIP). In fact, as mentioned in [23], SIP has become a de-facto signaling protocol for the Internet and NGN.

When media servers, like streaming servers are involved in a multimedia session, we may have a session control protocol specifically for media streaming sessions. In this regard, RTSP is another session control protocol to manage streaming sessions. These two protocols are explained below.

#### 2.2.1.1 The Session Initiation Protocol (SIP)

SIP is designed to provide signaling functionality for real-time communications, such as VoIP, instant messaging (IM), video, conferencing/collaboration, and others. It is used to create, modify and terminate multimedia sessions in IP networks.

The Internet Engineering Task Force (IETF)[24] is behind standardization of the SIP protocol, which was initially approved as RFC 2543 in March 1999 [25]. It has, however, evolved to include new features and the current SIP standard that is used for VoIP systems is RFC 3261 [26], and it is referred to as the core SIP specification. Because of the increased interest and tremendous contributions to SIP, IETF formed a separate SIP working group in 1999 for SIP. However, the specific needs of SIP developers and service providers have led to an increasing number of new working groups that handle SIP standardization efforts for specific applications [22]. One of the major working groups is the SIP Project Investigation (SIPPING) working group. The SIPPING group was formed in 2002 to analyze the requirements for application of SIP to several different tasks. It is this working group that came up with extensions of SIP for the IMS environment. There is also another working group that focuses on SIP for Instant Messaging and Presence Leveraging (SIMPLE).

Through the various extensions, SIP has evolved to the current stage where it can now

be used to provide interesting services like session transfer [27] and event notification and subscription [28]. These features is discussed in detail in Chapter 3. Because the protocol is text-based, SIP is flexible and easy to extend. SIP's extensibility allows for new headers to be added without changes to pass-through proxies or even client user agents that use SIP. New methods can also be defined as needed. For example, the authors in [29] defined a new method to implement a task they called *Split a SIP session over multiple devices (SSIP)*, which helps the management of SIP sessions on different devices from one of the user agents. Another very common SIP extension is the SIP INFO method [30], defined for the purpose of extending the functionality of SIP for carrying session-related control information (e.g account balance information). In general, SIP with its extensions are key elements of many new applications in the Internet and Telecom world. For details of the SIP signaling protocol, see [22, 31, 32].

In a SIP-based communication, different components are involved. The main ones are Proxy, Registrar, Application Server, and SIP User Agent (UA). Proxies are similar to the standard telecommunication switches, and their task is the routing of connection requests to the appropriate end device for proper communication. Another important component in the SIP-based communication system is the SIP Application Server (AS), which is used to store the service logic.

The basic idea behind the development of ASs is to provide one or more value-added multimedia services to the caller, the callee or both. In processing the application logic, ASs provide any of the following functions: originate, terminate or route SIP messages. Because of this, a SIP AS can operate in different modes. When it generates or terminates a SIP request, it operates in a SIP UA mode. When it stays in the middle of a SIP session so as to track all SIP messages, it operates as a B2BUA (Back-to-Back User Agent). A B2BUA is concatenation of two SIP User Agents connected by some application-specific logic. Figure 2.1 shows how a B2BUA handles SIP messages. Finally, an AS can also act as a SIP redirect server and a SIP proxy server, depending on the service they provide to the user.

SIP-based sessions are initiated using the SIP INVITE request. The SIP INVITE request contains among other things the media capabilities of the caller together with the ports that it is going to use to accept media. For the purpose of media negotiation, UAs send their media capabilities using the Session Description Protocol (SDP). The SDP protocol is explained later in this chapter. SIP sessions are setup by using a three-way handshake communication between the end points much like the TCP protocol. So, after receiving the INVITE, the other communicating party responds with an OK message and the initiator of the communication then sends an ACK message to establish the communication.

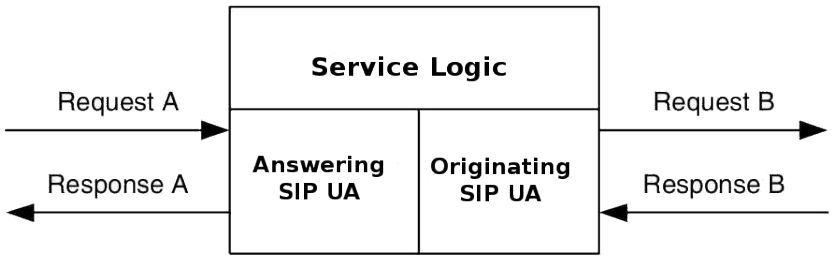



Figure 2.1: A SIP B2BUA logical view

As an IP-based protocol, SIP also utilizes an addressing scheme similar to email addresses. Each user in a SIP network is identified by a unique name (also called SIP account) and the SIP domain where he registered to get his services.

There are various open source implementations of the SIP stack available to application developers. The common ones are the Java API for Integrated Networks (JAIN) SIP, developed as a Java API, and the eXosip stack written with the C programming language. The JAIN SIP API itself has different high-level implementations (also called *reference implementations or RIs*). JSIP (also known as NIST-SIP) is the popular RI of the JAIN SIP API. Actually we used the Mobicents Service Development Platform (SDP) [33] to develop SIP based ASs. Mobicents is a Java-based service delivery platform for quick development, deployment and management of next generation network applications [34]. It is JAIN SLEE 1.0 compliant, and adopts data from various IP protocols for use as Java objects through its various resource adapters. Service Logic Execution Environment (SLEE) is a well known concept in the telecommunications industry for developing and deploying network services [35]. The use of Mobicents as a service development environment for NGN is explained in detail in Chapters 5 and 6.

#### 2.2.1.2 The Real Time Streaming Protocol (RTSP)

In video streaming, the client and server communicate using a streaming control protocol called RTSP [36]. The RTSP protocol is a media control protocol and does not handle media delivery by itself. The delivery of media is mostly done through RTP [37], which is described later in this chapter. Similar to real-time communication, the server sends technical information about the media to be streamed during media setup using the SDP protocol and the client checks this information in order to see if it can decode the stream or not. During the course of streaming, the client and server also exchange QoS information using the Real Time Control Protocol (RTCP) [38].

RTSP messages can be transported through either a TCP or UDP connection, whereas the data (RTP packet) is mostly transmitted through UDP. The RTSP message formats share a similar syntax to HTTP. Table 2.1 presents the RTSP methods as specified in the standard document but the most important ones are: DESCRIBE, SETUP, PLAY, PAUSE and TEARDOWN. Below is the syntax that all RTSP requests should follow (where SP stands for *space* and CRLF for *carriage return* (*CR*) and a *line feed* (*LF*)):

Method SP Request-URI SP RTSP-Version CRLF

RTSP responses should follow the following syntax:

RTSP-Version SP Status-Code SP Reason-Phrase CRLF

As in HTTP, the Status-code is a 3 digit result code and the first digit defines the class of response. There are 5 values for the first digit, as shown below:

1XX: Informational - Request recieved, continuing process
2XX: Sucess - The action was successfully received, understood, and accepted
3XX: Redirection - Further action must be taken in order to complete the request
4XX: Client Error - The request contains bad syntax or cannot be fulfilled
5XX: Server Error - The server failed to fillfill an apparently valid request

| RTSP Method | Description |
|---|---|
| OPTIONS | Is used to get supported methods and features by the server. |
| DESCRIBE | Retrieves the description of a presentation or media object identified by the request URL from a server. |
| ANNOUNCE | Is used to update the session description in real-time. |
| SETUP | Is used to setup media session with the server. |
| PLAY | Is used to start media delivery by the server |
| PAUSE | Is used to pause current media delivery. |
| TEARDOWN | Is used to terminate media session. |
| GET_PARAMETER | The GET_PARAMETER request retrieves the value of a parameter of a presentation or stream specified in the URI. |
| SET_PARAMETER | This method requests to set the value of a parameter for a presentation or stream specified by the URI. |
| REDIRECT | The server informs the client to connect to another server at another location. |
| RECORD | This method initiates recording a range of media data according to the presentation description. |

Table 2.1: RTSP methods

Basically a streaming session starts with a DESCRIBE command from the client (asking the server to describe the media). The response to the DESCRIBE request should contain

all media initialization information for the video. The server sends this information using the SDP protocol. Listing 1 shows a typical response to a DESCRIBE request.

**Listing 1** Sample DESCRIBE response

```
Content-Type:application/sdp
x-Accepted-Retransmit:  our-retransmit
x-Accepted-Dynamic-Rate:  1
Content-Base:  rtsp://146.231.123.99:554/samplevideo.mp4
v=0;
o=StreamingServer 3479449657 1251491821000 IN IP4 146.231.123.99
s=/samplevideo.mp4
u=http:// e=admin@
c=IN IP4 0.0.0.0
t=0 0
a=control:*
a=range:ntp=0-112.04527
m=video 0 RTP/AVP 96
a=3GPP-Adaptation-Support:1
a=control:trackID=3
a=rtpmap:96 MP4V-ES/90000
a=mpeg4-esid:1
a=fmtp:96
profile-level-id=1;
config=000001b589130000010000000120 0c4fc6a85885da041e14630000001b30001007;
m=audio 0 RTP/AVP 97
a=3GPP-Adaptation-Support:1
a=control:trackID=4
a=rtpmap:97 streamtype=5;
profile-level-id=15;
mode=AAC-hbr;
config=1210;
SizeLength=13;
IndexLength=3;
IndexDeltaLength=3;
```

The important parts in the SDP message as shown in Listing 1 are the `c=` and `m=` lines. These attributes provide the connection type and media information respectively. A separate `m=` lines should be stated for each media type and contain information about the type of media (audio or video), the port through which the server will send the particular media, the transport type and codec code. Mostly the `m=` line is accompanied by an `a:control` line that contains the URL of the specific media element, also called track. The RTSP specification also defines other attributes such as `a:range` for the purpose of providing time-related information about the media.

After the client receives a response for the DESCRIBE request, it sends separate SETUP request related to each track. In the SETUP request, the client needs to specify the port through which it wants to receive the media. In its response to the first SETUP request, the server creates a `SessionID` and includes it in the response. The client and server

use the `SessionID` in all further communications. A sample SETUP request is shown in Listing 2.

If we want the media to be delivered to a different device or client other than the client that initiated the streaming session, we can use the `destination` parameter in the transport section of the SETUP request as specified in Listing 2. This is an important parameter that we can use to develop value-added streaming services, like stream transfer. However, there is a problem in using this parameter. In the first place, different versions of the RTSP specification refer to it by different names. Version 1.0 [36], for example, refers it as `destination`, while version 2.0 of the RTSP specification, which is still an Internet draft, refers it by the name `dest_addr` [39]. On the other hand, most of the open source streaming servers we investigated do not support this parameter. The basic idea for the introduction of this attribute is that if an RTSP client wants the server to send the media to a different location, then it should put the address of the destination machine (client) that it wants the media to be sent to in this parameter. If the server supports this feature, then it sends the media to the specified destination when the media delivery begins, but it continues to send the RTSP responses to the RTSP client. We used this feature to develop a streaming service for SIP UAs that do not support the RTSP protocol and is explained later in Chapter 3.

**Listing 2** Sample SETUP command

```
SETUP rtsp://146.231.123.99:554/samplevideo.mp4/trackID=4 RTSP/1.0
CSeq:2
Transport:  RTP/AVP;unicast;client_port=4588-4589;destination=146.231.123.98
```

Considering how media sessions are set up in SIP-based communications, we observe an issue with the session setup of the RTSP protocol. If we compare the streaming session setup with SIP session setup, the streaming session setup seems a bit awkward. From a SIP session setup we can see that a multimedia session setup can be made using one command (INVITE request). In the same manner, once the streaming client knows the different tracks contained in a stream, then it should be able to issue one SETUP command to set up the session. In other words, it would be good if the streaming client can send one SETUP request instead of two (or more depending on the number of tracks) and the server can also send the `SessionID` when it responds to the SETUP request. We want to propose this as an RTSP protocol extension for the SETUP command and for this purpose we define an attribute called `moreTrack` that can be included in the SETUP command for each additional track.

A sample SETUP command with this new feature is shown in Listing 3. The client can still send a SETUP command for one track as usual, like when the user wants to set up

an audio only streaming session.

**Listing 3** The proposed RTSP SETUP command

```
SETUP rtsp://146.231.123.99:554/samplevideo.mp4/trackID=3 RTSP/1.0
CSeq:2
Transport:  RTP/AVP;unicast;client_port=4588-4589;destination=146.231.123.98
moreTrack:  rtsp://146.231.123.99:554/samplevideo.mp4/trackID=4 RTSP/1.0
Transport:  RTP/AVP;unicast;client_port=4590-4591;destination=146.231.123.98
```

When the client is done with the SETUP request, it sends the RTSP PLAY command to demand the starting of the media delivery. Upon receiving the PLAY command, the server then starts sending the media to the address the client specified in his SETUP request. The response to the PLAY command may carry a header called `RTP-Info` which is used to set RTP-specific parameters in the PLAY response. Two of the parameters we may get in this header are URL and rtpmap.

Finally, RTSP clients use the TEARDOWN command to terminate a streaming session. One major problem of the RTSP protocol related to session termination is the lack of appropriate command for the server to specify the end of a session, like the BYE command in the SIP protocol. The streaming server only informs the client about the end of a session using the RTCP BYE message. However, as a session control protocol, it would have been good if RTSP does this. This problem makes it challenging to develop some value-added services, like *Recommendation* service. Different solutions have been proposed to curb this problem.

An Internet draft [40] has been published to extend the use of ANNOUNCE for sending the *end of stream* signal from the server to the client but it is not included in the recent draft RTSP specification (RTSP 2.0). In fact, the Internet draft also mentioned the limitation of the proposal in that it can be used only if the RTSP connection between server and client is persistent. As mentioned before, RTSP can be transported using UDP and the connection between the server and the client may not persist.

Microsoft has also defined a new attribute called `EndOfStream` that the server can use together with the RTSP SET_PARAMETER command to inform the client that the last RTP packet is sent and the session has ended [41].

What we consider as a good solution is proposed by IETF in the new Internet draft for RTSP (RTSP 2.0). IETF defined a command called PLAY_NOTIFY to send asynchronous messages from the server to clients. The PLAY_NOTIFY can be used to notify *end of stream* by setting the `Notify-Reason` header to `end-of-stream`. However, since it is yet an Internet draft and is not defined as a standard, all of the existing open source streaming servers analyzed do not support it.

The summary of the basic RTSP message communication between client and server is shown in Figure 2.2. The OPTIONS command is normally used to get information about the commands supported by the server.

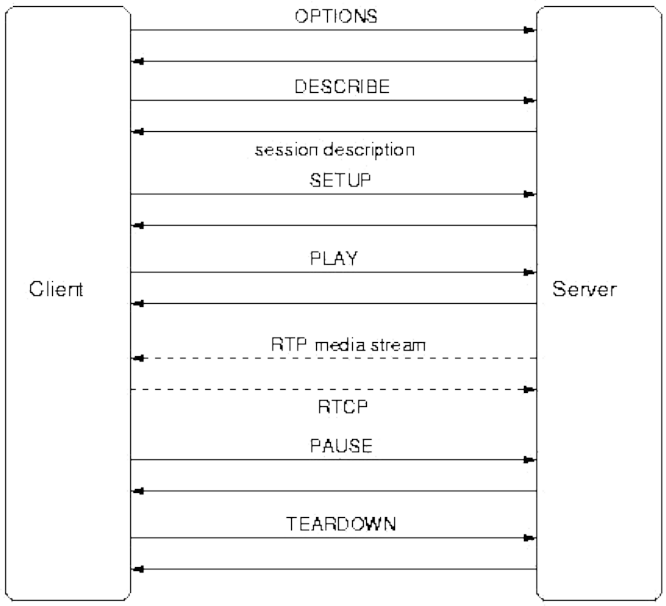



Figure 2.2: RTSP message flow

#### 2.2.1.3 The Session Description Protocol (SDP)

As mentioned above communicating parties in a SIP session send media related information, transport addresses, and other session descriptions to the other party using the SDP protocol [42]. This protocol is defined by the MMUSIC working group of IETF as RFC 2327. Because SDP was originally defined for the purpose of describing multicast sessions, a more specific protocol for use in a standard SIP communication is extended from the base SDP and it is called the SDP *offer/answer* model and is defined as RFC 3264 in [43]. The *offer/answer* model is also adopted by IMS standard bodies.

Similar to SIP, SDP is also text-based protocol and is designed to be used in a wide range of networked environments and for different applications. The type of information that an SDP session description includes are:

- Session name and purpose
- Time(s) the session is active

- Number and type of media comprising the session
- Information needed to receive those media (addresses, ports, formats, etc.).

Each element in an SDP body should be put in a separate line in the form of: *<type>=<value>*. The `value` part could be a single value or a number of values delimited by a single space or a free format string all together. Session descriptions are categorized as session-level, media-level, and timing information. Listing 4 shows a sample SDP offer session description.

**Listing 4** A sample SDP offer

```
v=0
o=Zelalem 4521839516 4880742302 IN IP4 146.231.123.112
s=SIP seminar
i=A Seminar on the Session Initiation Protocol
u=http://www.ru.ac.za/sip e=zelalem@convergence.ru.ac.za
c=IN IP4 146.231.123.124
t=2873367456 3878404796
a=recvonly
m=audio 59185 RTP/AVP 0
a=rtpmap:0 PCMU/8000
m=video 36677 RTP/AVP 31
a=rtpmap:31 H261/90000
m=video 38308 RTP/AVP 32
a=rtpmap:32 MPV/90000
```

In most cases, the information contained in SDP should be sufficient to enable communicating parties to join a session. However, if the need arises, SDP can also be extended to support new features. The protocol particularly defines an attribute line referred with the `a` attribute for the purpose of extending SDP. When an application wants a feature that is missing in the standard SDP, it can add an `a` line containing the information that the application wants to attach. The attribute may be defined with key and value as in `a=rtpmap:96 L8/8000` or with key only as in `a=sendrecv`. We have defined new attributes for the purpose of representing content markers and discussed them in Chapter 6.

Because there is no exhaustive open source SDP parser available for use, we have developed our own SDP parser based on a Java-based open source SDP API called *JAIN-SDP* to parse SDP messages. The parser is explained in Chapter 3.

### 2.2.2 Protocols for Media Transport and Feedback

As mentioned before, there are different transport protocols, but we consider RTP in this thesis and discuss how it supports the development of value-added services.

#### 2.2.2.1 The Real Time Protocol

The Real Time Protocol (RTP) is used as a transport protocol for real time data of any type. RTP provides an end-to-end network transport mechanism that is suitable for applications transmitting real time data, such as audio or video over IP networks [38]. It can be used for both multicast or unicast network services. It defines a standardized packet format for delivering real-time data. RTP allows the receiver of a media stream to play out the media at a proper pace.

One of the interesting design considerations of RTP is that it can be used by any type of media format including new media formats, without the need to modify it. For this purpose the RTP protocol includes a header (field) called `Payload Type (PT)` that relates to the type of codec used to encode the media contained in the RTP packet. This will help the receiver to easily generate the data by using an appropriate decoder. To this effect, IETF defines application-level documents that specify the media profile for different applications which contain one or more payload types. A media profile defines the codecs used to encode the payload data and their mapping to payload format codes that is included in the PT field of the RTP header. RFC 3551 [44] presents a set of static payload types and a mechanism for mapping between a payload format, and a payload type identifier for audio and video conferences. Applications other than audio and video conferences can also use these codes to communicate with each other. RFC 3551 also specifies some payload codes as dynamic payload type, leaving the coding assignment to implementers. As per the RFC, payload type numbers *96 to 127* are exclusively reserved for dynamic assignment.

When using dynamic payload types, the mechanisms for defining dynamic payload type bindings should be incorporated inside the SDP message that describes the multimedia session, using the `a=rtpmap` format. For example, the line `a=rtpmap:98 L16/16000/2` in an SDP refers to an audio codec named `L16` with payload type `98` (and sampling rate of `16000` and number of channels `2`).

RTP also uses `timestamp` and `sequence number` for media synchronization. These RTP headers allow a receiver to distinguish between lost packets and periods of time when no data was transmitted (e.g. silence suppression). The `timestamp` header gives the sampling instant of the first byte in an audio/video packet. The first byte of the first packet is assigned a random initial value and is different for each RTP stream. Because a given video frame could be broken down into several different packets, several packets may have the same timestamp. Because of this, the `sequence number` header is an important header that is used for synchronization. So, in real-time communication and

video streaming the receiver places incoming RTP packets in a buffer according to their timestamp and sequence number and starts playing them when the buffer is full. Sequence numbers increase by one for each RTP packet transmitted but `timestamp` increase by the time "covered" by a packet, which also depends on the sampling instance of the codec used. We used the `timestamp` information to implement bookmarking service and it is explained in Chapter 6.

#### 2.2.2.2 Real Time Control Protocol

Because most of the time RTP is transported over UDP, data delivery can not be guaranteed. For this purpose RTP uses another protocol, called Real Time Control Protocol (RTCP), to exchange periodic transmission of statistics or reports for the purpose of monitoring the delivery of data [36]. These reports help ensure QoS and also aid synchronization of multiple streams. RTCP reports are of two types. The reports from the sender side are called *Sender Reports* (SR), and the reports from the receiver are called *Receiver Reports* (RR). In addition to the SR and RR packet types, RTCP also has *Source Description* (SDES) and *End of Participation* (BYE) message types. The BYE message serves to shut down the stream or to indicate the end of a stream. The use of BYE message to identify the end of stream for the purpose of initiating a recommender system is discussed in Section 5.1.2. RTCP is also used to provide mapping between the timestamp of the media streams in a given session to a wall clock.

If a given communication contains different types of media, each media type will have its own RTP and RTCP sessions. Therefore, in a video communication, we will have two RTP sessions and two RTCP sessions, one for audio and another one for the video part.

### 2.2.3 User Profile Management Protocols

Personalization is one area where we need to utilize user profile data. Personalization involve delivering personalized media to users based on their request and also notifying the existence or arrival of media that may interest them by the system automatically. There are different protocols that can be used to achieve this, but we present below those that we utilized in this research.

#### 2.2.3.1 The XCAP Protocol

For different reasons, it is customary to use XML documents to store profile data for the development of different applications. As a result of this, the IETF has defined an HTTP-

based protocol called XCAP for the purpose of manipulating profile and configuration data stored as XML document in a central location (server) [45]. XCAP allows applications to access XML documents that are common to all users or XML documents that affect the service of a given user. XCAP also allows us to retrieve or update a piece of information or the whole XML document. Most applications in NGN environment, including IPTV also utilize XCAP for managing profile data.

XCAP is a set of conventions for mapping XML documents and document components into HTTP URLs, rules for how the modification of one resource affects another, data validation constraints, and authorization policies associated with access to those resources [45]. It allows clients to create, read, update and delete configuration data stored in a server. The protocol uses URIs to identify XML documents or elements, attributes and values in XML documents.

**Listing 5** The HTTP GET syntax

```
GET http://localhost:8080/XCAPRoot/Application(k)/user(j)/document(i)/~~/xcap-caps/friends/friend1 HTTP/1.1
```

**Listing 6** Sample XCAP document

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xcap-caps>
   <friends>
      <friend1>g08s0002@cs.ru.ac.za</friend1>
      <friend2>g09m3245@cs.ru.ac.za</friend2>
   </friends>
</xcap-caps>
```

XCAP defines two logical roles: XCAP client and XCAP server, also called XML Document Management Server (XDMS). An XCAP client uses the HTTP PUT, GET and DELETE methods to manipulate XML documents in an XDMS, which are basically called *CRUD operations* (create, read, update, and delete). Some examples of XML documents that needs to be stored on the network side in XDMS include presence authorization rules, static presence information, contact and group lists, and policy data. Listing 5 shows an example of a URL required for the retrieval of an element from an XML document presented in Listing 6 using the HTTP GET command.

Because an XDMS server manages configuration documents of different applications, IETF defines the concept of *application usage* [45] (also called *appusage*). An application usage defines a convention about the structure and constraints of the XML data for a specific application. It also defines well-known URIs to bootstrap access to the application's data. We discussed the application usage define by TISPAN for IPTV in Chapter 4.

#### 2.2.3.2 The Diameter Protocol

Diameter is a protocol used for communicating authentication and authorization related information. It is also used for carrying accounting information (like charging). The protocol is defined as a base protocol and a set of different applications. Diameter is standardized by IETF and the base Diameter is defined in [46]. Application designers can come up with a new Diameter application by extending the base protocol and adding new commands and/or attributes. Basically, Diameter Applications are extensions to the basic functionality that are tailored for a particular usage of Diameter in a particular environment [13]. In this regard, 3GPP also defined a Diameter application to extend the base Diameter protocol to be used for the `Sh` interface of the IMS architecture and published it in [47]. The IMS architecture is described in Chapter 4. The application defined by 3GPP is called the *Diameter Application for the Sh interface*.

## 2.3 Basic Video-Oriented Services and Challenges Related to their Development

Multimedia services represent real-time interactive audio video (e.g. video call or video conferencing) services, streaming of live audio video (e.g. broadcast TV), and streaming of stored audio video (e.g. VoD). We can broadly categorize video services as real-time video services and video on-demand services and refer to them as basic video services. Related to these categories, we also have value-added video-oriented services. Value-added video-oriented services can be developed by combining them with other Internet services. For example, we have investigated the possibility of developing an eLearning system that uses video communication together with a streaming service and presented it in [48]. Another value-added service related to real-time video services is videomail. As they will provide insight for building the envisaged environment, we present the characteristics of videomail and also the issues related to streaming services in this section.

### 2.3.1 The Videomail Service

A videomail service is usually invoked when the user is not able to answer a call. This may happen when the user's device is not registered or if the user does not want to answer the communication request. IMS provides support for checking if the callee is registered or not. This is discussed in Chapter 4. On non-IMS environments, however, the AS needs to include a logic to check if it needs to initiate the recording of the message. In both cases,

the procedure followed when a videomail service is started is that the videomail system plays an announcement to inform the caller that the user is not available and requests him to leave his message. If the caller wants to leave a message, the system starts to record the call and later on informs the user about the availability of the videomail.

Normally, we develop these functions as separate modules or components of the videomail system and in minimum we can announcement server, recorder, and player. The problem is that these components may use different protocols and involve separate codec negotiations. So, the main issue of developing a videomail service is finding an environment that supports the different protocols that the different components of the service use. Figure 2.3 shows the flow diagram of a particular videomail service implementation.

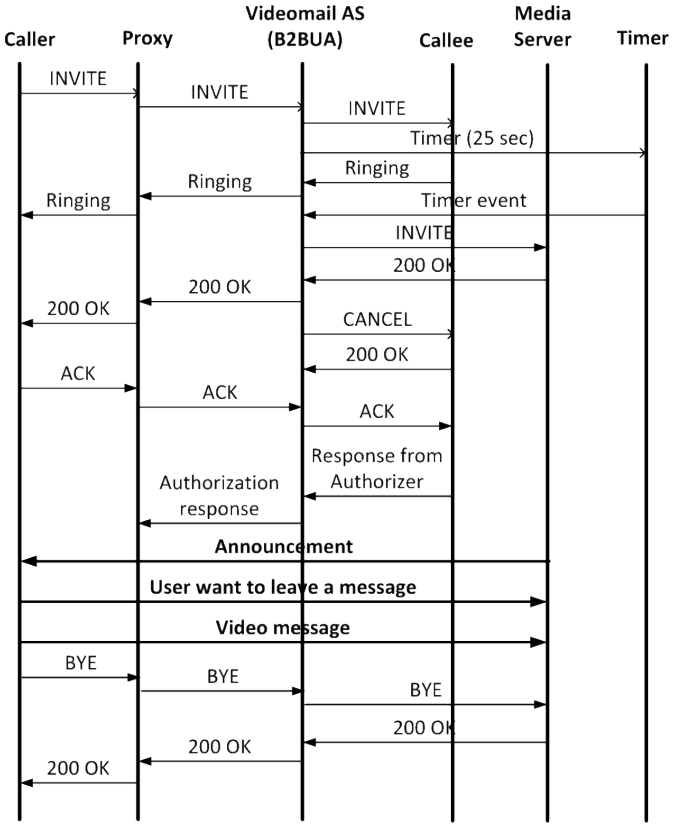



Figure 2.3: Flow diagram for non-IMS videomail system

### 2.3.2 Streaming Service

There are two types of video streaming services: progressive streaming or progressive download and true streaming (also called real-time streaming). Progressive streaming is implemented using the HTTP protocol and most video sharing sites on the Internet use

it. Because of the inherent nature of the Internet, the transmission speed may not be constant and the client in the case of progressive streaming uses a buffer to store enough data before it starts displaying. With true streaming, the client starts watching the file almost as soon as it begins downloading. In this thesis we consider true streaming, which we call it simply streaming. In streaming video, the server provides users with a VCR like functionality, and users can fast-forward or rewind as well as pause and resume the video. We identified different issues related to the development of streaming services.

One issue we identified that is related to streaming service is the challenge of developing a streaming service for clients that do not support the RTSP protocol, like the SIP UAs. The problem is that most of the open source streaming servers do not support the `destination` parameter, that is mentioned in Section 2.2.1.2, and which is used to to request the delivery of a stream to another device other than the devise initiated the session. Of all the open source streaming servers investigated, only Live555 [49] support this parameter. Actually, even in Live555 this feature is disabled by default but can be enabled by inserting the "`#define RTSP_ALLOW_CLIENT_DESTINATION_SETTING 1`" directive at the beginning of the `RTSPServer.cpp` file. The Darwin Streaming Server (DSS), and VideoLan Streaming Server (also called VideoLan Client or VLC), the two popular open source streaming servers, do not support the use of this parameter. DSS returns an `Invalid Code` error message if it receives a session initiation request with this parameter and VLC just ignores it.

Another issue we identified relates to the support for timestamp attribute by open source streaming servers. One important feature that RTSP and streaming servers needs to support is the feature to provide the current play time. This is an important feature for developing services like Bookmarking. The RTSP protocol proposes the use of GET_PARAMETER command for requesting the current play time of a streaming session. Specifically one can use this command together with the `range` parameter to obtain the current media position. Again the open source streaming servers investigated do not support this command. DSS responds with a 500 error code, while the other servers just ignore it. Detail explanation of these and other issues related to open source streaming servers is presented in [50]. In general, open source streaming servers have issues when it comes to supporting streaming related services in the context we want them to work. For this purpose we developed an RTSP Proxy and Relay unit and present it in Chapter 5.

#### 2.3.2.1 Stream Switching

With the current implementation of RTSP protocol, what we can do to switch to another media is to terminate the current media session and then set up and play the new media session to start watching the new media. Even though it is small, this technique takes some time before the new media is presented and does not produce a good user experience. Especially if the media type (the number of tracks in the video and codecs used for the tracks) is the same between the current media and the one we want to switch to, it does not make sense to tear down the current session and set up up another one. What we need to have is a seamless stream switching so as to provide a good Quality of Experience (QoE).

Seamless stream switching is obtained when the switch is performed in such a fashion that media playback is minimally disturbed [51]. We can achieve this by using the existing session's connection to get the new media, avoiding the delay introduced because of the session tear down and setting up a new session. As a result it would be good if the client can request the server to replace (switch) the current media with a new one instead of issuing a separate session setup request.

Various proposals have been made on how RTSP can be improved or use other techniques to accommodate a stream-switching feature, but instead of proposing a standard solution, most of the solutions are designed as proprietary solutions. For example, Friedrich et. al. [52] show how media insertion, especially ad-insertions, can be done using a proprietary technique using the SIP INFO method. Apart from the proprietary nature of their proposal, the media is also coming from different servers. As a result, we cannot use this technique for media switching, because in our case the media may come from the same server. On the other hand, Blackberry Limited [53], also defined a method for the purpose of switching media feeds by defining new methods in their RTSP API [54]. They defined an `RtspContentControl` object that has a `switchStreamToUrl` method that can be used for the purpose of switching media. This is also a proprietary solution.

Some researchers have also considered other aspects of stream switching. For example, the work presented in [55] presents a proposal for RTSP that shows how separate sessions with different bit-rates can be set up at the beginning of a session for the purpose of switching to the appropriate session later on based on available bandwidth. The server can then deliver the appropriate media through the connection created at the beginning of the session based on the feedback it gets from the client. This proposal is resource intensive because for each user, the streaming server should establish different sessions from the beginning and wait until the user indicates his interest to make a specific session

active or not.

A recent Internet draft by Phillippe [51] proposes a new RTSP header, called `switch-stream` that the client can use it to indicate its interest to switch the current media with another content. The new header is used with the RTSP PLAY command. The author describes various scenarios that we can use this technique to switch a content, including when there is a content switching request coming without content description. This technique solves the major problem of media switching but we think there are still other scenarios that this proposal does not entertain. For example, there are cases where the switching of media is time-based, meaning the switching should happen after a certain time. This is particularly the case for playlists with play time (duration) attached to the list of media in the playlist. A good example of this case is the Personalized Channel (PCh) service presented in the IPTV specification and the above proposal does not handle media switching of this sort. The work we did related to Personalized Channel service is discussed in Chapter 5.

On the other hand, especially from the point of view of inserting a media (like an advertisement or a video clip about some event), we think there is still another use case where the media switch is required and needs to be done temporarily. That is, the switching is required just to see a small clip (like an ad) and then the user wants to go back to the current stream to resume watching where he left off. In this scenario the streaming server should resume the main media automatically when the server finish playing the temporary media instead of expecting another media session initiation and setup. For this purpose we want to make a new proposal for switching media, and defined new RTSP commands related to handle the above scenarios. The commands are discussed in Chapter 6.

## 2.4 Summary

This chapter presents background information about digital video that is particularly important for the development of basic and value-added video-oriented services. The chapter also provided critical review of the protocols required to implement video-oriented services that needs to be delivered over a Telco network. As a major video-oriented service in IMS, discussion was made regarding the issues related to the delivery of streaming service. Specifically, we presented the issues related to the RTSP protocol. The problems are not only with the protocols, but also with open source streaming servers. Open source streaming servers do not support some of the features that are defined in RTSP which are important for the development of innovative video-oriented services for IMS. We briefly highlighted the solutions that we wanted to propose for the various problems presented in the chapter.

In the next chapter, we present part of our experimental work that led us to the identification of some of the issues presented in this chapter. Other issues that were considered as requirements for the envisaged environment are also presented there.

# Chapter 3

# Video Service Development Experiments: Basic Video Services

As mentioned in Chapter 1, as part of understanding the problem of developing video-oriented services and selecting an appropriate development environment, we developed various video-oriented services using different service development deployment platforms and we present part of the work in this chapter. The service development exercise helped us to identify various issues related to basic video services and we present how we dealt with them in this chapter. This chapter specifically discusses the development of videomail and streaming related services. The chapter also gives background information about the technologies used to develop the services.

## 3.1 Videomail Service for iLanga

The Computer Science Department of Rhodes University was using a communication system called *iLanga*, and the videomail system was developed to upgrade this communication system.

### 3.1.1 The iLanga Communication System.

The Computer Science Department at Rhodes University has been using Asterisk together with other VoIP and non-telephony software for communication activities for a long time. This integrated communication system, called iLanga, was developed by the then VoIP research group at the Department. iLanga is implemented by utilizing various VoIP and non-telephony technologies to extend the capabilities of Asterisk [56]. iLanga, in addition

to Asterisk, uses the SIP Express Router (SER) as a SIP Proxy and Registrar, OpenGK, an implementation of the H.323 Gatekeeper for support for non-SIP UAs, and MySQL database, to provide a comprehensive communication service.

iLanga also includes a flash-based management and control web interface that allow users to easily set up their profile and communication devices, access their voicemail, and update their credit information for the prepaid modules. So, as a first experiment, we embarked on a project to include a videomail service into iLanga. The next section provides background information about Asterisk, the main component of iLanga, and what we also used to develop our videomail component.

### 3.1.2 The Asterisk Open Source PBX System

Asterisk is an open source Private Branch Exchange (PBX) that gives all functionality of high-end business telephone systems [57]. It provides a voicemail service through its core system. However, support for video related services is still in its infancy. The following sections discuss Asterisk's internal structure, its video capabilities and how we developed an application for the videomail service.

#### 3.1.2.1 Basic Concepts in Asterisk

Asterisk does most of its tasks by reading the different configuration files, the most important being the `extensions.conf` file, which contains the dialplan of Asterisk. The dialplan tells Asterisk how to handle incoming and outgoing calls.

Extensions, which are the basic concepts in any PBX system, have bigger meaning in Asterisk. They define the steps that Asterisk will take when a call comes to that extension. Each of these steps can be associated with applications (both built-in and others that can be developed by service developers) that Asterisk needs to execute.

Asterisk comes with special extensions for the purpose of representing some actions and events related to user defined extensions. These include the `h` (hangup), the `i` (invalid) and `s` (when a call comes to unknown extension) extensions. If we specify an action for an `h` extension in a given context, then Asterisk will execute the statements associated with the extension when the call hangup. We usually use the `h` extension to perform clean up activity, for example, delete temporary files to record a videomail.

The following example shows how extensions are defined. The statement shows that Asterisk executes the `Answer` application (specified in the action part) as a first task

(because it was defined with priority 1, specified in the middle) when the extension `1234` is called (the first part in the statement).

```
exten => 123,1,Answer()
```

Another fundamental concept in Asterisk is `channel`. A channel represents a connection between the Asterisk system and some telephony endpoint and contains information about the connection. Channels are either inbound or outbound. An inbound channel is created when a call comes into Asterisk. When Asterisk initializes an application in a dialplan, it passes the instance of the current channel as a parameter through the `ast_channel` data structure so that the application can do anything to the channel, such as playing a sound, accepting DTMF input, hanging up the call, and so forth.

Other concepts in Asterisk include `peers, users` and `friends` and can be referred in [58].

#### 3.1.2.2 Developing Applications for Asterisk

One can extend Asterisk's capabilities either using the Asterisk Gateway Interface (AGI) or by writing an application using the `C` programming language. If we follow the later route, the applications we develop need to be compiled as shared libraries (with `.so` extension) and be stored in a specific directory in Asterisk's directory structure.

When writing applications for Asterisk, we must confirm to the Asterisk application template. Detailed discussion of Asterisk's application development is found in [57].

#### 3.1.2.3 Video Support in Asterisk

Users can make video calls in Asterisk by changing certain configuration settings on their UA and Asterisk's configuration files. Asterisk supports most of the common codecs and file formats for audio communication. However, for video communication, it only supports *H.261, H.263* (and *263+*) and *H.264* video codecs.

With regard to video recording, although Asterisk can dump the contents of the RTP packets including some timing information in files related to the codecs used, like *.h263, .h263p*, and *.h264*, these files cannot be played back or streamed properly as ordinary video files [59]. As a result, an initiative was undertaken by some members of the Asterisk video service development user community to develop a pluggable application that can record and play back video calls in *mp4* video file format. This video recording and playback

application, called `app_mp4`, is an important work to help service developers build various video-related services for Asterisk, like the videomail systems we are describing in this chapter. The following section describes how we develop the videomail service.

### 3.1.3 Development of the iLanga Videomail (iVideoMail) System

As mentioned above, the iVideoMail system is developed based on the `mp4_app` application. The system also uses other Asterisk built-in applications. We start our discussion by explaining the `app_mp4` application.

#### 3.1.3.1 The app_mp4 Video Recorder and Playback Module

Installation of the `app_mp4` application for use in Asterisk can be found from [57] and one can follow the steps presented in the document to install the application. After the `app_mp4` application is set up and integrated into Asterisk, we can use its two functions, namely the `mp4play` and `mp4save` for playing back and recording of *mp4* files related to video calls, respectively. This is done by calling the functions at the appropriate place in the dialplan.

Both the `mp4play` and `mp4save` functions require the file name of the video message that we want to play or record to be supplied as a parameter. If one wants to play or record new video messages every time, he needs to develop an AS that provides the proper filename dynamically. So for the purpose of using these modules for developing the videomail for iLanga, we developed other modules and integrated them into iLanga. The following section presents the different tasks we did to develop videomail system and integrate it into iLanga.

#### 3.1.3.2 Preliminary Tasks

As a first task to creating the necessary environment for the smooth running of the system, we created the proper directory structure to represent users' videomail boxes for storing and archiving video messages in different formats. This was done under the */var/mail/videos/* folder. Each user's videomail box has three sub-folders, *Inbox* (for storing video messages), *FlashFiles* (for storing video messages in flash format that can be retrieved using the web interface) and *Archive* (for storing viewed messages). For example, the path to the video message mailbox of a user with extension number 1000 is: */var/mail/videos/1000/Inbox*.

Because Asterisk has a voicemail system, the videomail system needs to be invoked only when a video call is not answered. So, a videomail system needs to check if a call is not answered and also if the call was a video call.

We used the Dial command that Asterisk uses to forward calls to users to check if a call is not answered. This command takes a number of parameters, one of which is `timeout`, which specifies the number of seconds that we want Asterisk to wait for the user to answer or after which Asterisk stops ringing the callee's phone and moves the control to the next line in the dialplan. We can then check the status of the last Dial command to see if the call was answered or not using the built-in variable called `DIALSTATUS`. Status codes `NOANSWER` and `CHANUNAVAIL` are particularly important for our case.

To check if a call is a video call or not, we used Asterisk's built-in function called `CHANNEL` that returns the video codec that the caller wanted to use if there was one. If the call is an audio call, the function returns null. The following statement is used to store the result of the function in a variable that will be used to inspect if a call is a video call or not.

