Towards a Toolkit for creating video-oriented services for Mobicents

Zelalem S. Shibeshi¹, Alfredo Terzoli², Karen Bradshaw²

Department of Computer Science
Rhodes University, P. O. Box 94, Grahamstown 6140

Tel: +27 46 6038247, Fax: +27 46 6361915

email: zelalems@rucus.ru.ac.za¹; {A.Terzoli, K.Bradshaw}@ru.ac.za²

Abstract—The research community seems to have settled on the selection of an IMS core testbed for testing IMS services. The “Open IMS Core” has been the de facto standard for IMS core by the academic research community for a long time. On the other hand, there is no single service development platform that has become a de facto standard for multimedia service development for IMS. One important subset of video oriented services in the IMS environment is IPTV and there is no obvious choice for IPTV service development from the open source community. The IPTV specification lists the use of various protocols to implement an IPTV service. There are also different options for implementing these protocols. For example, using the different platforms that implement these protocols and combining them to develop and implement IPTV services. This, however, has disadvantages such as lack of consistency and incompatibility with each other. A relatively recent open source service development environment, Mobicents, could solve these problems. Mobicents brings together implementations of different IPTV protocols, and provides a platform for service developers. Mobicents is still a complex framework and requires a steep learning curve. Programmers who do not know about telecom protocols or the Mobicents programming model should be able to use its functionalities and develop video-oriented services easily. In this paper we discuss the design of a Toolkit to ease the development of video-oriented services in Mobicents¹.

Index Terms—Multimedia service, Multimedia service development environments, IPTV, Mobicents

I. INTRODUCTION

Because video provides good user experience, most services on the Internet, including the mushrooming social networks, include video as part of their service. A recent article published on BBC website [1] revealed that YouTube receives 2 billion hits per day in 2010. This number might have increased in the past one year. Because of this huge demand, Telcos are also trying to tap new technologies in the field of multimedia service delivery by embracing IPTV, that can be used to provide a wide variety of video-oriented services. IPTV is more than just what its name implies and includes various types of additional services to make the TV a better entertainment and communication environment. The IPTV architecture allows one to develop video services by including different features that enable transmission of a variety of content either in real time, on demand, with the ability to stop and go.

The services that IPTV provides could not have been considered with the previous technologies of broadcast TV. Nevertheless, despite all its promises, the IPTV implementation has 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 could be the technical challenge that IPTV pose for service developers. The specification defines various components that need to be implemented to provide the basic IPTV service. And going deeper into the components one can see that some of the components can even be further broken down based on the required functionalities and protocols required to implement them. In fact, to create a full-fledged IPTV service, one needs to utilize various protocols some of which can be found as stand alone services or enablers and requires the understanding of how to compose all these things. For this purpose we have embarked on a project to develop a toolkit with which service developers, without having any prior knowledge of the telecom protocols can develop IPTV services more easily and with minimal effort.

As a first step in developing the toolkit, we have examined development environments that have implemented the different protocols that an IPTV service developer requires. Considering the open source development and deployment environments, the above mentioned protocols are implemented by various institutions and companies and there are many compatibility issues when trying to use these components and tools. As a result, it would be beneficial if one were to obtain an implementation of at least the majority of these protocols from one source so as to avoid the incompatibility issues mentioned above. In this regard, we conducted an investigation to identify a proper development environment on which we could base our toolkit for IPTV service. This paper presents the different development environment that are available for service developers and shows how the Mobicents [2] environment helps in the implementation of innovative and converged services for IPTV users.

¹This work is being carried out in the Distributed Multimedia Centre of Excellence at Rhodes University, with financial support from Telkom, Tellabs, Stortech, Eastel, Bright Ideas 39, and THRIP.
The paper is organized as follows. Section II describes the IPTV architecture and its services, while section III presents the development of open source development tools and environments. Section IV describes the Mobicents service development environment and section V presents how Mobicents can be used to develop IPTV components, followed by a discussion in Section VI. Finally, Section VII presents our conclusion and future work.