```
exten => s,1,Set(videocodec=${CHANNEL(videonativeformat)})
```

#### 3.1.3.3 Components of the iVideoMail System

The iVideoMail system contains various modules to perform different tasks and are developed as Asterisk ASs. For the purpose of allowing users to get their video messages on the iLanga web interface, the iVideoMail system includes a transcoding module that converts the video messages from *mp4* to *flv* format which the iLanga web interface uses. The iVideoMail system also sends email to users to inform about the availability of a new video message. In general, we developed and integrated six applications, which are: `app_checkGreetingMessage()`, `app_getNextCounter()`, `app_getNextFile()`, `app_sendMail()` `app_moveFile()`, and `app_convertVideo()`.

##### Playing the Greeting Message

Each user calls a special extension to record his own greeting message that will be played when the system needs to record a videomail for him. A file named *greetingsmsg.mp4* is created and stored in users videomail folder. So this module checks for the availability of this file.

If there is a greeting message then asterisk plays the greeting message using the `mp4play` module. If there is no personal greeting message created by the user, then Asterisk plays the default greeting message.

If the caller wants to leave the message then control passes to the video recording part that starts by checking the sequence number to be used for naming the video message. This is done with the `app_getnextcounter()` application.

**Recording the Video Message** A video message is saved using file name format: *message_ xxxx–from_ caller_ no.mp4*. For example, the file *message_ 0001-073010165.mp4*, specifies the first video message for a particular user from a caller who used the phone number *0730101651* to leave the message. We define a file named *nextFileNo* for each user and store the last sequence number we used in it. The `getnextcounter` function is used to read the value of the serial number from this file and then increment and save it again into the file. The code snippet in Listing 7 shows the steps that Asterisk follows to record a video message.

**Listing 7** Partial view of video recording steps in the dialplan

```
exten => s-vidrecord,1,NoOp(checkGreetMsg())
exten => s-vidrecord,n,GotoIf($["${greetingmsgexist}" = 1]?dial1,dial2)
exten => s-vidrecord,n(dial1),mp4play(/var/mail/videos/${to}/greetings.mp4); User
specific greeting message from his folder
exten => s-vidrecord,n,Goto(continue) ;No need to play public greeting message
exten => s-vidrecord,n(dial2),mp4play(/var/mail/videos/greetings.mp4); Public greeting
message
exten =>s-vidrecord,n(continue),Read(dtmfs,,1) ;check if he wants to leave message
exten =>s-vidrecord,n,GotoIf($["${dtmfs}" != "1"]?s-hangup,1)
exten =>s-vidrecord,n,getnextcounter()
exten => s-vidrecord,n,Set(fname=/var/mail/videos/${to}/inbox/${fnamewoext}-${callee}
$ {fileextension}) ;concatenates the file name with extension
exten => s-vidrecord,n,mp4save(${fname})
exten =>s-vidrecord,n,GoTo(h,1) ;perform clear up operation
```

**The Different Ways of Accessing Videomail in iLanga** One of the advantages of iVideoMail system is that it gives the user the ability to see his video messages using different techniques. There are three ways that a user can access his videomail: by calling a special extension in Asterisk, from a streaming server using a streaming client, and through the iLanga web interface.

As mentioned above, for the purpose of using the iLanga web interface, the system converts video messages into *flv* format because the *Flash Player* in the iLanga web interface does not support the *H.263/H.263+* codecs.

For the purpose of accessing the message from streaming servers, the videomail system sends the RTSP URL link to the user by email. These last two tasks are performed by

the `app_convertVideo` and `app_sendMail` applications and explained below.

**Converting the Video Message** The `app_convertVideo` uses the FFMPEG [60] media framework to convert the video message into *flv* format. It uses the Unix `system()` function to issue the FFMPEG command to convert the *mp4* file into *flv* format. The code snippet in Listing 8 show this step.

**Listing 8** Code extract from convert video application

```
....
strcpy(inputfile,"/var/mail/videos/");
strcat(inputfile,exten);
strcpy(outputfile,inputfile);
strcat(inputfile,"/inbox/");
strcat(inputfile,file name);
strcat(inputfile,".mp4"); //The file name just recorded*/
strcat(outputfile,"/flash/");
strcat(outputfile,file name);
strcat(outputfile,".flv");
strcpy(command,"sudo ffmpeg -i ");
strcat(command,inputfile);
strcat(command," -f flv -r 25 -b 560000 -s 610x340 ");
strcat(command,outptfile);
i=system(command); //execute the command
....
```

**Sending an Email** The app_sendMail application uses the Linux *Simple SMTP (sSMTP) mail agent* for sending email to users through an SMTP mail server. sSMTP uses the Simple Mail Transfer Protocol (SMTP). The sSMTP mail agent has a configuration file that needs to be set to access a mail server. The mail agent is designed to be interactive and is initiated by submitting the email address that we want it to use to send an email as a parameter. The sSMTP agent will then start and wait for us to type our message, which needs to be formatted similar to the one presented in Listing 9:

**Listing 9** Sample sSMTP content

```
To:  zelalemss@gmail.com
From:  iVideomail@convergence.ru.ac.za
Subject:  test email
CRLF
hello world!
```

When we finish typing the content of the email, we need to press *Ctr + D* and the agent will send the email based on our configuration. To make this process automatic, we can

use the UNIX I/O redirection technique to inform the sSMPTP agent to get the content from a text file as shown below.

```
ssmtp zelalemss@gmail.com  < msg.txt
```

So, the `app_sendMail` application creates a text file and put all parts of the email into the file for the mail agent to use it before it issues the above command. The application then issues the command using the UNIX `System()` function similar to the one mentioned in the previous section. The subject line contains the phrase *You have new videomail* and the body of the message is similar to the following:

```
You have new videomail from 0730101651.  You can access it by calling extension
1000 or using any media player or RTSP enabled web browser using the link:
rtsp://streaming.convergence.ru.ac.za:5554/7500/message0001-073010165.mp4
```

**Video Message Delivery and Post-Delivery Tasks** As mentioned before, users can use different technologies to get their messages. When the user uses a SIP UA to call the special extension to get his message, the iVideoMail delivers all unseen video messages one by one using the `mp4play` function. The system accesses each video message with the help of the `app_getnextfile` application. Once a message is presented to the user the system archives it using the `app_movefile` application. These two applications use UNIX file manipulation commands to do their task.

We also used the *Festival Speech Synthesis System (FSSS)* [61], which is integrated with Asterisk, to let the user know about the number of emails he has, and also to ask if he wants to play them. The configuration file for FSSS is attached as Appendix B.

**Lessons Learnt** The videomail system just explained shows how open source tools can be used to create a converged videomail system. In conventional videomail systems where the videomail system is designed for a particular technology, like a PBX, users will not normally have the possibility to get their messages using a different channel. The iVideoMail system shows how this can be changed by including streaming functionality to the system.

The videomail system still lacks dynamic notification features. For some reason, users may not check their emails, and especially for urgent messages, it would be good if the system alerts the user the next time he boots up his device or register. Asterisk do not support this feature and we could not include it in the iVideomail system. Because IMS provides the notification of users registration status, which is an important aspect to

develop dynamic notification systems, the videomail system needs to be migrated to an IMS based environment. This is one of the reasons we selected the IMS framework as an infrastructure (backbone) for the service development environment that we envisaged to develop.

In addition to this, the iVideoMail also lacks extensibility. For example, if we want to use a different protocol to send the mail or use a different streaming protocol for the streaming part, we have to re-write the applications again. This means, the system should have a mechanism to include support for new protocols.

## 3.2 Selective Dissemination Information (SDI)-Based Notification

Selective notification, also called *Selective Dissemination of Information (SDI)*, is an old technology that has been used in libraries for a long time for the purpose of providing personalized notification about the arrival of new items [62].

In this section we present the SDI-based notification system that is developed for Asterisk and discussed in [63]. The system notifies to users the availability of videos that may interest them in two different ways. In the first case, notification is done automatically by the system. This is especially happens when a new video comes into the system and the user is online. When a new video comes, the SDI module, checks which user may be interested in the new video and sends a SIP MESSAGE command to the user to inform him about the video and how he can access it. The text message includes a code of the new video. The user then dials a special extension to access the new video and then supply the code when the IVR system asks for the code of the new video. The sequence diagram for notifying the arrival of new videos to the user is presented in Figure 3.1.

The other type of notification is in the form of a recommendation that is made after a user finishes watching a video. At the end of any video streaming, the system asks the user to rate the video and based on his rating, the recommendation module will choose similar videos to him. The system uses the different attributes of the video to check a matching video to recommend. Figure 3.2 presents the sequence diagram that show the different parts of the system to deliver the service.

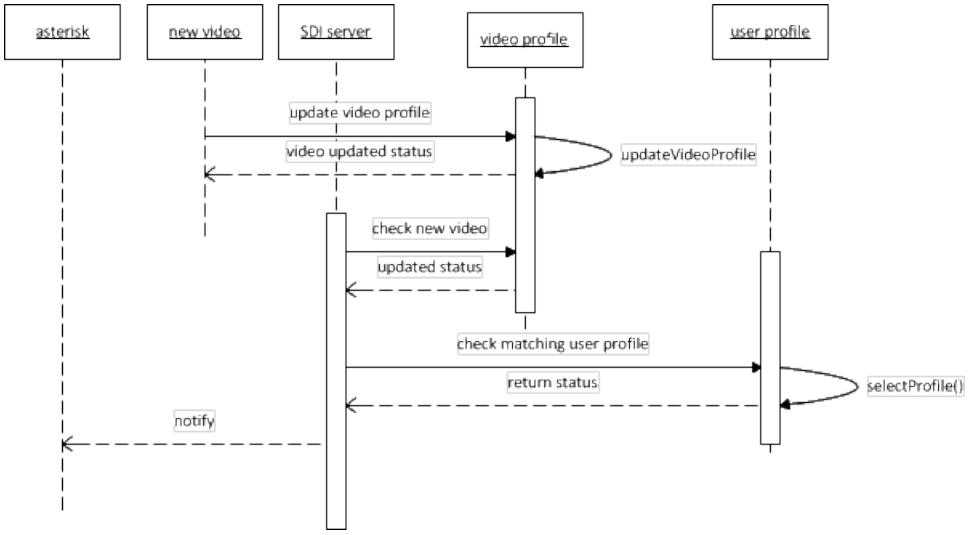
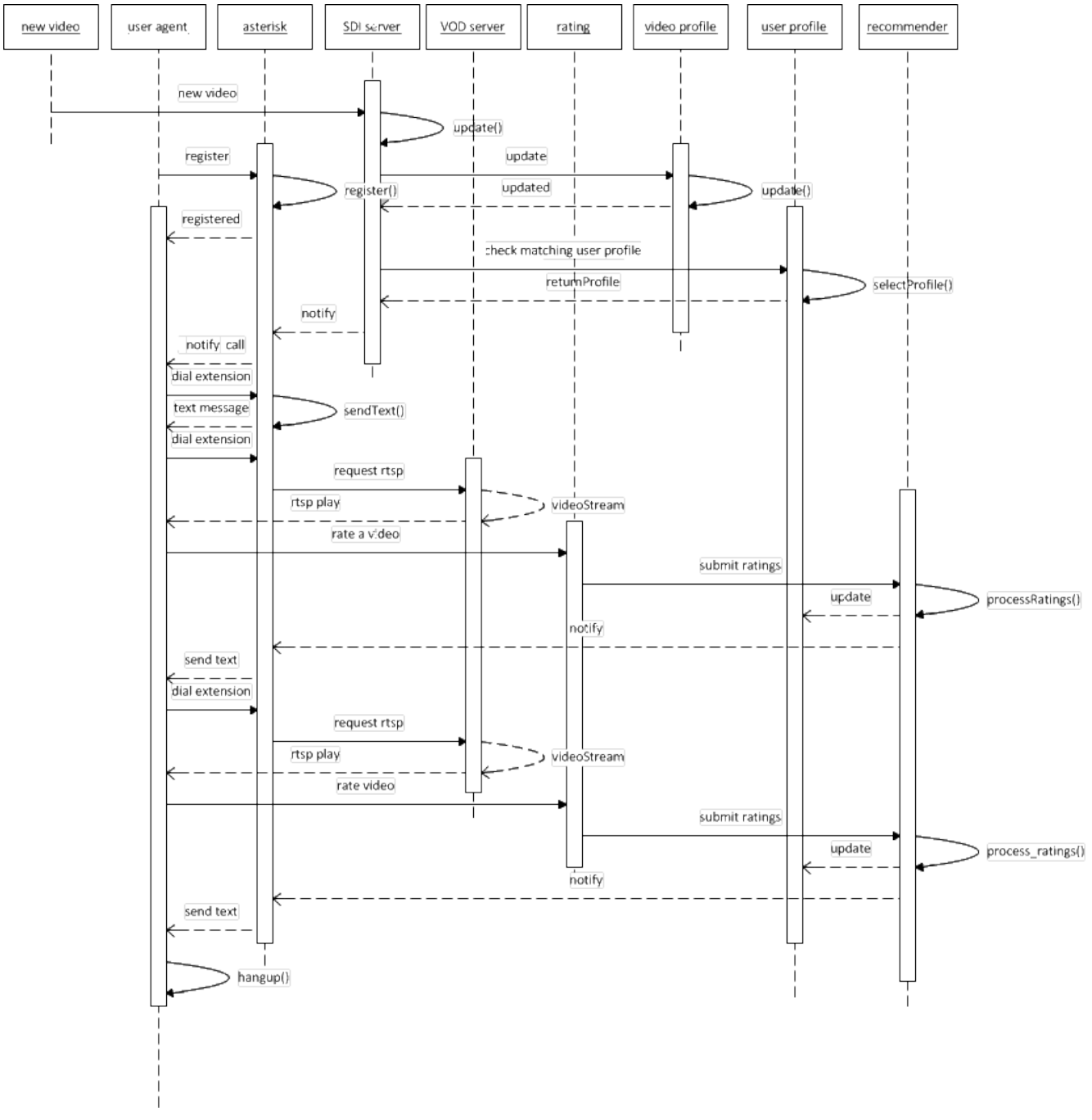



Figure 3.1: Sequence diagram for SDI based recommendation of new video



Similar to the problem with iVideoMail, one problem we observed with this service is the non-availability of support for automatic notification on device bootup. This is especially good for a user who is not online when a new video comes in. If Asterisk had supported this, we could have developed a true SDI notification system that provides notification to the user about the availability of a new video when he registers his device.

## 3.3 Streaming Service for SIP User Agents

Most of the current SIP UAs do not support the RTSP protocol, which is important to access a streaming service. In this section we present a system that we developed to allow SIP UAs without RTSP capability to access streaming service. This is especially important because one of the reasons for the introduction of NGN/IMS is giving users access to all their services from anywhere and using any device. To achieve this we developed an AS that has RTSP functionality and able to initiate streaming session on behalf of the SIP UAs.

As mentioned in Chapter 2, only the Live555 stream server supports the use of `destination` parameter of the RTSP protocol that is important for this service. However, Live555 only plays video files that are encoded with the MPEG video codec, and if we want to use Live555 as our streaming server, then we have to convert the video files that we want to stream into a file that can contain `mpeg` encoded video, for example, a *mov* file. At the time of the experiment, our investigation of open source SIP UAs for the support of the above mentioned codec led us to the realization that only the SIP-Communicator UA supported this codec. As a result, we used this UA to test this service. This was a major challenge to develop the service. The following sections present how this service is developed.

### 3.3.1 Architecture and Components of the System

The system architecture for this service is shown in Figure 3.3.

The system has two components, one that handles the SIP session and the other that handles the media session. The AS is developed to serve as a SIP terminating UA. The component that handles the media session is known *as Streaming Server Controller (SSC)*. Basically, the SSC contains an embedded RTSP client. For simplicity, we included the SSC into the SIP AS.

The following sections present the different components of the system.

#### 3.3.1.1 The SIP Terminating Application Server

After we realized the limitations of Asterisk, we moved to the other service development environment that was being investigated by our research group, which is the Mobicents Service Development Platform (SDP). So, the SIP terminating application server is implemented using Mobicents JAIN SLEE. Services in Mobicents are developed in the form of a Service Building Block (SBB) , and SLEE is a container for SBBs.As a result, we created an SBB called `StreamingSeviceSBB` to handle the basic SIP events necessary for this application. The service is developed in such a way that users send a SIP INVITE request to initiate the service by including an address that relates to the channel number which is also associated with a VoD file.

When an INVITE message comes to the AS the first time, it checks a look-up table to extract the URL related to the requested channel. The AS then supplies the SDP of the INVITE message and the URL that relates to the requested channel to the SSC by initiating the SSC's `createMediaSession` method. The method use the information to create the RTSP DESCRIBE command and send it to the streaming server. When the SSC gets the response from the streaming server, it forwards the track information of the media session to the AS. The AS then uses this information to form an SDP that it is going to attach to the SIP OK message and sends the OK message as a response to the INVITE request to the client. When the AS gets an ACK from the client, it requests the streaming server to start playing the media using the `startMediaSession` method of the SSC and the user starts to get the media immediately.

SBBs has a life-cycle method called `setSBBContext`, which is run by SLEE when the SBB is initially loaded into SLEE. The `setSBBContext` method is where we defined global variables that the SBB needs for its task. We also created an object of the SSC and retrieve and store the SLEE's profile table facilities into `profileFacility` variable for the purpose of storing the channel information in this method. We used a simple XML file to store the different channels and associated URLs as shown in Listing 10.

#### 3.3.1.2 The Streaming Server Controller

The SSC handles all communications with the streaming server. When the AS initiates the `createMediaSession` request, the SSC contacts the streaming server to creates a

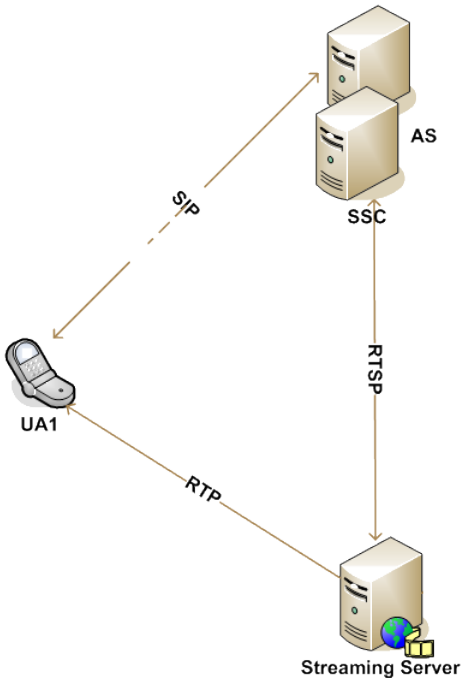



Figure 3.3: Accessing streaming session from a SIP UA

**Listing 10** Sample channel definition

```
<?xml version="1.0" encoding="UTF"?>
<channels>
   <channel>
      <name>Channel1</name>
      <url>rtsp://146.231.123.97:5554/channel1 </url>
   </channel>
      <name>Channel2</name>
      <url>rtsp://146.231.123.97:5554/channel2 </url>
   </channel>
</channels>
```

streaming session and during the session setup stage, it puts the address of the SIP UA in the `destination` parameter when sending the SETUP request to the streaming server.

The SSC has different methods for sending requests and handling responses for all the basic RTSP commands. Because the second SETUP request contains the *session id* that comes from the server, it is handled differently. The SSC also generates a random sequence number to be used as a sequence number and puts it in the `CSeq` parameter when it sends the first request (DESCRIBE request) and then increments it by one for subsequent requests. The SSC uses another class, called the `SDPManager` to process SDP messages coming. The SDPManager class is explained below.

#### 3.3.1.3 The SDP Manager

For the purpose of processing SDP messages that come from the streaming server and also from the SIP UA, we have created a class called `SDPManager,` based on the Java API for SDP. This class is used to parse and also create SDP messages. We defined different methods to handle media attributes that are specific to streaming sessions and included them in the `SDPManager` class. For example, the `CheckForDstParamter` and `parseTrackInfo` are used to check if the `destination` parameter is received with the SETUP command and to get track information from the response of a DESCRIBE request respectively. We also defined `parseSessionInfo` method to get the *session id* of an RTSP session.

The `SDPManager` class also has other methods to get a list of matching codecs between the communicating parties (e.g. the SIP UA and the streaming sever). The code of the `SDPManager` class is included in the DVD accompanying this thesis.

On the other hand, for the purpose of representing streaming sessions, we defined a class called `SDPInfo`. This class consists of information related to both audio and video streams and provides methods like `getAFormats()` to get a list of audio formats and `getVFormats()` to get a list of video formats that come through the SDP. The `SDPInfo` class is attached in Appendix C.

### 3.3.2 Message Flow Between the Different Entities

Figure 3.4 shows the high-level interaction of the different entities (UE, AS and Streaming server) involved in the provision of a streaming service to SIP UA. Because we embedded the SSC inside the AS, the message flow between the AS and the SSC is just a method call. The bold pink line in Figure 3.4 shows the SIP message flow. As there are two

different sessions (SIP and RTSP), the AS should make sure that the session initiation requests for the two sessions are synchronized so that in case a response from one session is delayed, it would be able avoid the breakdown of the other session. As shown in Figure 3.4, the AS uses the SIP TRYING message to ensure this .

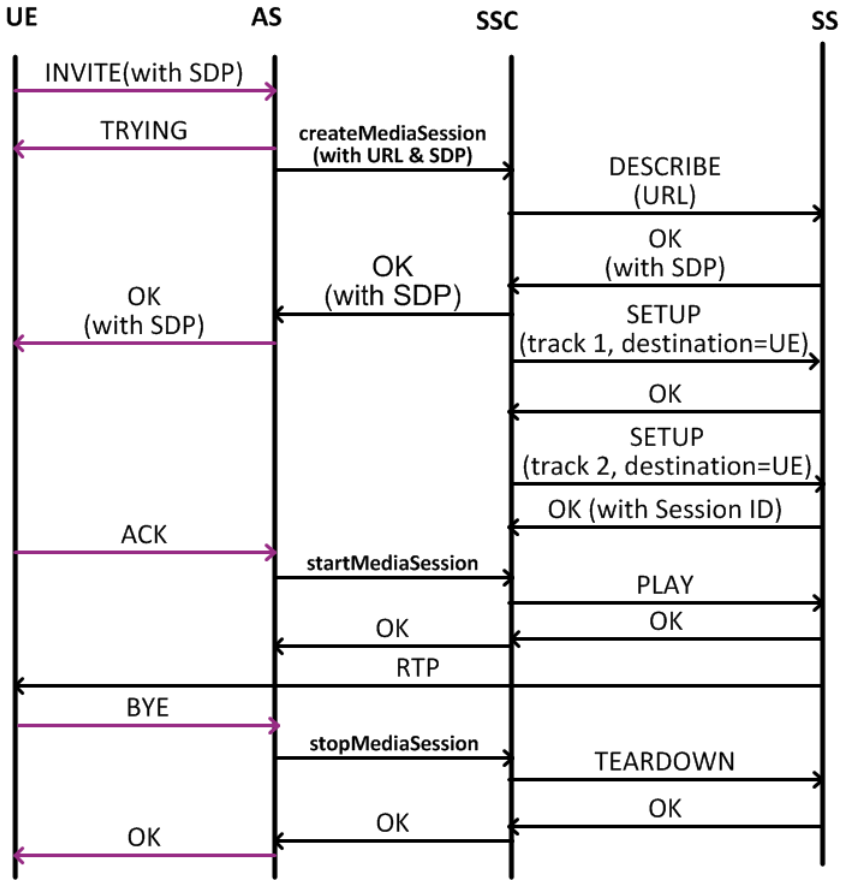



Figure 3.4: Message flow between UA, AS/SSC and streaming servers

### 3.3.3 Adding The Session Transfer Feature

Since a streaming session consists of two different sessions, the SIP and RTSP sessions, session transfer should involve the transfer of both sessions. To transfer the streaming session, we can use the SETUP command together with the `destination` parameter to transfer a streaming session to a different location (destination).

On the other hand, session transfer for a SIP-based VoIP system is mostly done using the SIP REFER method [27]. There are two different types of session transfer methods in SIP, which are *attended* and *unattended* session transfer methods. A good description

of these techniques is given in [26]. In our case we used the unattended session transfer using the SIP REFER method for the transfer of SIP sessions.

The main aspect we need to give attention to when developing this service is the decision of which session needs to be transferred first. The following section describes the different aspects of this service.

#### 3.3.3.1 Initiating the Session Transfer

Session transfer request can be sent in two different ways. We can send a session transfer request without specifying the destination device and the AS does the identification and transfer the session to another registered device of ours. We can also specify which specific device we want the session to be transferred to.

Because all multimedia sessions in IMS are set up using the SIP INVITE command, the session transfer should also be initiated with a SIP session transfer request. In our case, because the SIP-Communicator UA, that we used for testing the streaming service, does not support the SIP REFER method, we used the SIP Re-INVITE method to signal the session transfer request to the AS. A Re-INVITE message contains a tag in the `to` field of the INVITE command. For this purpose, the AS includes a method called `checkReinvite` to check if an INVITE is a new INVITE or Re-INVITE.

#### 3.3.3.2 The Stream Session Transfer Process

When the AS gets the Re-INVITE request, it does the following:

- instructs the SSC to pause the current streaming session,
- initiates a SIP session with the other user agent, and
- instructs the SSC to change the direction of the stream to the new UA.

The last two tasks are not mutually exclusive and can be done in parallel. Figure 3.5 shows the flow diagram of the streaming session transfer request.

#### 3.3.3.3 Locating the Transfer Device and Modifying the Media Session

The Mobicents SDP comes with SIP services like Proxy and Registrar in addition to various examples that we can use. We also relied on the Registrar and Proxy services for

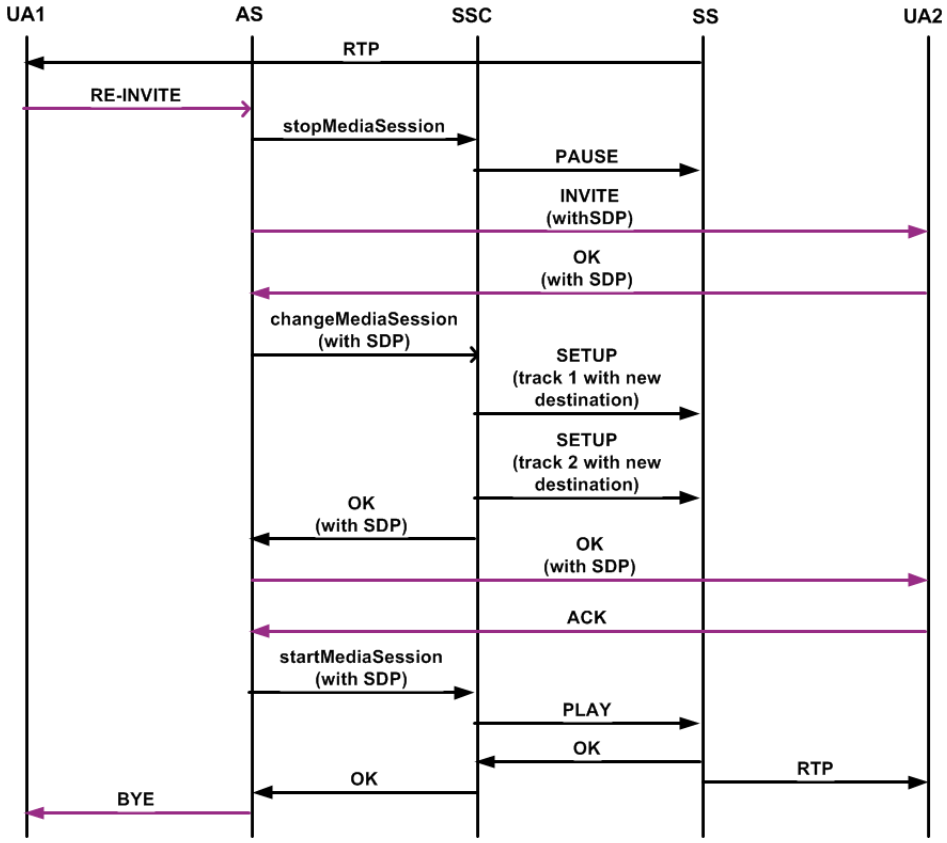



Figure 3.5: Message flow for streaming session transfer

identifying the destination device. The Mobicents Registrar service stores bindings in the form of mapping: `AOR - {ContactAddress-BindingData,...}`, where the AOR returns all registered devices of a particular user. So, the AS checks the entries in the Registrar (`locationSBB`) for the caller's Address of Records (AORs). The AS particularly searches the list for a different device from the one that the user used to initiate the service. The current device address can be found in the `contact` header field of the SIP INVITE request. The code snippet in Listing 11 shows how the registration entries are queried.

Once the AS gets the proper address of the user's devices, then it creates a SIP INVITE request and send it through the Proxy to the user's other devices. Because the device that is going to accept the transfer starts a media session with the streaming server, the AS attaches the SDP information of the streaming server in the INVITE request.

Once the transferee responds to the SIP session transfer request, the AS initiates the media session transfer procedure, by invoking the `mofidyMediaSession` method of the SSC. The subsequent communication between the transferee, the AS and the streaming server to setup the media session with the transferee is similar to the one mentioned in

**Listing 11** Accessing the registration entries

```
private URI isUserAvailable(URI uri) throws
    SipSendErrorResponseException {
String addressOfRecord = uri.toString();
URI target = null;
Map bindings = null;

try {
        bindings = getLocationSbb().getBindings(addressOfRecord);
} catch (LocationServiceException e) {
        log.error(e.getMessage(), e);
} catch (TransactionRequiredLocalException e) {
        log.error(e.getMessage(), e);
} catch (SLEEException e) {
        log.error(e.getMessage(), e);
} catch (CreateException e) {
        log.error(e.getMessage(), e);
}

if (bindings != null & !bindings.isEmpty()) {
        Iterator it = bindings.values().iterator();
        while (it.hasNext()) {
                RegistrationBinding binding = (RegistrationBinding) it.
                   next();
                ContactHeader header = null;
                try {
                        header = getHeaderFactory().createContactHeader(
                            getAddressFactory().createAddress(binding.
                               getContactAddress()));
                } catch (ParseException e) {
                  log.error(e.getMessage(), e);
                }

                if (header == null) { // entry expired
                        continue;
        }

        Address na = header.getAddress();
        target = na.getURI();
        break;
}

if (target == null) {
        throw new SipSendErrorResponseException("User temporarily
           unavailable", Response.TEMPORARILY_UNAVAILABLE);
}}