II. IPTV Architecture and Services

There are various proposals of implementation standards for IPTV by different standards bodies. The major IMS standards bodies are: the 3rd Generation Partnership Project (3GPP) [3], European Telecommunications Standards Institute (ETSI) Telecommunications and Internet converged Services and Protocols for Advanced Networking European Telecommunications Standards Institute (TISPAN) [4], and the Telecommunication Standardization Sector of International Telecommunications Union (ITU) [5]. However, the specification that has received the greatest emphasis for the development and testing of IPTV service is the IMS-based IPTV architecture proposed by ETSI-TISPAN. We also considered the architecture of IPTV specified by this standards body in this paper.

Figure 1 presents the IPTV architecture as specified in [4]. As shown in Figure, the major components of this architecture specific to IPTV are: Service Discovery and Selection, Service Controller, and Media Control and Delivery. Apart from this, the other components in the architecture include the Transport Function which is responsible for transport, network attachment and resource reservation tasks, and the IMS core.

![Figure 1: Functional architecture of IPTV](image)

The IMS core [6] contains all the core elements of IMS/NGN architectures specified by the various standard bodies. It controls the IMS networks, and manages the routing of messages and sessions in the network.

The function of the IPTV components and the major IPTV services one can expect from IPTV are presented in the following sections.

A. Functional description of IPTV components

1. The Service Control Function (SCF)

As any SIP-based application server, the IPTV Service Composition Function (SCF) controls the IPTV session and service. In general, the SCF performs service authorization and credit control, selection of the relevant IPTV Media Function, and forwarding the request to the MCF. Basically, the service logic is stored in this unit. The SCF uses SIP, HTTP, and Diameter protocols to deliver the above mentioned functions. Optionally, it can also support the XCAP protocol.

2. The Service Selection and Discovery Function (SSF/SDF)

This unit is responsible for service discovery and selection functions. After registration, the first step in accessing an IPTV service by any IPTV User Equipment (UE) is the identification and selection of a service that the user desires to access. This is done by contacting these units. Specifically these units provide the following functions:

- The SDF provides information about the user’s IPTV services and where the user can select the services. The UE uses SIP to communicate with SDF.
- Once the UE obtains the service description, it contacts the SSF to retrieve relevant information about the specific IPTV service, like the URL of the media to initiate the session. SSF uses HTTP protocol.

3. The Media Function (MF)

The MF is responsible for controlling and delivering media to the UE. The control unit of the IPTV MF is the Media Control Function (MCF) and the delivery unit is the Media Delivery Function (MDF). In the case of Content on Demand IPTV service, the MCF is responsible for initiating the media session by contacting the MDF that is supposed to serve the particular user and it should be developed as a SIP AS.

B. Major IPTV Services

As mentioned before, IPTV is not just a single service, rather it is a combination of various services. The major IPTV services are Content on Demand (CoD), Personalized Video Recorder (PVR) and Broadcast (BC) services. These services can be delivered in a variety of ways and personalization is one of the innovations that distinguish IPTV from traditional TV products. In order to provide a personalized IPTV service, the architecture provides a mechanism for user profile management by a centralized unit.

Another interesting feature of IPTV is notification. IPTV utilize the SIP event-notification and subscription facility to implement this feature. For example using this feature, one can develop an IPTV service that provides dynamic content insertion and content switching based on a certain type of event.

All of the above services can be developed in a variety of ways. The specification, however, dictates that service developers use specific protocols for interoperability purpose. The following lists the protocols specified in the IPTV specification.

C. Protocols used to provide the various IPTV services.

 SIP/SDP: SIP, apart from its standard use, is also used for service discovery in the IPTV architecture. SDP is used to describe SIP based sessions.

RTSP: Used for media set-up and control.