return target;
}
```

the previous section. After the media starts to play on the new device, the AS then sends a SIP BYE message to the device that initiated the session transfer. Figure 3.6 shows the architecture for this service.

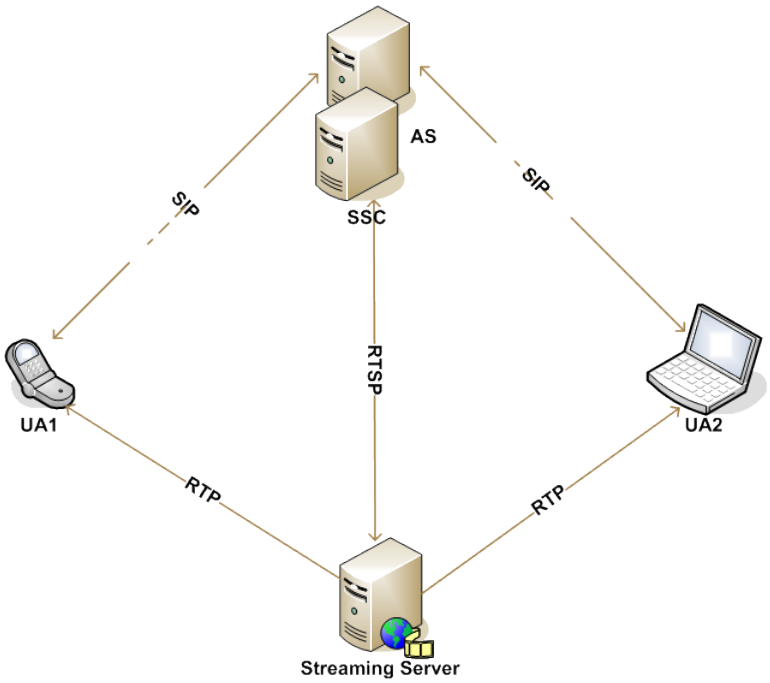



Figure 3.6: Architecture of streaming session transfer service

The identification of a user's devices that are online is a feature that different services may require and it is something that needs to be considered as an enabler for the envisaged environment. The part that synchronizes the SIP and RTSP sessions can also be developed as one of the enabler for streaming related services.

## 3.4 Summary

In this chapter we have presented our work related to basic video service development. These include iVideoMail, the videomail service for iLanga and SDI based recommendation system for Asterisk. The chapter also discussed streaming services that we developed for SIP UAs without RTSP capability.

To develop the iVideoMail service, the first step we took was to identify the basic requirements of a full-fledged videomail service. The work demonstrated how we can create a videomail delivery system that utilizes different technologies to deliver video messages.

But, it also demonstrated how important is dynamic notification for video-oriented services.

The discussion related to the streaming service surfaced issues related to RTSP and open source streaming server implementations. We have seen, for example, the issues surrounding streaming session transfer.

The work we presented in this chapter allowed us to make major conclusions. The development of both the iVideomail and SDI based notification services showed the lack of support from Asterisk for developing services related dynamic notification. There are also other video-oriented services (like converged services) that Asterisk cannot support. This shows the need for an open and extensive service development and delivery environment. In general, our investigation of the different technologies in the area of NGN/IMS for the supporting the development of video-oriented services led us to the conclusion that IMS and IPTV are good options. The next chapter presents our background information related to these two architectures: IMS and IPTV.

# Chapter 4

# The IP Multimedia Subsystem and IPTV as a Basic Video-Oriented Service Delivery Infrastructure for Telcos

This thesis presents our work related to the development of an easy to use open source-based video-oriented service development and deployment environment for a Telecom environment that adheres to open standards. As mentioned in Chapter 1, we considered IMS as a reference implementation of NGN. Therefore, this chapter is dedicated to giving background information about IMS and how it supports video-oriented service delivery. The chapter discusses the architecture of IMS and the features exposed by the architecture to service developers. Service delivery always involves the identification of users. So, another discussion area of the chapter is on user identities in IMS.

One particular architecture that is proposed for video-oriented service development for IMS is IPTV. We also chose the IPTV architecture as a basis for our video service delivery environment and this choice is well motivated in this chapter. The chapter presents the current state in IPTV service development, standardization efforts, development platforms for IPTV service and issues related to the development of IPTV services.

## 4.1 Overview of the IP Multimedia Subsystem (IMS) Platform

The flexibility, power and popularity of IP has attracted the interest of different standards development organizations and there are many standard development organizations

(SDOs) that produce standards for the different communication platforms available for service providers. Because IMS is developed to meet the communication requirements of different governments and international organizations in the telecommunications environment, various stakeholders are involved in the standardization of its specifications. We start our discussion by presenting the standardization efforts for the IMS platform.

### 4.1.1 IMS Standardization Efforts

The IMS was originally defined by the 3rd Generation Partnership Project (3GPP) [64] for mobile networks. The 3GPP project, itself, was initiated by a group of telecommunications standards bodies, the major one being the TISPAN, which is the technical committee of the European Telecommunication Standards Institute (ETSI) [65]. ETSI is the organization that is behind the definition and standardization of the Global System for Mobile Communications (GSM) and General Packet Radio Service (GPRS) mobile network architectures. TISPAN was formed by ETSI with the goal of standardizing NGN for fixed network access based on IMS. A good discussion of IMS standard bodies is given in [17]. Because of its popularity, we also consider the standard of IMS developed by 3GPP.

The 3GPP takes advantage of the standards of IETF. When an Internet protocol, developed by IETF, lacks some essential functionality for its use, 3GPP submit such requirements to the IETF for amendment of standards and IETF produces a different RFC for those requirements. An example of such a process is the RFC 4083 [66] that documents 3GPP's requirements for extensions to the SIP protocol [66].

The 3GPP publishes standard documents as *releases*. Since its inception, TISPAN and 3GPP have produced various releases of the IMS specification. A modified version of IMS architecture, called *Common IMS*, was defined in the recent release of 3GPP (Release 8), in collaboration with 3GPP2[1], and the industry also seems to have settled in using this version of the IMS standard. We also consider Release 8 in this thesis. There are two basic concepts in any IMS network: the Home and Visited Networks. We first discuss about these concepts before looking at the IMS architecture.

[1]The 3GPP2 is another standard body particularly established to produce standards for 3G technology based on CDMA2000 (Code Division Multiple Access 2000) and other related American National Standards Institute (ANSI) standards.

### 4.1.2 The Home and Visited Networks

IMS makes sure that users are able to access all their services even when roaming. For this purpose, IMS defines *home network* to refer to the user's operator network and the other networks that IMS users get their service from while roaming are called *visited networks*. Figure 4.1 shows the distinction between a home and visited network. The home network is the core network supporting the IMS services that hold the IMS subscription and the visited network is the network currently providing the user with connectivity to his IMS services.

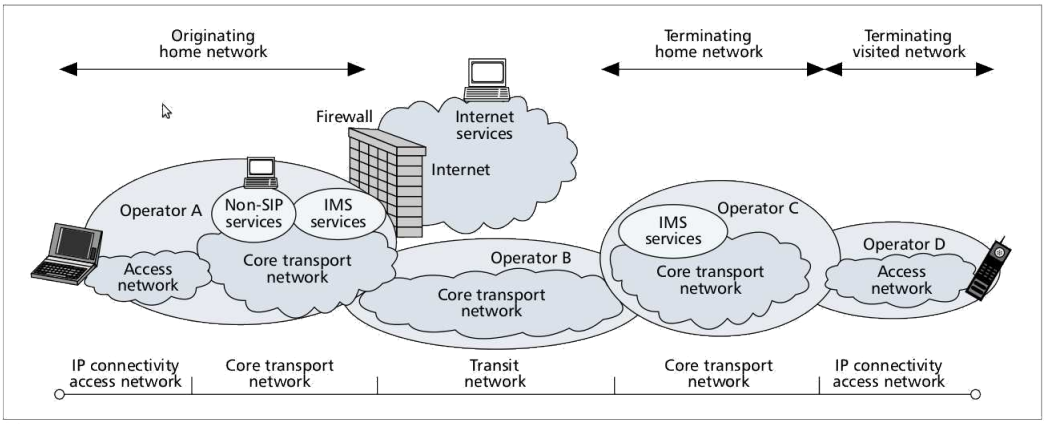



Figure 4.1: Network partitioning in IMS. Source: [7]

The following section presents the IMS architecture.

### 4.1.3 The IMS Architecture

The 3GPP has developed a list of requirements that IMS should provide solutions to. Nevertheless, in terms of architecture, 3GPP does not standardize nodes, rather it standardizes functions [13]. This implies that the IMS architecture is basically a collection of functions linked by standardized interfaces. Implementers are free to combine different functions into a single node (e.g., into a single physical box). They can also split a single function into two or more nodes. This is where the work reported in this thesis becomes important.

In general, IMS was built on top of the following basic architectural requirements:

- IP Multimedia Sessions – IMS is defined as a multimedia delivery platform.
- IP connectivity – User devices have to have IP connectivity to access the services deployed in the IMS platform.

- Ensuring QoS – IMS is designed to ensure an end-to-end QoS.

In IMS, the UE negotiates its capabilities and expresses its QoS requirements during session setup or session modification procedures. This is done through SIP and its accompanying protocol, the SDP.

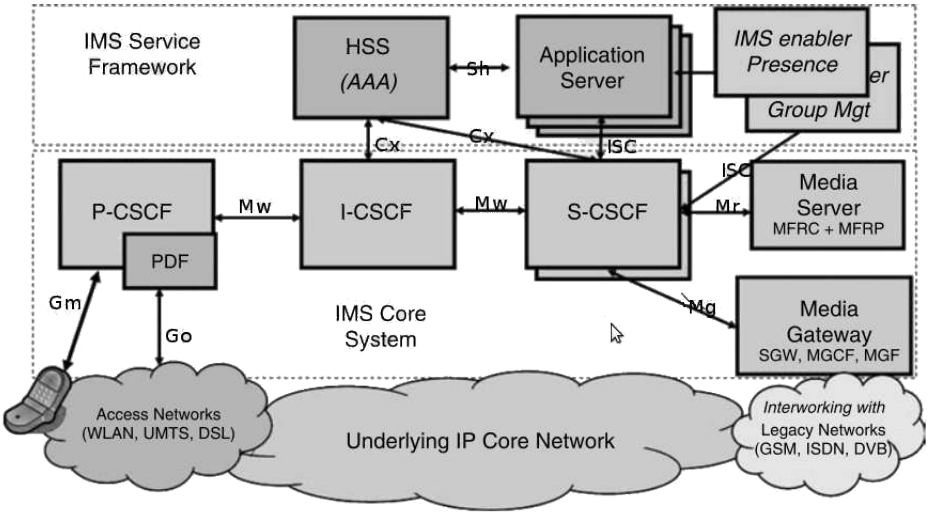



Figure 4.2: IMS architecture. Source: [67]

Using the above basic requirements, the 3GPP has introduced an IMS reference architecture that is widely adopted. Figure 4.2 presents this architecture with the most important components together with the interfaces they use.

### 4.1.4 Key Functions of the IMS architecture and Their Interfaces

As can be seen from Figure 4.2, the IMS architecture has three distinct and separate layers, the Application layer, control layer and the transport layer. In the middle is, the control layer, also called the IMS core, which basically consists of the different Call Service Control Functions (CSCFs) and the Home Subscriber Server (HSS). Some authors put the Media Function Resource Controller (MFRC) and Media Function Resource Processor (MFRP) in a separate layer between the IMS core and the service layer and call it Media layer. In the IMS architectural diagram shown in Figure 4.2, the mobile phone presented on the left bottom corner represents another important component called the User Equipment (UE). The UE can be any device with IMS client installed on it.

As mentioned before IMS does not define nodes but rather functions. In general, as mentioned in [68], the IMS key functionalities can be roughly classified into the following six main categories .

- Session management and routing family (CSCFs);
- Databases (HSS, SLF);
- Services (application server, MRFC, MRFP);
- Interworking functions (BGCF, MGCF, IMS-MGW, SGW);
- Support functions (PCRF, SEG, IBCF, TrGW, LRF);
- Charging.

The interfaces between the different components of the IMS architecture presented in Figure 4.2 indirectly represent the protocols used in those interfaces. As mentioned above, the 3GPP adopted SIP as a signaling protocol and defined new headers for using it in IMS. The newly defined headers include `P-Asserted-Identity, P-Access-Network-Info`, and `P-Visited-Network-Id` among others [69]. The `P-Asserted-Identity` header permits the SIP AS to know that the user was authenticated by the IMS core network, and `P-Access-Network-Info` provides the control application with information about the access technology being used by the client (e.g. xDSL, UMTS, WiFi). This is especially good for video services, because the AS needs to know the kind of connection that the user is connected to in order to provide an appropriate content type. The same header may also include information about the location of the user (e.g. a cell ID). This may help the application decide, e.g. which content server is optimal to deliver the content and also to provide location-based services (like adverts suitable for the location where the user is accessing his service). There are also other headers like the `P-Visited-Network-Id` that provide the name of the network into which the user might be roaming.

Different components of the IMS architecture use SIP to communicate with each other. Table 4.1 shows which of the IMS interfaces use SIP and also which ones use other protocols.

| IMS reference point | Protocol Used | Remark |
|---|---|---|
| ISC, Gm, Mw, Ma, Mr | SIP | S-CSCF <-> AS, UE <-> P-CSCF, (P-CSCF, I-CSCF, S-CSCF), I-CSCF <-> AS and S-CSCF, MRFC |
| Sh, Cx, Dx | Diameter | AS <-> HSS, (I-CSCF, S-CSCF), HSS, and (I-CSCF or S-CSCF) <-> SLF |
| Ut | HTTP, XCAP | UE <-> AS |
| Mp | H.248 | MRFC <-> MRFP, IPTV uses a different media control protocol |

Table 4.1: Protocols used by IMS reference points

Diameter is another protocol used in IMS. For this purpose 3GPP also defines a Diameter application to extend the base Diameter protocol to be used for the `Sh` interface and specified in [47]. The application defined by 3GPP is called the *Diameter Application for Sh*.

The Sh interface is the standard interface that ASs use to interact with the HSS. The The HSS is the central repository or database for user-related subscription information that the other nodes of IMS use for handling of calls and sessions. ASs use the Sh interface for two purposes: one is to query or update a user's data stored on HSS, and another is to subscribe to and receive notifications when a user's data changes at the HSS. With this feature an AS can subscribe to get notification when the user's registration status changes to provide services like videomail notification. This is a basic requirement for creating personalized video services and we also discuss other use cases related to this in Chapter 6.

There are also other interfaces in the IMS architecture that uses the Diameter protocol, which include the `Cx, Dh, Dx, Rf, and Ro` interfaces. These interfaces utilize different Diameter applications.

Generally speaking, as mentioned in [67], the central session control protocols in IMS are SIP and Diameter. An IMS application server should support at least these two protocols in order to provide a proper service in IMS.

From the point of view of video services and the scope of our work, we present the main components of the IMS architecture in the following sections.

#### 4.1.4.1 The Call Session Control Function (CSCF)

The CSCF, which represent three different components, is considered as the core of IMS. IMS uses SIP as signaling protocol and these components are SIP-based servers and process SIP messages. According to [70], the CSCF is responsible for establishing, monitoring and releasing multimedia sessions. It also manages the user's service interaction. Based on the functionality they provide, there are three types of CSCF: Serving-CSCF (S-CSCF), Proxy-CSCF (P-CSCF) and Interrogating-CSCF (I-CSCF). Detailed explanations of these servers are presented in [13]. Because of its importance, we present the S-CSCF in the following section.

**The S-CSCF** The S-CSCF is a proxy server that controls the entire communication session of a user and is responsible for handling registration requests, making routing decisions for SIP messages and maintaining session states and also for storing the service profile(s). In fact, it is this unit that initiates IMS services for users by forwarding their requests to ASs. To achieve this, the S-CSCF, upon receiving a SIP request, checks if an application server is to be contacted before sending the request to its destination. The decision is made based on the user's service preference.

A user's service preference, also called *a service profile*, is configuration information that helps the S-CSCF to decide when and, in particular, which application server to contact when a user sends a SIP request or a SIP request comes to the user from another place. The S-CSCF, using the Diameter protocol, downloads the service profiles associated with a particular public user identity from the HSS when the user is registered in IMS. Service profiles, and private and public identities of users, are described later in this chapter. For this purpose, the S-CSCF intercepts all UE-originated and UE-terminated session initiation requests and session transactions [68].

The S-CSCF also serves as a SIP Registrar Server and maintains binding information between the user location (i.e., the IP address of the terminal the user is logged on) and the user's SIP address of record. In addition to this, the S-CSCF is used to enforce policies of the network operator, like prohibiting an unauthorized user from establishing certain types of session. ASs need to utilize these information to provide the required service.

In general, The S-CSCF is always located in the home network.

#### 4.1.4.2 Databases

One of the main advantages of IMS is the provision of personalized service to users through different access networks. To achieves this, IMS stores user subscription and service-related data in a central location, called HSS. The main data stored in the HSS include user identities, registration information, access parameters, service-triggering information and authentication keys that other nodes use for handling calls and sessions[70]. For example, ASs can query the HSS to get registration status of users.

IMS also supports other types of databases. The IPTV architecture, for example, includes another database called the UPSF for the purpose of user profile management for IPTV services. This is discussed in detail later in this chapter.

#### 4.1.4.3 Application Servers

In IMS, ASs are SIP-based servers and are defined in the TS 23.218 standard document [71]. IMS ASs provide all functionality that basic SIP ASs provide.

The 3GPP does not standardize how an AS should be programmed, rather it only standardizes the signaling and administration interfaces that ASs use. The AS is defined as having interfaces with S-CSCF and HSS. It uses the IMS Service Control (ISC) interface to communicate with S-CSCF using the SIP protocol, and the Sh interface to communi-

cate with HSS using the Diameter protocol. So an AS should make use of at least these two interfaces to qualify as an IMS AS.

While executing the service logic, an AS may also communicate with the HSS to get additional information about a subscriber or to learn about changes in the subscriber's profile [13].

An AS can originate, terminate or transit a SIP request. Because value-added services are usually session stateful and feature capabilities that go beyond basic call proxying, most of the time, ASs are developed based on B2BUA frameworks. ASs can be located either in the operator's network or elsewhere in a third party network.

#### 4.1.4.4 Media Resource Function (MRF)

In traditional telecommunication networks (circuit-switched networks), service tones and recordings are all provided via the switch and other external elements (such as voice response units). However, in NGN/IMS, we do not have switches, and media-related services are provided with a component known as the Media Resource Function. The MRF provides the home network with the ability to play announcements, mix media streams (e.g., in a centralized conference bridge), transcode between different codecs, obtain statistics, etc. [13]. The MRF is subdivided into the MRFC (MRF Controller) and MRFP (MRF Processor). The MRF, however, does not provide streaming and broadcasting services and IPTV is defined for this purpose. IPTV uses a different type of media resource function called the Media Function and this is discussed later in this chapter.

## 4.2 User Identification and Service Initiation in IMS

As mentioned before, the decision to invoke which AS and when is determined through service triggering information and is related to the initial SIP request. The S-CSCF gets service triggering information in the form of initial filter criteria from HSS.

Defining the set of services to provide to users is the operator's task and it is not the concern of this thesis. However, since service development requires a good knowledge about user identities and filter criteria, the next sections discuss about identifications in IMS and how filter criteria are defined and used.

### 4.2.1 Identification in IMS

In IMS, users are identified with a user identity and not related to a specific line or device. Not surprisingly, services are also not tied with certain server and they have identifications that allows them to change their location at different times. The following sections present the user and service identities.

#### 4.2.1.1 User Identities

In IMS, user identities are of two types: private or public user identity. The private user identity, also called *IP Multimedia Private Identity (IMPI)*, is a unique user identity that is assigned by the network operator and is used for registration and authorization purposes, while the public user identity, also called *IP Multimedia Public Identity (IMPU)*, is the identity that other people use for requesting communication with the user [68]. Users may not know their Private User Identity because it might be stored in a smart card, in the same way as an International Mobile Subscriber Identifier (IMSI) is stored in a SIM (Subscriber Identity Module) card [13].

On the other hand, an IMS user can have one or more public user identities. A Public User Identity can take the form of either a SIP URI (e.g. *sip:zelalem@convergence.ru.ac.za*) or a TEL URI (*tel:+27-234-567-890*). The fact that IMS supports multiple access technologies makes it possible that an IMPU can concurrently be shared between multiple fixed and mobile devices and the possibility to concurrently assign an IMPU to multiple devices is one of the interesting features of IMS.

The idea of having different identities for a given user is very good especially for video services, like IPTV, because TVs are communal devices that many users can use to watch programs. So, for example, for the purpose of parental control, the different users who use the same TV in a household may use different public user identities, like *family* to refer the entire family and *dad* to refer to the super user (father). The IPTV set-top box (STB) can be configured to use the default user that represents the household (for example, *sip:familyOfXYZ@specialtv.com*) to log into the IMS domain. It can also be configured to make the credential of a personal user (i.e., *sip:dad.familyOfXYZ@specialtv.com*) allowing access to all the services that are restricted to children. IPTV has its own user data stored at different places and we discuss them later in this chapter.

The HSS, as a main database for all data related to a subscriber, stores the Private User Identity and all the Public User Identities allocated to a user.

#### 4.2.1.2 Service Identities

Similar to IMPU, the 3GPP (in its Release 6) also introduced a new type of identity called *Public Service Identity (PSI)* which is an identity assigned to a service hosted in an Application Server. Similar to IMPUs, PSIs can be identified using a SIP URI or a TEL URI and they are also stored in HSS. However, unlike user identities, PSIs can be stored either as a distinct PSI or using a wild card. A distinct PSI contains the PSI that is used in routing. However, a wild carded PSI represents a collection of PSIs (services).

The next section describes how IMS service initiation is done using filter criteria.

### 4.2.2 IMS User Profile and Service Triggering

#### 4.2.2.1 Introduction

An IMS user profile is a profile that is related to a user through his private and public user identities and also stores the different service profiles related to the user. Basically a user profile is organized as presented in Figure 4.3 is the corner stone for service provisioning. As can be seen from the figure, another important aspect that service providers consider are service profiles. A service profile store information related to a particular service, which includes the user's public identifications that can access that particular service and zero or more initial and shared initial filter criteria. The shared initial filter criteria are filter criteria used by different users but they are unique to each network operator. The initial filter criteria (also called *ifc*), however, are developed based on the template defined by the 3GPP and can work with any network operator. In general, service provisioning in IMS follows the following three fundamental steps [68]:

1. Defining possible service or service sets;
2. When a user subscribes to a service or modifies his subscription, creating user-specific service data in the form of initial filter criteria related to the service;
3. Passing an incoming initial request to an AS that contains the service logic.

The next section presents how initial filter criteria are defined and used in IMS.

#### 4.2.2.2 The Initial Filter Criteria (ifc)

Initial filter criteria are criteria that the S-CSCF uses to check if a user's request warrants the provision of a service that is specified in the criteria. It is a data structure that

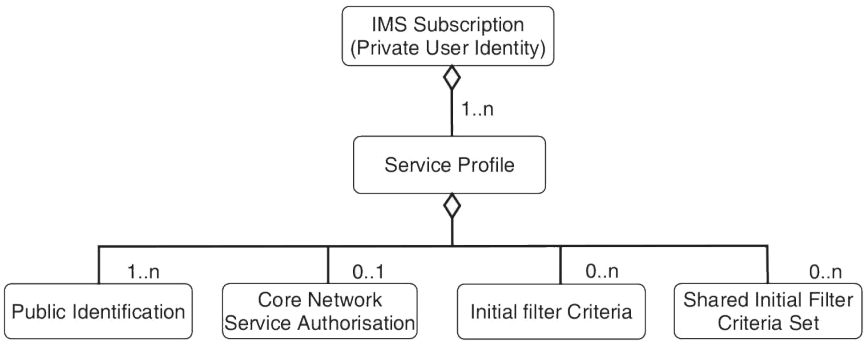



Figure 4.3: The structure of IMS user profile. Source: [68]

contains both the criteria and the action part (the AS to be invoked). Figure 4.4 shows the structure of initial filter criteria. As it can be seen from the figure, an ifc contain the following fields: a priority, zero or one trigger point that contains one or more service point triggers, and information about the AS that the S-CSCF needs to invoke if the criteria are met.

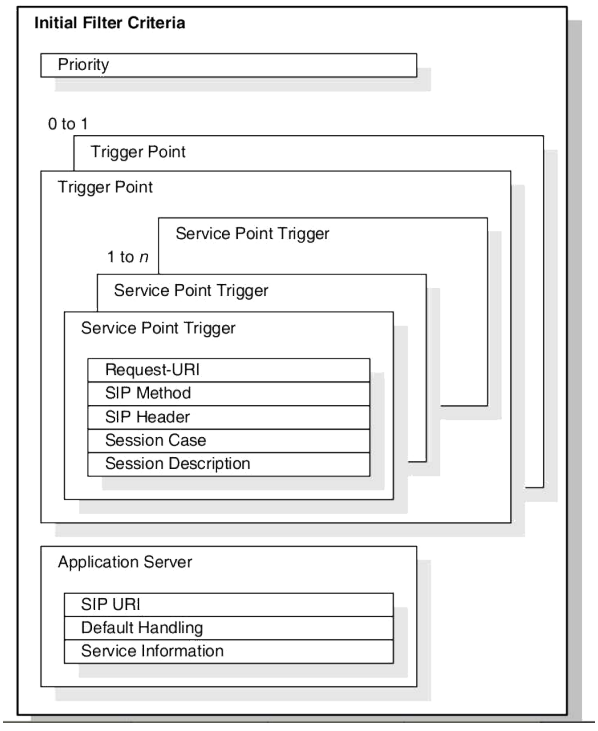



Figure 4.4: The structure of initial filter criteria. Source: [13]

The priority tells the order in which IMS assess these criteria compared to other filter criteria that are part of the same user profile. Accordingly, the S-CSCF choose the filter criteria with the highest priority first. We can have several ASs executed one after the

other based on the priorities.

Another important aspect of ifc is the service trigger point (STP). Service trigger points, as the core of ifc, are expressions composed of a collection of individual filters, also called *Service Point Triggers (SPT)*. STPs are created based on the different parts of the SIP request that needs to be checked for the purpose of service initiation. STPs may also relate to session cases, that is whether or not a service should be triggered if the session is an originating session or terminating session. It could also be directed to unregistered UE (also called *terminating unregistered*), and may also be based on a match to the incoming SDP.

Basically, STPs are used to check:

- The value of Request-URI
- The method of the SIP request (e.g., INVITE, OPTIONS, SUBSCRIBE, etc.)
- The presence or absence of any SIP header
- A partial or full match between the contents of any SIP header
- The session case (i.e., whether the SIP request is originated by the user that is being serviced, or addressed to the user who subscribed to the service). The served user also can either be registered or unregistered at the time the request arrives at the S-CSCF. Accordingly, we have four possible session cases, which are: Originating Registered, Terminating Registered, Originating Unregistered, or Terminating Unregistered
- The session description (i.e., any partial or full match on any SDP line).

Different STPs can also be combined by means of logical expressions (*AND, OR, NOT*) to form a complex service trigger point. For example, a Trigger Point can be used to express two STPs as: (`Method = INVITE`) AND (`Request-URI = sip:user@ruconvergence.com`). If no service trigger point is defined for a service profile then all SIP requests are unconditionally forwarded to the AS.

With regard to the type of requests that would be checked for STPs, only initial SIP requests are subject to treatment by filter criteria. These include REGISTER, INVITE, SUBSCRIBE, MESSAGE. This is one advantage we have over other service deployment frameworks like Asterisk. For example, we can develop a service that can be invoked when the user registers.

The last part in the ifc is the Application Server component and the "SIP URI" field relates to the address of the AS that is going to receive the SIP request if the conditions described in the service trigger point are met. Description about the other fields in this group can be seen from [68].

In general, the service configuration techniques discussed above are especially important to develop video services easily. For example, one could develop a videomail AS and choose the terminating unregistered session case for the initiation criteria of the AS, unlike to what we had done for iVideomail where we included a code in the AS to check if the user is not registered. The ifc can also check if the call is a video call for us by checking if the session includes video codecs in the SDP. This way the service developer focuses only on the logic of the service because it gets various support from the IMS core.

### 4.2.3 Video-Oriented Service in IMS

Because the standard IMS didn't include the popular video services like streaming and broadcasting, and considering the ever-increasing demand for these services on the Internet, the telecommunications industry defined and adopted IPTV as the next generation television [72]. In fact, IPTV is more than what its name implies and can be used to deliver a wide variety of video-oriented services. The following section discusses IPTV in general, and how we can use IPTV to develop a personalized video services in the IMS environment in particular.

## 4.3 The IP Television (IPTV) Technology

As a major stakeholder, ITU-T formed a focus group on IPTV, named ITU-T FG IPTV, and defined IPTV as "multimedia services such as television/ video/ audio/ text/ graphics/ data delivered over IP based networks managed to provide the required level of QoS/QoE, security, interactivity and reliability" [73]. The fact that IPTV provides QoS makes it a good choice for delivering video-oriented services because video is a medium that requires QoS. Since the Internet can not guarantee QoS, IPTV services are best delivered through Telecom networks, such as NGN.

As a result, he ITU-T FG IPTV decided that IPTV architecture shall be defined for both NGN and non-NGN environments, and within the NGN-approach, defined IPTV with both IMS and non-IMS approaches [73]. Other standard bodies that are involved in the standardization of IPTV are presented below.

### 4.3.1 IPTV Standardization Efforts

There are many standard organizations for IPTV each focusing on certain aspect of the technology. ETSI TISPAN is one of the main organizations behind the standardization of IPTV specifications. TISPAN has also defined standardized IPTV services. Other standard bodies include OIPF [74], DVB [75], ITU-T, and ATIS [72].

Because of the scope of the work reported in this thesis, the specifications that are important for us are the ones developed by TISPAN and OIPF. The following sections presents the IPTV standards developed by these standard bodies.

#### 4.3.1.1 The ETSI TISPAN IPTV Standards

ETSI TISPAN is a very active standard body on the development of IPTV standards. TISPAN produced standards both for IMS-based and non-IMS based IPTV. It has defined several specifications that are related to the requirements, architecture and services of IPTV. The group defined requirements for IMS-based IPTV in [76], its architecture in [77] and services in [78]. They introduced IPTV to the NGN architecture as one of IMS enabled services in their specification Release No 2 (2008). Release 3 [79] added new services by combining other IMS features like voice, data, presence and messaging into IPTV. From our assessment, we reached into conclusion that the list of requirements for IMS-based IPTV described in TISPAN's specification is not exhaustive. Because some of the requirements are also specified at a higher level and need detailed discussion, our first work for considering IPTV as a base architecture for the envisaged environment was to identify new requirements and develop a comprehensive IPTV-based video-oriented service requirement. We present this work in Chapter 6.

#### 4.3.1.2 The Open IPTV Forum (OIPF) IPTV Standards

The Open IPTV Forum [74] is a standard body that focuses on standardizing the user-to-network interface (UNI) of IPTV, both for a managed and a non-managed network. OIPF has standards on user identity management, user equipment profile and media handling that other standard bodies can adapt. In fact, ETSI TISPAN bases its user profile data structure on the ones defined by OIPF.

As specified in [80], "the ETSI and OIPF standards complement each other; where OIPF defines how the consumer electronics equipment should interact with the service, and does not delve too deeply into the network aspects, that is exactly what TISPAN defines, leaving the user equipment a mere footnote at the end of the service delivery chain".

### 4.3.2 Advantages of Using IPTV

There are many advantages of using IPTV. TV stations, be it cable TV providers or traditional terrestrial and satellite signal providers, have limitations as to the number of TV channels that they can provide, which is dictated by the range of frequencies available for broadcasting. This is so because Broadcast networks feed all channels to users at the same time and this is what limits traditional TV's from providing unlimited number of channels. IPTV networks, however, transmits only the channel that the user wants to watch and there is no limit as to the number of channels we can have. This is a very big advantage especially as high definition video become more and more common.

Another interesting aspect of IPTV is its interactive nature. As mentioned in [81], interactive IPTV has essentially shifted TV watching from a completely passive activity to an active one - introducing lean-forward services into a traditional lean-back environment.

One big advantage of IP-based networks in general, and IMS in particular, is the opportunity for integrating different types of services or what we call converged services. In this regard, the advantage of using the IMS-based IPTV architecture to develop a converged service is mentioned in various articles including [82, 83, 84, 85, 86]. Some of the advantages presented in these papers include single sign-on, user subscription management, unified charging and service personalization. In fact, services on IMS can be easily blended with other communication services to bring up new and innovative services. We can have a service that displays the caller-id of a call that comes to a user while he is watching TV. Users can also check out items that are being advertised on their TV directly from their TV, as presented in [87]. In general, as mentioned in [80], the resilience and ubiquity of the IP infrastructure is what makes IPTV so suitable for video delivery in IP networks.

These are some of the reasons why we also considered the IPTV architecture as a basis for our video-oriented service development environment.

### 4.3.3 The ETSI-TISPAN IPTV Architecture

In this section we present the functional architecture of IMS-based IPTV as specified by ETSI TISPAN and discuss the main functions of its components and the reference points that the components use. Most of the time IMS components are presented as a black box and labeled as IMS core. Figure 4.5 shows the IPTV architecture as specified in [79].

The two rounded dotted boxes (added to the architecture for the purpose of emphasis) contain the main functional units of the IPTV architecture. The Core IMS contains the CSCFs we discussed in Section 4.1.4 and it provides functionality for authentication,

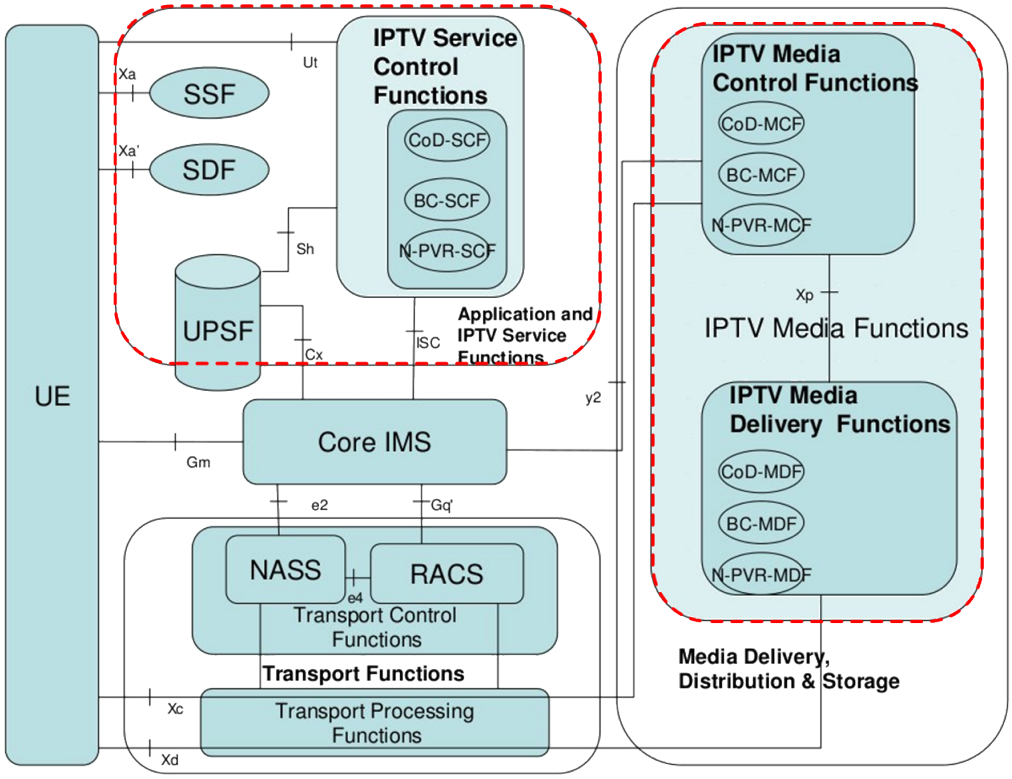



Figure 4.5: IPTV functional architecture. Source: [79]

authorization, and signaling for the setup of IPTV service and content delivery. The User Profile Service Function (UPSF) provides the same functionality as the HSS in the standard IMS architecture. The following sections provide description of the different IPTV specific components.

#### 4.3.3.1 The User Equipment (UE)

The UE represents different types of devices (Set-top boxes, PCs, mobile devices or smart phones) with the proper signaling and media processing capability. An IPTV UE has to support at least three interfaces, which are the Gm, the Xc, and Xd interfaces. The protocols used by these interfaces are RTSP, RTP and HTTP respectively. In addition, the UE may support the Ut interface to interact with SCF for the purpose of service profile configuration.

#### 4.3.3.2 The IPTV Service Control Functions (SCF)

The service control unit in the case of IPTV is called the IPTV *Service Control Function (SCF)*. This unit is divided into three similar units related to the three major services

of IPTV, which are Content on Demand (CoD), Broadcasting (BC) and Network-based Personalized Video Services (N-PVR). The SCFs related to these services are CoD-SCF, BC-SCF, and N-PVR-SCF and each of these SCFs are implemented as a SIP AS. For the purpose of delivering personalized service to users, the SCF may contact the UPSF using the Diameter protocol through the Sh interface.

#### 4.3.3.3 The Service Discovery and Selection Functions (SD&SF)

Service discovery, also called *service attachment*, is accomplished when the UE contacts the SDF through the Xa interface, to get information about the user's IPTV services so that the user can select a specific service. The information retrieved may refer to the address of the service server or portal that provide the user with a description of the available services. Once the UE obtains the information from the SDF, it contacts the SSF to retrieve relevant information about a specific IPTV service, like the URL of the media, to initiate the session. The UE uses the HTTP protocol to communicate with SSF through the Xa reference point. The SDF is implemented as a SIP AS.

#### 4.3.3.4 The IPTV Media Function (MF)

Similar to the standard IMS architecture, the IPTV specification also includes separate components (functions) for media delivery and control, which collectively are called the IPTV Media Function (MF). The control unit of the IPTV MF is MCF and the delivery unit is called the MDF. These functional units are responsible for the delivery of all type of media that the user subscribes to: live TV channels, CoD, or recorded video. The media control and delivery units for each of the basic IPTV services are CoD-MCF, BC-MCF, and N-PVR MCF for controlling media and CoD-MDF, BC-MDF, and N-PVR MDF for delivering media.

The MDF is usually implemented with a full-fledged media server, but for CoD services, streaming servers can be used. In fact, looking at the standard documents, we can see that RTSP is specified as media control protocol for the Xc interface, especially for streaming related services. But, various research has also been carried out to see if a different media control protocol can be used for the Xc reference point.

Some researchers suggested the use of SIP instead of RTSP, because they said it simplifies session management tasks. In this regard different IETF Internet drafts have been published [88, 89, 90] to demonstrate how SIP can be used as a media control protocol in this kind of context. Various researchers have also reported their experience with regard

to using SIP as a media control protocol. The authors in [91] showed how a new SIP header (called *SIP-MEX*) together with an XML document as an attachment to the SIP INFO message can be used to send media control commands to the MCF. On the other hand, Shiroor [92] suggested the integration of SIP and RTSP to create a comprehensive media control protocol.

We did not consider these proposals because an IPTV service is expected to implement trick modes, especially for CoD services, and SIP do not support these functionality, and to use SIP as a media control protocol would be like re-inventing the wheel. As a result, we stick to the RTSP protocol.

On the other hand, the protocol for the Xp reference point is not defined in the IPTV standard specification. In fact, it is presented as "not defined" in the *Summary of Protocols Used* section of the standard document [78]. But from the implicit discussions made in the specification, it seems that for CoD related services, the MDF and MCF communicates using the RTSP protocol. For example, the specification mentions that the RTSP ANNOUNCE method can be used as a means of notifying the events that are happening at the server side to the UE or MCF. Because of this and the required trick mode features mentioned above, most of the time researchers also use the RTSP protocol for the Xp interface[78, 93] and it has become a de-facto media control protocol especially for CoD services. This also implies that streaming servers are good options for use as MDF.

However, as mentioned before, implementation wise, there are still problems for using open source streaming servers as an MDF unit. Because the Streaming Server Controller that we presented in Chapter 3 still lacks some features that a proper MDF should have, we developed another proxy, called *Streaming Proxy and Relay Component* to be used together with streaming servers to serve as an MDF unit [50]. In addition to providing the required feedback to the MCF, the Proxy is also designed in such a way that it should be easy to include additional features to develop innovative IPTV services. The Proxy is a major contribution of this thesis and is discussed in detail in Chapter 5.

The following section presents the main IPTV services that are specified in the different IPTV standard documents.

### 4.3.4 Standard IPTV Services

We organizethe IPTV services specified in the ETSI-TISPAN standard documents as Basic IPTV and Personalized IPTV services.

#### 4.3.4.1 Basic IPTV Services

As mentioned before, the IPTV specification defined three major IPTV services: Content on Demand (CoD), Broadcasting (BC), and Network-based Personal Video Recorder (N-PVR), and we consider them as basic IPTV services. The following sections present these services in brief.

#### 4.3.4.2 Content on Demand (CoD)

Similar to the Video on Demand (VoD) service, the CoD service gives users the facility to use trick mode features, like fast forward, pause and resume. The UE can initiate a CoD service using the SIP INVITE request but uses the RTSP protocol for trick mode functionality.

With regard to the setting up of a CoD session, the specification offers two methods for media initiation, referred to as Method 1 and Method 2. The distinction relates to where the RTSP media session setup commands (specifically DESCRIBE and SETUP) are initiated from. In the Method 1 session setup technique, the session is initiated and the setup originates from the MCF. In Method 2, the session setup request is initiated by the UE.

**Broadcasting** This service is similar to the traditional TV broadcasting service, but in the case of IPTV, the user has more control over the service and can provide feedback and participate in voting. With IPTV, users can pause live broadcasts and create an effect of what is known as *time-shift* broadcast delivery through the use of the a Personalized Video Recorder (PVR).

**Network-based Personal Video Recorder** N-PVR is a service that allow users to record TV programs and watch them at their convenient time. Users can request content recording service from anywhere and can make the request even when they are off-line. The UE use a SIP MESSAGE to send a PVR content capture request. This component can be used to develop other video-oriented services that require video recording like videomail.

#### 4.3.4.3 Extending Basic IPTV Services

Because the IPTV architecture uses SIP as a signaling protocol, it is easy for service developers to come up with innovative services. For example, a service that allows users

to transfer an IPTV service into another device (also called *session transfer*) can be developed using the SIP session transfer feature.

As mentioned in one of the TISPAN's specification [76], the IPTV solution should also enable interaction/integration with Internet services (e.g. e-mail and web).

#### 4.3.4.4 Personalized IPTV Services

While the HSS data is quite sufficient for storing user profile for standard communications services, it is not sufficient for providing personalized IPTV services. For this purpose, TISPAN defined the data structure presented in Figure 4.6 and related to different service profiles for service developers to use. The following section presents the IPTV User Profile.

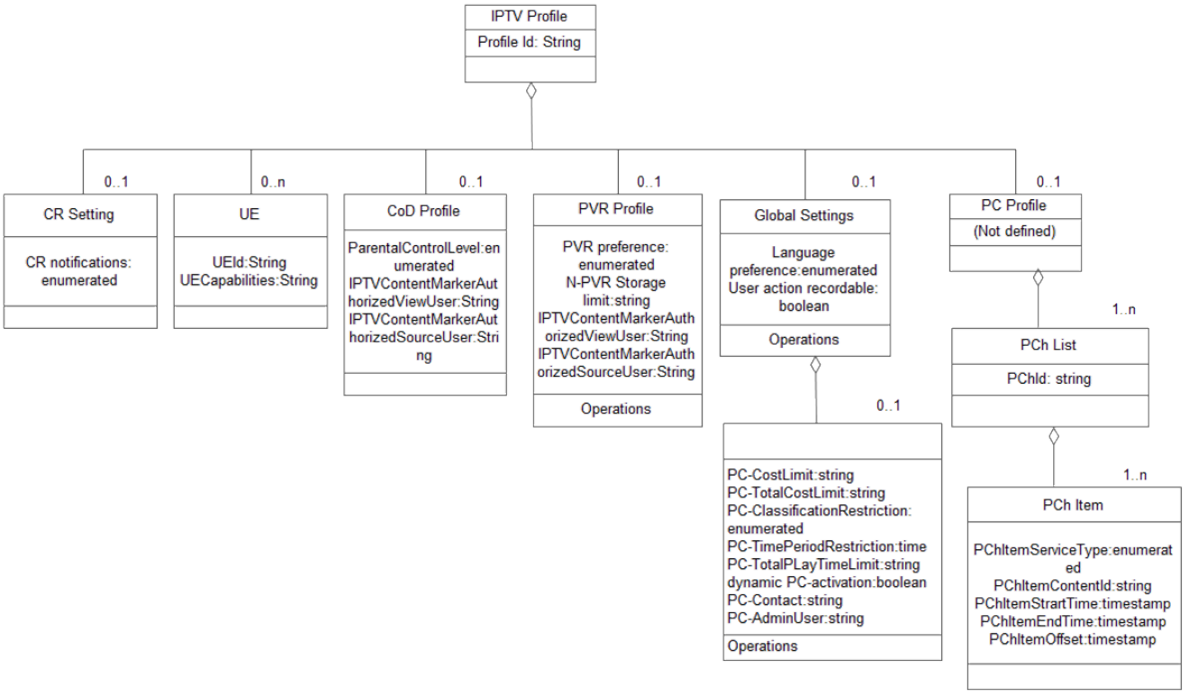



Figure 4.6: The IPTV user profile data model. Source: [77]

**The IPTV User Profile** The IPTV user profile contains basic user information, and also service related information, e.g. subscriptions, bookmarks, activities, parental control, etc. User actions related to service purchase and consumption are also included in the user profile. Similar to IMS subscription information, IPTV user profile information is also stored in a central location but we use different technologies. The IPTV specification proposes the storage of user profile data in XDMS and for this purpose TISPAN defined an *appusage*.

**The IPTV Application Usage** The first thing that application usages must include is the definition of a namespace for the evaluation of XCAP URIs. In addition, each XCAP application usage needs to define an Application Unique ID (AUID), MIMI Type, XML Schema and Data Constraints, Data Semantics, Authorization Policies, and Resource Interdependence.

ETSI TISPAN defined `org.etsi.ngn.iptv` as an AUID for IPTV appusage, and presented the schema file related to the application usage in a file named *iptvprofile.xsd* [94]. The *iptvprofile.xsd* schema uses other schema from TV Anytime [95] and MPEG [96]. The *iptvprofile.xsd* schema file specified in [94] is attached as Appendix D.

From our investigation, the schema provided by TISPAN lacks some data elements that we think are important for the delivery of personalized services and we proposed their inclusion into the user profile data structure. This is explained in Chapter 5.

#### 4.3.4.5 Personalized IPTV Service Examples

**Content Recommendation and Notification** The task of content recommendation is to predict the rating that a user would give to a particular content based on his preference. This is done automatically by matching the user profile with the profile of the video files. Videos are basically represented with metadata such as tags, subtitles and other textual information (like genre) related to the video and the matching is done between the user profile and the metadata representing the videos.

There are some attempts for the development of Recommendation Systems (RS) for IPTV by different researchers [97, 98] . Samsung has also introduced a technique called *video on the web* that can be used to recommend videos on the web to an IPTV user [99].

Considering the advantages of RSs and the attempt made by different researchers, we want to recommend the inclusion of a recommendation engine as one component in the IPTV architecture. We call this engine the *RecommendationAndFeeback* engine and it has an interface to SCF. Details about this engine is given Chapter 6.

**Bookmarking** According to the the IPTV specification, the SIP INFO message is used to set bookmarks and the SIP MESSAGE request can be used to retrieve bookmark information. A special form of this service is when the user wants to make bookmark and pause the session and resume from another device which is called *park and pickup video* service.

Another special form of bookmarking is when the user want to watch a special section of a video (e.g. watch the goals of a match), which can be implemented with two bookmark positions (the starting and end of the special section of the video). The TISPAN specification defined the term *content marker* to refer to this type of sections in videos.

**Content Marker** Content markers are used to assign and store an identification for a specific region of a video. A user specifies the region that he wants to be marked by a content marker by assigning a content marker id and also can include other information organized in an XML document. ETSI TISPAN defined a schema for storing information about IPTV content marker in [94]. The content marker information defined in the schema include: *ContentMarkerID*, *ForbiddenViewUser* (to specify users forbidden to view this content marker), *StartTimeOfIPTVContentMarker*, *EndTimeOfIPTVContentMarker* and *Rank* (to specify a user's favorite rating for the content marker). We believe a brief description of the content marker is important and we want to propose the inclusion of a *Description* field in the schema.

On the other hand, it is also be good if content providers also include content markers together with their content so that users can have the option of watching the segment they are interested in, instead of watching the entire movie or video. For this purpose, we want to extend the RTSP protocol to include a feature to indicate the existence of content markers in videos. Its implementation is explained in Chapter 6.

Similar to Bookmarking command, we use the SIP INFO command and the SIP MESSAGE command to define and store content markers.

### 4.3.5 IPTV Service Development Tools

The main task in building an IPTV service development environment is the identification of the different components required to provide IPTV specific services. The major IPTV components that are required to provide a basic IPTV service are the SDF, SSF, SCF, MCF and MDF. As discussed in Section 4.3.3, most of these components are SIP based ASs and we have seen how the Mobicents SDP can be used to develop a SIP AS.

On the other hand, as mentioned before, for the purpose of media control, we developed a middle layer, in the form of a streaming proxy, that intermediates between the streaming client (the UE or MCF) and the streaming server. We used the Apache MINA multimedia framework to develop the Streaming Proxy and Relay Component. A good discussion about this framework is give in [100].

Other multimedia frameworks reviewed in this research are presented in [50].

### 4.3.6 Codec Requirements

For interoperability purposes, an IPTV client needs to support the following video and audio codecs [101]:

> Video: MPEG-2 and H.264.
> Audio: MPEG-2/4 - AAC v2, AMR-WB+, AC3 and enhanced AC3.

So, we can assume that UEs support these codecs when developing an IPTV service.

## 4.4 Summary

This chapter presented background information about the IMS and IPTV architectures by focusing on the technologies and protocols used to develop innovative video-oriented services using these architectures. We have identified issues related to their architecture and also with the user profiles defined in the standard documents. The discussions presented in the chapter made it clear that video-oriented service development in general, and the IMS architecture in particular are complex and this would be a challenge for potential service developers from the Internet. There is thus clearly a need for an easy-to-use service development and deployment environment that can simplify the service development activity of video-oriented services in the Telcom world. Service developers need to be assisted with various service building components so as to minimize the need to understand the various protocols and technologies.

In the next chapter we present our experimental work that we did to identify missing elements from the standard IPTV architecture and also discuss our findings.

# Chapter 5

# Video Service Development Experiments: Open Source MDF, Personalized and Converged Services

As mentioned before, the IPTV architecture is proposed for use to develop various types of video-oriented services in IMS. We made a series of experiments to see if the architecture fulfills its promises and and also to check if there are gaps in the architecture. In this chapter our work related to basic and personalized IPTV services are discussed. We also present a service called *converged demo* to show how converged services that include video can be developed in IMS.

One important component of the IPTV architecture is the Media Delivery Function (MDF). As mentioned before, streaming servers are used to serve as MDF but they have issues. As a result, our first task related to IPTV was to find a solution that allow open source streaming servers to work as a proper MDF so as to come up with an open source IPTV Media Delivery Function. We start the chapter by presenting this work.

## 5.1 Development of the Open Source Media Delivery Function

One of the objectives of this thesis is to build an open source video-oriented service development and delivery environment. This include building an open source MDF. However, streaming servers in general, and open source streaming servers in particular, have issues

when it comes to working as media servers controlled by a unit other than the client that consumes the media.

For this purpose we propose the inclusion of another component - a Streaming Proxy and Relay Component, that should be part of the IPTV MF and mediate between the MCF and streaming servers. Because timing information, which is important for the development of services like bookmarking, is stored in RTP packets, the Streaming Proxy and Relay Component also needs to inspect RTP packets. Our original proxy (the SSC), presented in Chapter 3, was developed only to handle RTSP messages and did not handle RTP packets. As a result, we developed another proxy that also relays RTP packets. One advantage we get with a component like this is that the functionalities missing from streaming servers can be implemented in it.

The following section presents the RTSP proxy and relay unit.

### 5.1.1 The Streaming Proxy and Relay Component

We based the Streaming Proxy and Relay Component (we refer to it hereafter as simply *the Proxy*) on an open source project [102]. The project was actually a frozen project with lots of issues, but also handles RTP. We fixed the problems and also included additional features that are important for developing advanced IPTV services, such as Bookmarking and Personalized Service Composition in the Proxy.

Figure 5.1 presents the Proxy in the context of the new modified MF that can be used to deliver streaming media from any open source streaming server.

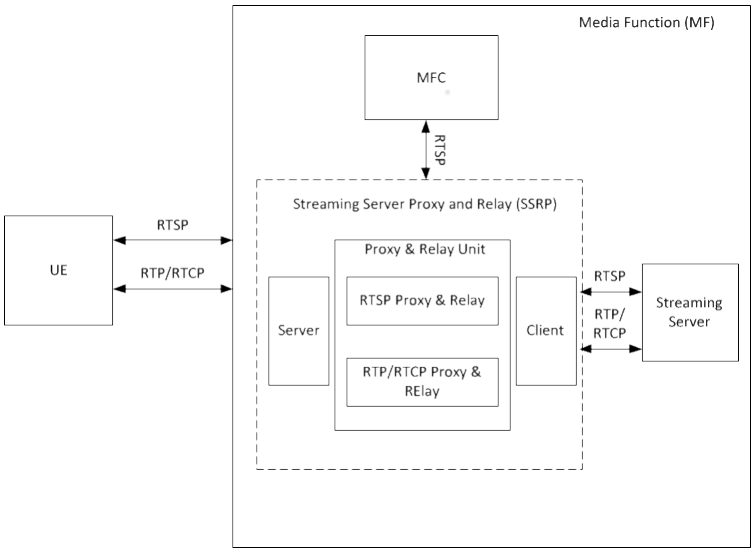



Figure 5.1: Block diagram of the modified Media Function

The Proxy has different request handlers for both the RTSP and RTP/RTCP sessions on both the client and server sides. On the client side, RTSP session is created to allow MCF to communicate with it while the RTP/RTCP session is created to communicate with the client (UE).As depicted in the figure, requests coming through these sessions are handled by the *ServerUnit* . On the streaming server side, both sessions (RTSP and RTP/RTCP) are created by the Proxy to communicate with the streaming server and are handled by the *ClientUnit*. The Proxy is developed based on the Apache Multipurpose Infrastructure for Network Applications (MINA) Framework mentioned in Chapter 4.

#### 5.1.1.1 Modules of the Proxy

In the Apache MINA Framework, an input is processed at different levels through different components, the main components being `IoService, IoFilter` and `IoHandler`. The Proxy is also developed based on these components, and explained below.

**I/O Services in the Proxy** The IoService class provides basic I/O service and manages I/O sessions within MINA. It is responsible for handling low level I/O operations. The Proxy has three `IoServices`: `RtspService`, `RtpClientService`, and `RtpServerService`. They are used to handle the RTSP and RTP sessions.

An `IoService` is created by associating it with a transport type and an address. Once a service is created, it should be associated with `IoFilters` and `IoHandlers`. The code snippet in Listing 12 shows how the `RtspService` is created and attached with `IoFilters` and `IoHandlers`. The `ClientSide()` constructor in the last line of the listing creates an `IoHandler` object that handles all RTSP requests from the client side.

**Listing 12** RTSP service creation code snippet

```
Service service;
service = new Service( "RtspService", TransportType.SOCKET, new InetSocketAddress(
netInterface, port ) );
Reactor.getRegistry().bind( service, new ClientSide(), new RtspClientFilters() );
```

**The Proxy Packet/Message Filters** In the context of MINA, an `IoFilter` class is used to filter I/O requests and events before they are transferred to the `IoHandler`. `IoFilters` are used to reject messages that do not qualify to move to the next level. Filters are organized as a pipeline. For example, in last line of Listing 12, the attribute `new RtspClientFilters()` is used to create a pipeline that will contain different filters. The Proxy has the following filters on the client side: `IPAddressFilter` to filter blacklisted IP

addresses, `RtspCodecFilter` to pass proper RTSP messages and discard erroneous ones, and `AuthenticationFilter` to implement authentication. Filters should be ordered as to reflect the sequence in which we want the messages to be filtered. Listing 13 shows how different filters are appended to build the `RtspClientFilters`.

**Listing 13** RTSP filter creation

```
public void buildFilterChain( IoFilterChain chain ) throws Exception
{
addIpAddressFilter( chain );
addRtspCodecFilter( chain );
addAuthenticationFilter( chain );
}
```

After passing through the filters, a message is next delivered to an `IoHandler`. `IoHandlers` are used to implement the business logic, in our case, proxying incoming messages from the client to the server and vice versa. All `IoHandlers` have a method called `messageReceived` that is called when a message is submitted to them. This is where we put application logic to examine the message and determine the appropriate action. A message is an assembly of IP packets at a particular time that is handled by a particular message handler. If an `IoHandler` decides to pass the message to another handler, then it initiates the `messageRecieived` method of the next handler. `IoFilters` also have `messageReceived` method to inspect the message. The code snippet in Listing 14 shows how the `iPAddressFilter` inspects the messages it has received and passes it to the next level.

**Listing 14** Message passing from one filter to another

```
@Override
public void messageReceived( NextFilter nextFilter, IoSession session,
   Object message )
throws Exception
{
        if ( !provider.isBlocked( ( (InetSocketAddress) session.
           getRemoteAddress() ).getAddress() ) ) {
                nextFilter.messageReceived( session, message );
        } else {
                blockSession( session );
        }
}
```

**Sessions in the Proxy** As mentioned before, a streaming session is created when the streaming server receives a SETUP command from the client. However, in the Proxy, we have a Java i/o session (that we called `IoSession`) even before the SETUP command is received. Actually, we have two separate and different i/o sessions: one on the client side

and another on the server side that are used to exchange RTSP messages. The Proxy treats all these sessions differently and the proxying of the messages on these sessions is handled by different `IoHandlers`.

**Proxying the I/O Sessions** An `IoSession` on the client side is created when the first message arrives at the Proxy. When this happens, a `ProxyHandler` object, responsible for proxying RTSP messages for this session is created and attached to the `IoSession` as an attribute. Since the Proxy has not started communicating with the streaming server by this time, an `IoSession` is created on the server side and a handler, called `ServerSide`, is attached to it. A reference to the server side `IoSession` is also put on the `ProxyHandler` object.

The `ProxyHandler` object has `passToClient` and `passToServer` methods to forward messages received from the client to the server and from the server to the client respectively. For reasons mentioned before, the SETUP requests are handled differently: the Proxy uses `passSetupRequestToClient` and `passSetupRequestToServer` methods for this purpose.

**Proxying the Streaming Sessions** As mentioned before streaming session is created when setup request is received by the server. So, the Proxy creates a streaming session (defined by a `ProxySession` object) that represents the streaming session on the client side when it receives the first SETUP request from the client. The Proxy creates its own *session id* (a 64 bit random number) to refer to this session.

When a response to the SETUP request comes from the server, the Proxy gets the server's *session id* and records it in the `ProxySession` object. In addition to this, when a SETUP response comes from the server, a `Track` object is created for each track in the streaming session and the information is attached to the `ProxySession` object. The code snippet in Listing 15 shows how this is done.

**Listing 15** Code listing showing the creation of Track object

```
Track track = proxySession.addTrack( (String) clientSession.getAttribute( "setupURL"
),
transport.getSSRC() );
```

The code of the `addTrack()` method is shown in listing 16. The `Track` object is similar to the `ProxyHandler` and `ProxySession` class, in that it is used to proxy RTP/RTCP packets using the methods `forwardRtpToClient` and `forwardRtcpToServer`.

**Listing 16** The addTrack method

```
public synchronized Track addTrack( String url, String serverSsrc )
{
        Track track = new Track( url );
        if ( serverSsrc != null )
                track.setServerSSRC( serverSsrc );
        trackList.put( url, track );
}
```

As we developed the Proxy to augment open source streaming servers, we tried to include the various features that we said are missing from streaming servers and RTSP protocol and these are explained below.

### 5.1.2 Changes Made to the Proxy

The Proxy already supports the `destination` parameter, and as such it also supports streaming session transfer. In this section, we only present the other features that we included in the Proxy.

#### 5.1.2.1 Support for End of Stream Notification

As mentioned before, the recent RTSP Internet draft introduced a new command called PLAY_NOTIFY to signal end of stream. There are also other proposals for end of stream notification, but the open source streaming servers that we investigated do not support any of the recommendations.

We extended the Proxy to utilize the information from RTCP to send end of stream notification. We modifiedthe module that relays RTCP packets (the `Track` object) to send the end of stream notification to the module that forwards RTSP messages (`ProxySession`) when it gets the RTCP BYE message. Accordingly, the `ProxySession` object sends an end of stream message to the MCF as an RTSP command. We used the RTSP ANNOUNCE command as suggested by [103] to send the `End-Of-Stream` message to the MCF.

To achieve this, we also modified the `Track` class to include a reference of the `ProxySession` so as to send the RTSP message from within a `Track` object. Listing 17 shows how the `Track` class uses `ProxySession` to send the `End-Of-Stream` message to the MCF.

**Listing 17** The RTSP end of stream implementation

```
ProxyHandler ph=null;
if (packet.getType().getValue().intValue()== 203)
        ph=proxySession.getProxyHandler();
if (ph!=null)
{
        RtspRequest rtspRequest = new RtspRequest();
        rtspRequest.setVerb(RtspRequest.Verb.ANNOUNCE);
        rtspRequest.setNotice("2101 End-Of-Stream");
        ph.passToClient(rtspRequest);
}
```

#### 5.1.2.2 Support for Bookmark Command

An AS needs to get the *current play time* from the server, if it has to support the Bookmark request. The current play time of a stream can be found from the `timestamp` field of the RTP packet that is going to be presented to the user when the bookmark request comes. Because media servers give a random starting `timestamp` value for the first packet (instead if zero), we have to use the `timestamp` of the first RTP packet in the calculation of the current play time. On the other hand, a `timestamp` value is also related to the `clock_rate` of the codec of the RTP packet that is being examined for this request. So we also utilized the `clock_rate` information that we get from the SDP. `Clock_rate` is implicit for static audio/video (A/V) profile codecs and one can use a table to check the `clock_rate` value for each codec. For codecs with dynamic A/V profile on the other hand, we get the `clock_rate` from the `rtpmap` attribute of the SDP. Therefore, we record the `clock_rate` together with the `track_number` of the track we want to use for bookmark at the beginning of the stream. For obvious reason, we used the video track of the stream to record these data.

Once we have the different values just mentioned, to get the current play time of the current media position we subtract the last `timestamp` from the first `timestamp` and divide it by the `clock_rate`. We defined the `getBookmarkPositon` method in the `Track` object to handle this request.

After calculating the current play time, the Proxy forms a response, and send the response back to MCF as a message body to the 200 OK response message. Listing 18 shows the code in the Track class to do this.

**Listing 18** The getBookmark implementation

```
RtspMessage request = proxyHandler.request;
RtspResponse response;
response.setCode(RtspCode.OK);
response.setSequenceNumber (request.getSequenceNumber()+1);
response.setCommonHeaders();
Double BM = (timestamp1stPacket - timestampCurrentPacket) / clockeRate;
String bookmarkPosition = String.format("%.3d",BM);
response.setHeader("bookmarkPosition", bookmarkPosition);
proxyHandler.passToClient(response);
```

## 5.2 Development and Delivery of Personalized Video Services Using the IPTV Architecture

As mentioned before, the development of personalized services requires the use of user profile, which are stored in XDMS. As a result, the first task in developing a personalized video service for IPTV was to implement the IPTV appusage defined by TISPAN. The following section describes the steps we followed to implement the IPTV appusage.

### 5.2.1 Development and Deployment of the IPTV AppUsage Using Mobicents

Mobicents offers a built-in XDMS server that we used to test the IPTV appusage that we implemented. In Mobicents, application usages are represented by a class that extends the `AppUsage` abstract class. This class defines the `AUID`, `default_doc_namespace`, and the `MIMITYPE` attributes. The class has different constructors, but all of them have a `validator` object as one of their parameter. The `validator` object is used to validate the schema that the appusage is based on. We used the default constructor that only accepts a `validator` object.

The definition of an appusage also requires the definition of an object factory which should implement the `AppUsageFactory` abstract class. It also requires the definition of a class that is used to deploy the appusage into the XDMS server by extending the `AppUsageDeployer` class. Listing 19 shows the parameters that needs to be set in the appusage class. Listing 20 shows an implementation of the object factory.

The main files of the IPTV appusage are attached in Appendix E and the entire package of the appusage is included in DVD attached to the thesis. ASs access or modify user profile data using XCAP requests.

The extraction of user related data by application servers is not the final goal. The

**Listing 19** IPTV application usage parameters

```
public static final String ID = "org.etsi.ngn.iptv";
public static final String DEFAULT_DOC_NAMESPACE = "urn:org:etsi:ngn:
   params:xml:ns:iptv";
public static final String MIMETYPE = "application/vnd.etsi.iptvprofile+
   xml";
private static final String AUTH_ONLY_DOCUMENT_NAME = "iptvprofile";

public IPTVUEProfileAppusage(Validator schemaValidator) {
        super(ID, DEFAULT_DOC_NAMESPACE, MIMETYPE, schemaValidator,
           AUTH_ONLY_DOCUMENT_NAME);
}
```

**Listing 20** IPTV application usage object factory implementation

```
private Schema schema = null;

public IPTVUEProfileAppusageFactory(Schema schema) {
        this.schema = schema;
}

public AppUsage getAppUsageInstance() {
        return new IPTVUEProfileAppusage(schema.newValidator());
}

public String getAppUsageId() {
        return IPTVUEProfileAppusage.ID;
}

public AppUsageDataSourceInterceptor getDataSourceInterceptor() {
        return null;
}
```

extracted data (an XML document) needs to be adopted to the programming language we want to use through a technique called *parsing*. We used JAXB [104], a lightweight XML parser to do the parsing. To test the appusage, we created a default well-formed XML document and upload it into the XDMS server using a Java class. The Java class used to create the default XML file is attached as Appendix F. The classes generated by the JAXB framework to represent the IPTV profile schema are also included in the DVD attached to the thesis.

### 5.2.2 The Personalized Dynamic Video Delivery System

In addition to a component that handles the user profile management, the development of a personalized service delivery also involves the handling of subscription and notification of events and the work discussed in this section presents our experiment in this regard. This service was developed before we implemented the IPTV application usage mentioned

above. As a result, a different application usage and preference document was defined for the purpose of specifying user's preferences.

The dynamic video delivery system, published in [105], was developed to make subscription of special video alerts (like breaking news alert) in order to get notification when the video is available. When the video is available, there are two main tasks that the system has to do. The first is the notification of the arrival of new video through the event notification sub-system and the other is the switching of the media seamlessly, preferably using the existing connection. The architecture of this service is presented in Figure 5.2.

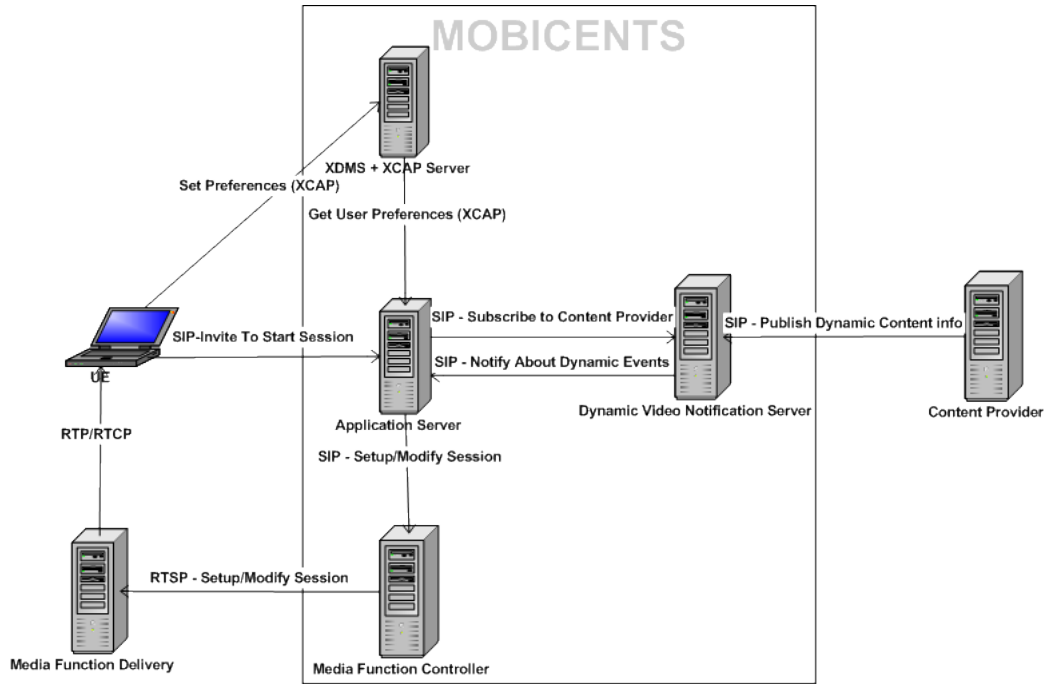



Figure 5.2: The dynamic video delivery service Architecture

As one can see, the major components required for this service are the Dynamic Video Notification Server (DVNS), the XCAP/XDMS server, the IPTV AS and the MF. For simplicity we present the Streaming Server and the Proxy, detailed in Section 5.1.1, as MDF in the figure.

As mentioned in Chapter 2, IETF developed an RFC [28] for the purpose of event subscription and notification. In the RFC IETF defined a concept called the SIP *event package* [28] which is an application-specific notification technique that defines a set of state information to be reported by a notifier to a subscriber. Accordingly, various application-specific event packages have been defined to extend the generic SIP event package. We also modified the SIP event package used for presence service in Mobicents for our purpose.

In any event notification system, there are two main parties involved, which are Event

State Subscribers (ESSs) and Event State Publishers (ESPs). Because, different users can subscribe to the same kind of event, another unit involved in such a system is *the State Agent* (also called *network aggregation point*), which is responsible to aggregate publication information from different publishers and provide them to subscribers in an appropriate manner. In the service we are discussing, video content providers are ESPs. They publish information about the video events using the SIP PUBLISH request. The DVNS is the State Agent and receives published information and sends it to ASs (ESSs) so that they notify users based on their preferences. The content of the publication request, an XML document, is carried as a SIP payload.

The following sections describe the different modules of our system.

#### 5.2.2.1 The Media Notification Sub-System

The first thing we did in the development of this service was to define a custom event with different headers for the purpose of subscription and notification and explained below.

**The Event Package Headers and the DVN Document** As specified in the SIP event package framework specification [28], custom event packages that extend the SIP event package must specify their event type in the `event-type` header and content type in the `content-type` header. The event type we designed for our service is called `dvn` (short for dynamic video notification). The `content-type` header is set to `application/dvn+xml`. All the other SIP headers and behaviors specified in [28] remain the same in our event package. The information about the dynamic event is specified in an XML document, which we called it *DVN document*, and is carried as part of the SIP PUBLISH, SUBSCRIBE and NOTIFY payloads. Listing 21 shows a sample DVN document.

As it can be seen from Listing 21, the root `dvn` element contains an `id` attribute which identifies the content provider; in our case, `sip:convergence@ru.ac.za`. DVN documents can have one or more `media` elements that describe the events related to a type of media such as *sport* or *news*.

The `type` attribute of the `event` element takes an integer value regarding the type of event in the context of the particular media type. For instance, in the context of sport, a type value of `1` signals a goal in soccer or hockey; a try in rugby and so on, and a `2` may indicate a secondary event such as a penalty, a red card, etc. The `time` attribute takes the time at which the event was published, or when it started happening. The `ttl (time-to-live)` field, on the other hand, indicates, in seconds, for how long the event is relevant. If this time passes, the event should be discarded.

**Listing 21** Sample DVN document

```
<dvn id="sip:convergence@ru.ac.za">
   <media type="sports" >
      <event type="1" time="2010-07-21T09:44:00" ttl="3600">
         <description> Goal: Inter Milan have scored a goal in the UEFA
         Quarter Final game. The score is now Inter millan 2 to 1 Real
            Madrid <description>
         <link>rtsp://convergence.ru.ac.za:554/uefa-21072010.mp4</link>
      </event>
   </media>
   <media type="news" >
      <event type="1" time="2010-07-21T09:46:00" ttl="7200">
         <description> Breaking news: 39 Chille miners rescued
         successfully after 69 days trapped underground</description>
         <link>rtsp://convergence.ru.ac.za:554/news1-21072010.mp4</link>
      </event>
      <event type="2" time="2010-07-21T09:46:00" ttl="6500">
         <description> Oil leak stopped in Gulf of Mexico</description>
         <link>rtsp://convergence.ru.ac.za:554/news2-21072010.mp4</link>
      <event>
   </media>
</dvn>
```

Finally, the `description` element contains human-readable information about the event which eventually be displayed to the user by the MDF or by the AS when notifying the user about the event.

**The DVN Application Usage and User Preference Documents** We name our appusage `dvnservices` and this is what we use in our XCAP URIs to perform CRUD operations on DVN preference documents (which we call `dvnprefs`) from the XCAP server. Every user has his own preference document and so their SIP identity needs to be included in the URI as per the XCAP specifications. For instance, we use the URI *http://convergence.ru.ac.za:8080/mobicents/dvn-services/users/ sip:shange@ convergence.ru.ac.za/i* to add, modify or delete the entire `dvn` preference document for the user *shange@convergence.ru.ac.za*. Listing 22 shows a sample user preference document.

In the preference document, shown in Listing 22, the user specifies the video media type that he is interested to get in the `type` attribute of the media element. The user can optionally specify from which content provider he wants to get the video event notifications.

**The Event Subscription and Notification Modules** As mentioned above, Mobicents includes a *presence service* which runs on top of a SIP event package implementation. So, we also extended this event package to develop our service. We developed two SBBs for our service, which are `DvnPublicationSBB` and `DvnSubscriptionSBB`. The

**Listing 22** Sample user preference document