Diameter: Used for authentication and charging

XCAP: Used to set, retrieve and modify user profile

HTTP: Used for retrieving information from the SSF.
III. OPEN SOURCE SERVICE DEVELOPMENT ENVIRONMENTS

As our focus in this paper is on service development environments for IPTV, we only present freely available open source development and deployment environments to deliver the service that the IPTV specification specifies. As discussed in [6] and [8], there are various specialized service development environments for IMS.

Comparison of SIP based service development and deployment environment is presented in [6] and [8]. Nevertheless, not all development tools provide support for all the protocols that IPTV service developers need. So, in addition to the environments mentioned in the articles referenced above, there are also tools that can give support for the development of XCAP client/server and Diameter client/server, the two most important protocols required for IPTV service development. The environments analyzed and presented in the table are Kamailio (an open source SIP server), OpenXCAP (an open source XCAP server), Asterisk (an open source IP PBX), and SailFin (open source SIP Servlet execution environment). The MCF component should also have an RTSP capability and we compare the platforms to check if they support this protocol. The interface or protocol between MCF and MDF is not specified in the specification and it will be difficult to compare the platforms for conformity of this interface. The following table presents the analysis of these open source service development tools. The platform used to test these tools is Linux.

<table>
<thead>
<tr>
<th>Technology/Feature</th>
<th>Kamailio</th>
<th>OpenXCAP</th>
<th>Asterisk</th>
<th>Mobicents</th>
<th>SailFin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming language used</td>
<td>C</td>
<td>Phyton</td>
<td>C</td>
<td>Java</td>
<td>Java</td>
</tr>
<tr>
<td>Technology used</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>JAIN-SLEE, Servlets</td>
<td>Servlets</td>
</tr>
<tr>
<td>Supported protocols</td>
<td>SIP,XCPA</td>
<td>XCAP</td>
<td>SIP, IAX</td>
<td>SIP/HTTP, XCAP, Diameter(^2)</td>
<td>SIP/HTTP</td>
</tr>
<tr>
<td>Support for media control and delivery</td>
<td>No</td>
<td>No</td>
<td>Yes – MGCP(^3)</td>
<td>Yes – with various protocols(^3)</td>
<td>No</td>
</tr>
<tr>
<td>Other features</td>
<td>No</td>
<td>No</td>
<td>timers, traces and alarms</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Technologies for the development of IPTV services.

As can be seen from Table 1, only Mobicents provides support for most of the IPTV service development protocols. But there is also an option of combining the different platforms or tools to develop the same service. Nevertheless, as described in [8], it is always disadvantageous to use different platforms for various reasons. First, the different platforms depend on several independent libraries that service developers have to install before using these tools. Another obvious challenge is the programming language requirement that these tools pose for the developer. On the other hand, even when the services are correctly installed, their appropriate behavior needs to be configured. The configuration issues that these tools bring is explained in [8].

Accordingly, from the above analysis, it is evident that Mobicents gives better support than the other open source platforms for the development and deployment of IPTV services. The next section presents the different technologies used in the Mobicents service development environment.

IV. THE MOBICENTS SERVICE DEVELOPMENT ENVIRONMENT

Mobicents is a Java based service delivery platform for quick development, deployment and management of next generation network applications [9]. It is JAIN SLEE 1.0 compliant, and adopts data from various IP protocols for use as Java objects. This functionality is achieved through JAIN-SLEE components called Resource Adaptors (RA) [10]. Some of the resource adaptors in Mobicents are SIP, HTTP and Diameter RAs. Mobicents uses JAIN SLEE units known as Service Building Blocks (SBBs) for composition and building of services. Figure 2 shows the different components of Mobicents.

SBBs are reusable objects that can both send and receive events, and are responsible for processing events based on application-defined logic as well as their own internal state.

The event router in the SLEE forwards events to SBBs that have explicitly shown interest by “registering” themselves for those events. SBBs also fire events to signal other SBBs.