```
<dvnprefs>
   <media type="sports">
      <event type="1" expires="2010-07-21T10:00:00"/>
      <event type="2" expires="2010-07-21T11:00:00 "/>
      <publishers>
         <publisher id="sip:convergence@ru.ac.za">
         <publisher id="sip:sports.convergence@ru.ac.za">
      </publishers>
   </media>
   <media type="news">
      <event type="1" expires="2010-07-21T11:00:00"/>
      <publishers>
         <publisher id="sip:news.convergence.ru.ac.za">
      </publishers>
   </media>
</dvnprefs>
```

`DvnPublicationSBB` is developed to provide the publication service by handling the headers and the content that are delivered through SIP PUBLISH request, and the sip event package handles the rest for us. The `DvnSubscriptionSBB`, on the other hand, handles incoming SIP SUBSCRIBE requests and it sends SIP NOTIFY requests using the JAXB context.

**Notifying the User** The last part of this service is the notification of the event to the user. One way to do this is through the MDF, where the AS requests the MDF to send the notification as a banner on top of what the user is watching and ask the user if he wants to switch to the new media. The `libvlc` media player has a feature to display text overlays on top of video through its `mediacontrol_display_text` function and the new video alert can be sent as a banner to the client in this way. In our case, we use the SIP Message command to send the notification.

Once the user shows interest in the new media, the media switching follows as presented in the next section.

#### 5.2.2.2 Media Switching

The AS through the MCF sends the media switch command to the Proxy. We have extended the RTSP OPTIONS command to send the media switch request (that we define as `switchto`) to the Proxy by including the URL of the new media in the request.

The Proxy then initiates a new media session with the server and sends and receives the RTSP requests and responses by itself to perform the media setup. When the media setup

is done, the Proxy informs the MCF so that the MCF sends the SIP re-INVITE command to the UE in order for the UE to reset the RTCP parameters of the existing session to reflect the attributes of the new media session. After this, the Proxy issues the RTSP PLAY request to ask the streaming server to start the delivery of media and the media delivery would begin. Figure 5.3 shows the call flow diagram that shows the interaction between the different components.

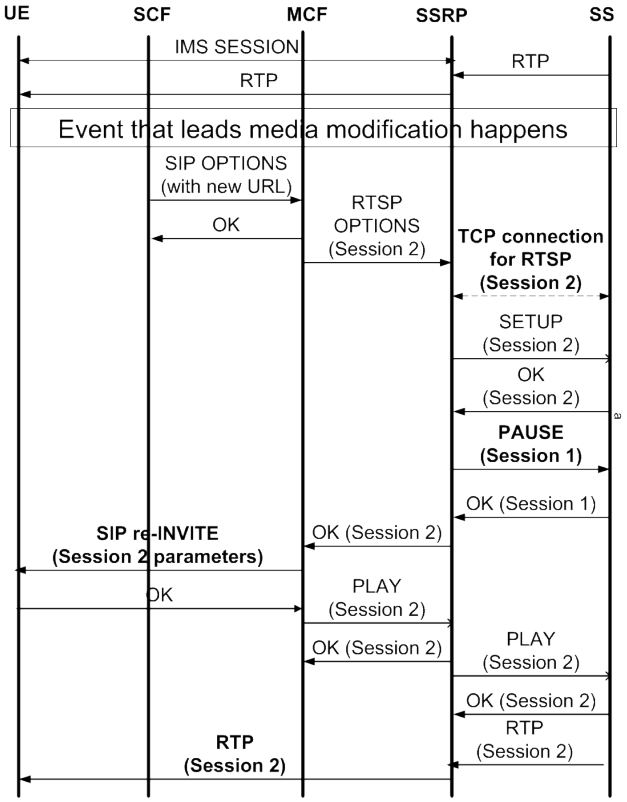



Figure 5.3: Call flow diagram for media switching using existing link

The media switching feature that we discussed in this section can be extended to develop other similar services, like personalized channel and we can consider it as an service enabler for the envisaged environment.

## 5.3 Development of a Converged Video Service

Another aspect that we wanted our environment to support was the provisioning of converged services. We are encouraged to include this requirement because the Mobicents SDP that we chose as a basis for our service development environment has been used as a showcase to develop various services that bring together technologies from telecommunications and the Internet world.

For example, the Mobicents development team provided a sample converged application named *Converged Shopping Demo* [106] where users use it to order furniture from a website and get it delivered to them after the system confirms that the order actually comes from them through a VoIP call.

The example application is developed based on the JBoss Seam Web Framework which helps an application's web interface to communicate with the Mobicents core. The Seam Framework basically integrates different technologies, including EJB3 Beans, JSF and jBPM. At the back end, the example uses the SIP and TTS (text-to-speech) resource adapters together with the Mobicents Media Server to communicate with the user.

The first thing that users need to do to use the *Converged Shopping Demo* is to create an account. Each account maps a user name with a SIP address. Accordingly, when the user logs into the website, the system authenticates the user based on his credentials. Once authenticated, a user can browse through a catalog of furniture items and put items that he is interested in to a shopping cart. When the user finishes choosing items from the website, he place an order for the items by pressing the checkout button. The system then initiates a call to the user through the help of JAIN SLEE to finalize the order.

When the user answers the call, the system plays an announcement requesting him to key in a digit to confirm the purchase and also enter the desired delivery date for the order using his keypad. If the user confirms the purchase using DTMF, then the call terminates and the order will appear under the *My Orders* tab in the system's web page. The message that comes from the web page to Mobicents is packaged as an event that contains the required information to make the call and delivered to the AS through the event router.

The Convergence Research Group at Rhodes University, of which the author is a member, has modified this example to provide converged video-oriented service. The idea was to use the basic functionality of the example and create a system that allows registered users to browse through a web catalog of videos instead of furniture items and request a trailer for a video they are interested in before buying it. In the modified system, when a user requests a trailer, the system makes a call similar to original example to get confirmation of the request from the user. Once the request is confirmed the system then streams the video to the user's device so that after viewing the trailer, the user can decide if he wants to add the video to the shopping cart. This system is explained in [17] and Figure 5.4 shows the architecture of the system. A video clip that shows how the system works was put on YouTube and got more than 1500 hits [107].

The development of the converged service allowed us to make the following conclusions.

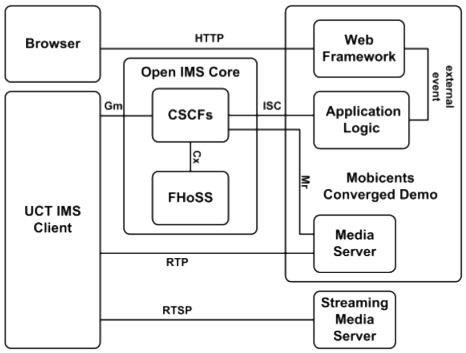



Figure 5.4: The converged shopping demo architecture. Source: [17]

The component that handles user authentication using SIP call involves different technologies. The component,developed as an SBB (`CallControlSbb`), specifically uses two protocols, which are used to achieve the following functionality:

- establishing a network connection between the system and the user's UA using SIP, and
- setting up a media path with Mobicents Media Server using MGCP.

This SBB can be considered as an enabler for video-oriented converged services. For example, it can be used in video delivery systems with parental control support, where the system use it to contact the parent (parental control authorizer) to curb the parental control restrictions by getting his feedback.

The converged video service presented above can also be modified to create an extended Electronic Program Guide (EPG) that can be put on IPTV service providers website and users can chose to view trailers, for example, before requesting a particular VoD item. If the requested trailer has a parental control restriction, the system would initiate a call to the parental control authorizer, using the `CallControlSbb`. Once the user is authenticated then the streaming of the trailer can continue. This can be considered as a service enabler (call control enabler). The use of a *call control enabler* is further discussed in Chapters 6 and 7.

## 5.4 Summary

This chapter presented our experimental activities and findings related to IPTV service development. The problem with the use of open source streaming servers and deficiencies of the RTSP protocol is one of the challenges for delivering innovative video-oriented

services in IMS. A close investigation of the RTSP protocol leads us to the conclusion that IETF defined it for use in a client/server environment and not for use in an environment where we have media controller in between the client and the server. This can be seen from the fact that any streaming issue and problem in a streaming session is reported through the RTCP protocol (a protocol used for communicating feedback information) and not through RTSP (the session control protocol).

As presented in the chapter, the RTSP Proxy is developed to solve the above issues and was explained how it can help us in building an open source IPTV service development and delivery environment. The main advantage of the Proxy is its openness. As a result, it can be used to test and prove new concepts. In this regard, we have showed how media switching can be implemented with a parallel media setup for SIP-based clients. We also showed how the Proxy handles Bookmark requests. Even though the proposal needs to be further expanded, we can consider the Proxy as one infrastructure of the environment that we envisaged to create.

As mentioned before, personalization is what makes IPTV flourish as a service. In this regard, we have shown how IPTV can be used to provide personalized video services. We have also seen how we can combine the web and telecommunication infrastructures to develop and deliver a converged service.

In general, the work we did in this chapter shows how complex is developing a personalized video service for NGN/IMS and shows how imminent is the need for an easy to use service development environment to develop video services.

Through our experiments presented in this chapter and Chapter 3, we identified the issues related to the existing IPTV architecture and also implementation related issues with RTSP. The experiments also helped us to see new requirements for video-oriented services. In the next chapter we compile and present requirements of the new video-oriented service development and delivery environment together with our design of the environment.

# Chapter 6

# Requirements and Design of the Environment

As mentioned before, we base our environment based on the IPTV architecture. The service development activity that we presented in the previous chapters helped us to identify missing components from the current IPTV architecture and pinpoint issues related to protocol implementations.

Our experiments not only did show us missing components, but also helped us to identify new service requirements.

Some of the IPTV service requirements listed in the IPTV standard documents are high level requirements and no implementation detail is given. As a result, as a first task to consolidate and finalize our work, we compiled our analysis of the issues of the existing architecture and revisited the requirements from the different standard documents that required a further investigation, to build a list of requirements for the environment that we envisaged to build. We believe these requirements will be considered in future standard documents. The main focus is on requirements relating to IPTV streaming services.

Based on our compiled requirement specification, we also modified the current architecture and presented it in this chapter. Therefore, another discussion point of the chapter is the design of the environment, which includes the architecture of the environment and algorithms of the major services.

## 6.1 Requirements of the Environment

We start our discussion by presenting a summary of existing IPTV service requirements and their sources. We then present the new requirements that we identified through our experiments of video-oriented services and our analysis of the problems of the current architecture and technological trends.

### 6.1.1 Existing IPTV Service Requirements

We discussed about IPTV specifications from the different standard bodies in Section 4.3.1. For ease of reference we summarized and present them in this section. Table 6.1 presents the main IPTV specifications that relate to IPTV service and the architectural requirements of the different standard bodies, together with their focus areas.

| Standard Organization | Document name | Focus of the Document |
|---|---|---|
| OMA[108] | Mobile Broadcast Services Requirements | IPTV services for Mobile Broadcasting technologies |
| TISPAN[65] | IPTV functions supported by the IMS subsystem; Service Layer Requirements to integrate NGN Services and IPTV | IPTV services for IMS and NGN |
| OIPF [74] | Service and Platform Requirements;Services and Functions | IPTV service for Open Internet |
| ATIS [72] | IPTV Architecture Requirements | QoS-related issues |
| ITU-T [73] | IPTV Service Use cases; IPTV Service Requirements; Functional Requirements and Architecture NGN | IPTV for NGN |

Table 6.1: IPTV standard organizations and their focus area

For our purpose, we considered the requirements specified by TISPAN because it has become a de facto standard for IPTV by the research community. However, the requirements from all standard bodies are not complete. As mentioned in [109], IPTV requirements are evolving. As a result, we made an effort to identify new requirements. We used different techniques to come up with new requirements.

### 6.1.2 Considerations for New Requirements Discovery

We considered different aspects to come up with new requirements. The considered both functional and non-functional requirements that also have an impact on the choice of implementation technology.

Accordingly, the first consideration we made is that service developers need a consistent programming environment to develop video-oriented services. For example, the system should allow the handling of different types of events in a similar way.

Another aspect we considered was that the environment also needs to provide support for the development of innovative video-oriented services that we get on the Internet such as recommendation and notification.

Because the trend in the communication world is convergence, we need to build an environment that supports the development and delivery of convergence services.

Accordingly, we came up with a long list of requirements that the environment should support. The following section presents summary of the extended IPTV service requirements. The entire list is presented in Appendix G. We believe the requirements list is a good resource for future IPTV related standard document development.

### 6.1.3 Summary of the Extended IPTV Service Requirements

For the purpose of presentation, we organized the requirements into two categories, one with requirements related to the services that the environment should provide and another with requirements related to user profile data.

#### 6.1.3.1 Service Related Requirements

- Advanced streaming: requirements related to the provision of this service include the automatic recovery of a session after an unwanted/unforeseen interruption (e.g. due to power loss), and controlling of the session from other devices. The system should also allow the receiving of metadata together or separate from the media. The system should make it easy for users to follow links that come with the video (e.g. to check out items on the screen from an advert). The system should also support capability based session transfer and initiate automatic transfer of a session when the user boots a device with better capability than the one he is using to watch a program. Other requirements that relate to advanced streaming are content switching and bookmarking, which have their own detailed requirements and are presented below.

- Content switching: seamless switching and resumption of the previous stream after switching are the main requirements in this category. Resumption of the previous session is needed when we want to insert media temporarily (e.g. insertion of an advertisement or important announcements in a given session).

- Bookmarking: support for the provision of pre-configured bookmarks in addition to the traditional bookmarks that the user would request the system to place.

- Content marker: similar to bookmarking, the system should allow the provision of content markers in videos delivered to users (like the chapters in DVDs).

- Notification and recommendation: based on users' configuration, the system should provide notifications that include content guide for new services, new CoD availability, incoming calls, etc. Notifications could have priorities and the delivery method also depends on the priorities. The system should support the automatic delivery of important notifications. User should be able to configure the system about their notification interest, and the system should also learn from their watching habits to send reminder by itself. The system should consider device capability when making the recommendation.

- Special clips delivery: the system should support the automatic playing of special clips on system boot up on special occasions, like birthday and wedding anniversary. The delivery of the special clips should consider device capability and the user's status.

- Parental control: In addition to the ordinary parental control services, the system should allow the definition of a parental control authorizer that can view what other users under him are watching, and also allow the lifting of the restrictions on users on specific occasions when contacted by the system.

- Converged services: The system should allow the blending of TV services with other telecommunication and Internet services, like voice or video calls, presence, emails and web pages, etc.

- Time-shift TV: requirements related to this include *catch-up* TV where the system should allow users to go live. The system should also support what we called *true time-shifting*, which happens especially when the user wants to resume a time-shifted TV where the system provides a stream from the PVR while at the same time recording the channel into the PVR because the user may want to pause again. An interesting scenario that we identified related to true time-shifting is dynamic (automatic) time-shifting where the system automatically initiates this service when an important call (that we call a VIP call) is received and automatically resumes the session when the call is ended.

#### 6.1.3.2 User Profile Related Requirements

If we compare the detailed service requirements presented in Appendix G and the IPTV user profile specified in the TISPAN specification [94], we see the need for amendments in the user profile. The reason for this is that we have service requirements that require some elements in the user profile for their provision but those elements are missing from the current user profile. Table 6.2 presents some of the requirements that led us to the identification of new elements in the IPTV user profile.

| Req. No | Description of requirement | New user profile/ element |
|---|---|---|
| RM-02 | The IPTV service should be able to recommend appropriate clips on system boot up based on users profiles – like happy birthday and happy anniversary clips. | Date of birth, Date of anniversary |
| NO-02 | The IPTV solution shall provide a mechanism to assign a priority to notifications. Accordingly, the IPTV solution shall provide a mechanism to guarantee the delivery of critical notification messages with minimal delay (like the use of SMS or other delivery mechanisms that the user chooses). | Telephone numbers for SMS (like Tel No1, Tel No2, etc), Email addresses (like email1, email2 etc), etc |
| ST-01 | The IPTV system shall allow capability based session transfer. | The user should be able to set default transfer device. |

Table 6.2: List of user profile extensions

In general, the main extensions to the IPTV user profile include: default transfer device, accept/block recommendation, critical notification delivery style (automatic, send me message, user confirmation, etc.), critical notification delivery method (EMAIL, SMS, etc.), off-line reminder method (email or SMS), and auto-record programs list (list of programs that the user wants to be recorded when his device is off).

Based on the above requirements, we designed the system, which is presented in the following section.

## 6.2 Design of the Environment

The main objective of the design process was to identify standard components and their interfaces that implement the requirements described in the previous section. It is cus-

tomarily to consider different aspects when designing a system and the following section presents the things we considered when designing our environment.

### 6.2.1 Design Considerations

Since some of the requirements can be implemented through the functionality of the IMS core, we only considered requirements whose implementation should be done through an AS[1]. In fact, looking at the detailed requirements presented in the different documents, we can see that there are requirements that need to be implemented by the UE as well.

Since some services contain logic that can be used by other services, a service could be implemented by a collection of other services (each of them could be implemented as a separate AS). As a result, another design consideration is code-reuse and we identified functions common to all services and considered them as services that need to be provided by the environment. In fact, we focused more on defining primitives rather than services. An example of this type of feature is device capability checking, which is used by different services.

The use of appropriate design patterns is yet another design consideration. Figure 6.1 summarizes the various aspects that we considered to come up with the new architecture.

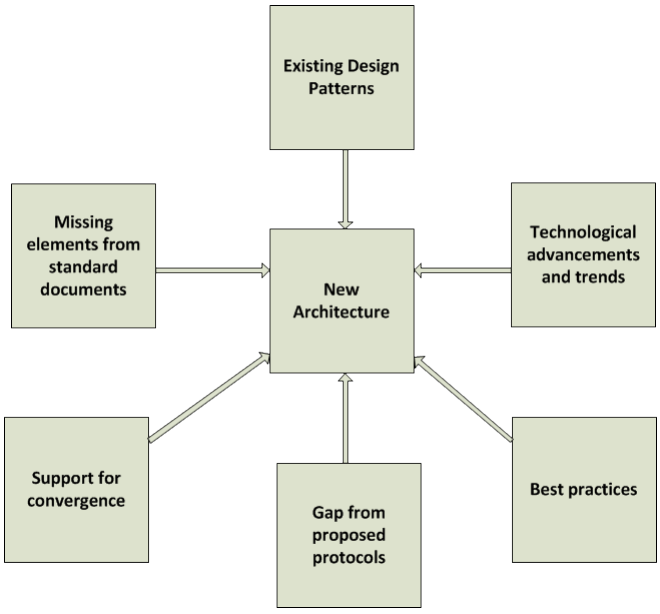



Figure 6.1: Design considerations for building the new architecture

[1]In this chapter we use the term AS to refer to any piece of code (module) with a certain logic and that does a meaningful work

### 6.2.2 Assumptions

The major assumption we made, especially with regards to protocol extensions, is whatever we proposed for the server side will also be implemented in the client, if there is a need.

Another assumption that we made is that appropriate processing speed and memory will be made available for the effective running of the environment and its modules.

### 6.2.3 Architecture of the System

A software architecture is a blueprint of a system that specify high-level functionality. Because we considered IPTV as a reference architecture for our environment, we developed the architecture of the environment based on the TISPAN IPTV reference architecture.However, as mentioned above, there are many differences between the reference architecture and our modified architecture. For example, the standard specification mentions that the SCF is a SIP AS and could represent any kind of SIP-based AS. But if we want to implement the above requirements, we may need ASs that are not SIP-based. This means the SCF needs to be specified in a more detailed manner, and for this purpose we elaborated the SCF according to its functionality so as to show the different components that need to be developed separately and present the message communication and protocols that the different components use.

In terms of architectural style, we followed the shared data and shared service style [110] because IPTV services require similar data, like the current user session state, and they also share some functionality (services), like authentication and authorization. As mentioned before, we considered re-usability and maintainability as our two main architectural design principles.

#### 6.2.3.1 Major Components of the Environment

After analyzing the different services and requirements, we came up with four major groups of services, which can be implemented as components or subsystems. The major components or subsystems identified in this research are:

- Session Initiation and Modification Manager
- User Profile Manager
- External Service Handler

- Event Subscription and Handling Manager.

Detailed explanations of these subsystems and the modules that relate to them is presented in subsequent sections.

Figure 6.2 presents a high-level view of the architecture of the environment, which is based on the IPTV reference architecture. The main difference between this architecture and the IPTV reference architecture developed by TISPAN is the changes we made to the SCF. Among other things, the modified SCF has an interface to external services. An example of a use case that requires an interaction of the SCF with external services is the handling of a request for a PVR service by SMS. The SCF, based on the user's profile, could also need to update the user's Facebook or Twitter account to post what the user is watching etc. A diagram that shows the details of the modified SCF is presented in Figure 6.3.

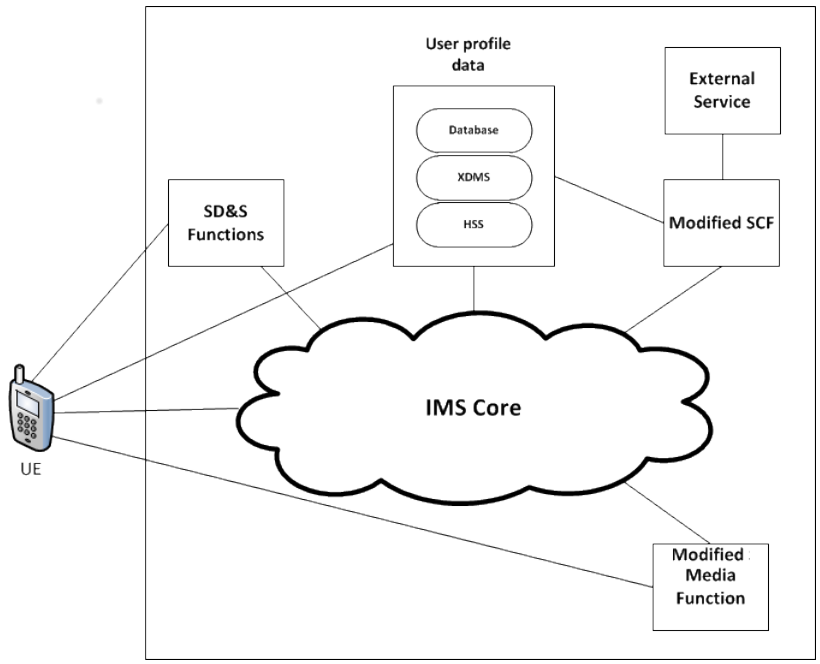



Figure 6.2: Architecture of the environment

#### 6.2.3.2 The Modified SCF

The Modified SCF in the high-level architecture represents a SIP AS or a group of ASs that implement the different service logic and may also use different protocols. Even when the different ASs use different protocols, the service development environment needs to provide a consistent message exchange interface that the ASs use to communicate among

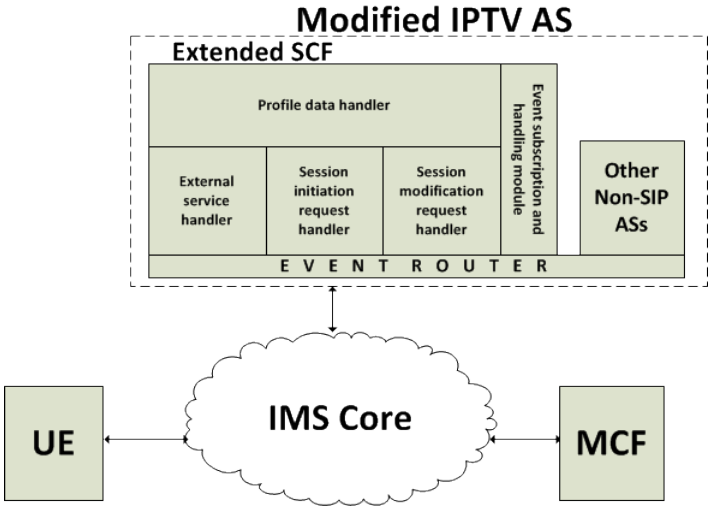



Figure 6.3: Major components of the modified SCF

themselves. For this purpose we propose a component that present protocol messages to ASs in a consistent manner, and we call this component an *event router*. The event router is responsible for routing SIP and non-SIP messages (events) back and forth between external entities and ASs and also between ASs in a consistent way. The event router also support event subscription and notification.

#### 6.2.3.3 The Modified MF

Our modified architecture is also different from the reference architecture in relation to the MF. In the modified architecture, the reference point between the MCF and MDF specifies an extended media session, which could represent one or more media sessions, where we can add, remove, pause or resume media sessions. Internally the implementation is made through different sessions. In fact, as discussed before, we also included the Proxy as an intermediary between the MCF and MDF especially for streaming related services.

#### 6.2.3.4 User Profile Related

A mentioned before, IPTV brings its own unique requirements with regard to user profile data creation and management. The user profile data can be static or dynamic.

With regard to user profile data format, the data can be stored in different formats. We have seen how HSS and XDMS are used to store users profile data. From the requirements we have presented in this chapter, we can also see that we need other user profile data formats for the purposes of delivering recommendation and notification services, and

they need to be stored in relational databases. So, component that is responsible for handling all communications to the different profile data stores using different protocols and technologies is required. This is explained in Section 6.2.6.

The following section presents the detail design activities that we carried out.

### 6.2.4 Process Design and Basic Functionalities

The modified IPTV architecture presented in Figures 6.2 and 6.3 show the major components of our environment. In this section section we present the algorithms required to implement them.

Because delivering a service may involve the coordination of various modules or components, the first activity that we carried out was to investigate how the processing of these components should be done. As a result, we first present the process styles that we identified for the realization of the coordination of the different modules that make up the service development and deployment environment.

#### 6.2.4.1 Choosing the Message Communication Style

IPTV services follow a pipeline and event-based processing styles. For example, when initiating a service, the request has to pass through a set of processes (filters) before the user gets the service he requests. An important aspect to consider when building a component that implement this is that the different filters that process the INVITE command need to know if the previous filter in the pipeline has processed the request and also check the outcome of the process if they have to process the request or not. Because, we can have a new filter in the future, our design should allow the insertion or removal of filters easily. Figure 6.4 shows how the session initiation handler is handled.

On the other hand, when the user is about to finish watching a video, a timer module could alert the AS to request a similar video from the recommender module so as to set up a connection for the new video to start when the current video ends. This shows the need for event-based processing. Because of the event-based nature of the service development environment, one of the task in the design stage was to identify and define all types of events or alerts that we need to design as event classes during implementation. The following section presents the events and notifications that we identified.

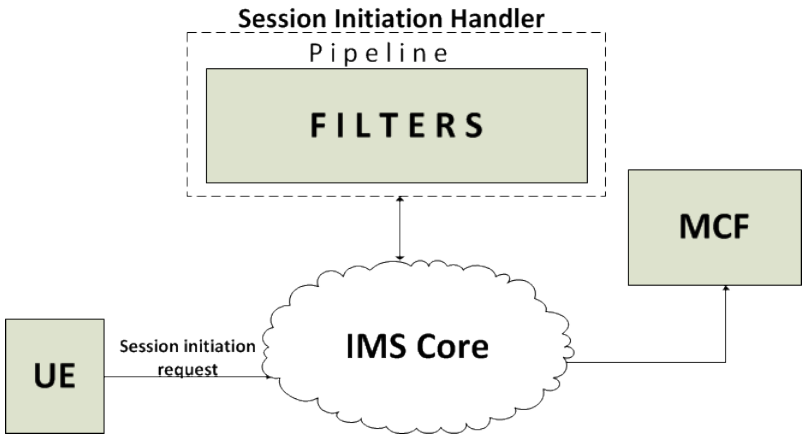



Figure 6.4: Session initiation handling

#### 6.2.4.2 Designing the Message Communication: Events and Notifications

Most of the events we identified are media stream-related events and one event relates to presence information.

- End of stream. This event happens at the end of a stream and is referred to as EOST.
- Beginning of Content Marker. This event happens at the beginning of a content marker in a stream and is referred to as BOCM.
- End of Content Marker. This event happens at the end of a content marker in a stream and is referred to as EOCM.
- Playlist transition. This event happens at the end of every media stream in a playlist and is referred to as EOPT.
- End of playlist. This event happens at the end of a playlist and is referred to as EOPL.
- Stream terminated. This event happens when a stream is inactive and is referred to as STTE.
- New content. This event happens when new content arrives and is referred to as NCON. For the change in the availability of content (i.e if the new content is removed again) this event with a special code is delivered?
- Program reminder. This event happens when a program series that the user is watching is going to come on screen, and is referred to as PREM.

- Special content. This even happens when a special content like breaking news is being presented, or a goal of a football match or match analysis is presented, and is referred to as SCON.
- Important event. This event happens when an important event like emergency announcement happens as presented in the system requirements specification no. NO-05 (Annex G), and is referred to as IMEV.
- Incoming call. This event happens when a call come while the user is watching a program, and is referred to as INCA. For VIP calls special code can be attached to the event.
- Presence related events (notifying friends about the channel a user is watching).

#### 6.2.4.3 Algorithm Description of Major Components and Identified Services

In this section we present the algorithms of the different services. A service can be implemented by one AS or a group of ASs.

**6.2.4.3.1 Bootstrap Service Coordinator** Our analysis of the requirements of the video-oriented service development and deployment environment led us to the identification of different services that need to be delivered by the system automatically on system bootup, that we collectively call *Bootstrap Services*. The Bootstrap Service Coordination module is responsible for initiating all services that need to be provided when the user boots up his device. There are different scenarios where the user boots up his device. The two main scenarios are "first device bootup" and "booting up other devices while he is logged in using another device". The bootstrap services related to these scenarios are also different. One service that can be delivered automatically during first device boot up is *Autoresume Service*, which resumes playing a streaming session that was not terminated properly. Other services related to this scenario include playing appropriate video clips on special days, like the user's birthday and wedding anniversary. On the other hand, as mentioned in the requirement section, the system could initiate session transfer if the user boots up a device with high capability while he is watching a program using a device with less capability. For example, if a user boots up a device with very high definition device while watching a video using a device with single definition video capability, the system should identify this and ask the user if he wants to transfer the session to the high definition device and deliver the high definition version of the video if he wants. We call this service as *Auto Transfer Streaming Session*.

The implementation of bootstrap services require that the AS get the registration status of the user that it wants to serve. Assuming that the AS that provides this service is subscribed to the user's registration status (from HSS), the AS, when it gets the registration status notification, first of all checks if the user is already being served with another device. If that is the case, then it executes the logic for the Auto Transfer Streaming Session service. On the other hand, if this is the first device that the user boots up today, then the AS checks the following when it gets the notification about the user's registration information from HSS.

- is today the birthday or wedding anniversary of the user?
- was there a stream that was not terminated properly in the last IPTV session?, and
- is there notification for a program that the user needs to watch today?, etc.

If one of the above satisfies, then the AS sets up a session to provide the appropriate stream to the user. If two or more of the above applies, then the AS provides the appropriate streams based on the user's configuration settings, one after the other. The user can cancel the delivery of the streams at any point. The data (e.g. the URL of the stream) required for the delivery of the bootstrap service is stored in a database and is described in detail in the data design section of this chapter.

**6.2.4.3.2 Session Initiation and Modification Manager** The major tasks of an IPTV service development are session initiation and modification. IPTV service initiation involves the coordination of various activities.

When a service initiation request comes to the system, the AS responsible to handle the request performs a set of actions, which include checking if the user is authenticated, has enough credit in his account, and has Parental Control (PC) clearance related to the PC rating of the content before setting up the session. These features can be implemented in the same AS or could be developed as a stand alone AS. For the purpose of extensibility we suggest implementing them as separate ASs. Authorization and accounting tasks are common to telecommunication systems and we do not provide further detail here. Because the parental control authorization involves different technologies and protocols, we explain how it should be implemented in the next section. The PC authorization module is activated only if the user is authenticated and has enough credit in his account. The PC authorization module may contact the PC authorizer to get feedback and decide if the user should be allowed or not. After getting the feedback, the parental control authorization module returns the control to the AS by supplying the authorizers feedback, and the AS

then allows or denies the user to watch the content. The detail PC authorization process is explained in the next section.

Session modification is the other major task in IPTV service provision, and it happens when a change of media is required in a given session. Services that require session modification can be initiated from the same machine on which the current service is running or could be externally initiated. For this purpose, an IPTV service environment should allow the sharing of IPTV sessions so that other services can get access and modify them and this component is responsible for facilitating such communication.

**Parental Control Authorization** During a new channel or channel switch request, if the user is on a parental control restriction, then the AS contacts the parental control authorization module. The first thing the parental control authorization module should do is to check if the system needs to contact the parental control authorizer to request authorization. If the logic for initiating a call to the parental control authorizer is met, then the parental control authorization module initiates a call to the authorizer and, after receiving the authorizer's feedback, returns the result back to the AS. The contacting and receiving of feedback from the authorizer needs to be developed as a separate module in order to make changes when a new way of interaction with the authorizer is available.

**Media Switch** As mentioned before, what we call a streaming session is composed of SIP and media sessions. The media session is controlled by RTSP. So, when the AS receives a media switch request, it forwards the request to the MCF because the MCF handles the media session and is also involved in the SIP dialog. The MCF then performs two things: it changes the media session (with the new media parameters) and then inform the AS to change the SIP session to reflect the parameters of the new media.

In most cases, the media server for all the media that the user receives would be the same server. For the media change to work smoothly, all communicating participants need to be aware of the change. This means that, since the change is initiated by the MCF, the UE should also be aware of the change. SIP has a command to update a session when we want to change the media attributes when the communicating devices are the same. As mentioned before, this can be done by issuing the SIP re-INVITE command with the new parameters. However, RTSP do not have this type of feature. Since RTSP is the media control protocol for IPTV CoD services, it is also ideal to expect RTSP to make the same provision.

The use of the RTSP OPTION command that we presented in Chapter 5 for this purpose is not extensible and for this purpose we propose a new RTSP command, called

REPLACE, to be used for switching media. The REPLACE command takes the new media URL as a parameter. As there are other services that depend on REPLACE command, the REPLACE command also take two more parameters. The second parameter is used to specify when the media replacement needs to be done. The accepted values are `immediately` or `when current media ends` represented by 0 and 1. If the two media types are not the same, the command should respond with a `parameter mismatch` error code. Otherwise, the server responds with the media attributes of the new media. The third parameter is used to specify if we want the media change to be permanent or temporary. The acceptable values are 0 for `permanent` and 1 for `temporary`. After getting response to the REPLACE command, the client (in our case, the MCF) issues the PLAY command and the server starts to deliver the new media immediately.

Temporary insertion of media is a special form of media switch and is explained in the next section.

**Temporary Insertion of Media** There are many use cases for temporary video insertion. A VIP call coming while the user is watching a TV program is a good example that requires the temporary video insertion service.

As mentioned above, we proposed the use of the RTSP REPLACE command for media switch. For temporary media insertion, we use this command with the third parameter assigned a value 1. This command will make the streaming server resume the paused media when the temporary clip ends automatically. The response to this command will contain the media attributes of the temporary media. If the response is OK, then in the same manner as the standard REPLACE command, the MCF issues the PLAY command to start the delivery of media.

Since the media controller and the AS need to know when the original media resumes, the streaming server should indicate this through a command to the MCF. This needs to be done because the client should reset the media attributes for the media that is going to be resumed. As mentioned in Chapter 2, the RTSP protocol has a command called PLAY_NOTIFY to send asynchronous notifications. We also propose the use this command with a Notify_Reason set to Resume. The command should include the session id of the session that is going to be resumed. When the streaming server gets a response to this command from the media controller, it will automatically resume playing the paused media from the last playable position (I-frame).

**Bookmarking a Media** As mentioned before, the standard documents, proposed the use of RTSP GET_PARAMETER command for getting the current media position for

the purpose of creating services like Bookmark. But it is not supported by most open source streaming servers. In fact, when we have media with random access properties (i.e., a media stream that uses the *I*, *B*, and *P* frames), the position that we get as a bookmark position, even using the GET_PARAMETER command, could be a position of other frames (i.e. the B and P-frames). If this happens, then when we want to play the media starting from the bookmark position next time, the client could have difficulty to decode the media because B and P frames do not have full information about a frame and depend on previous frames. In other words, the client may find it difficult to recreate the frames until it gets the next I-frame because the B and P-frames depend on other previous frames. The streaming server may also start presenting from the next I-frame, missing out some of the frames in between. Because of this, the server may also complain and give us an error message. For this purpose, we want to propose a new RTSP command called BOOKMARK that returns the position of the last I-frame and then we can use this as the bookmark position.

An algorithm to implement this in our Proxy is to have a variable and record the position of I-frames as last I-frame position and submit the value of the variable when the bookmark request comes. In fact, the use of the Proxy for this purpose has additional advantage. We can develop a service called *Auto Bookmark* easily. Auto Bookmark is a service that records the last bookmark position of a video that a user have been watching before abnormal termination of the video. This is discussed in detail in the implementation section. Media servers can also keep the position of I-frame(s) in an index file or certain type of data structure and return the position of an I-frame with time-frame close to the time of the request.

**Handling Preconfigured Bookmarks and Content Markers** We have come up with and motivated requirements (as presented in Appendix G) for the delivery and use of pre-configured bookmarks and content markers that need to be provided by IPTV content providers (to produce the same type of user experience that users get from DVDs).

The delivery and storage of content marker information can be done in different ways. The IPTV specification specifies that this information should be stored separately from the media and also suggests the use of a different protocol to deliver them. However, we believe this could be done better (especially for preconfigured content markers) if we can include some features into the streaming control protocol (RTSP) to communicate this information. It could also be done through the SDP protocol. The client can access this information during the session setup (for example, as a response to the DESCRIBE request by the client or in our case by MCF). The RTSP PLAY command together with

the Range parameter can then be used to start playing from a bookmarked position or to play a range specified by the content marker. In fact, the parameters for the PLAY command could also be the name of the content marker if we want to play a particular section of the video, or the bookmark name to start playing from the bookmark position. This means an attribute also need to be defined for the PLAY command. The bookmark and content marker data can still be stored in a separate file because the current container files may not support the format of these data.

Related to the proposal of extending RTSP and SDP for content markers, we also want RTSP to signal the beginning and ending of content markers. The events BOCM and EOCM represent the beginning and end of content marker events respectively. The notification of these events could be done through the PLAY_NOTIFY command as mentioned before.

**Session Transfer** As mentioned before, both SIP and RTSP protocols support session transfer with the help of REFER and SETUP commands respectively. However, similar to the problem mentioned for the Bookmark command, there is a problem when we try to transfer a media with random access features (i.e. containing the I, B, and P frames) from a position different from an I-frame. For this purpose we propose an RTSP command called *TRANSFER* that is used to request the server to transfer a streaming session and start presenting the media from the last I-frame. This can be implemented similarly to the BOOKMARK command that we defined above. The streaming server then sends the video starting from the last I-frame.

**Recommending Similar Videos** The recommendation for similar videos needs to be initiated only when it appears that the user likes the video that he is currently watching, by watching it right to the end. It would be good, though, if the system checks for similar videos just before the current video ends so that by the time the user finishes watching the current video, the system can introduce recommended videos immediately. For this purpose, if the user chooses to receive recommendations then, when a CoD is delivered, the AS l sets a timer to get reminded just before the video ends so that it contacts the recommender module to get a list of similar videos. For simplicity purpose, we set the timer to remind the AS one minute before the end of the media session. However, to be a more realistic, the amount of time we used to set the timer to initiate the recommender service can be determined using different network parameters like latency and throughput or can be configured manually.

After contacting the recommender module and get the list of recommendations the AS

sends the result of the recommendation to the user through the SIP MESSAGE command when the video ends. A different command with an XML payload can also be defined for this purpose. After the user chooses a video from the list, then the session is established the same way as a session initiation for a new media. The user is also asked to provide a rating for the media he has just finished watching so as to dynamically update his profile for recommendations.

**Notification of Events** When any of the events that a user is subscribed to happens the notification system should consider the user's presence status to make the notification. For example, if the user sets his status to *Don't Disturb*, then there should not be any notification.

The way the AS notifies the user should also depend on the priority of the event. If an event or message has a high priority, then the AS should notify the user using all communication channels and through all registered devices, especially if the user has not switched on his tv and does not acknowledge receiving the message. This is explained in detail in the next section.

**Important Event Handling** Important events are events such as emergency announcements, VIP calls and other events specified as important events by the user or system provider. When such an event happens, the AS performs the procedure described for temporarily insertion of media by switching the current stream with the new content. Most of the time, the notification of other events, which are not considered as important events, is done through a text overlay feature provided by streaming servers.

**Message Dispatching and Handling** As mentioned before, for the purpose of sending reminders and notifications, the IPTV system may need to use a module that sends email to users. We have seen how we integrated an email client for iVideomail system as presented in Chapter 3. In fact, the IPTV system also needs to send SMSs for urgent reminders and notifications, and as a result the system needs to use an SMS gateway as well. The module that handles SMSs can also be used to get control commands from users. For example, a user can send an SMS to initiate the recording of a particular program while he is away. It has also become common practice that users send their feedback (comments) by SMS, which could be analyzed to rate the program that they are watching. So, the email client and SMS gateway are important components. In general, we need a component for sending and receiving messages to the user, with different message dispatching techniques based on the priority of message and information in the user profile.

**Presence Support** As mentioned before, the initiation of some of the services require that the user should have a specific presence status (e.g. notification of events). On the other hand, depending on the user's preference, the system can update the user's presence status (e.g. to show his friends what the user is currently watching). So Presence Manager is one important subsystem for an IPTV service development and deployment environment.

**Advanced PVR (True Time-shifting)** The IPTV standard document specifies the use of a PVR system for the purpose of delivering time-shifted tv. It is a system used to record a program that the user wants to watch later and present the video to the user at his convenience. When the user wants to watch the recorded video, the AS sends a PLAY request to the PVR to start playing the recorded video. The recording has also a limit and the PVR stops recording when the limit reaches. There is a use case that is not handled by the current standard. If the user wants to start recording a program to watch it at a later time but before the PVR reaches its recording limit the user wants to start watching the video, the PVR should continue to record while delivering the recorded program to the user. That way the user will not miss any part of the program. The current standard do not mention this case. We believe this is particularly good to implement services like VIP Call Answering, where the system will automatically start to record the program that the user is watching when a VIP call comes through his tv. When the call ends, the system should start to play the redirected media from the PVR while pushing the program into the PVR. The AS needs to issue a different command to achieve this, which could be something like RESUME. We call this procedure as Advanced PVR. The call flow diagram in Figure 6.5 shows the procedure for Advanced PVR.

### 6.2.5 Design Model

The previous section presents the algorithmic realizations of the different components and subsystems that need to be implemented. It did not show the structure of the different classes that are needed to implement the components. Because they provide a good visualization of the internal structure of a system, we provide the class diagram of the components and their interfaces in this section. The classes are categorized according to their functionality and use. We have classes that ASs need to inherit and implement and we have classes that can be used to implement the MCF.

Figure 6.6 presents the class diagram of the modified IPTV AS and Figure 6.7 shows the class diagram related to MCF. Implementation of some of these classes is presented in

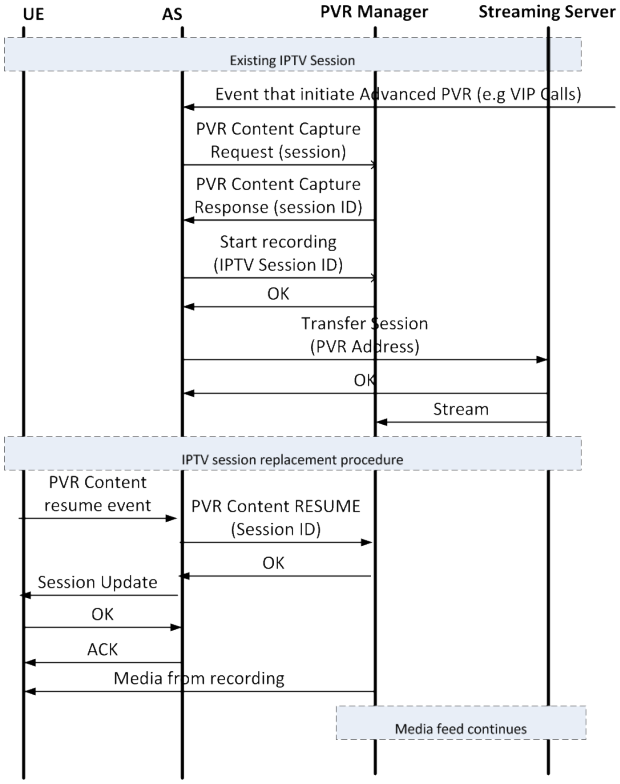



Figure 6.5: Advanced PVR procedure

Chapter 7.

As mentioned before, there are several types of data that are required for the provision of IPTV services. The following section presents our design activity related to the data storage and retrieval requirements of the environment.

### 6.2.6 Designing the Data Access Layer

Most of the time the service profile and user profile data are stored in HSS. However, as discussed in the requirements section of this chapter, there are also data specific to some of the IPTV services and stored at different place. In this section we present the design of the data layer responsible for dealing with the different data types. The types of data include both static and dynamic data.

As per our analysis of the requirements, we have identified data items that need to be stored in XDMS and also in relational databases.

With regard to XDMS based data items, we modified the different sections of the TISPAN IPTV user profile schema based on the user profile extension requirements presented in Section 6.1.3. We also created new profile types. One profile type that we created is the

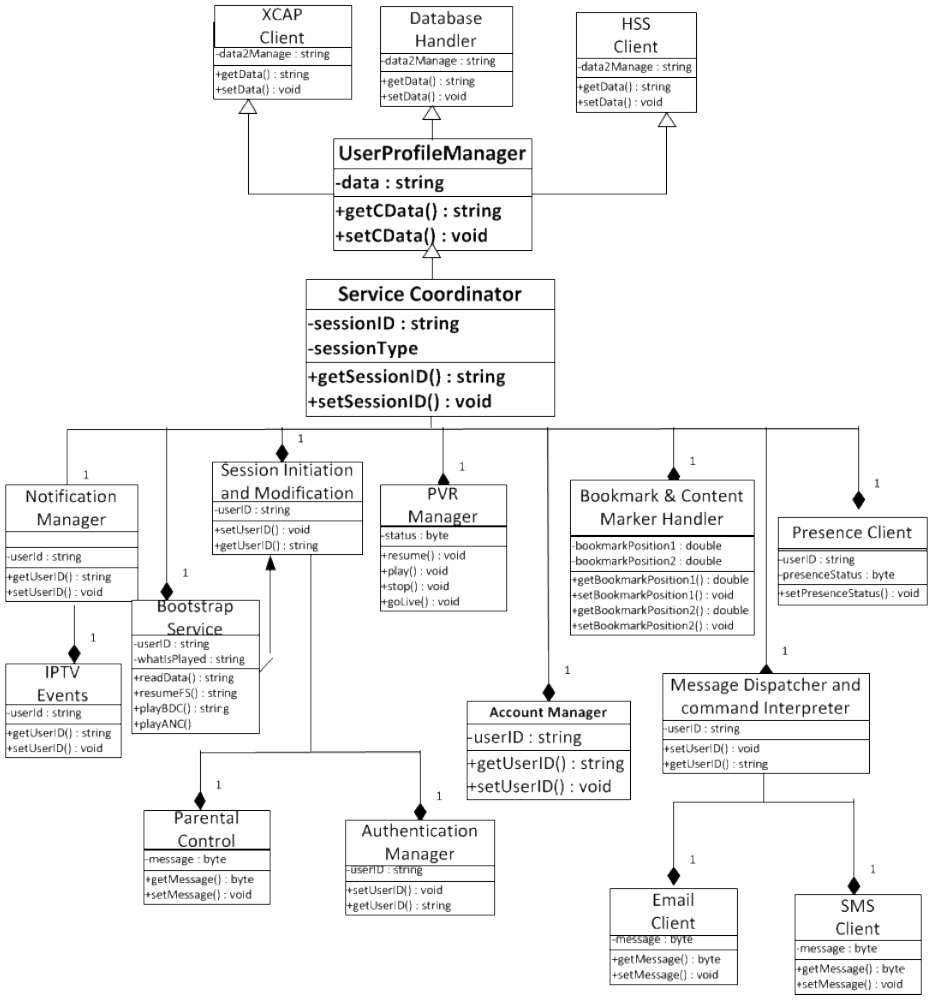



Figure 6.6: Class diagram for the modified IPTV AS

PersonalProfile profile type. Listing 23 shows the new elements of the PersonalProfile profile type. The modified IPTV user profile schema that shows all the new data types and other modifications is presented in Appendix H.

**Listing 23** Partial view of modified user profile schema

```
<xs:complexType name="tPersonalProfile">
   <xs:sequence>
      <xs:element name="Telephone" type="xs:string" minOccurs="0"/>
      <xs:element name="Email" type="xs:string" minOccurs="0"/>
      <xs:element name="Birthdate" type="xs:date" minOccurs="0"/>
      <xs:element name="Anniversary" type="xs:date" minOccurs="0"/>
   </xs:sequence>
</xs:complexType>
```

As mentioned before, we used relational database specifically for the purpose of developing and using personalized services, like notification and recommendation services. We choose the relational database for its efficiency in processing queries and relating data from

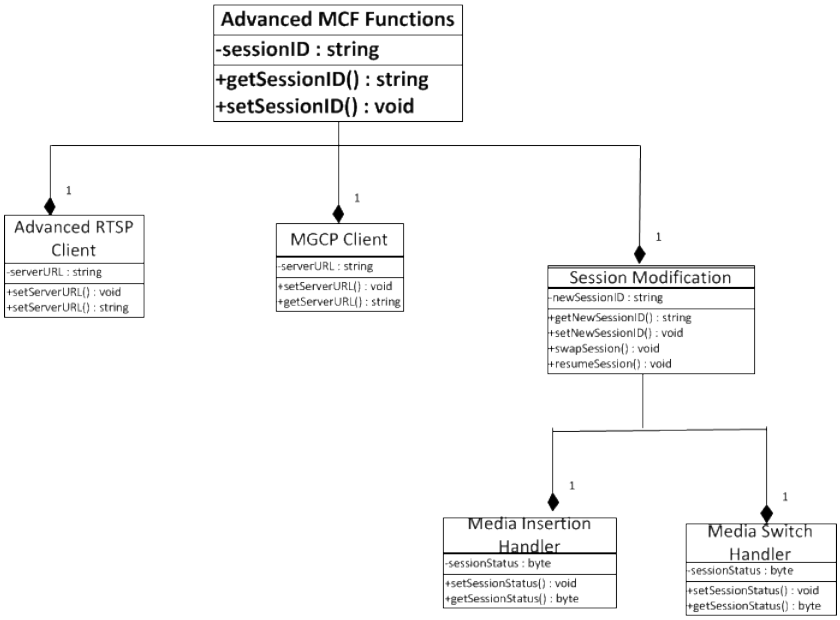



Figure 6.7: Class diagram for the modified MCF

different tables. As a result, we have organized the data items we identified as important for the building of the environment as database tables. Table 6.3 shows the snippet of the tables identified in this thesis. The table presented in Appendix I shows the data dictionary of all the tables.

The User Profile Manager class, shown in Figure 6.6 is responsible for handling the different types of data and data manipulation tasks that involve different technologies by hiding the complexities of data manipulation. It provide this feature by providing similar methods required for CRUD operations for the different storage technologies to the AS. It also includes methods for the purpose of subscription and notification of user profile data related events.

| Tables | Field | Type | Null | Default | Remark |
|---|---|---|---|---|---|
| Users | User Id | String | No | | IMPU |
| | ... | | | | |
| Video | Video Id | String | No | | |
| | Director Id | String | No | | |
| | ... | | | | |
| Reminder | User Id | String | No | | |
| | Program Id | String | No | | |
| | ... | | | | |
| ... | | | | | |

Table 6.3: Snippet of the schema of tables identified in this thesis

## 6.3 Summary

The previous chapters presented our experimental work to identify issues related to the current architectures and technologies. The experiments led us to the identification of various problems and gave us insights to the develop the requirements for new and innovative services. We summarized the outcome of those activities in this chapter by compiling the requirements of the environment and developing the architecture of the envisaged environment. Basically, this chapter has presented one of the major part of our work, which is the compilation of the requirements of the environment and presented its design. We specifically focused on the requirements that were not discussed in detail in the standard documents and also the new requirements that we identified from our experimentation and analysis of the different standard documents.

The design activity produced the architecture of the environment. We also presented the different components that will help service developers to develop services easily and fast. In general, we have presented the algorithm of the different subsystems and modules that service developers may need to utilize to develop IPTV services.

# Chapter 7

# Selection and Augmenting of the Environment

The previous chapter presented the requirements and design of the environment that we want to develop. This chapter concludes our selection activity and how we augment the selected environment through prototypical implementation of the major components. The chapter clearly shows how we implemented the different implementation issues presented in the previous chapter such as the message communication mechanism between the different components.

As part of our selection activity, we have motivated how Mobicents can be used as a basis for developing our environment. In fact, the use of Mobicents as an extensible open source service development platform for NGN/IMS is also justified by other researchers [111, 17, 112]. The issues that we considered as implementation challenges when designing the environment can be handled by Mobicents easily. Since we used Mobicents as a basis for our service development and deployment environment, we start the chapter by presenting some of the advantages we get from using Mobicents as a basis for our environment. This is followed by the actual implementation of some of the components that we developed to augment it. Finally, we present how the RTSP extensions proposed in this thesis can be implemented using the Proxy and how it fits into our environment.

## 7.1 The Advantages of Using Mobicents

There are many advantages we get by using Mobicents as as a basis for our service development and deployment environment.

### 7.1.1 How Mobicents Adapt Protocol Messages

One interesting feature of Mobicents is its protocol agnostic nature. Mobicents adapts protocol messages from various IP protocols as events through different Resource Adapters (RAs), which constitute an abstract interface layer that allows ASs to access external resources [113]. As mentioned before, the different ASs that service developers need to develop could use different protocols but the service development environment needs to provide a consistent interface for ASs to access these protocol messages. ASs receive protocol messages (which could be SIP messages, database responses, XCAP messages, etc.) just as regular events with a similar structure (as Plain Old Java Object or POJO). In its current implementation, the Mobicents service development platform supports most of the protocol requirements of our video-oriented service development environment. It is also easy to to develop a RA for a new protocol that we may have in the future.

### 7.1.2 How Mobicents Support Message Communication

Once a message passes through a RA, it is dispatched to interested SBBs by the help of the event router. The event router is the module responsible for creating new service instances and delivering events to all interested service building blocks (SBBs). Each component in the service delivery environment can subscribe to the events that it's interested in and receives messages accordingly. Figure 7.1 shows how ASs interact with Mobicents RA through the help of the event router. The event router is also used to transmit events (messages) between SBBs with SLEE.

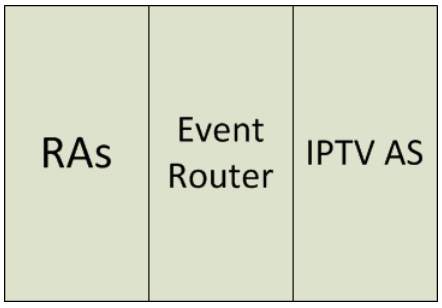



Figure 7.1: Interaction of the AS with other entities

### 7.1.3 How Mobicents Support Information Sharing Between Different Applications

One major issue discussed in the design section of this thesis as implementation issue was the problem of implementing information sharing across different sessions and also how different SBBs would share information when they need to process the same request one after the other. Mobicents solves the problem of coordination of different sessions through the use of the parent-child relationship of service building blocks (SBBs) and also solves the problem of information sharing using the features of activity context. Activity contexts are components that represent a dialog in Mobicents. SBBs attach themselves to an activity context to get access to the messages that belong to the dialog. So, all SBBs that want to process the same request can be attached to the activity context and retrieve or send message using it. On important thing about activity contexts is that we can define variables on them to share information (the outcome of a certain process) between SBBs who attached themselves to it. For example, we can set a variable on an activity context to represent the completion of processing a request by a particular SBB in the pipeline.

## 7.2 Realization of the Environment Through Mobicents

As mentioned before, in Mobicents, services are developed as service building blocks (SBBs). So, our main task in the implementation stage was to build reusable SBBs that service developers can use to develop different services. We have presented how we changed a furniture order and delivery system into a video streaming system in Section 5.3. So, service developers can also change the SBBs presented in this chapter to include other business logics to develop their own service. The implementation of SBBs that are important for the coordination of different types of session is also presented in this thesis. Another important thing that service developers need is the Pipeline that they use to create their own filters. A Pipeline is also used to share information between high level SBBs.

### 7.2.1 Organizing the Environment According to Main Components

As discussed in Section 6.2.3.1, the main functionalities that the environment needs to support are delivering Bootstrap service, handling of session initiation and modification,

and external service requests through the help of event router. As a result, as our first task, we organized the environment based on the main components presented in the design section of the previous chapter and developed the main components as root SBBs.

Accordingly, we developed root SBBs for coordinating the `Bootstrap Services`, for handling `Service Initiation` requests, and for handling `Service Management`. Because all services need to use users profiles, these root SBBs inherit from the `ProfileData` root SBB that contains the logic to communicate with XDMS and other databases to retrieve and update user and service profiles. In general, the three root SBBs coordinate the three main functions of the environment and under each of the root SBBs, we have other (child) SBBs that relate to specific service. So, for example, the Bootstrap Services root SBB handles the subscription and handling of users registration status and when it is notified about the registration of a user, it initiates any service that needs to be initiated as a bootstrap service like the `Auto Resume Last Failed Session`, which are developed as child SBB to it. We developed the root SBBs as Abstract classes with basic functionalities and service developer can later include service logic to develop a custom service as they wish. Service developers can also come up with new type of bootstrap services and can include it easily. The whole SBB tree is shown in Figure 7.2.

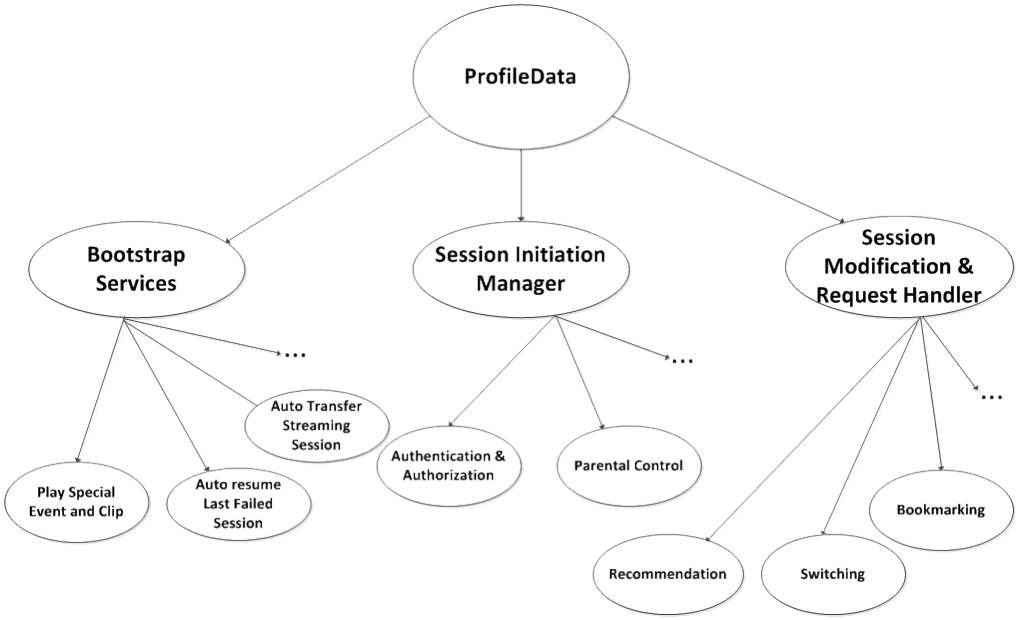



Figure 7.2: SBB tree of the personalized service development environment

### 7.2.2 Defining Root SBBs

We define a root SBB by specifying a request that initiates the SBB as `initial-event` in its descriptor file. This is done by setting the event as initial-event in the XML descriptor of the event handler of the SBB (saved as *sbb-jar.xml)*. For example, because all INVITE requests have to be processed by the `ServiceInitiationSBB`, we declared the SIP INVITE event as `initial-event` in its *sbb-jar.xml* file. Listing 24 shows the content of the sbb-jar.xml file of the `ServiceInitiationSBB` SBB.

**Listing 24** The sbb-jar.xml file of ServiceInitiationSBB

```
<event event-direction="Receive" initial-event="True">
    <event-name>InviteEvent</event-name>
    <event-type-ref>
        <event-type-name>javax.sip.message.Request.INVITE</event-type-
           name>
        <event-type-vendor>net.java.slee</event-type-vendor>
         <event-type-version>1.2</event-type-version>
    </event-type-ref>
    <initial-event-selector-method-name>
        InitiateService
     </initial-event-selector-method-name>
</event>
```

The sbb-jar.xml file also contains a *callback* method that will handle the event. In Listing 24, the callback method is `InitiateService`, which is declared using the `initial-event-selector-method-name` tag and invoked by JSLEE when the initial event is received.

The following sections present the implementation of the different components that are defined and presented in previous section.

## 7.3 Implementation of the Main Components and Basic Services

As mentioned above we have organized the main functionalities of the environment using three root SBBs specified as Bootstrap Service Coordinator, Session Initiation, and Session Manager. In this section we present the prototypical implementation of the components and some of the basic services under these SBBs, some of which are called *filters*.

### 7.3.1 The Bootstrap Service Coordinator

In order to deliver services that need to be initiated at bootstrap, the Bootstrap Service Coordinator (developed as `BoostrapService` SBB) needs to get the registration status of the user and then deliver the services based on the user's preferences. For this purpose it may need to contact other components as well. As mentioned in Section 6.2.4.3, there are two scenarios related to this service. The scenarios relate to which device the user boots up. If it the user boots up his device while he is logged in with another device, the Auto Transfer service is initiated. However if it is the first device then other sets of services need to be initiated. The following sections present how the three major activities of this SBB are implemented for the scenario representing first device boot up happens. The Auto Transfer service is not implemented and is discussed in the future work section of the thesis.

#### 7.3.1.1 User Registration Status Subscription Module

The registration status of UEs can be obtained by subscribing to the SIP REGISTER event from HSS. Mobicents includes an example SBB for subscribing to user registration status from HSS and we also utilized it to develop this SBB.

The first action an SBB needs to do before it can subscribe to any event is to connect to HSS. We use a configuration file to store the basic Attribute Value Pairs (AVPs) required for establishing the connection between the AS and HSS. The AVPs we used include `originIP` and `destinationIP`.

After establishing connection with HSS, the SBB creates a subscribe notification request, `Subscribe-Notifications-Request (SNR)`, and use the `sendMessageSync` method to send the request for user registration status update. The `Subscribe-Notifications-Request` requires the user's public id `(UserPublicId)` among other things. The `createSubscribeNotificati` method is used to create the subscription request, getting most of the data from the configuration file. The following code fragment shows how the request is created and sent.

```
Request message = createSubscribeNotificationsRequest(user)
Answer ans = (Answer) provider.sendMessageSync( message );
```

If the subscription is successful then the AS get a response with result code 2001.

#### 7.3.1.2 User Subscription Status Notification Processing Module

The notification data that come from HSS is referred to as `Push-Notification-Request (PNR)` and is a structured data that include the status code in addition to other information related to the user, like the `UserPublicId`. The SBB, however, can request to get only specific data. For example, if the SBB wants to get only the `User State`, it can do so by setting the `DATA_REFERENCE_AVP` field of the request to 11. A value of 0 means that the AS wants all the reference data.

Accordingly, when the user registers, and the AS receives the notification data from HSS, it initiates the `processRequest` method. If the response comes with all the data, then the AS needs to extract the specific information that it wants to process appropriately. For example, the following code is used to extract the `User State` from the response that contains all the reference data.

```
AvpSet avps = request.getAvps();
String userData = avps.getAvp( DiameterShCodes.USER_DATA_AVP ).getOctetString();
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
Document doc = (Document) builder.parse(new InputSource(new StringReader(userData)));
String userState = doc.getElementsByTagName("IMSUserState").item(0).getTextContent();
if(userState.equals("1"))
{
   . . .
}
```

The user's public identity can be found using the following statement:

```
String userPublicIdentity = avps.getAvp( DiameterShCodes.USER_IDENTITY_AVP).
getGrouped().getAvp( DiameterShCodes.PUBLIC_IDENTITY_AVP ).getUTF8String();
```

#### 7.3.1.3 Accessing Other Important Data for Delivering the Service

The activation of each of the Bootstrap services also depends on other service related information. For example, the Bootstrap Service Coordinator activates the `PlaySpecialClips` SBB only if today is the user's birth date or his wedding anniversary. So, once the Bootstrap Service Coordinator is initiated, it gets the user profile data from XDMS as explained in Section 5.2 in order to decide on the initiation of the different services.

As mentioned before, we have extended the IPTV user profile schema to include `Birthdate`, `Anniversary`, and `LastFailedSession`, which are important information for initiating bootstrap services. Accordingly, the Bootstrap Service Coordinator first checks if today is the user's birthday or anniversary and initiates the above mentioned SBB.

The `LastFailedSession` element of the user profile, is a structure and contains information like the `URL` of the video of the last failed session if there was a failed session, and `LastFailedSessionBookmarkPosition` among other things. The SBB retrieves and records these information as follows:

```
UserProfile up;
Date dob = up.getDateOfBirth();
Date ann = up.getAnniversary();
String lfs = up.getLastFailedSession();
String lfsbp = up.getLastFailedSessionBookmarkPosition();
```

Accordingly, the Bootstrap Service Coordinator forwards the information to the different SBBs that contain the service logic for the different bootstrap services that the user is subscribed to. The SBBs related to bootstrap service `are PlaySpecialEventClip` and `AutoResumeLastFailedSession`. The services are developed as child SBBs to the `BoostrapService` SBB. After the system plays the birthday or anniversary clips, it delivers the last failed stream, if there is one, through the `AutoResumeLastFailedSession` SBB. After the streams from the failed sessions are played, the system clears the information from the user profile.

Because users may have their own language preference, the birthday and anniversary clips are stored in a table called `SpecialClips`. The implementation of new video notification involves different technologies and explained below.

### 7.3.2 Session Initiation Handler

The thesis is full of services that show the implementation of session initiation without the use of filters. So, in this section we present how filters are implemented for the purpose of session initiation.

As mentioned before session initiation involves the execution of various filters, which include: authentication and authorization, parental control, device capability checking, and charging and account information. Basically, the user is allowed to watch the channel that he requested, if he is authenticated, has enough credit, and has PC clearance. Of course, his device should also be capable of supporting the delivery of the video he requested.

Based on the user's configuration information, the system may also initiate other modules, like a feedback manager after the session is initiated.

In general, for reasons mentioned in the previous section, the main thing one needs to consider when developing filters is providing a way of sharing the outcome of the process of each filter to other filters in the pipeline. As mentioned before, Mobicents allows developers to achieve this using techniques called *variable aliasing* and *custom activity context*.

For our environment, information sharing happens at two levels. The root SBBs need to share the session they are handling so that other SBBs can modify and include additional service at run time and also the SBBs under the root SBBs may also need to share information (to know the outcome of the process of request by the previous SBB in the pipeline). As a result, we have different pipelines. Figure 7.3 shows how different processes share information at different level. This is a major contribution of this thesis. The pipelines can easily be modified by modifying the configuration files (service descriptors). For sharing of information between the root SBBs we use a special form of activity context called *Null Activity Context*. The implementation of both types pipelines is the same.

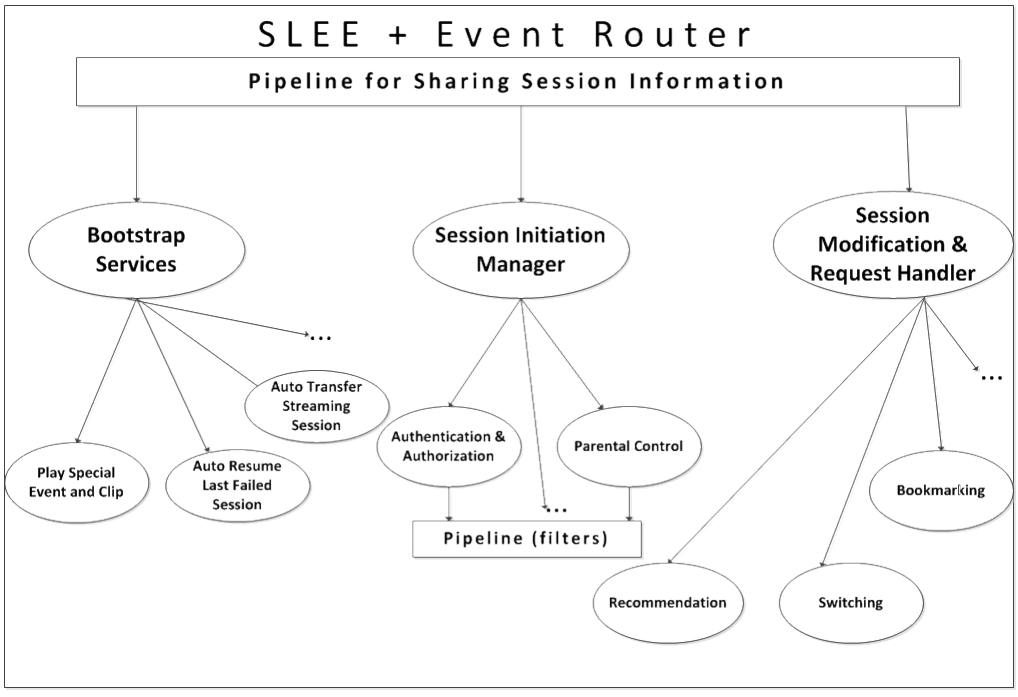



Figure 7.3: SBB tree showing the different pipelines

#### 7.3.2.1 An Example Pipeline from the Mobicents Service Development Platform

The Mobicents team come up with an example that shows the use of custom activity context to share data between SBBs in [114]. The example demonstrates how a call control service composed of call blocking, call forwarding and voicemail services that run in the order presented here utilizes custom activity context to share data between them. The way the example works is that the first filter (call blocking) checks if the caller is blocked by the user and if the number is not blocked, it forwards the call to the next filter (call forwarding service). The call forwarding service then checks if the user provides a call forward address and if so, it forwards the call to that address, otherwise, it forwards the call to the next filter, which is the voicemail service. The voicemail service checks if the user has subscribed to the service and if so, it requests the caller to leave a message and then records the message to the user. Figure 7.4 shows how variable aliasing is used to share data between the different components of the call control service.

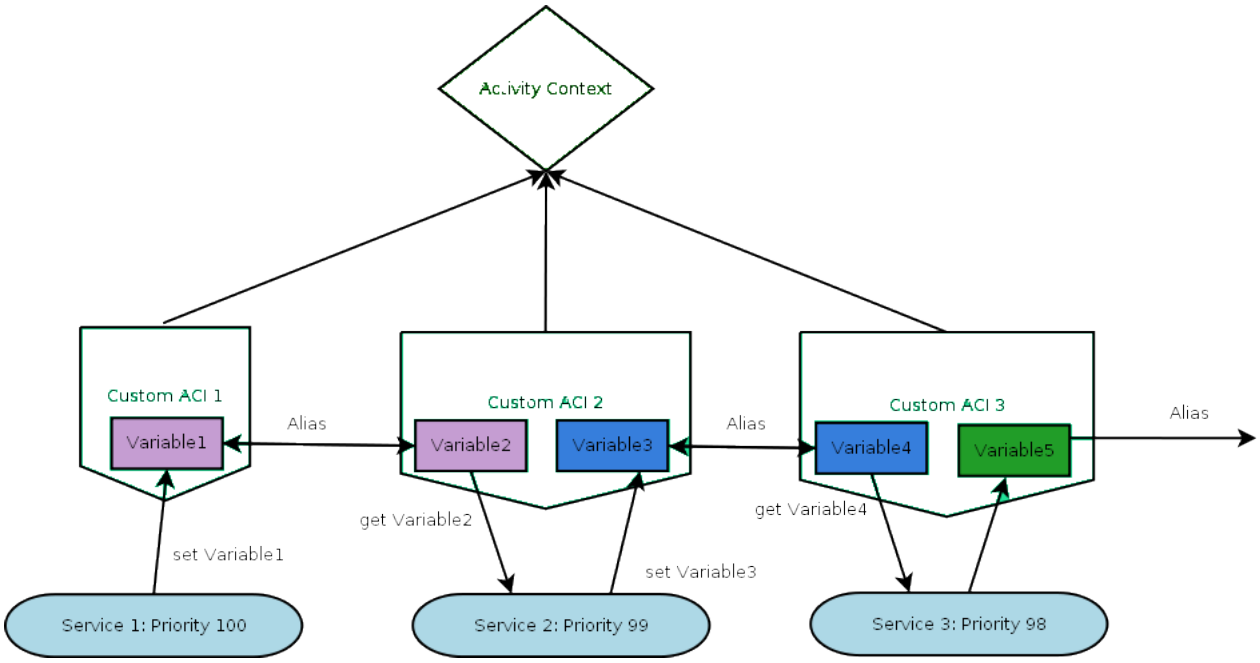



Figure 7.4: Variable aliasing example. Source: [114]

#### 7.3.2.2 Implementation of the Pipeline for Session Initiation

We also used the variable aliasing technique to develop the pipeline and share the outcome of the processes of the request by the different SBBs used to filter the request. Specifically we use custom ACI for this purpose. Custom ACIs and variable aliases are defined in

the service descriptor of the SBB (sbb-jar.xml file). The sbb-jar.xml file for the session initiation filters is presented in Listing 25.

One advantage we have when using ACIs is that we can include and remove filters easily and the system always run smoothly. As we can see from Listing 25, we can use anonymous names, like `inviteProcessedByPreviousFilter`, to refer to an SBB that processes the message just before the current SBB. This is a very useful design principle for video-oriented services such as the IPTV service where we may want to add or remove a particular filter based on the user's configuration information. The following sections discuss the implementation of the different filters used in this thesis.

**7.3.2.2.1 User Authorization and Credit Checking Services** The user authentication and credit checking features are developed by the Mobicents team and other researchers [115, 116, 117]. As a result, these modules are not discussed here. The other filters, particularly the PC service uses different technologies and is a bit complicated. We present this service in in the following section.

**7.3.2.2.2 Parental Control Procedures** The basic procedure for parental control is presented in the IPTV specification [94]. However, as mentioned in the requirements section, this module may need to contact PC authorizer and this is what makes it a bit complex. Figure 7.5 shows the call flow diagram when the PC authorizer needs to be involved. For simplicity, the AS also includes the work of the MCF in the call flow diagram. The flow diagram shows the flow of messages when the response from the authorizer is positive. As it can be seen from the diagram, there are two main tasks that the AS does to deliver this service, which are the contacting of the parental control authorizer and the allowing or denying of the service based on the feedback it gets from the authorizer. For the purpose of contacting the parental control authorizer to get his feedback, we utilized the code from the Converged Video service, which is presented in Section 5.3. The PC Call Controller SBB (`PCCallControlSBB`) is the component responsible for establishing a connection between the PC Authorizer and Media Server and also supplies the Media Server with a text message to play as media (e.g. User XXX is going to watch video YYY and parental authorization is required. Would you allow him to watch the video now?). The Authorizer supplies his feedback by pressing a key in his keyboard. The Media Server then forwards the feedback back to the initiating SBB. The `PCCallControlSBB` then forwards the decision to the AS.

**Listing 25** Custom ACI variables for service initiation filters

```
<sbb>
   <description />
   <sbb-name>AuthenticationSbb</sbb-name>
   <sbb-vendor>za.ac.ru.convergence</sbb-vendor>
   sbb-version>0.1</sbb-version>
   <sbb-classes>
   ...
      <sbb-activity-context-interface>
      <sbb-activity-context-interface-name>
         za.ac.ru.convergence.service.authorization.authentication.
            authenticationSbbActivityContextInterface
      </sbb-activity-context-interface-name>
      </sbb-activity-context-interface>
   </sbb-classes>
   ....
   <activity-context-attribute-alias>
   <attribute-alias-name>inviteFilteredByAuthentication</attribute-alias
      -name>
   <sbb-activity-context-attribute-name>filteredByMe</sbb-activity-
      context-attribute-name>
   </activity-context-attribute-alias>
</sbb>
<sbb>
   <description />
   <sbb-name>BalanceCheckingSbb</sbb-name>
   <sbb-vendor>za.ac.ru.convergence</sbb-vendor>
   <sbb-version>0.1</sbb-version>
   <sbb-classes>
      <sbb-abstract-class>
      ...
      <sbb-activity-context-interface>
      <sbb-activity-context-interface-name>
         za.ac.ru.convergence.service.authorization.BalanceCheking.
            BalanceChekingSbbActivityContextInterface
      </sbb-activity-context-interface-name>
      </sbb-activity-context-interface>
   </sbb-classes>
   <activity-context-attribute-alias>
   <attribute-alias-name>inviteFilteredByAuthentication</attribute-alias
      -name>
   <sbb-activity-context-attribute-name>filteredByPreviousSBB</sbb-
      activity-context-attributename>
   </activity-context-attribute-alias>
   <activity-context-attribute-alias>
   <attribute-alias-name>inviteFilteredByBalanceChecking</attribute-
      alias-name>
   <sbb-activity-context-attribute-name>filteredByMe</sbb-activity-
      context-attribute-name>
   </activity-context-attribute-alias>
</sbb>
<sbb>
   <description/>
   <sbb-name>PCSbb</sbb-name>
   <sbb-vendor>za.ac.ru.convergence</sbb-vendor>
   <sbb-version>0.1</sbb-version>
   <sbb-classes>
   <sbb-abstract-class>
      ...
      <sbb-activity-context-interface>
      <sbb-activity-context-interface-name>
         za.ac.ru.convergence.service.authorization.PC.
```

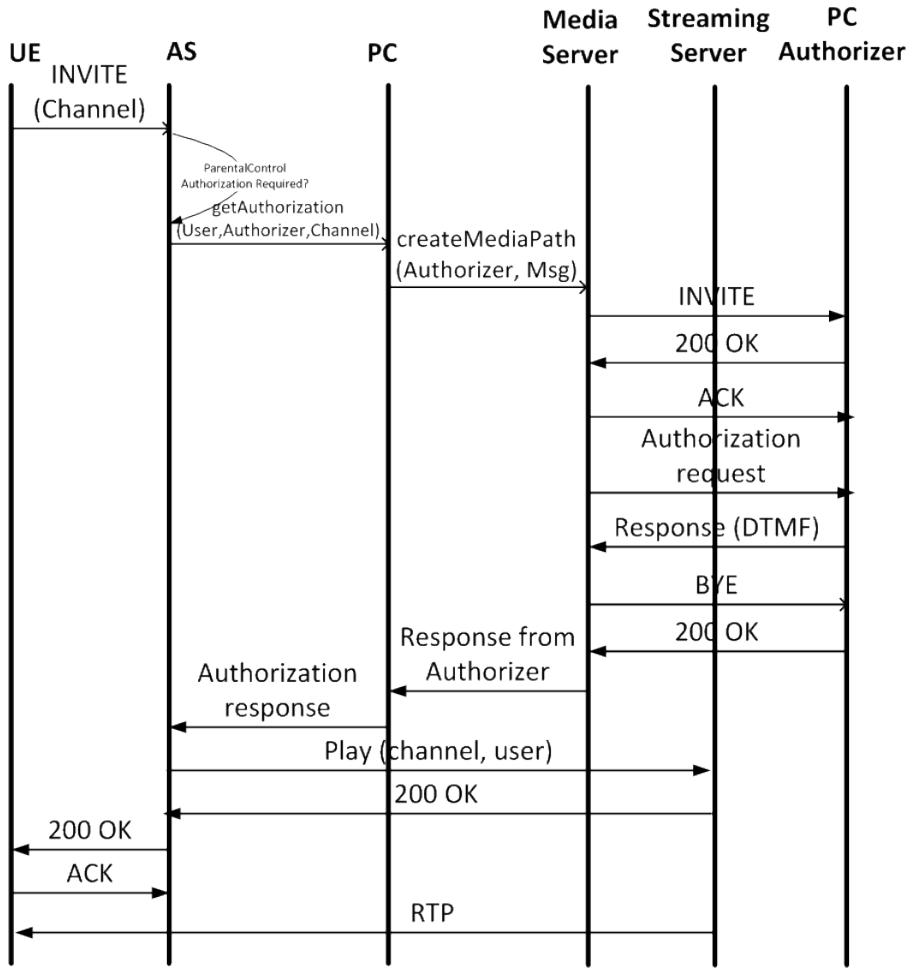



Figure 7.5: Parental control procedures

### 7.3.3 Service Manager

One basic principle we used to implement the extensibility of the environment is the separation of session initiation from session modification. As a result, all services that lead to session modification (both signaling and media) are coordinated by the Service Manager. The Service Manager is developed as an SBB and handles all requests other than the SIP INVITE request, which is handled by the Session Initiation Manager. Services like Stream Switching, and Recommendation and Notification are developed as child SBBs to this SBB. For the purpose of media session modification, the Service Manager also needs to communicate with the MCF. Two services that relate needs to be handled by Service Manager are new video notification and Bookmarking service.

The following section presents implementation of new video notification service.

#### 7.3.3.1 New Video Notification

There are two modes in which this service can be developed, push and pull mode. The push mode is where the notification comes from the AS serving the user automatically, whereas in the pull mode the user asks for the availability of new video.

**Push Mode Video Notification** As mentioned before, in Mobicents, notifications are communicated by firing events inside SLEE. This implies, for non-standard events, like the arrival of a new video, one needs to first define custom events to communicate to JSLEE services. Events are defined by XML descriptors that describe the event class. The work of event descriptors is to tie the logical *event id* (*name, vendor*, and *version*) with an *event class*. An event class can contain different custom events and holds information necessary to process the events (like the URL of a new video or callID of a new call).

When discussing about the converged demo, we presented how an event that is generated from a website comes into SLEE through the SEAM framework. We can use the same technique to fire events from content providers website to ASs. When the content provider enters details of the new video on the Content Distributor Network (CDN) web page, then the system can be configured to send notification to JAIN SLEE. We can also implement this solution using the Repository Architectural Style where the repository notifies the insertion of a new item (video) in the repository to interested parties (in our case the SEAM Framework that eventually notify the AS through JSLEE). The following shows how a Converged Shopping Demo style of notification can be developed.

The Converged Shopping Demo uses the Mobicents JCA (Java Connector Architecture) RA to transmit communication messages between JAIN SLEE and the SEAM framework. The JCA is basically used to communicate information between different applications packaged in different forms. Figure 7.6 shows the communication of SEAM and SLEE.

As mentioned before, events are carried as objects. As a result, for the notification of new video, we need to define an event class to carry information about the availability of the new video. Listings 26 and 27 present the custom event descriptor and event class for the notification of the arrival of new video. As per the specification of SLEE, there are methods that any custom event class needs to include, and they are shown in the class. On the JAIN SLEE side, the signaling of the new video can be implemented through one of the available resource adapters, like the HTTP RA, and by extending one of its commands.

Once the AS is aware of the availability of the new video, it notifies the user about the new video by sending a text message and specifying when the new video is going to be

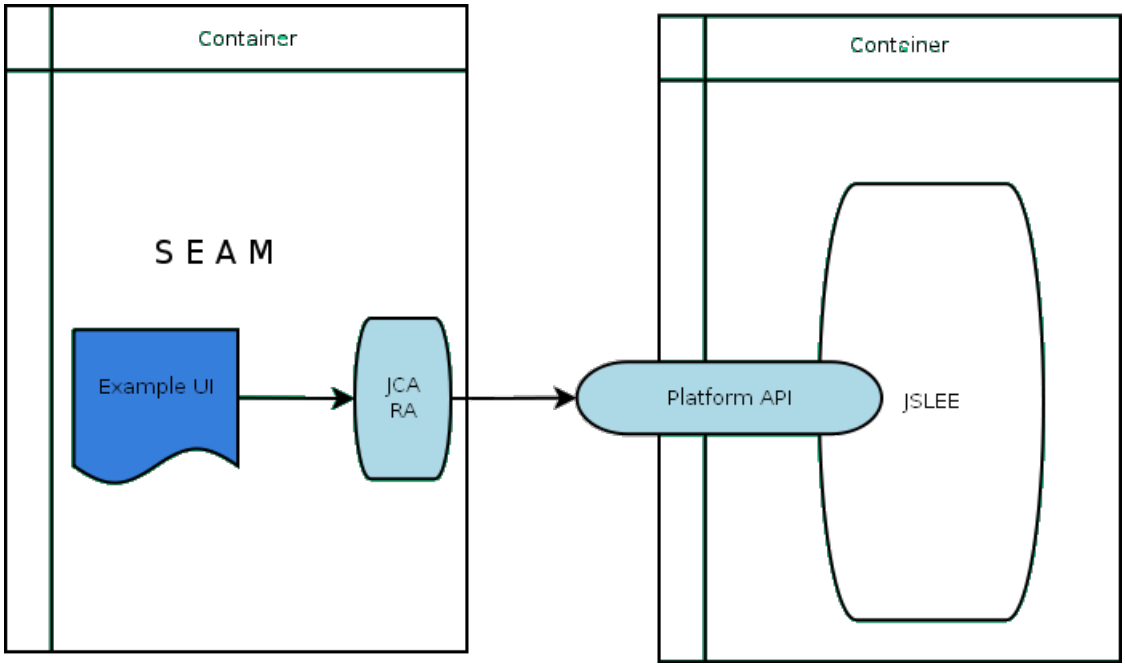



Figure 7.6: External service initiation example. Source: [118]

available. This is discussed in [63]. The AS does this for all new videos. That is, the AS sends a SIP MESSAGE command to the user if there are videos that the user is not notified about.

**Listing 26** Event descriptor for notification of new video

```
<event-definition>
   <description>Fired from New video notification Page. Initiates
      NewVideoHandlerSBB</description>
   <event-type-name>za.ac.ru.convergence.service.dvn.NEW_VIDEO</event-
      typename>
   <event-type-vendor>za.ac.ru.convergence</event-type-vendor>
   <event-type-version>1.0</event-type-version>
   <event-class-name>za.ac.ru.convergence.service.events.NewVideoEvent</
      event-class-name>
</event-definition>
```

**Pull Mode Video Notification** The pull mode of video notification happens when the user (the AS on behalf of the user) explicitly requests the availability of new video. In order to avoid notifying the user about the availability of the same video several times, the AS responsible for notifying about new videos checks when the user was last notified before issuing the query for new video. This information is recorded in the `LastDateNewVideosNotified` field in the `Users` table, which is presented in Appendix I. After the user is notified about the new video(s), the AS then updates the

**Listing 27** Event class for notification of new video

```
public class NewVideoEvent implements Cloneable, Serializable {
        private static final long serialVersionUID = 1L;
        private long id;
        private String title, description;
        private String actor, director;
        private integer yearReleased;
        private String url;
        private byte pc_level;
        public CustomEvent(COD_Profile cod) {
                this.id = cod.id;
                this.title = cod.title;
                this.description= cod.description;
                this.actor = cod.actor;
                this.yearReleased= cod.yearReleased;
                this.url = cod.url;
                this. pc_level = cod. pc_level;
        }
        public boolean equals(Object o) {
                if (o == this)
                  return true;
                if (o == null)
                  return false;
                return (o instanceof CustomEvent) && ((CustomEvent) o).
                   id == id;
        }
        public int hashCode() {
          return (int) id;
        }
        public Sring getTitle() {
          return title;
        }
        public String getDescription() {
          return description;
        }
        public String getActor() {
          return actor;
        }
        public String getDirector() {
          return director;
        }
        public int getYearReleased() {
          return yearReleased;
        }
        public String getURL() {
          return url;
        }
        public Object clone() {
        CustomEvent clonedCustomEvent = new CustomEvent(this.getTitle(),
            this
        .getDescription(), this.getActor(), this.getDirector(), this.
           getYearReleased(),
        this.getURL());
          return clonedCustomEvent;
        }
}
```

`LastDateNewVideosNotified` field of the `Users` table. For each new video, the AS receives the profile of the video which contains its details, like description, and URL.

The module responsible for preparing the EPG can also include the new contents specifically suggested for the user in a separate section.

#### 7.3.3.2 Implementation of Bookmarking Service

The implementation of the Bookmark and the Auto Bookmarking services depend on the streaming server's capability for providing the current media position when requested. As mentioned before open source streaming servers do not support this feature and we implemented this capability on the Proxy. In fact, the Proxy is important for the development of Auto Bookmark even if streaming servers support the provision of the current media position. The following section present how the Proxy fit into the environment in general.

## 7.4 Support from the Proxy for the Development of Innovative Video-Oriented Services

One area that the Proxy will be helpful is for the implementation of the RTSP extensions. As mentioned before, we have made several proposals for extending the RTSP protocol so that it can be used as a proper media control protocol. In fact, one objective of this thesis is to come up with an open source video service development and delivery environment and, as mentioned before, open source streaming servers lack some feature to serve as full-fledged MDF. So, we needed to use the Proxy in the environment to augment open source streaming servers so as to use them to build a complete open source environment. In this regard, we can say that the Proxy is one of the main components of the environment. In this section we present how we implemented the REPLACE command that we recommended for media switch (especially for recommendation purpose).

To show how the Proxy supports the development of innovative video-oriented services, we present the development of Bookmark and Auto Bookmark services..

### 7.4.1 Implementation of the RTSP REPLACE Command for Media Recommendation Service

We configured the MCF to send the REPLACE command (with the URL of the media to be recommended) to the Proxy a few minutes before the media ends. The MCF uses the Timer utility of Mobicents to get a reminder to send the command. As mentioned before, REPLACE has a parameter that specifies if the replacement should be done immediately or when the current media ends. In our case we used the REPLACE command with the later option. So, when the Proxy receives the REPLACE command, it initiates a new session. A new thread (using the ServerSide class) was defined to do this. The Proxy stores the information about the new session on a variable in the ProxySession class for swapping the session later. When the current media ends, it sends a notification message to the MCF using the RTSP OPTIONS command. When it gets a response from the MCF, it then swaps the current session with the session associated with the REPLACE command and send the RTSP PLAY command on the new session to the streaming server to start delivering the new media automatically. The user then start to watch the new video using the same connection that he was using to get the previous media. Figure 7.7 shows the flow diagram for the implementation of the RTSP REPLACE command.

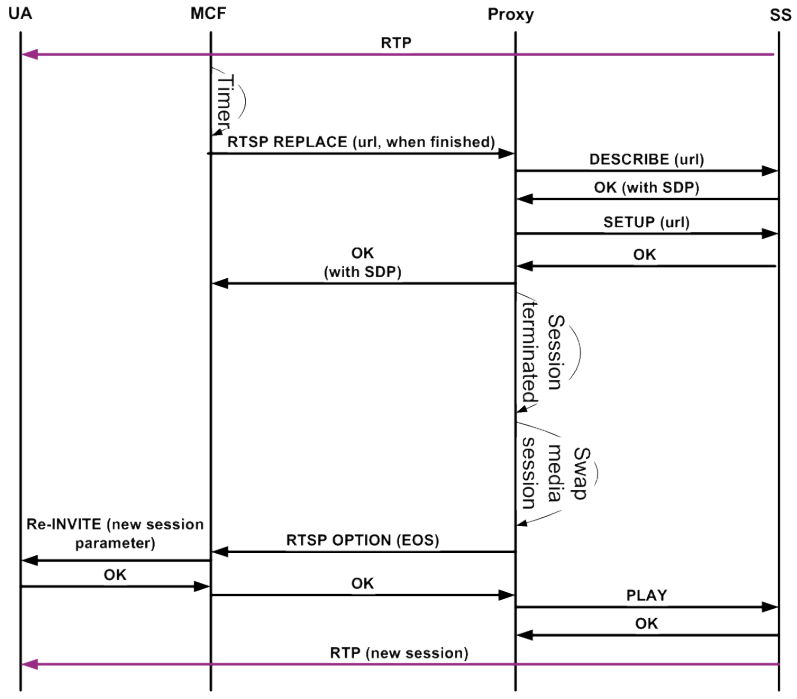



Figure 7.7: Flow diagram for the implementation of the RTSP REPLACE command

### 7.4.2 Modifications Made on the Proxy to Support Bookmarking and Auto Bookmarking Service Development

As mentioned before, the position that we put a bookmark for a media should be a position that can be played without a problem when we want to resume watching the video next time. In other words, it should be a position of an I-frame. We implement this feature in the proxy as follows.

Because GOP coding is only supported by certain codecs like H263 and H264, we implemented this feature for these codecs. So, if the video is coded using one of these codecs (which are actually the default codecs for IPTV), the following happens.

In the module of the Proxy that forwards RTP packets to the client, we included a code that checks if an RTP packet that it receives is an I-frame or not. If it is an I-frame, then it save the frame's timestamp in a variable called the `lastIFramePosition`. An I-frame, also called *the IDR slice* of an RTP payload for an H264 encoded packet, can be identified using the first byte of the payload. The value of first byte of an H264 encoded packet should be 1 for I-frame packet. So, when a user sends a Bookmark request, the Proxy retrieves the value from the `lastIFramePosition` to calculate the bookmark position and returns it to the AS. The following code fragment is used to store the timestamp of the last I-Frame.

```
byte [] payload = packet.getPayload();
byte frametype = payload[0];
if (frametype ==1)
   unsignedInt LastBMP = packet.getTimestamp();
```

The AutoBookmark module also uses the same technique to record the timestamp of last I-frame before a stream terminates abnormally. As mentioned before, an AutoBookmark service is a service that constantly updates the last time a media is delivered properly before the system crashed. So, if the media was not terminated properly (i.e. if the proxy lost contact with the media server), then we record the `LastIFramePosition` together with the `URL` in the user profile for the purpose of providing AutoResume service. In socket programming, the detection of a lost connection is called *heartbeat monitoring*. Apache MINA, the network application framework that the Proxy was developed, has a mechanism to check if a connection is lost. So, we used this mechanism to detect if the Proxy lost the connection from the server. This is done by setting up an `IdleTime` for the I/O Session that we want to detect if it gets idle. Listing 28 shows how we set an idle time for a session. As a result, when the session terminates abnormally, MINA fires

a `sessionIdle` event and control transfers to the handler. Listing 29 shows an example of a `sessionIdle` handler.

**Listing 28** Setting an idle time for a session

```
acceptor.getSessionConfig().setIdleTime( IdleStatus.BOTH_IDLE, 10 );
```

**Listing 29** Idle session event handler

```
@Override

public void sessionIdle( IoSession session, IdleStatus status ) throws
    Exception

{

        System.out.println( "IDLE " + session.getIdleCount( status ));

        ProxyHandler proxyHandler = (ProxyHandler) ( session.
            getAttribute( ProxyHandler.ATTR ) );
        Track track = proxyHandler.getRelatedTrack();
        int currentTime = track.getFirstTimestamp().intValue() - track.
            getCurrentTimeStamp().intValue()
        RtspRequest request = new RtspRequest();
        request.setVerb("ANNOUNCE");
        request.setHeader("Session", sessionid );
        request.setSequenceNumber(lastsequencenumber+1);
        request.setNotice("Abnormal Termination Of Session");

}
```

The following section presents how the new video notification service is developed.

## 7.4.3 The Extended User Profile

### 7.4.3.1 The Extended User Profile and Accessing Technologies

As mentioned in the requirements section of this thesis, the user profile needs to be modified to include user profile data (attributes) that are important for other services, like email address, telephone number, and birth date. We modified the schema to include these attributes and created the appusage and the parsers so that services can access the user profile with these attributes. The modified appusage and the XML parser are included in the DVD attached at the end of this thesis. Mobicents comes with an XCAP client and we use it to extract user profile information from the XDMS server. Besides this, there is also a code for a JDBC client for the purpose of interacting with any database using the JDBC connectivity. All these clients are part of the environment and are included in the attached DVD.

## 7.5 Summary

This chapter concludes our work of compiling the video-oriented service development and deployment environment. The chapter discusses how the different aspects of the environment can be implemented and presented the actual implementation of the different components of the environment.

The discussions presented in the chapter made it clear how an event-based delivery environment, such as Mobicents, is a good development and delivery environment for video-oriented services, such as IPTV.

The chapter also demonstrated how the Proxy can be used to implement the new protocol extensions that we came up for RTSP until they become standards. In general, we can conclude that Mobicents, with all the different components that we developed, together with the Proxy can be an easy to use service development environment to develop personalized and converged video-oriented services for IMS. The next chapter presents testing of the environment showing how we configure and used to develop video-oriented services.

# Chapter 8

# Testing and Discussion

This chapter presents the results of testing of the environment by showing the proper interaction of the different components of the environment with other units like the IMS core and streaming server that are necessary to provide a service to the user and also by showing how the environment supports the development of innovative services.

Because our work is entirely on the server side, the testing of the environment needs to be done by developing the services that we expect from the environment and accessing the services (checking if the functionality are provided) using an appropriate UE (an IPTV client).

Because there is no proper open source IPTV client, we developed a stub to simulate the requests from UEs for some of the tests. For example, for requests that require an extended RTSP client, we simulated the requests using the command line client from LIVE555. We also did this to the other protocol requests by putting a stub in the AS to simulate requests as coming from the UE or other appropriate places. The following section present the whole process of the testing activity.

## 8.1 Functional Testing

We have presented several functionality of the environment in this thesis. For example, we presented the iVideoMail service, which with little modification, can serve as PVR (please see Section 3.1). The Personalized Dynamic Video Delivery system is another system in which we demonstrated how the different components that we build for the environment can be used to deliver a personalized video service (see Section 5.2.2). As mentioned in Page 86, we also showed how the Proxy can be used to develop an open source MDF

and published the work in a Journal. In this chapter we present functionalities of the environment that have not been presented before. The purpose of this chapter is to show the use of the main functionality of the environment.

### 8.1.1 Test Cases Considered

We present our test to demonstrate the new functionality of the environment from two perspectives. We demonstrate how an AS can use external components like XDMS and HSS and also show how the new media-related features like Autoresume can be used. We use the following hypothetical cases to demonstrate the environment.

#### 8.1.1.1 Test Case One (Media Server-Related)

One of the services presented as Bootstrap Service is the Autoresume service. This service is initiated automatically when the user boots up his device and there was also a failed session (i.e. abnormal termination of the video that the user was watching) in his last usage of the system. This could happen if a streaming server that was serving the user was terminated abnormally (e.g. because of power failure). To simulate this, a streaming server was forced to terminate, so that the Proxy could identify the situation and sends a message (stream terminated or STTE event) to the MCF. For the purpose of simplicity, we combine the MCF and the AS in this chapter. As a result, the the Proxy sends the event to the AS. When the AS receives the message that a stream has terminated abnormally, it properly terminates the SIP session and stores the required information in a table (Last Failed Sessions) so as to resume the stream next time the user boots up his device.

#### 8.1.1.2 Test Case Two (Complex Service Creation)

A user profile with the current date as the user's birth date, among other attributes, is created to demonstrate bootstrap services. The AS subscribes to the user's registration status with HSS. When the UE boots up, HSS informs the AS about the user's registration status. The AS contacts XDMS to get the user's profile and checks if today is his birthday and, if so, initiates a Happy Birthday video clip from the media server.

### 8.1.2 Setting up the Environment

To test the above services, we needed to configure the environment with the required services and user data. As a result, we first show how we configured the environment and created the required data.

#### 8.1.2.1 Downloading and Installing Servers

The major components of our testbed for the series of experiments presented in this chapter are: the Open IMS Core, Mobicents, the VLC streaming server, and the RTSP Proxy and Relay unit. The Open IMS Core was set up and installed on a desktop computer that runs Dual Core Intel 2.66 GHz PC with 2 GB RAM. We also installed VLC and Darwin Media servers on the desktop computer.

The installation of the Open IMS Core requires the setup and configuration of a database and a DNS server. The most common servers proposed by the Linux community for this purpose are MySQL database and the BIND9 DNS server. Accordingly, we also chose MySQL 5.1 as a database server and BIND9 as a DNS server. The default domain name open-ims.test was used for the testbed.

Mobicents comes with two different programming technologies, which are JAIN SLEE and SIP Servlet. For each technology Mobicents provides different download options for service developers to use. We downloaded and installed the binary version of the Mobicents JAIN-SLEE 2.7.0 integrated package. The package contains two servers: the default Mobicents server and also the presence server. In order to include the Diameter functionality, we used the Mobicents Diameter Server 1.3.1 FINAL binary package. We used the Eclipse IDE to run the Mobicents JAIN-SLEE default server and also to test our services.

For testing of user profile-related features, we used the Mobicents SIP Presence Integrated binary package as an XDMS server, which is stored in the presence package of the JAIN SLEE default server. The IPTV app usage that we developed and presented in Chapter 5 is installed on the XDMS server. We installed both the Mobicents JAIN-SLEE and Presence servers on a Lenovo i5 Quad Core laptop with 2.4 GHz processor and 4GB RAM that runs the Ubuntu 14.10 LTS operating system. To run the the Mobicents SIP Presence Integrated server as a separate instance, we needed to issue the following command:

./run.sh -c presence -Djboss.service.binding.set=ports-01 -Djboss.messaging.ServerPeerID=1

Figure 8.1 show the two instances of Mobicents.

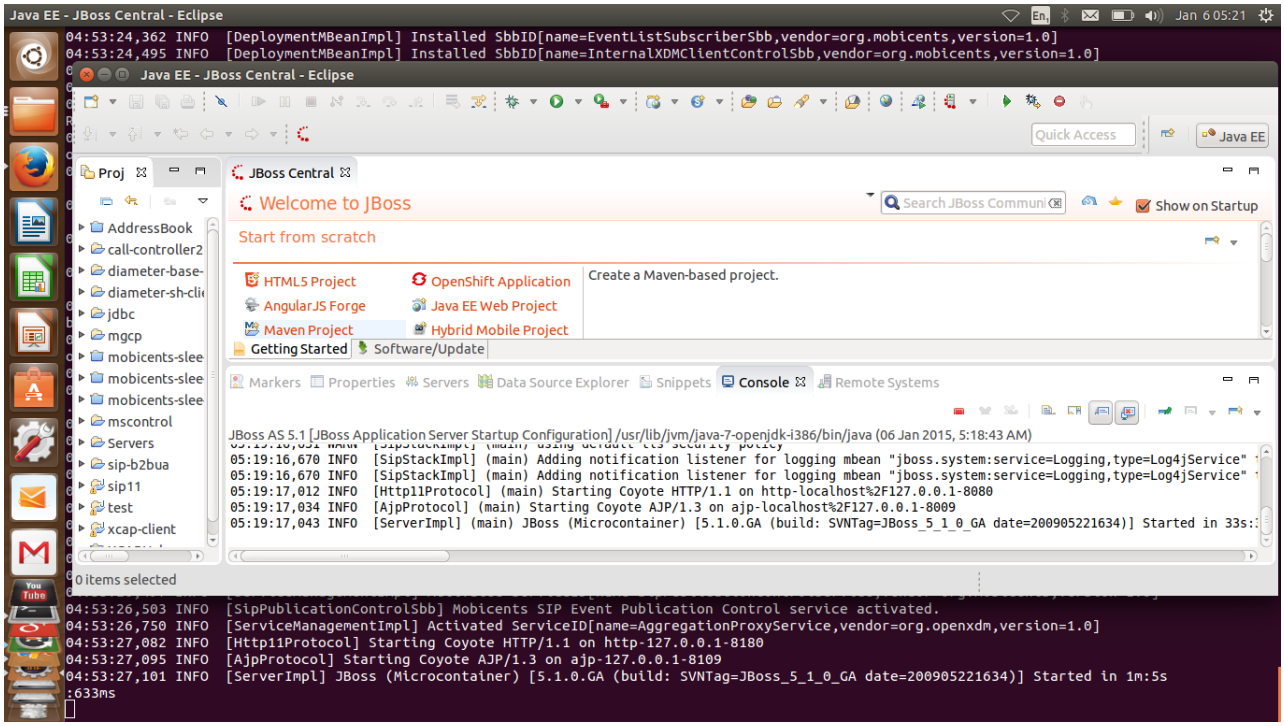



Figure 8.1: Multiple instances of Mobicents

We used the Netbeans IDE to run the Proxy on the same machine that runs the IMS core and the media servers.

#### 8.1.2.2 Integration of Mobicents and OpenIMS Core

In order for the ASs that run on Mobicents to interact with the IMS Core, we needed to configure them through the FHoSS web console. These are ASs that need to accept requests from the UE through the IMS Core. Figure 8.2 shows the application server configuration for a test service. The AS is also configured to subscribe users registration status from the HSS when it starts.

#### 8.1.2.3 Setting up the User Profile

The user profile management module is the core of our environment. It provides services that most of the other components need. For example, the services grouped as Bootstrap Services need to check the user's birth date or anniversary among other information to provide their service. So, the first part of the test is to create a user profile with appropriate data and check if it is accessible. The following section presents this.

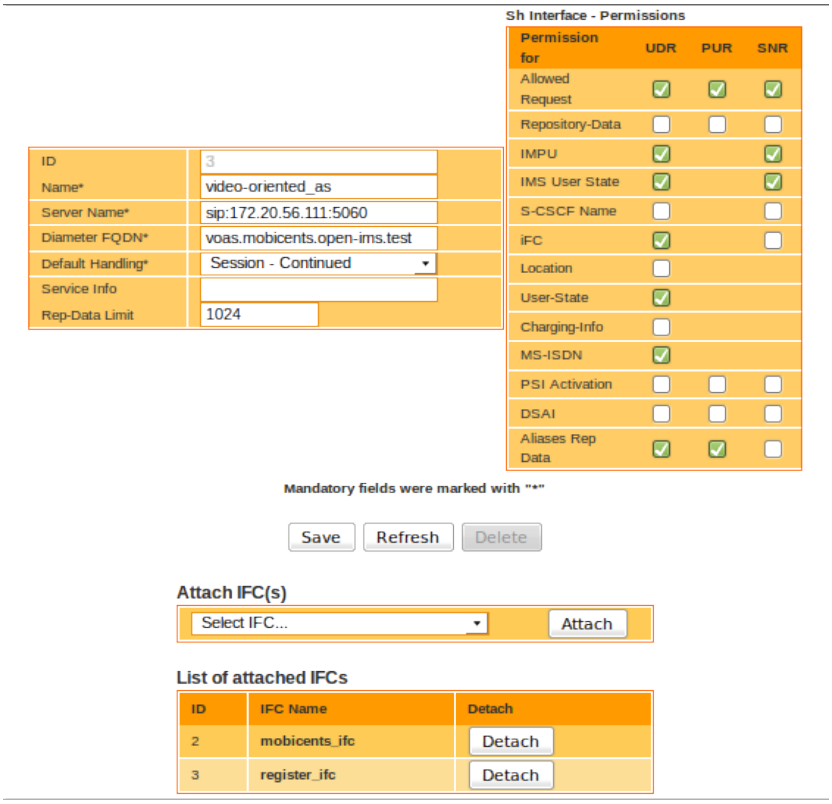



Figure 8.2: Application server configuration in Open IMS Core

**Creating a Default User Profile** As mentioned before, a Java program, attached as Appendix H, was written to create a default user profile for testing purposes. The program uses the JAXB generated classes to create an xml file with default values. The default user profile contains values for the user's birth date, telephone number and email address. The section of the user profile data that is used in this section is shown in Listing 30.

An SBB was created to load the default user profile.

**Creating the Relational Database Tables** As discussed in Chapter 6, we have identified different tables that are important for providing personalized services. Mobicents recently included the JDBC RA to allow SBBs to interact with databases. We used the Mobicents example SBB [119] to create the Special Clips table.

### 8.1.3 Functions Tested

In this section we present how the services that we used for testing were set up and run.

**Listing 30** Partial view of the user profile data

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<IPTVProfile xmlns="urn:org:etsi:ngn:params:xml:ns:iptv"
xmlns:ns2="urn:org:etsi:ngn:params:xml:ns:iptvueprofile"
xmlns:ns3="urn:tva:metadata:2005" Profileid="Zelalem001">
    <uEProfile>
        <ns2:UserEquipmentID>pc00001</ns2:USerEquipmentID>
                <ns2:userEquipmentclass>Pc</ns2:UserEquipmentclass>
                <ns2:SupportedEncodings>
                        <ns2:AudioEncoding>
                                <ns2:Encoding>
                                        <ns3:Name preferred="true">MPEG
                                            -4 Main Audio Profile</ns3:
                                            Name>
                                </ns2:Encoding>
                        </ns2:AudioEncoding>
                        <ns2:AudioEncoding>
                                <ns2:Encoding>
                                        <ns3:Name preferred="false">AMR-
                                            WB</ns3:Name>
                                </ns2:Encoding>
                        </ns2:AudioEncoding>

                                   <ns2:VideoEncoding>
                                <ns2:Encoding>
                                        <ns3:Name preferred="true">H.264
                                             Baseline Profile</ns3:Name>
                                </ns2:Encoding>
                        </ns2:VideoEncoding>
                        <ns2:VideoEncoding>
                                <ns2:Encoding>
                                        <ns3:Name preferred="false">H
                                            .263</ns3:Name>
                                </ns2:Encoding>
                        </ns2:VideoEncoding>
                </ns2:SupportedEncodings>
        </UEProfile>
        <Globalsettings>
                <UsersActionnecordable>false</UsersActionRecordable>
                <RecommendationOnorOFF>On</RecommendationOnorOFF>
                <CriticalNotificationDeliveryMethod> sms
                        </CriticalNotificationDeliveryMethod>
        </Globalsettings>
        <PersonalProfile>
                <Telephone>"27730101051</Telephone>
                <Email>zelalemsss@gmail.com</Email>
                <Birthdate>"1968-05-05</Birthdate>
        </Persona1Profile>
        <coDProfile>
                <Parentalcontrol>2<lParentalcontrol>
        </coDProfile>
</IPTVProfile>
```

#### 8.1.3.1 Testing the Autoresume Service

The Autoresume service has two parts: the first is the detection of abnormal termination of the stream and recording of the information into the appropriate place; the second part is the resumption of the failed session when the user boots up. The playing of the failed session when the user boots up is similar to other bootstrap services and is explained later. In this section we present how the first part of this service was tested.