![Network Functions](image)

Figure 2: The JAIN SLEE Architecture (adapted from [10]). Service logics are structured hierarchically in an SBB tree weight assigned to enable SLEE to determine the order of execution. A hypothetical SBB tree is presented in Figure 3. The developer is responsible for defining the weight and including this information, together with the events of interest to each SBB, in the deployment descriptor. In addition to the service component model, the SLEE also defines management interfaces used to administer the container and the service components executing within, and a set of standard facilities, which provide common features, such as timers, traces and alarms, to JAIN SLEE components.

Mobicents also provide an easy to use SIP Servlets execution environment as an alternative to JAIN SLEE. Mobicents’ SIP Servlets deliver a consistent, open platform on which to develop and deploy portable and distributable SIP and Converged J2EE services. The following section

\(^2\) Based on the Jdiameter stack.

\(^3\)Does not support RTSP natively. A separate application needs to be installed to give Asterisk RTSP support.

\(^3\)There is a stand-alone RTSP server and support specific file type. However, there is no RTSP adapter.
shows how one can use Mobicents to develop the IPTV service components.

This section discusses how Mobicents can be used to provide an IPTV service called Personalized Service Composition (PSC).

The recent specification of IPTV presents an IPTV service called Personalized Service Composition (PSC) used to compose multiple IPTV sessions and allow the user to watch them dynamically without any intervention. This service is advantageous especially for disabled people as another person can configure the IPTV sessions thereby allowing them to watch without requiring constant supervision.

In this section we only show the implementation of this service from the point of view of the communication between the UE, SCF and MCF. For the purpose of simplicity no mention is made of how the user profile is processed: issues related to authentication and charging are also not mentioned. The support of Mobicents for XCAP is shown in [11] and for Diameter in [12].

The IPTV specification demands that the request for this type of service be submitted using an INFO event package together with SMIL protocol, but to date no open source INFO event package implementation is available. As a result, for the sake of simplicity, we used a SIP MESSAGE command to pass this request. We defined a specific message structure for this purpose. Consequently, a user wishing to create the service composition sends a SIP MESSAGE with the following structure: *PSC*SendchannelsTo-myUE@imstv.ru.ac.za*firstMediaDuration-firstMedia; nextMediaDuration-nextMedia; thirdMediaDuration-thirdMedia# etc. where xMediaDuration is the time xMedia should play. Figure 3 shows a flow diagram of the service. The media switch aspect is described in [11] and for Diameter in [12].

To provide the above example at least two SBBs must be implemented: a root SBB (that handles registration and checks if users are allowed to access the IPTV services), and the PSC SBB that handles the logic of the PSC service. This also implies that the developer has to provide three deployment descriptors, one for the entire service (service.xml), and one for the different SBBs. The PSC service also has to implement different interfaces, including an interface to access the SLEE facilities (CMP fields and Timer facility). Of course, there will be different event handler methods as well.

The same is true for the MCF. In addition to the SBB that handles the SIP messages, a special interface has to be implemented to communicate with the RTSP proxy for RTSP support.

The above explanation covers the different interfaces one needs to understand and implement in Mobicents in order to develop simple services like the one presented above. And this is just the beginning. There are other interfaces and relationships that must be implemented and developed when creating complex services. The following section summarizes the major issues surrounding the Mobicents development environment and shows how a toolkit for IPTV service development solves this problem by giving a high level of functionality to the toolkit.

VI. DISCUSSION

The Mobicents environment has implemented several of the required protocols. This is a major advantage for service developers. Nevertheless, as can be seen from the project website, JAIN SLEE is a more complex specification than SIP Servlets, and is known to be heavyweight with a steep learning curve[13]. However, the protocol agnostic nature of JAIN SLEE makes it a superior choice to SIP Servlets. On the other hand, the SIP Servlet environment also lacks a rigid component model that separates call control from the business logic classes and persistent layer [13]. In fact, the ease of SIP Servlets is also debatable, because a SIP Servlet requires that developers understand much detailed control information of the SIP protocol, which adds to the learning curve of development skills[14].