**Simulating the Session Failure** We simulated this event by killing the streaming server while the user is watching a video from the server. When this happens, the Proxy senses the problem and sends a message to the AS. In the current implementation of RTSP and under a normal circumstance when this happens, the Client (UE) continues to send RTCP until the timeout is reached (the default value is 60 sec).

**Simulating the STTE Message** As mentioned in the design section, once the Proxy senses the abnormal termination of the stream, it sends the STTE message to the AS. Mobicents supports the development of custom events, but this was not developed because of time constraints. As a result, we used the SIP OPTIONS command to transmit the STTE message to the AS.

**The recording of the Auto Bookmark Entry** When the requirement for this service is met, i.e the abnormal termination of the stream happens, the AS records the bookmark position that it received from the Proxy into the user's profile data into the XDMS server. The AS includes a method, called `createAutoBookmarkEntry`, that writes the information (the URL and bookmark position of the media) into the XDMS server.

**Delivering a Good User Experience** In the current implementation of RTSP and under normal circumstance when a session terminates abnormally, the client (UE) continues to send RTCP until the timeout is reached (the default value is 60 sec). Because this would create a bad user experience, we configured the Proxy with 150ms idle time so that when the idle time reaches, it sends the stream terminated event to the AS to inform about the abnormal termination of the session. We believe this would provide a good QoE.

**Flow Diagram of the Autoresume Service** Figure 8.3 shows the entire communication flow of the different components that are involved.

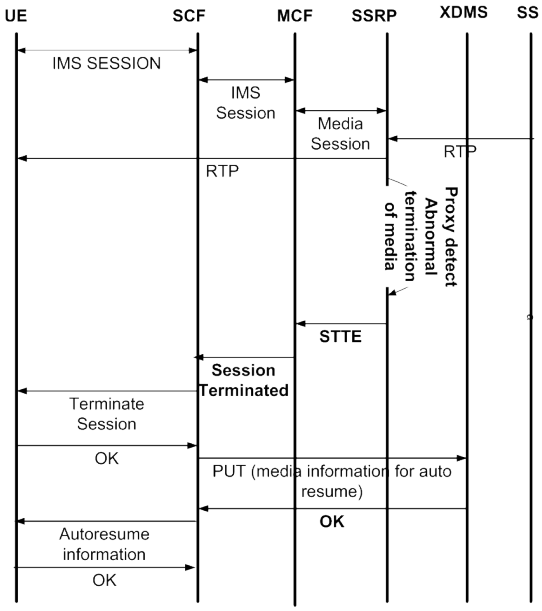



Figure 8.3: Flow diagram of autoresume services

#### 8.1.3.2 Testing the Bootstrap Service

The Bootstrap Service is a collection of services that are provided to the user by the system when the user boots up his device. It includes services like *Play Special Event Clips* and *Autoresume Last Failed Session*. In this section we are going to show how the Play Special Event Clips service works. We created a default user profile for user "Zelalem" with his birthday as the current date.

When the UE boots up, the HSS sends the event to the AS. The AS then checks the user profile to see if any of the bootstrap services can be provided to the user. We used the Mobicents XCAP client as a child SBB to our AS to retrieve the user's profile. As mentioned before, the first service provided by the Bootstrap Service is the delivery of special clips if the current date matches with the user's birthday or anniversary. In order to do this, the system contacts the mysql database. The same way we used the XCAP client, we also used the Mobicents JDBC client from Mobicents example to achieve this. Because the user may switch on his tv several times on his birthday, this service is provided only when the user boots up his device the first time on his birthday. The following is the call flow diagram when the user boots up his device on his birthday the first time.

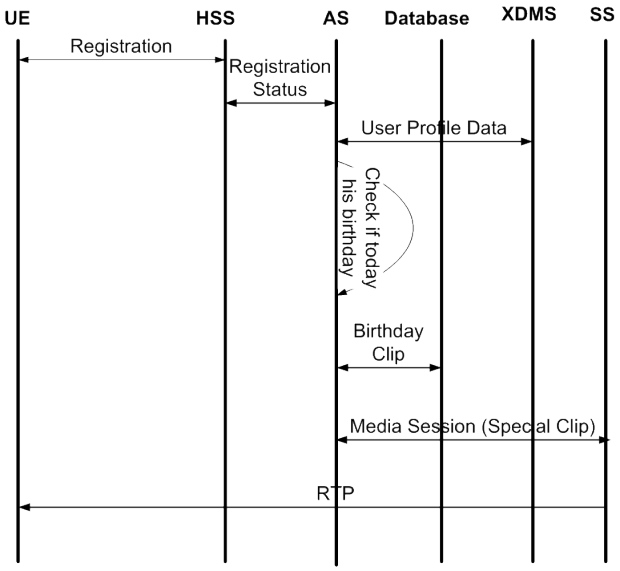



Figure 8.4: Flow diagram of play special clip service

## 8.2 Summary

In this chapter we have presented how the different components of the environment work together. Because our work is done on the server side, we used functional testing to test the functionality of the environment. Some of the interactions were simulated by developing a stub because of lack of a proper open source IPTV client. The results of the tests show how extensible is the environment for developing innovative video-oriented services. We have seen how the different components work together to develop and provide various video-oriented services, including the new use cases identified in the thesis. The Proxy also helped us to prove the extensibility of the environment by allowing us to implement the new RTSP extensions in the Proxy. The next chapter concludes the thesis and also discusses possible future work.

# Chapter 9

# Contributions and Future Work

The demand and growth of video services in the Internet is one reason why television companies and Telcos are migrating to IPTV, an architecture that allows the development of various personalized video-oriented services. The thesis motivated how the IPTV architecture can be used as a base architecture for general, innovative video service development. Through a series of experiments, we came up with new requirements and designed and developed the envisaged environment. As the main objective of the thesis, we selected and augmented a service development and deployment environment based on free and open source tools that can ease the development of video-oriented services in a Telco context. This chapter presents our major findings and contributions together with our conclusion and future work.

## 9.1 Major Findings and Contributions

As mentioned before, one of the reasons why this research was carried out was to identify issues related to the development of IPTV services and coming up with a service development environment that will ease the development activities of service developers. The thesis identified various issues that might have contributed to the limited expansion of video-oriented services in the Telecom world and came up with an environment to facilitate the development and deployment of these services using open source tools. The findings and contributions relate to different aspects of video service development and accordingly they are organized and presented in the same manner.

### 9.1.1 Architecture-Related Findings

The first contribution is the decision on how the environment should be designed. This included choosing an appropriate architectural style. Following a major software design principle, we analyzed the requirements and came up with different components. We decided to build the environment using components for various reasons. One advantage of component-based software development is that service developers only need to know the interface to these components, while the implementation of the components can also be changed if there is a need. For example, as we mentioned before, there are various algorithms to develop a recommender service, and depending on the context of the problem, we can implement the recommendation module using a specific algorithm.

To ease the service development activity, another decision we made was to develop service building blocks or reusable functions that can be used as service enablers to compose various services. We did this because there are services that need to implement the same feature. For example, most video-oriented services needs to include a logic to modify the attributes of the session of the service at some point, which involves modifying the signaling and media sessions. This functionality can be considered as an enabler that service developers can utilize.

Finally, the third important contribution in relation to architecture-related decisions was the actual identification and development of video service enablers and components. We have identified new components that the IPTV reference architecture from the standard bodies did not include. The new components are clearly specified and their interface with the existing components is discussed. One example is the Notification and Recommendation Engine.

### 9.1.2 Implementation Aspects and Decisions

The most important contribution related to the design and implementation of the environment is the selection of Mobicents as a basis for the composition of our environment because Mobicents provides the programming styles that the environment should support. Our analysis of the requirements led us to the identification that the development environment needs to support services that are developed using event based programming style and should run in pipeline mode.

Mobicents is an event-based service development platform and uses the same style to transmit any kind of event through its event router. This makes it easy, for example, to coordinate different sessions.

An aspect of the implementation issue that needed to be addressed by the environment was the support for pipeline-based service composition.IPTV service initiation involves the execution of various processes, referred to as filters. The processing of these filters need to be done in a pipeline fashion and information should be shared between the different processes involved in the service initiation. We have seen that Mobicents also supports this aspect. In fact, Mobicents allows an easy configuration of services for users, and we can easily include any new filter in the system or remove a given filter from the system.

On the other hand, especially for the purpose of developing a converged service, we needed an environment that supports protocols not only specified by the IMS and IPTV standard specifications but also other protocols from the Internet. This is another advantage we get from Mobicents. The fact that Mobicents supports different protocols through its resource adapter is a major advantage for service developers to have a consistent environment and programming model. It is also not difficult to develop resource adapters for new protocols.

In general, we found Mobicents to be a good service development and deployment platform for IMS-based IPTV services. We defined and implemented various filters and pipelines to implement the environment.

### 9.1.3 Media Protocol Related Findings

The IPTV standard is developed by assuming that the different protocols borrowed from the Internet would work in the IMS setting, but our findings show otherwise. This is particularly true for RTSP, where the IPTV specification for CoD service relies much on it. Even if RTSP is defined as a session control protocol, it works properly only in a client-server environment and not for settings similar to the IPTV architecture where it needs to be used to initiate and handle a streaming session by a third party component (the MCF) instead of the client. In this regard, a major contribution of the thesis is the realization of the deficiency RTSP. The RTSP protocol does not exhaustively handle all session management aspects that are expected from a session control protocol. Some of the media control aspects are also handled by the RTCP protocol, which is used by the client and server and not the media controller (the MCF).

As a major finding, the thesis identified the areas where the RTSP protocol needs improvements. We hope the proposals made in this thesis will be debated and be taken as standards for IPTV-based streaming services. We also hope that the MMUSIC working group of the IETF will also consider standardizing the various proposals submitted by

different people as Internet drafts so that device manufacturers and service developers can consider them.

### 9.1.4 The Proxy as an Immediate Solution to Build an Open Source MDF

Most of the open source streaming servers do not support the features of RTSP that are important for media control in the IPTV context and they do not work as a proper MDF, and a work-around was required to solve this problem. As a result, another contribution related to media delivery and control is the RTSP Proxy and Relay Unit that supplement the deficiencies of open source streaming servers. We presented how the Proxy can augment open source streaming servers to serve as an MDF. In fact, it was also a good showcase to demonstrate implementation aspects of the RTSP protocol extensions proposed in this thesis. We implemented and included the RTSP extensions that we proposed into the Proxy and this helped us to have a proper MDF with all the new features using open source streaming servers to test innovative video-oriented services.

### 9.1.5 New Session Management Techniques

One reason for users to choose IMS is for guaranteed QoS. In the case of video-oriented services, this translates to how quickly the system deliver media to users requests. One service that requires efficient user response is the temporary insertion of media. As discussed in the thesis, this task, under the current standard of RTSP, can be implemented by tearing down and setting up the different sessions related to the service. Nevertheless, if we know in advance when the temporary media is to be inserted, it is better to set up the temporary media in advance and delay the delivery until the time arrives to switch the media. Therefore, the solution we proposed for problems of this kind is to create different connections on the server side and consider only the main media connection on the client side. This is another contribution of this research. The technique can be used for services that involve a playlist and also for channel zapping service. We implemented this technique in the Proxy.

### 9.1.6 Modified Data Structure

The IPTV standard document specified a data structure to hold user profiles for the purpose of delivering personalized services. As discussed in this thesis, the user profile

from the standard body is not exhaustive. Especially if one wants to develop converged services, the default user profile that is specified in the standard document needs to be modified to include the information necessary for these services. As a result, we have modified the user profile to support the development of converged services and this is another contribution of the thesis.

In a similar manner, related to the the delivery of notification and recommendation services, we have also identified the requirements for new user profile information and as a result defined and implemented them using a relational database.

### 9.1.7 New Service Use Cases and Terminology

A final contribution of this research is the introduction of new use cases and terminologies. We have coined the term *advanced time-shifting* to describe the resumption of video watching from the PVR while a live program is pushed to the PVR. On the other hand, we have also defined new use cases and explained how they can be implemented. Examples of new service use cases include *Autoresume* and *Bootstrap* services.

## 9.2 Recommendations for Future Research

The service development and deployment environment presented in this thesis solves most of the problems raised at the beginning of the research. However, our discussions also exposed areas that need further investigation and as such it can be considered as a checklist for new research initiatives so as to come up with a comprehensive video service development environment.

One recommendation that stands out in this thesis is the need for definition of video-oriented service enablers for the purpose of service development. We discussed how the recording and playback module we described with the videomail service can be used to develop a PVR. We saw how the call control module that is developed with the converged service can be used to develop a parental control module. In fact, we also mentioned at different times the need for extensibility. As a result, research has to be carried out to exhaustively identify and develop video service enablers and make them available for the research community as open source APIs.

These all imply that there is a need for a design pattern that can assist service developers to see the relationship between different components and modules. In fact, the fact that

the architectural style of the environment mixes both event based and pipeline based styles implies we need a special design pattern. We believe this requires further investigation.

Another area that our research exposed is the lack of proper open source IPTV client to test and prove new concepts. We suggest the development of an extensible open source IPTV client that can be used by the research community to prove their concepts. It should support all client side features expected from an IPTV client and should also be extensible. One feature that IPTV UEs need to include is a mechanism for the UE to turn on and off picture-in-picture features, like the sign language broadcasting, from TV programs.

## 9.3 Conclusion

The main purpose of the research was to investigate the issues surrounding video-oriented service development and delivery in the Telco context, and identify and augment an open source based service development environment that eases the development of video-oriented services. We identified various issues and presented our solutions in this thesis. In this chapter we concluded our work by summarizing and presenting the major contributions. As discussed in this chapter, the major contribution of the thesis is the selection of Mobicents for use as a basis for the realization of the envisaged environment. Mobicents support the pipeline and event-based programming style and it is also easily extensible.

The thesis has identified a number of issues related to video-oriented service development, especially related to the protocols we use. Solutions are proposed for protocol related issues and a Proxy was developed to augment the problem of open source streaming servers and also to proof our concepts. Another finding relates to the identification of the major protocols that are important for video-oriented service development.

As mentioned before, when it comes to video-oriented services, not only SIP, but diameter is also an important protocol to develop a video-oriented AS.

The chapter also highlighted future research areas that need to be tackled.

# Bibliography


[1] JoVE. JoVE : Peer Reviewed Scientific Video Journal (Methods and Protocols on Video). Available Online, June 2012. URL: http://www.jove.com/.

[2] BBC. YouTube at five - 2bn views a day. Available Online, June 2012. URL: http://news.bbc.co.uk2/hi/technology/8676380.stmw.

[3] Cisco Systems, Inc. The Zettabyte Era: Trends and Analysis [Report]. Available Online, May 2015. URL: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visualnetworkingindexvni/VNI_Hyperconnectivity_WP.html.

[4] Accustream Media Research. Online Video Spend Tops $2.12B in 2008. Available Online, July 2010. URL: http://www.marketingcharts.com/interactive/online-video-spend-tops-212b-in-2008-225-growth-forecast-in-2009-7955/vo.

[5] Cisco Systems, Inc. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 20152020 [White Paper]. Available Online, February 2016. URL: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html.

[6] Cisco systems, Inc. Home Page. Available Online, December 2014. URL: http://www.cisco.com/.

[7] Keith Knightson, Naotaka Morita, and Thomas Towle. NGN Architecture: Generic Principle, Functional Architecture, and Implementation. *IEEE Communications Magazine*, 43:49–56, 2005.

[8] International Telecommunication Union (ITU). Y.2001: General Overview of NGN. Available Online, December 2004. URL: https://www.itu.int/rec/T-REC-Y.2001/e.

[9] E Burger. RFC 4483: A Mechanism for Content Indirection in Session Initiation Protocol (SIP) Messages. Internet Engineering Task Force, May 2006.

[10] Thomas Magedanz. Service Delivery Platform Options for Next Generation Networks, approved within the national German 3G Beyond Testbed. In *Southern African Telecommunication Networks and Applications Conference (SATNAC) 2004 Proceedings*, Spier Wine Estate, Western Cape, South Africa, 2004.

[11] Imen Grida Ben Yahia and Emmanual Bertin. Next Generation Networks Services and Management . In *Proceedings of International conference on Networking and Services, 2006. ICNS '06*, pages 15–19, Slicon Valley, CA, July 16-18, 2006.

[12] Chae-Sub Lee and Dick Knight. Realization of the next-generation network. *IEEE Communications Magazine*, 43:34–41, 2005.

[13] G. Camarillo and M.A. Garcia-Martin. *The 3G IP multimedia subsystem (IMS): merging the Internet and the cellular worlds*. John Wiley & Sons, Ltd, New York, NY, USA, third edition, 2008.

[14] T. Russell. *The IP Multimedia Subsystem (IMS)*. McGraw-Hill Education (India) Pvt Limited, Nagar, India, 2007.

[15] ABI Research. ABI Research - Technology Research Market. Available Online, July 2010. URL: http://www.abiresearch.com/home.jsp.

[16] Mohammad Ilyas and Syed Ahson. *IP Multimedia Subsystem (IMS) Handbook*. CRC Press, Inc., Boca Raton, FL, USA, 2009.

[17] M Tsietsi. *A Structural and Functional Specification of a SCIM for Service Interaction Management and Personalisation in the IMS*. PhD thesis, Rhodes University, South Africa, August 2011.

[18] Andreas Gebhardt. *Rapid Prototyping*. Carl Hanser Verlag GmbH & Co. KG, Munich, Germany, 2003.

[19] Craig Larman and Victor R. Basili. Iterative and Incremental Development: A Brief History. *IEEE Computer*, 36(6):47–56, June 2003.

[20] Videolan Organization. Libvlc. Available Online. URL: http://www.videolan.org/vlc/libvlc.html.

[21] R. Copeland. *Converging NGN wireline and mobile 3G networks with IMS*. Informa Telecommunications and Media. CRC Press, Inc., London, UK, 2009.

[22] Gonzalo Camarillo. *SIP Demystified*. McGraw-Hill Professional, Pennsylvania Plaza, NY, USA, 2001.

[23] F.A. Aagesen and S.J. Knapskog. *Networked Services and Applications - Engineering, Control and Management: 16th EUNICE/IFIP WG 6.6 Workshop, EUNICE 2010, Trondheim, Norway, June 28-30, 2010, Proceedings*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010.

[24] IETF. About the IETF. Available Online, May 2009. URL: http://www.ietf.org/about/.

[25] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg. SIP: Session Initiation Protocol. RFC 2543 (Proposed Standard), March 1999. Obsoleted by RFCs 3261, 3262, 3263, 3264, 3265.

[26] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261 (Proposed Standard), June 2002. Updated by RFCs 3265, 3853, 4320, 4916, 5393, 5621, 5626, 5630, 5922, 5954, 6026, 6141.

[27] R. Sparks. The Session Initiation Protocol (SIP) Refer Method. RFC 3515 (Proposed Standard), April 2003.

[28] A. B. Roach. Session Initiation Protocol (SIP)-Specific Event Notification. RFC 3265 (Proposed Standard), June 2002. Updated by RFCs 5367, 5727.

[29] Min-Xiou Chen, Chen-Jui Peng, and Ren-Hung Hwang. SSIP: Split a SIP session over multiple devices. *Computer Standards & Interfaces*, 29(5):531 – 545, 2007.

[30] S. Donovan. The SIP INFO Method. RFC 2976 (Proposed Standard), October 2000. Obsoleted by RFC 6086.

[31] Uri Baniel. SIP/IMS Specifications For Dummies eBook. Available Online, June 2010. http://www.sipknowledge.com/.

[32] Kamailio. Kamailio SIP Server. Available Online, December 2010. URL: http://www.kamailio.org.

[33] Red Hat. Mobicents - The Open Source SLEE and SIP Server. Available Online, May 2011. URL: http://www.mobicents.org.

[34] JBoss. Mobicents: The Open Source Communications Platform. Available Online, June 2014. URL:http://www.mobicents.org/.

[35] OpenCloud. About JAIN SLEE. Available Online, June 2012. URL: https://developer.opencloud.com/devportal/display/OCDEV/JAIN+SLEE.

[36] H. Schulzrinne, A. Rao, and R. Lanphier. Real Time Streaming Protocol (RTSP). RFC 2326 (Proposed Standard), April 1998.

[37] Audio-Video Transport Working Group, H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 1889 (Proposed Standard), January 1996. Obsoleted by RFC 3550.

[38] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550 (Standard), July 2003. Updated by RFCs 5506, 5761, 6051, 6222.

[39] H. Schulzrinne, A. Rao, R. Lanphier, and M. Westerlund. Real Time Streaming Protocol 2.0 (RTSP). Available Online. URL: http://tools.ietf.org/html/draft-ietf-mmusic-rfcrfc2326bis-28.

[40] Thomas M. Zeng and P. Greg Sherwood. Internet draft: RTSP Announce Method. Available Online, February 2005. URL: http://tools.ietf.org/html/draft-ietf-mmusic-rtsp-announce-01.

[41] Microsoft Corporation. [MS-RTSP]: Real-Time Streaming Protocol (RTSP) Windows Media Extensions. Available Online, January 2010. URL: http://msdn.microsoft.com/en-us/library/cc245238

[42] M. Handley and V. Jacobson. SDP: Session Description Protocol. RFC 2327 (Proposed Standard), April 1998. Obsoleted by RFC 4566, updated by RFC 3266.

[43] J. Rosenberg and H. Schulzrinne. An Offer/Answer Model with Session Description Protocol (SDP). RFC 3264 (Proposed Standard), June 2002.

[44] H. Schulzrinne and S. Casner. RTP Profile for Audio and Video Conferences with Minimal Control. RFC 3551 (Standard), July 2003. Updated by RFC 5761.

[45] J. Rosenberg. The Extensible Markup Language (XML) Configuration Access Protocol (XCAP). RFC 4825 (Proposed Standard), May 2007.

[46] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, and J. Arkko. Diameter Base Protocol. RFC 3588 (Proposed Standard), September 2003. Updated by RFCs 5729, 5719.

[47] 3GPP. TS 29.329: Sh Interface based on the Diameter protocol;Protocol details; Version 9.2.0. Third Generation Partnership Project, June 2010.

[48] M. Tsietsi, Z. Shibeshi, A. Terzoli, and G. Wells. An Asterisk-Based Framework for e-Learning using Open Protocols and Open Source Software. In *Innovations for Digital Inclusions, 2009. K-IDI 2009. ITU-T Kaleidoscope:*, pages 1 –5, 312009-sept.1 2009.

[49] Live555. Internet Streaming Media, Wireless, and Multicast technology, services, and standards. Available Online, July 2010. http://www.live555.com/.

[50] Zelalem S Shibeshi, Alfredo Terzoli, and Karen L. Bradshaw. An RTSP Proxy for Implementing the IPTV Media Function Using a. Streaming Server. *Informatica: An International Journal of Computing and Informatics*, 36:37–45, 2012.

[51] Philippe Gentric. Internet draft: RTSP Stream Switching. Available Online:, February 2003. http://tools.ietf.org/html/draft-gentric-mmusic-stream-switching-01.

[52] Oliver Friedrich, Robert Seeliger, Adel Al-Hezmi, Christian Riede, and Stefan Arbanowski. Prototyping Interactive and Personalized IPTV-Services on Top of Open IMS Infrastructures. In Manfred Tscheligi, Marianna Obrist, and Artur Lugmayr, editors, *Changing Television Environments*, volume 5066 of *Lecture Notes in Computer Science*, pages 204–208. Springer Berlin Heidelberg, 2008.

[53] BlackBerry. BlackBerry Limited. Available Online, June 2014. URL: http://www.blackberry.com.

[54] BlackBerry. BlackBerry Java SDK: Switching media feeds when using the RTSP protocol. Available online, June 2012. http://developer.blackberry.com/bbos/java/.

[55] Philippe Gentric. Internet draft: RTSP Stream Switching. Available Online:, February 2003. http://tools.ietf.org/html/draft-gentric-mmusic-stream-switching-00.

[56] Jason B. Penton and Alfredo Terzoli. iLanga: A Next Generation VOIP Based TDM-Enabled PBX. In *Proceedings of the 4th Southern African Telecommunications Networks and Applications Conference*, Spier Wine Estate, Western Cape, South Africa, 6-8 September 2004.

[57] Russell Bryant, Leif Madsen, and Jim Van Meggelen. *Asterisk: The Definitive Guide*. O'Reilly Media, Inc, Sebastopol, CA, USA, fourth edition, May 2013.

[58] J Van Meggelen, J Smith, and L Madsen. *Asterisk: The Future of Telephony*. O'Reilly Media, Sebastopol, CA, USA, first edition, 2005.

[59] Voip-info.org. Asterisk Video. Available Online, April 2009. URL: http://www.voip-info.org/wiki/view/Asterisk+video.

[60] Sourceforge.net. FFMPEG: A complete, cross-platform solution to record, convert and stream audio and video. Available Online, December 2011. URL: http://jffmpeg.sourceforge.net/.

[61] The Center for Speech Research Group (The University of Edinburgh). The Festival Speech Synthesis System. Available Online, June 2014. http://www.cstr.ed.ac.uk/projects/festival/.

[62] E.M. Heiliger and P.B. Henderson. *Library automation: experience, methodology, and technology of the library as an information system.* McGraw-Hill series in library education. McGraw-Hill, 1971.

[63] A. Musimwa and Z. Shibeshi. Development of SDI based VoD Recommender System. Honours thesis. University of Fort Hare, October 2013.

[64] 3GPP. AW: Tdoc S1-010232 - Standardisation of IMS services; Comments. Third Generation Partnership Project, September 2001.

[65] ETSI TISPAN. Home Page. European Telecommunication Standards Institute, July 2014. URL: http://www.etsi.org/tispan/.

[66] M. Garcia-Martin. Input 3rd-Generation Partnership Project (3GPP) Release 5 Requirements on the Session Initiation Protocol (SIP). RFC 4083 (Informational), May 2005.

[67] Thomas Magedanz and Fabricio Carvalho de Gouveia. IMS - the IP multimedia system as NGN service delivery platform. *Elektrotechnik und Informationstechnik*, 123(7-8):271–276, 2006.

[68] Miikka Poikselk and Georg Mayer. *The IMS IP Multimedia Concepts and Services.* John Wiley & Sons, Ltd., New York, NY, USA, Third edition, 2009.

[69] M. Garcia-Martin, E. Henrikson, and D. Mills. Private Header (P-Header) Extensions to the Session Initiation Protocol (SIP) for the 3rd-Generation Partnership Project (3GPP). RFC 3455 (Informational), January 2003.

[70] 3GPP. TS 23.002: Network Architecture. Third Generation Partnership Project, December 2008.

[71] 3GPP. TS 23.218: IP Multimedia (IM) Session Handling; IM Call model; Stage 2. Third Generation Partnership Project, December 2008.

[72] ATIS. The IPTV Interoprability Forum. http://www.atis.org/iif/index.asp.

[73] The Telecommunication Standardization Sector of ITU (ITU-T). Standardisation (ITU-T). Available Online, July 2010. URL: http://www.itu.int/ITU-T/.

[74] Open IPTV Forum. Available Online, June 2014. http://www.oipf.tv/.

[75] DVB. DVB - Digital Video Broadcasting, December 2012. http://www.dvb.org/.

[76] ETSI. TS 181 016: Service Layer Requirements to integrate NGN Services and IPTV, July 2009.

[77] ETSI. TS 182 027: IPTV Architecture;IPTV functions supported by the IMS subsystem. European Telecommunication Standards Institute, March 2011.

[78] ETSI. TS 181 016: IMS-based IPTV stage 3 Specification, February 2011.

[79] ETSI. TS 182.027: IPTV Architecture; IPTV functions supported by the IMS subsystem, June 2010.

[80] Johan Hjelm. *Why IPTV: Interactivity, Technologies, Services*. John Wiley & Sons, Ltd, New York, NY, USA, 2008.

[81] Marie-Jose Montpetit, Thomas Mirlacher, and Michael Ketcham. IPTV: An End to End Perspective (Invited Paper). *Journal of Communications*, 5(5):358–373, May 2010.

[82] Eugen Mikoczy, Dmitry Sivchenko, Bangnan Xu, and Veselin Rakocevic. IMS based IPTV services: architecture and implementation. In *Proceedings of the 3rd international conference on Mobile multimedia communications*, MobiMedia '07, pages 15:1–15:7, ICST, Brussels, Belgium, Belgium, 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[83] Oliver Friedrich, Adel Al-Hezmi, Stefan Arbanowski, and Thomas Magedanz. Evolution of Next Generation Networks towards an Integrated Platform for IMS-Based IPTV Services. In *Proceedings of the 2007 International Symposium on Applications and the Internet Workshops*, SAINT-W '07, pages 10–15seeliger, Washington, DC, USA, 2007. IEEE Computer Society.

[84] Oliver Friedrich, Robert Seeliger, and Stefan Arbanowski. Interactive and Personalized Services for an Open IMS-Based IPTV Infrastructure. In Seeliger Robert and ArbanowskiEditors Stefan, editors, *Seventh International Conference on Networking (ICN) 2008*, pages 302–307, 2008.

[85] Khalid Ahmad and Ali C. Begen. IPTV and video networks in the 2015 timeframe: The evolution to medianets. *IEEE Communications Magazine*, 47:68–74, December 2009.

[86] Dmitry Sivchenko, Bangnan Xu, and Deutsche Telekom. IPTV Services over IMS: Architecture and Standardization. *IEEE Communications Magazine*, 46(5):128–135, May 2008.

[87] T. Cagenius, A. Fasbender, J. Hjelm, Uwer Horn, Ivars, and N. Selberg. Evolving the TV experience: Anytime, anywhere, any device. *Ericsson Review*, 3:107–111, 2006.

[88] Steven Whitehead, Marie-Jose Montpetit, X, and Xavier Marjou. Internet draft. An Evaluation of Session Initiation Protocol (SIP) for use in Streaming Media Applications. Available Online, February 2006. URL: http://tools.ietf.org/html/draft-whitehead-sip-for-streaming-media-00.

[89] S Whitehead, M Montepetit, X Marjou, and S Ganesanan. Internet draft. Media Playback Control Protocol Requirements. Available Online. URL: http://tools.ietf.org/html/draft-whitehead-mmusic-sip-for-streaming-media-03. Expires: August 2008.

[90] Shanmugalingam Sivasothy, Gyu Myoung Lee, and Noel Crespi. Framework of media control for IPTV services. http://tools.ietf.org/html/draft-siva-iptv-media-01. Expires: September 2010.

[91] S. Sivasothy, G. M. Myoung, and N. Crespi. A unified session control protocol for IPTV services. In *Proceedings of the 11th international conference on Advanced Communication Technology*, pages 961–965, Gangwon-Do, South Korea, February 15-18, 2009.

[92] R. G. Shiroor. IPTV and VoD services in the context of IMS. In *International Conference on IP Multimedia Subsystem Architecture and Applications*, pages 1–5, December 6-8, 2007.

[93] E Mikoczy, D Sivchenko, B Xu, and J I. Moreno. IPTV Services over IMS: Architecture and Standardization. *IEEE Communications Magazine*, 46(5):128–135, May 2008.

[94] ETSI. TS 183.036: IMS-based IPTV stage 3 specification. European Telecommunication Standards Institute, March 2011.

[95] TV-Anytime Forum. Home Page. Available Online, June 2014. URL: http://www.tv-anytime.org/.

[96] The Moving Picture Experts Group. MPEG-7 Overview. Available Online, June 2014. URL: http://mpeg.chiariglione.org/standards/mpeg-7/mpeg-7.htm.

[97] Paolo Cremonesi and Roberto Turrin. Analysis of cold-start recommendations in IPTV systems. In *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, pages 233–236, New York, NY, USA, 2009. ACM.

[98] Riccardo Bambini, Paolo Cremonesi, and Roberto Turrin. A Recommender System for an IPTV Service Provider: a Real Large-Scale Production Environment. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 299–331. Springer US, 2011.

[99] Soohong Daniel Park, Cheolju Hwang, and Glenn A. Adams. Video Recommendation Based on User Preference on the Web. Available online, December 2007. URL: www.w3.org/2007/08/video/positions/SAMSUNG.pdf.

[100] Apache Software Foundation. The Apache Mina Software Foundation. Available Online, July 2010. http://mina.apache.org/.

[101] ETSI. TS 185.013: Codecs for customer network devices. European Telecommunication Standards Institute, May 2009.

[102] The RTSP Proxy. Available Online, January 2010. URL. http://rtspproxy.berlios.de/.

[103] T. Ura, K. Oku, H. Harada, and A. Kobayashi. Internet draft: RTSP 2.0 Asynchronous Notification. Available Online, August 2008. URL: http://tools.ietf.org/html/draft-stiemerling-rtsp-announce-01.

[104] Sun Microsystems. Java Architecture for XML Binding (JAXB). Available Online, June 2014. URL: https://jaxb.java.net//msdn.

[105] Z. Shibeshi, S. Ndakunda, A. Terzoli, and K. Bradshaw. Delivering a Personalized Video Service using IPTV. In *ICACT2011: Proceedings of the International Conference on Advanced Communication Technology*, 2011.

[106] JBoss. Converged Application Demo. Available Online, February 2011. URL: http://groups.google.com/group/mobicents-public/web/converged-application-demo?hl=en.

[107] M Tsietsi, Z Shibeshi, S Ndakunda, and R Musvibe. Mobicents Converged Demo with Video Streaming. Available Online, October 2009. URL: http://www.youtube.com/watch?v=YvzeYvWtDaI.

[108] Open Mobile Alliance (OMA). Home Page. http://www.openmobilealliance.org/.

[109] CISCO. The Evolving IPTV Service Architecture: White Paper. Available Online, June 2014. URL: http://www.cisco.com/c/en/us/solutions/collateral/serviceprovider/iptvsolutions-wirelinecarriers/net_implementation_white_paper0900aecd806530a4.htm.

[110] I. Sommerville. *Software Engineering*. International Computer Science Series. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2011.

[111] J. Deruelle. JSLEE and SIP-Servlets Interoperability with Mobicents Communication Platform. In *Next Generation Mobile Applications, Services and Technologies, 2008. NGMAST '08. The Second International Conference on* , pages 634–639, 2008.

[112] Gerardo Rojas Sierra, Oscar M Caicedo Rendon, Felipe Estrada Solano, and Julian A Caicedo M. Technical Criteria for Value-Added Services Creation, Execution and Deployment, on Next Generation Networks. In *AICT 2011, The Seventh Advanced International Conference on Telecommunications*, pages 123–129, 2011.

[113] Michael Maretzke. Implementing a JSLEE Resource Adaptor - A quick-starter guide . Available Online, June 2014. URL: http://maretzke.com/pub/howtos/mobicents_ra/index.html.

[114] Eduardo Martins, Bartosz Baranowski, and Alexandre Mendonca. Mobicents JSLEE Call Controller2 User Guide. Available Online, October 2009. URL: http://groups.google.com/group/mobicentspublic/web/jainsleeexamplecall-controller2.

[115] JBoss. Diameter Event Charging Service. Available Online, June 2014. URL: http://www.mobicents.org/diameter_event_charging.html.

[116] Moses Thizwilondi Nkhumeleni. MoBill: a framework for billing in next generation networks, 2010. Honours thesis. Rhodes University.

[117] JBoss. Mobicents Diameter User Guide. Available Online, February 2011. URL: http://hudson.jboss.org/hudson/job/MobicentsBooks/lastSuccessfulBuild/artifact/diameter/core/index.html.

[118] JBoss. Mobicents SIP Servlets (Shopping Demo). Available Online, February 2011. URL: http://www.mobicents.org/shoppingdemo.html.

[119] Eduardo Martins, Bartosz Baranowski, and Alexandre Mendonca. Mobicents JAIN SLEE SIP JDBC Registrar Example User Guide. Available Online, October 2011. URL: http://docs.jboss.org/mobicents/jainslee/2.7.0.FINAL/examples/sipjdbcregistrar/userguide/en-US/pdf/Mobicents_SLEE_Example_SIP_Services_User_Guide.pdf.

# Appendices

# Appendix A

## List of publications related to this research

M Tsietsi, Z. S. Shibeshi, A Terzoli and G Wells. "An asterisk-based framework for e-learning using open protocols and open source software". ITU-T Kaleidoscope: Innovations for Digital Inclusion Conference. Hotel Hermitage, Mar del Plata. Argentina. September 2009. 149-154.

Z. S. Shibeshi, A Terzoli, K Bradshaw. Developing a Videomail Service for iLanga, an Asterisk Based IP PBX. In SATNAC'09: Proceedings of the 12th Southern African Telecommunications Networks and Applications Conference. Ezulwini, Swaziland, 30 August to 2 September 2009

Z. S. Shibeshi, A Terzoli, K Bradshaw. Developing a Toolkit for Video-oriented services in the NGN environment. In SATNAC'09: Proceedings of the 12th Southern African Telecommunications Networks and Applications Conference. Ezulwini, Swaziland, 30 August to 2 September 2009 (Work in progress)

Z. S. Shibeshi, A Terzoli, and K Bradshaw. Streaming Session Transfer between Registered User Agents. In SATNAC'10: Proceedings of the 13th Southern African Telecommunications Networks and Applications Conference. Spier Estate, Stellenbosch, South Africa, 5 to 8 September 2010

Z. S. Shibeshi, A Terzoli, and K Bradshaw. Using an RTSP Proxy to implement the IPTV Media Function via a streaming server. In ICUMT'10: Proceedings of the International Congress on Ultra Modern Telecommunications and Control Systems. Moscow, Russia, 18 to 20 October 2010

R. Wertlen , I. Siebörger, M. Tsietsi, Z. Shibeshi, and A. Terzoli. Research Testbed Networks: Practical Tools for Service Delivery? In IDIA2010: Proceedings of the 4th International Development Informatics Association Conference. Cape Town, South Africa, 3 to 5 November 2010

Z. S. Shibeshi, S. Ndakunda, A. Terzoli, and K. Bradshaw. Delivering a Personalized Video Service using IPTV. In ICACT2011: Proceedings of the International Conference on Advanced Communication Technology. Phoenix Park, Gangwon-Do, Republic of Korea. February 13~16 2011.

Z. S. Shibeshi, A Terzoli, and K Bradshaw. Towards a Toolkit for creating video-oriented services for Mobicents. In SATNAC'11: Proceedings of the 14th Southern African Telecommunications Networks and Applications Conference. East London Convention Center, East London, South Africa, 4 to 7 September 2011

Z. S. Shibeshi, A Terzoli, and K Bradshaw. An RTSP Proxy for implementing the IPTV Media Function using a streaming server. Informatica: An International Journal of Computing and Informatics. Vol. 36, No. 1, pp 37–45, 2012.

R Wertlen, I Sieborger, M Tsietsi, Z Shibeshi, A Terzoli. Research Testbed Networks: Practical Tools for Service Delivery? Electronic Journal of Information Systems in Developing Countries. Vol. 50, No. 1., pp 1—14, 2012.

A. Musimwa, and Z. Shibeshi. Development of SDI based VoD Recommender system. Honours thesis. Nov. 2013. University of Fort Hare.

R. Muranganwa, and Z. Shibeshi. Development of a Dynamic Timeshifting System. Honours thesis. Nov. 2013. University of Fort Hare.

A. Musimwa, and Z. Shibeshi. Development of an extensible open source IPTV client for IMS environment. Proceedings of the 17th Southern African Telecommunications Networks and Applications Conference. The Boardwalk, Port Elizabeth, South Africa, 31 August to 3 September 2014. (Work in progress)

P. Ncube, and Z. Shibeshi. Design of an MPEG-7 based Multimedia Content Description and Management System (MCDMS) for Internet Protocol Television (IPTV). Proceedings of the 18th Southern African Telecommunications Networks and Applications Conference. The Arabella Hotel & Spa, Western Cape, South Africa, 6 to 9 September 2015.

# Appendix B

## Festival Configuration file

; Festival Configuration

;

[general]

;

; Host which runs the festival server (default : localhost);

;

host=146.231.123.87

;

; Port on host where the festival server runs (default : 1314)

;

port=1314

;

; Use cache (yes, no - defaults to no)

;

usecache=yes

;

; If usecache=yes, a directory to store waveform cache files.

; The cache is never cleared (yet), so you must take care of cleaning it

; yourself (just delete any or all files from the cache).

; THIS DIRECTORY *MUST* EXIST and must be writable from the asterisk process.

```
; Defaults to /tmp/
;
cachedir=/var/lib/asterisk/festivalcache/
;
; Festival command to send to the server.
; Defaults to: (tts_textasterisk "%s" 'file)(quit)\n
; %s is replaced by the desired text to say. The command MUST end with a
; (quit) directive, or the cache handling mechanism will hang. Do not
; forget the \n at the end.
;
;festivalcommand=(tts_textasterisk "%s" 'file)(quit)\n
;
;
```

# Appendix C

### The SDPInfo class

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package sip_as_example;

import java.util.Vector;

/**
 *
 * author zelalem
 */

public class SdpInfo {
  //Other parameters to include  from http://rubydoc.info/github/
      turboladen/sdp/master/frames
  private int communicatingDevice; //Phone or SS (if it is streaming
      server then session id is string).
  private static final int PHONE=1;  //This is one communicating device
  private static final int STREAMINGSERVER=2; //Another communicating
      device.
  private String IpAddress;
  private int aport;
  private Vector aformat; //better we make it array or a vector (
      especially for the recieving sdp)
  private int vport;
  private int timeFrom=0; //If I want to start from middle like for
      bookmarked movie.
  private int timeTo=0;
  private Vector vformat; //better we make it array or a vector (
      especially for the recieving sdp)
  private String address;
  private String addressType;
  private long sessionId;
  private String streamingSessionId;
  private String userName;
  private String audioTrack; //Value of the first track (will modify it
      to check if it actually is an audio) instead of assuming.
```

```
  private String videoTrack;
  private int sessionState; //INIT, READY, PLAYING, RECORDING, PAUSED (I
     'll have setter and getter methods.


 public SdpInfo() {
   IpAddress="";
   aport=0;
   aformat = new Vector(); //if array all 0 or vector to null.
   vport=0;
   vformat= new Vector();
   sessionId=0;
   address=null;
   addressType=null;
   userName=null;
   audioTrack=null;
   videoTrack=null;

 }

public void setStreamingSessionId(String id) { streamingSessionId=id;}
public void setIPAddress(String IP) { IpAddress=IP;}
public void setCommunicatingDevice(int deviceType) { communicatingDevice
   =deviceType;}
public void setAPort(int AP) { aport=AP;}
public void setAFormat(Vector AF) { aformat=AF;}
public void setVPort(int VP) { vport=VP;}
public void setVFormat(Vector VF) { vformat=VF;}
public void setAddress(String add) { address=add;}
public void setAddressType(String addType) { addressType=addType;}
public void setSessionId(long id) { sessionId=id;}
public void setUserName(String name) { userName=name;}
public void setAudioTrack(String track) { audioTrack=track;}
public void setVideoTrack(String track) { videoTrack=track;}
public String getStreamingSessionId() { return streamingSessionId;}
public String getIPAddress() { return IpAddress;}
public int getCommunicatingDevice() { return communicatingDevice;}
public int getAPort() { return aport;}
public Vector getAFormat() { return aformat;}
public int getVPort() { return vport;}
public Vector getVFormat() { return vformat;}
public String getAddress() {return address;}
public String getAddressType() {return addressType;}
public long getSessionId() {return sessionId;}
public String getUserName() {return userName;}
public String getAudioTrack() { return audioTrack;}
public String getVideoTrack() { return videoTrack;}

}
```

# Appendix D

### XML Schema for IPTV user profile

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
targetNamespace="urn:org:etsi:ngn:params:xml:ns:iptv"
xmlns="urn:org:etsi:ngn:params:xml:ns:iptv"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:uep="urn:org:etsi:ngn:params:xml:ns:iptvueprofile"
elementFormDefault="qualified"
attributeFormDefault="unqualified">

<xs:import namespace="urn:org:etsi:ngn:params:xml:ns:iptvueprofile"
schemaLocation="annexP-XML Schema for UE Profile.xsd"/>


<xs:element name="IPTVProfile">
        <xs:annotation>
                <xs:documentation> XML Schema for representing the IPTV
                   Profile object identified in TS 182 027 clause 7.3.1
                </xs:documentation>
        </xs:annotation>
        <xs:complexType>
                <xs:sequence>
                        <xs:element name="UEProfile" type="
                           uep:tUEProfile" minOccurs="0"/>
                        <xs:element name="GlobalSettings" type="
                           tGlobalSettings" minOccurs="1"/>
                        <xs:element name="BCProfile" type="tBCProfile"
                           minOccurs="0"/>
                        <xs:element name="CoDProfile" type="tCoDProfile"
                            minOccurs="0"/>
                        <xs:element name="PVRProfile" type="tPVRProfile"
                            minOccurs="0"/>
                        <xs:element name="Extension" type="tExtension"
                           minOccurs="0"/>
                        <xs:any namespace="##other" processContents="lax
                           " minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute name="ProfileId" type="xs:ID" />
                <xs:anyAttribute/>
        </xs:complexType>
</xs:element>
```

```
        <xs:complexType name="tBCProfile">
                <xs:sequence>
                        <xs:element name="BCServicePackage" type="
                           tBCServicePackage"
minOccurs="1" maxOccurs="unbounded"/>
                        <xs:element name="Extension" type="tExtension"
                           minOccurs="0"/>
                        <xs:any namespace="##other" processContents="lax
                           " minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>


        <xs:complexType name="tBCServicePackage">
                <xs:sequence>
                        <xs:element name="BCPackageId" type="
                           tBCServicePackageID" minOccurs="1"/>
                        <xs:element name="Description" type="
                           tBCServicePackageDescription" minOccurs="0"/>
                        <xs:element name="BCService" type="tBCService"
                           minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element name="Extension" type="tExtension"
                           minOccurs="0"/>
                        <xs:any namespace="##other" processContents="lax
                           " minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>


        <xs:simpleType name="tBCServicePackageID" final="list
           restriction">
                <xs:restriction base="xs:string">
                        <xs:minLength value="0"/>
                        <xs:maxLength value="16"/>
                </xs:restriction>
        </xs:simpleType>


        <xs:simpleType name="tBCServicePackageDescription" final="list
           restriction">
                <xs:restriction base="xs:string">
                        <xs:minLength value="0"/>
                        <xs:maxLength value="64"/>
                </xs:restriction>
        </xs:simpleType>


        <xs:complexType name="tBCService">
                <xs:sequence>
                        <xs:element name="BCServiceId" type="
                           tBCServiceID" minOccurs="1"/>
                        <xs:element name="QualityDefinition" type="
                           tQualityDefinition" minOccurs="0"/>
                        <xs:element name="Extension" type="tExtension"
```

```
                    minOccurs="0"/>
                <xs:any namespace="##other" processContents="lax
                    " minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
</xs:complexType>


<xs:simpleType name="tBCServiceID" final="list restriction">
        <xs:restriction base="xs:string">
                <xs:minLength value="0"/>
                <xs:maxLength value="16"/>
        </xs:restriction>
</xs:simpleType>


<xs:simpleType name="tQualityDefinition" final="list restriction
    ">
        <xs:restriction base="xs:unsignedByte">
                <xs:minInclusive value="0"/>
                <xs:maxInclusive value="1"/>
                <xs:enumeration value="0">
                        <xs:annotation>
                                <xs:documentation>
                                        <label xml:lang="en">SD<
                                            /label>
                                        <definition xml:lang="en
                                            ">Standard Definition
                                            </definition>
                                </xs:documentation>
                        </xs:annotation>
                </xs:enumeration>
                <xs:enumeration value="1">
                        <xs:annotation>
                                <xs:documentation>
                                        <label xml:lang="en">HD<
                                            /label>
                                        <definition xml:lang="en
                                            ">High Definition</
                                            definition>
                                </xs:documentation>
                        </xs:annotation>
                </xs:enumeration>
        </xs:restriction>
</xs:simpleType>


<xs:complexType name="tCoDProfile">
        <xs:sequence>
                <xs:element name="ParentalControl" type="
                    tParentalControlLevel" minOccurs="0"/>
                <xs:element name="Extension" type="tExtension"
                    minOccurs="0"/>
                <xs:any namespace="##other" processContents="lax
                    " minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
```

```
</xs:complexType>


<xs:simpleType name="tParentalControlLevel" final="list
   restriction">
        <xs:restriction base="xs:unsignedByte">
                <xs:minInclusive value="0"/>
                <xs:maxInclusive value="3"/>
                <xs:enumeration value="0">
                        <xs:annotation>
                                <xs:documentation>
                                        <label xml:lang="en">ALL
                                           </label>
                                        <definition xml:lang="en
                                           ">All contents</
                                           definition>
                                </xs:documentation>
                        </xs:annotation>
                </xs:enumeration>
                <xs:enumeration value="1">
                        <xs:annotation>
                                <xs:documentation>
                                        <label xml:lang="en">
                                           Level 1</label>
                                        <definition xml:lang="en
                                           ">Level 1 contents</
                                           definition>
                                </xs:documentation>
                        </xs:annotation>
                </xs:enumeration>
                <xs:enumeration value="2">
                        <xs:annotation>
                                <xs:documentation>
                                        <label xml:lang="en">
                                           Level 2</label>
                                        <definition xml:lang="en
                                           ">Up to level 2</
                                           definition>
                                </xs:documentation>
                        </xs:annotation>
                </xs:enumeration>
                <xs:enumeration value="3">
                        <xs:annotation>
                                <xs:documentation>
                                        <label xml:lang="en">
                                           Level 3</label>
                                        <definition xml:lang="en
                                           ">Up to level 3</
                                           definition>
                                </xs:documentation>
                        </xs:annotation>
                </xs:enumeration>
                <xs:enumeration value="4">
                        <xs:annotation>
                                <xs:documentation>
```

```
                                        <label xml:lang="en">
                                            Level 4</label>
                                        <definition xml:lang="en
                                            ">Up to level 4</
                                            definition>
                                    </xs:documentation>
                            </xs:annotation>
                    </xs:enumeration>
                    <xs:enumeration value="5">
                            <xs:annotation>
                                    <xs:documentation>
                                        <label xml:lang="en">
                                            Level 5</label>
                                        <definition xml:lang="en
                                            ">Up to level 5</
                                            definition>
                                    </xs:documentation>
                            </xs:annotation>
                    </xs:enumeration>
            </xs:restriction>
    </xs:simpleType>


    <xs:complexType name="tPVRProfile">
            <xs:sequence>
                    <xs:annotation>
                            <xs:documentation>
                                    Unit of the StorageLimitInVolume
                                        element is the GigaOctet
                            </xs:documentation>
                    </xs:annotation>
                    <xs:element name="PVRPreference" type="
                        tPVRPreference"/>
                    <xs:element name="StorageLimitInTime" type="
                        tNPVRStorageLimitInTime" minOccurs="0"/>
                    <xs:element name="StorageLimitInVolume" type="
                        tNPVRStorageLimitInVolume" minOccurs="0"/>
                    <xs:element name="Extension" type="tExtension"
                        minOccurs="0"/>
                    <xs:any namespace="##other" processContents="lax
                        " minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
    </xs:complexType>

    <xs:simpleType name="tPVRPreference" final="list restriction">
            <xs:restriction base="xs:unsignedByte">
                    <xs:minInclusive value="0"/>
                    <xs:maxInclusive value="1"/>
                    <xs:enumeration value="0">
                            <xs:annotation>
                                    <xs:documentation>
                                        <label xml:lang="en">
                                            Network</label>
                                        <definition xml:lang="en
                                            ">Recording is done
```

```
                                        in the network</
                                        definition>
                                </xs:documentation>
                        </xs:annotation>
                </xs:enumeration>
                <xs:enumeration value="1">
                        <xs:annotation>
                                <xs:documentation>
                                        <label xml:lang="en">
                                            User_Equipment</label
                                            >
                                        <definition xml:lang="en
                                            ">Recording is done
                                            on the user equipment
                                            </definition>
                                </xs:documentation>
                        </xs:annotation>
                </xs:enumeration>
        </xs:restriction>
</xs:simpleType>

<xs:simpleType name="tNPVRStorageLimitInTime">
        <xs:restriction base="xs:duration">
                <xs:minInclusive value="PT0H"/>
                <xs:maxInclusive value="PT1000000000H"/>
        </xs:restriction>
</xs:simpleType>

<xs:simpleType name="tNPVRStorageLimitInVolume">
        <xs:restriction base="xs:nonNegativeInteger"/>
</xs:simpleType>

<xs:complexType name="tGlobalSettings">
        <xs:sequence>
                <xs:element name="LanguagePreference" type="
                    tLanguage" minOccurs="0"/>
                <xs:element name="UsersActionRecordable" type="
                    tUserActionRecordable"/>
                <xs:element name="Extension" type="tExtension"
                    minOccurs="0"/>
                <xs:any namespace="##other" processContents="lax
                    " minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
</xs:complexType>


<xs:simpleType name="tLanguage">
        <xs:restriction base="xs:string">
                                <xs:annotation>
                                <xs:documentation>
                                  <definition xml:lang="en">ISO
                                      639-2 Language code</
                                      definition>
                                </xs:documentation>
                        </xs:annotation>
```

```
                        <xs:minLength value="3"/>
                        <xs:maxLength value="3"/>
                </xs:restriction>
        </xs:simpleType>

        <xs:complexType name="tExtension">
                <xs:sequence>
                        <xs:any processContents="lax" minOccurs="0"
                           maxOccurs="unbounded"/>
                </xs:sequence>
    </xs:complexType>

        <xs:simpleType name="tUserActionRecordable">
                <xs:restriction base="xs:boolean"/>
        </xs:simpleType>

</xs:schema>
```

# Appendix E

### IPTV App Usage

```
package org.convergence.ru.xdm.appusage.iptv;
import javax.xml.validation.Validator;
import org.mobicents.xdm.server.appusage.AppUsage;

public class IPTVUEProfileAppusage extends AppUsage{
    public static final String ID = "org.etsi.ngn.iptv";
    public static final String DEFAULT_DOC_NAMESPACE =
        "urn:org:etsi:ngn:params:xml:ns:iptv";
    public static final String MIMETYPE =
        "application/vnd.etsi.iptvprofile+xml";
    private static final String AUTH_ONLY_DOCUMENT_NAME = "iptvprofile";
    public IPTVUEProfileAppusage(Validator schemaValidator) {
      super(ID, DEFAULT_DOC_NAMESPACE, MIMETYPE, schemaValidator,
          AUTH_ONLY_DOCUMENT_NAME);}
}
```

# Appendix F

### The Java program used to generate defaul xml file

```
import java.util.*;

import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;

import generated.org.etsi.ngn.params.xml.ns.iptv.*;
import generated.org.etsi.ngn.params.xml.ns.iptvueprofile.*;
import generated.tva.metadata._2005.*;
import generated.tva.mpeg7.*;

public class TestJaxb {
//This class will create a default xml document for iptv profile, which
    will base for XCAP client.
//Will be reported. And will also be abstracted further


         public static void main(String[] args) {

                 TestJaxb test = new TestJaxb();
                 try {
                         JAXBContext jaxbContext
                         = JAXBContext.newInstance("org.etsi.ngn.params.
                            xml.ns.iptv:org.etsi.ngn.params.xml.ns.
                            iptvueprofile:tva.metadata._2005:tva.mpeg7");

                         IPTVProfile iptvProfile = new IPTVProfile();
                         TUEProfile ueProfile = new TUEProfile();
                         TCoDProfile codProfile = new TCoDProfile();

                         TGlobalSettings globalSetting = new
                            TGlobalSettings();
                         ueProfile.setUserEquipmentID("pc00001");
                         ueProfile.setUserEquipmentClass(
                            TUserEquipmentClass.PC);

                         List <ControlledTermType> videoEncoding = new
                            ArrayList<ControlledTermType>();
                         videoEncoding = test.getVideoEncoding();
```

```
                        List <TVideoEncoding> videoEncodingList = new
                            ArrayList<TVideoEncoding>();
                        videoEncodingList.add((TVideoEncoding)
                            videoEncoding);


                        List <ControlledTermType> audioEncoding = new
                            ArrayList<ControlledTermType>();
                        audioEncoding = test.getAudioEncoding();
                        List <TAudioEncoding> audioEncodingList = new
                            ArrayList<TAudioEncoding>();
                        audioEncodingList.add((TAudioEncoding)
                            audioEncoding);


                        TSupportedEncodings supportedEncodings = new
                            TSupportedEncodings();
                        supportedEncodings.setAudioEncoding(
                            audioEncodingList);
                        supportedEncodings.setVideoEncoding(
                            videoEncodingList);


                } catch (JAXBException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

}

    List <ControlledTermType> getVideoEncoding()
    {
            //In the future it'll read from elsewhere (configuration
                file may be)
            TermNameType nameType = new TermNameType();
            nameType.setValue("H.263");
            nameType.setPreferred(true);
            ControlledTermType encodingName = new ControlledTermType
                ();
            encodingName.setName(nameType);
            List <ControlledTermType> encoding = new ArrayList<
                ControlledTermType>(3);
            encoding.add(encodingName);
            nameType.setValue("H.263+");//also specified as H263
                -1998
            nameType.setPreferred(false);
            encodingName.setName(nameType);
            encoding.add(encodingName);
            nameType.setValue("H.264");
            nameType.setPreferred(false);
            encodingName.setName(nameType);
            encoding.add(encodingName);
            return encoding;
    }
```

```
        List <ControlledTermType> getAudioEncoding()
        {
                //In the future it'll read from elsewhere (configuration
                    file may be)
                TermNameType nameType = new TermNameType();
                nameType.setValue("PCMU");
                nameType.setPreferred(true);
                ControlledTermType encodingName = new ControlledTermType
                   ();
                encodingName.setName(nameType);
                List <ControlledTermType> encoding = new ArrayList<
                   ControlledTermType>(3);
                encoding.add(encodingName);
                nameType.setValue("PCMA");
                nameType.setPreferred(false);
                encodingName.setName(nameType);
                encoding.add(encodingName);
                nameType.setValue("GSM");
                nameType.setPreferred(false);
                encodingName.setName(nameType);
                encoding.add(encodingName);
                return encoding;
        }


}
```

# Appendix G

**Requirements of the environment**

| Req. No | Description | Specified in |
|---|---|---|
| Basic streaming | | |
| BS-01 | The system shall allow the delivery of video that is user initiated, AS initiated or third party initiated. | |
| | | |
| Advanced streaming | | |
| AS-01 | The system should allow recovering a session after an unwanted/unforeseen interruption (e.g. power loss. | New[1] |
| AS-02 | The system should provide a mechanism whereby the user can query for video based on the different content metadata (e.g. category of content, title, scheduled time, etc). | |
| AS-03 | The system should have ability to flag items for later viewing. | |
| AS-04 | The system should allow the user to define default channel. | |
| AS-05 | The system shall allow the controlling of the program from other devices. For example, the user should be able to enter comment using his mobile or a keyboard of his PC. | |

[1]New requirements defined by the authors

| | | |
|---|---|---|
| AS-06 | The system shall allow users to access a service through multiple UEs simultaneously. | |
| AS-07 | Users shall be able to access multiple services through a UE simultaneously. | |
| AS-08 | The IPTV solution shall provide the capability of receiving and correctly processing the metadata for content coming from the service/content providers. The metadata can come together with the video content or separately. | |
| AS-09 | The metadata delivery without the audiovisual contents of the IPTV service should be possible. | |
| AS-10 | The IPTV system shall associate presence information to IPTV channels or content by including information like program status (on air, delayed 5 minutes, current genre, currently running title, etc) – | |
| AS-11 | The system shall allow the user to rate the content after he finished watching. | New |
| AS-12 | The IPTV system shall allow users to follow the link related to the program that they are watching. For example, if the user wants to buy the boots that a player wore on a football match, then the user can point to the boot and go to the page that he can order the boot or go to the voting page related to a particular show. | |
| AS-13 | The IPTV system should allow the transcoding of video (from VoD or PVR systems) based on the UE's capability. | |
| | | |
| Content switching | | |
| CS-01 | The IPTV solution should support the reuse of reserved resources where appropriate (e.g. media channels reused for subsequent delivery of a different content with same characteristics upon a switch). | |

| | | |
|---|---|---|
| CS-02 | After channel switch, the system shall allow the user to go back to the previous channel with one button click or automatically. | New |
| | | |
| Session transfer | | |
| ST-01 | The IPTV system shall allow capability based session transfer. This means the system shouldn't transfer a session to a UE that is not capable of handling the session to be transferred. | New |
| | | |
| Content insertion | | |
| CI-01 | The IPTV solution shall allow the insertion of content, like advertisings into a video session (CoD). | |
| CI-02 | The IPTV system shall allow the temporary insertion of content, like an important video call, and when done with the inserted content, seamlessly resume the original content. This should happen for both BC and VoD. | |
| | | |
| Timeshifting | | |
| TS-01 | The system shall allow Catch-up on time shift TV, which is stopping the recording and resume watching the live streaming. | |
| TS-03 | The system shall allow "true timeshifting[2]", which is allowing the user to watch the channel from the recording while the system continues to record until the maximum record limit is reached[3]. | New |
| TS-02 | The system shall allow dynamic timeshifting for VIP calls and allow the automatic resumption of the stream after the call. | |

[2]We define the term "True timeshifting" to refer to the case where the PVR will deliver a time shifted stream while at the same time it record the live stream.

[3]The maximum record limit is set by the service provider, which could be disk volume or time

| | | |
|---|---|---|
| | | |
| Bookmarking | | |
| BM-01 | It may be possible for an IPTV Service Provider to offer pre-configured bookmarks. | |
| | | |
| Content marker | | |
| CM-01 | The user should be able to create and name content markers for viewed content or content that he will see in the future. | |
| CM-02 | The user should be able to watch a region of video that he set by a content marker directly. | |
| CM-03 | The content provider can divide the content into sections (with content markers) similar to DVD chapters and the user should be able to see those content markers during session setup | New |
| | | |
| | User profile[4] | |
| UP-01 | The user should be able to maintain his/her preferences regarding IPTV services as such and in combination with other available services to the user in the NGN network. | |
| UP-02 | It shall be possible to define users, within the subscription, with administration rights over other specified users. | |
| UP-03 | The IPTV subsystem shall support the capability for different profile based service personalization (e.g. advertisements, news, traffic information, retrieved from search engines, other). | |
| | | |
| | Recommendation | |

[4]User Profiles refers to all user data that an IPTV system will use to provide IPTV services [TISPAN 182 027]

| | | |
|---|---|---|
| RM-01 | The IPTV Content Recommendation Service should support mechanisms for an IPTV service provider to recommend IPTV contents related to specific user or service provider criteria, e.g.:<br>- Based on user profile, IPTV UE capabilities or user preferences.<br>- Related to the program being watched by the user.<br>- For upcoming programs, or popular movies, etc. | |
| RM-02 | The IPTV service should be able to recommend appropriate clips on system boot up based on users' profiles[5]. | New |
| RM-03 | The IPTV Content Recommendation Service shall support mechanisms for a subscriber to access and set his recommendation profile e.g. to switch on/off regarding receiving recommendations, or to set the criteria of accepting or blocking certain content recommendations and so on. | |
| RM-04 | The IPTV Content Recommendation Service should support mechanisms for an IPTV service provider to provide content recommendation information based on the presence status of IPTV UE or user (e.g. online, offline, or other presence information like willingness to be disturbed by recommendations). | |
| RM-05 | The system shall provide recommendation based on the rating of items from other individuals who liked what the user just watched (i.e. a recommendation similar to what Amazon does – something like "people who watched this also watched that video") | New |
| | | |
| | Notification | |

[5]Other user profiles refer to user's profile in other systems (like on social media)

| | | |
|---|---|---|
| NO-01 | Based on the user's configuration, the system shall provide notifications that include new services content guide, new CoD availability, incoming call, etc. | |
| NO-02 | The IPTV Solution shall provide a mechanism to assign a priority to notifications. Accordingly, the IPTV Solution shall provide a mechanism to guarantee the delivery of critical notification messages with minimal delay (like the use of SMS or other delivery mechanisms that the user chose). | |
| NO-03 | Similar to the traffic updates in Radio Data Systems (RDSs), IPTV channels should have different notification signals to notify the occurrence of important events (for example, news channels can have notification like breaking news or very important messages like state of emergency announcements; sport channels – goal score of important match, etc). Accordingly, the system should allow the user to subscribe to these notifications and get the content automatically based on his/her configuration. | New |
| NO-04 | The system should allow the user to set a reminder about a particular program so that he can be reminded about the occurrence of the particular program. The information from the system shall include at least; channel name, program title and description, actors, scheduled start and end times, genre, parental information and information regarding the location of the recording of the content if the user chooses to record. | |
| NO-05 | The system should build user's interest by tracking his behaviour (following what the user has been watching) for a TV series program and remind him/her if he/she forgets to watch the series and watches some other program next time. The user can opt for recording of the series and keep watching what he/she is watching currently. | New |

| | | |
|---|---|---|
| NO-06 | The system shall be able to notify users about the availability of content and also any changes in the availability of the content notified before. | New |
| | | |
| | Personalized Service Composition | |
| SC-01 | The system shall compose different tracks from different streams and deliver to the user (e.g a football match from a German tv with an English sound track from another channel). | |
| | | |
| | Parental control | |
| PC-01 | The system shall allow parental control based on content rating, content cost, time of day etc, | |
| PC-01 | It shall be possible to define users, within a subscription, with administration rights over other specified users to allow or deny watching specified materials based on content rating, content cost, time of day, etc, | |
| PC-01 | When a user requests an IPTV service, the system shall also allow the contacting of the person who has administrative right on the user to remotely lift the parental control restriction for that particular case. | |
| PC-01 | The parental control authorizer can see who is watching what under his/her subscription (e.g. tracking kids). | |
| | | |
| | Service discovery | |
| SD-01 | Applications should be able to access current schedule of channels. | New |
| SD-02 | The IPTV solution shall support personalized service discovery (e.g. the service discovery based on user preferences (subscription, habit etc.) and device capabilities). | |

| | | |
|---|---|---|
| | | |
| | Converged services | |
| CS-01 | The IMS-based IPTV solution shall allow the blending of TV services with other telecommunication services, like voice or video call, presence, and data services. | |
| CS-02 | The IPTV solution should enable interaction/integration with internet services (e.g. e-mails, web pages or other multimedia services, etc.) | |
| CS-03 | Call takeover (also called stealing or intercepting a call) from a TV (so the call will come to the TV – and the tv will automatically be paused and start to record). | New |
| | | |
| | Non-functional requirements | |
| NF-01 | The IPTV system should be easy to add new features and service. In order to compose different applications, as much as possible, applications and service components should work in asynchronous mode. | New |
| NF-02 | The system shall have a consistent service development environment with regard to programming language and programming style for the different services and sub services. | New |
| NF-03 | As much as possible the service development environment shall support asynchronous event handling. | New |
| NF-04 | As much as possible, service developers need to use the same programming style to develop services that require the different components of the environment. | New |
| NF-05 | Because of the direction of IPTV services towards providing a converged service, the environment should allow an easy adaptation of new protocols. In other words, it should be easy to plug-in new protocol stacks and the programming model should also be the same. | New |

| | | |
|---|---|---|
| NF-06 | The system should support an easy to use mechanism for message exchange b/n modules or components (e.g. the module responsible for PC should inform the module or service that want to use the outcome of the check (i.e. if the user passed the parental control check) in the chain). | New |

# Appendix H

### The modified IPTV user profile schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
targetNamespace="urn:org:etsi:ngn:params:xml:ns:iptv"
xmlns="urn:org:etsi:ngn:params:xml:ns:iptv"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:uep="urn:org:etsi:ngn:params:xml:ns:iptvueprofile"
elementFormDefault="qualified"
attributeFormDefault="unqualified">

<xs:import namespace="urn:org:etsi:ngn:params:xml:ns:iptvueprofile"
schemaLocation="annexP-XML Schema for UE Profile.xsd"/>


<xs:element name="IPTVProfile">
        <xs:annotation>
                <xs:documentation> XML Schema for representing the IPTV
                   Profile object identified in TS 182 027 clause 7.3.1
                </xs:documentation>
        </xs:annotation>
        <xs:complexType>
                <xs:sequence>
                        <xs:element name="PersonalInformation" type="
                           tPersonalProfile"/>
                        <xs:element name="UEProfile" type="
                           uep:tUEProfile" minOccurs="0"/>
                        <xs:element name="GlobalSettings" type="
                           tGlobalSettings" minOccurs="1"/>
                        <xs:element name="BCProfile" type="tBCProfile"
                           minOccurs="0"/>
                        <xs:element name="CoDProfile" type="tCoDProfile"
                            minOccurs="0"/>
                        <xs:element name="PVRProfile" type="tPVRProfile"
                            minOccurs="0"/>
                        <xs:element name="Extension" type="tExtension"
                           minOccurs="0"/>
                        <xs:any namespace="##other" processContents="lax
                           " minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
```

```
                <xs:attribute name="ProfileId" type="xs:ID" />
                <xs:anyAttribute/>
        </xs:complexType>
</xs:element>

        <xs:complexType name="tPersonalProfile">
                <xs:sequence>
                        <xs:element name="Telephone" type="xs:string"
                           minOccurs="0"/>
                        <xs:element name="Email" type="xs:string"
                           minOccurs="0"/>
                        <xs:element name="Birthdate" type="xs:date"
                           minOccurs="0"/>
                        <xs:element name="Anniversary" type="xs:date"
                           minOccurs="0"/>
                </xs:sequence>
        </xs:complexType>

        <xs:complexType name="tBCProfile">
                <xs:sequence>
                        <xs:element name="BCServicePackage" type="
                           tBCServicePackage"
minOccurs="1" maxOccurs="unbounded"/>
                        <xs:element name="Extension" type="tExtension"
                           minOccurs="0"/>
                        <xs:any namespace="##other" processContents="lax
                           " minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>


        <xs:complexType name="tBCServicePackage">
                <xs:sequence>
                        <xs:element name="BCPackageId" type="
                           tBCServicePackageID" minOccurs="1"/>
                        <xs:element name="Description" type="
                           tBCServicePackageDescription" minOccurs="0"/>
                        <xs:element name="BCService" type="tBCService"
                           minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element name="Extension" type="tExtension"
                           minOccurs="0"/>
                        <xs:any namespace="##other" processContents="lax
                           " minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>


        <xs:simpleType name="tBCServicePackageID" final="list
           restriction">
                <xs:restriction base="xs:string">
                        <xs:minLength value="0"/>
                        <xs:maxLength value="16"/>
                </xs:restriction>
        </xs:simpleType>
```

```
<xs:simpleType name="tBCServicePackageDescription" final="list
   restriction">
        <xs:restriction base="xs:string">
                <xs:minLength value="0"/>
                <xs:maxLength value="64"/>
        </xs:restriction>
</xs:simpleType>


<xs:complexType name="tBCService">
        <xs:sequence>
                <xs:element name="BCServiceId" type="
                   tBCServiceID" minOccurs="1"/>
                <xs:element name="QualityDefinition" type="
                   tQualityDefinition" minOccurs="0"/>
                <xs:element name="Extension" type="tExtension"
                   minOccurs="0"/>
                <xs:any namespace="##other" processContents="lax
                   " minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
</xs:complexType>


<xs:simpleType name="tBCServiceID" final="list restriction">
        <xs:restriction base="xs:string">
                <xs:minLength value="0"/>
                <xs:maxLength value="16"/>
        </xs:restriction>
</xs:simpleType>


<xs:simpleType name="tQualityDefinition" final="list restriction
   ">
        <xs:restriction base="xs:unsignedByte">
                <xs:minInclusive value="0"/>
                <xs:maxInclusive value="1"/>
                <xs:enumeration value="0">
                        <xs:annotation>
                                <xs:documentation>
                                        <label xml:lang="en">SD<
                                           /label>
                                        <definition xml:lang="en
                                           ">Standard Definition
                                           </definition>
                                </xs:documentation>
                        </xs:annotation>
                </xs:enumeration>
                <xs:enumeration value="1">
                        <xs:annotation>
                                <xs:documentation>
                                        <label xml:lang="en">HD<
                                           /label>
                                        <definition xml:lang="en
                                           ">High Definition</
```

```
                                        definition>
                                </xs:documentation>
                        </xs:annotation>
                </xs:enumeration>
        </xs:restriction>
</xs:simpleType>


<xs:complexType name="tCoDProfile">
        <xs:sequence>
                <xs:element name="ParentalControl" type="
                   tParentalControlLevel" minOccurs="0"/>
                <xs:element name="Extension" type="tExtension"
                   minOccurs="0"/>
                <xs:any namespace="##other" processContents="lax
                   " minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
</xs:complexType>


<xs:simpleType name="tParentalControlLevel" final="list
   restriction">
        <xs:restriction base="xs:unsignedByte">
                <xs:minInclusive value="0"/>
                <xs:maxInclusive value="3"/>
                <xs:enumeration value="0">
                        <xs:annotation>
                                <xs:documentation>
                                        <label xml:lang="en">ALL
                                           </label>
                                        <definition xml:lang="en
                                           ">All contents</
                                           definition>
                                </xs:documentation>
                        </xs:annotation>
                </xs:enumeration>
                <xs:enumeration value="1">
                        <xs:annotation>
                                <xs:documentation>
                                        <label xml:lang="en">
                                           Level 1</label>
                                        <definition xml:lang="en
                                           ">Level 1 contents</
                                           definition>
                                </xs:documentation>
                        </xs:annotation>
                </xs:enumeration>
                <xs:enumeration value="2">
                        <xs:annotation>
                                <xs:documentation>
                                        <label xml:lang="en">
                                           Level 2</label>
                                        <definition xml:lang="en
                                           ">Up to level 2</
                                           definition>
```

```
                                </xs:documentation>
                        </xs:annotation>
                </xs:enumeration>
                <xs:enumeration value="3">
                        <xs:annotation>
                                <xs:documentation>
                                        <label xml:lang="en">
                                           Level 3</label>
                                        <definition xml:lang="en
                                           ">Up to level 3</
                                           definition>
                                </xs:documentation>
                        </xs:annotation>
                </xs:enumeration>
                <xs:enumeration value="4">
                        <xs:annotation>
                                <xs:documentation>
                                        <label xml:lang="en">
                                           Level 4</label>
                                        <definition xml:lang="en
                                           ">Up to level 4</
                                           definition>
                                </xs:documentation>
                        </xs:annotation>
                </xs:enumeration>
                <xs:enumeration value="5">
                        <xs:annotation>
                                <xs:documentation>
                                        <label xml:lang="en">
                                           Level 5</label>
                                        <definition xml:lang="en
                                           ">Up to level 5</
                                           definition>
                                </xs:documentation>
                        </xs:annotation>
                </xs:enumeration>
        </xs:restriction>
</xs:simpleType>


<xs:complexType name="tPVRProfile">
        <xs:sequence>
                <xs:annotation>
                        <xs:documentation>
                                Unit of the StorageLimitInVolume
                                    element is the GigaOctet
                        </xs:documentation>
                </xs:annotation>
                <xs:element name="PVRPreference" type="
                   tPVRPreference"/>
                <xs:element name="StorageLimitInTime" type="
                   tNPVRStorageLimitInTime" minOccurs="0"/>
                <xs:element name="StorageLimitInVolume" type="
                   tNPVRStorageLimitInVolume" minOccurs="0"/>
```

```
                <xs:element name="Extension" type="tExtension"
                    minOccurs="0"/>
                <xs:any namespace="##other" processContents="lax
                    " minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
</xs:complexType>

<xs:simpleType name="tPVRPreference" final="list restriction">
        <xs:restriction base="xs:unsignedByte">
                <xs:minInclusive value="0"/>
                <xs:maxInclusive value="1"/>
                <xs:enumeration value="0">
                        <xs:annotation>
                                <xs:documentation>
                                        <label xml:lang="en">
                                            Network</label>
                                        <definition xml:lang="en
                                            ">Recording is done
                                            in the network</
                                            definition>
                                </xs:documentation>
                        </xs:annotation>
                </xs:enumeration>
                <xs:enumeration value="1">
                        <xs:annotation>
                                <xs:documentation>
                                        <label xml:lang="en">
                                            User_Equipment</label
                                            >
                                        <definition xml:lang="en
                                            ">Recording is done
                                            on the user equipment
                                            </definition>
                                </xs:documentation>
                        </xs:annotation>
                </xs:enumeration>
        </xs:restriction>
</xs:simpleType>

<xs:simpleType name="tNPVRStorageLimitInTime">
        <xs:restriction base="xs:duration">
                <xs:minInclusive value="PT0H"/>
                <xs:maxInclusive value="PT1000000000H"/>
        </xs:restriction>
</xs:simpleType>

<xs:simpleType name="tNPVRStorageLimitInVolume">
        <xs:restriction base="xs:nonNegativeInteger"/>
</xs:simpleType>

<xs:complexType name="tGlobalSettings">
        <xs:sequence>
                <xs:element name="LanguagePreference" type="
                    tLanguage" minOccurs="0"/>
```

```
                        <xs:element name="UsersActionRecordable" type="
                            tUserActionRecordable"/>
                        <xs:element name="Extension" type="tExtension"
                            minOccurs="0"/>
                        <xs:any namespace="##other" processContents="lax
                            " minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>


        <xs:simpleType name="tLanguage">
                <xs:restriction base="xs:string">
                                        <xs:annotation>
                                        <xs:documentation>
                                          <definition xml:lang="en">ISO
                                              639-2 Language code</
                                              definition>
                                        </xs:documentation>
                                </xs:annotation>
                        <xs:minLength value="3"/>
                        <xs:maxLength value="3"/>
                </xs:restriction>
        </xs:simpleType>

        <xs:complexType name="tExtension">
                <xs:sequence>
                        <xs:any processContents="lax" minOccurs="0"
                            maxOccurs="unbounded"/>
                </xs:sequence>
    </xs:complexType>

        <xs:simpleType name="tUserActionRecordable">
                <xs:restriction base="xs:boolean"/>
        </xs:simpleType>

</xs:schema>
```

# Appendix I

**Data dictionary**

| Tables | Field | Type | Null | Default | Remark |
|---|---|---|---|---|---|
| Users | User Id | String | No | | IMPU |
| | First Name | String | No | | |
| | Last Name | String | No | | |
| | Last Date New Videos Notified | Date/Time | No | | |
| | Last Date Bootstrap Service Delivered | Date/Time | | | |
| | | | | | |
| Video | Video Id | String | No | | |
| | Director Id | String | No | | |
| | Title | String | No | | |
| | Genre | String | No | | |
| | Year Released | Date/Time | No | | |
| | Date acquired | Date/Time | No | | |
| | Metadata | String | No | | Keywords of the video |
| | | | | | |
| Directors | Director ID | String | No | | |

| | | | | | |
|---|---|---|---|---|---|
| | First Name | String | No | | |
| | Last Name | String | No | | |
| | | | | | |
| Actors | Actor Id | String | No | | |
| | First Name | String | No | | |
| | Last Name | String | No | | |
| | | | | | |
| VideoActor | Id | String | No | | |
| | Video ID | String | No | | |
| | Actor Id | String | No | | |
| | | | | | |
| | | | | | |
| UserPreference | User ID | String | No | | |
| | Genre | String | No | | |
| | | | | | |
| Rating | User Id | String | No | | |
| | Video Id | String | No | | |
| | Rating Value | Number | No | | |
| | | | | | |
| Reminder | User Id | String | No | | |
| | Program Id | String | No | | |
| | Scheduled Date | Date/Time | No | | |
| | Scheduled Time | Date/Time | No | | |
| | Expiry Date | Date/Time | Yes | | |
| | | | | | |
| VIP List | User Id | String | No | | |

| | | | | | |
|---|---|---|---|---|---|
| | First Name | String | No | | |
| | Last Name | String | No | | |
| | | | | | |
| Special Clips | Birthday Clip | String | No | | URL of the clip |
| | Anniversary Clip | String | No | | URL of the clip |