In general, in addition to learning the various protocols that are required to develop an IPTV service, a developer is required to learn complex frameworks like Mobicents to
A. Issues with Mobicents SDP

- The different protocol requirements of IPTV require its development to include several different SBBs. In addition to a deployment descriptor for the entire service, a separate deployment descriptor also has to be supplied for each component. As the service becomes more and more complex, so too does the composition. Figure 4 shows a particular SBB entity tree.

- For complex services an SBB might need to implement several interfaces and methods to properly provide the services.

- Mobicents uses the Java project management tool Maven along with its associated POM (Project Object Model) files to describe projects, resources and services, and to deploy these components onto the JBoss server. The extended dependencies, and dependency clashes results in Maven being far too complex for average user.

The above points demonstrate how complex the Mobicents development framework is. Nevertheless, as Mobicents brings together most of the required protocol implementations for IPTV, we consider it to be a step closer to achieving our aim of developing a toolkit. The toolkit would hide the above complexities and provide an interface to service developers with a generic interface to access and compose the services based on the protocols that Mobicents provides.

B. Issues with media processing

1. Issues with the SDP protocol

The IPTV specification suggests the use of H263-1998 (200) as the video codec for IPTV. Because the payload type of this codec falls under the dynamic payload type in [15], different implementations give it different payload values. As a result a stringent matching algorithm may produce an error. So media negotiation has to be done with great care.

On the other hand, SDP does not include session id when using streaming servers, contrary to the SDP we get in a SIP session, which does contain session-id. Again care must be taken when processing SDPs for different sessions.

2. Issues with streaming servers

Different streaming servers differ with respect to how they specify some of the RTSP parameters. For example Darwin and VLC present the media control parameters of different tracks of a streaming session differently. VLC sends the track information as a=control:track #, whereas Darwin sends it as a=control:TrackID=# (where# is the track id number). So again a simple matching algorithm may produce the wrong result.

On the other hand, if one wants to bookmark a streaming session, the server needs to be queried to get the current play time. The recent RTSP protocol specifies that get parameter with only a Range format and no value can be sent to a streaming server to request the current media position. Nevertheless both Darwin and VLC do not support this feature and produce an error.

In general, the toolkit, in addition to avoiding the complex programming framework of Mobicents, will also abstract all protocol related issues away from the developer.

C. How the toolkit can help service developers

Any video-oriented service contains two types of message processing: the SIP session message processing and the media session message processing. Focus will be given to the media session message processing in this paper.

With regard to the SIP signaling process, the toolkit will process all service requests as a connection object and the developer will be provided with methods to manipulate the connection object that is obtained when a request arrives or when he/she create a request himself/herself. The toolkit will also provide an easy to understand response and status message sent back to the application.

1. Architecture of the toolkit

Figure 5 presents the architecture of the toolkit.

![Figure 5. Architecture of the toolkit](image)

2. Description of major components in the toolkit

As can be seen from Figure 5, the toolkit has three layers. The lower layer is composed of protocol specific components. Basically, except for the RTSP and SIP protocols the components in the lower part consist of clients for the different protocols that communicate with the translator component, which is found in the middle layer. Because the RTSP and SIP components also receive request and are required to respond to client request they are developed as servers for the respective protocol.

The translator layer translates the protocol specific messages into a toolkit message by creating an object for the messages and forwarding them to the event router. As mentioned above, for example, the SIP Invite request will be translated into a newConnection request message.

The top layer is the event router that fires the event coming from the middle layer and also accept messages from the application.

3. Programming model for the development of the toolkit

The different SIP based message processing functionalities are provided as APIs that the developer needs to implement the abstract classes. Developers can use generic Java to develop services.
4. Examples of using the toolkit
For the example presented in Section V, the VoDSCF and VoDMCF abstract classes of the toolkit can be used.

The VoDSCF will have an onNewConnection event handler method, which when instantiated will check the user's credit before it forwards the message to the VoDMCF object. The connection object can also be used to modify SIP sessions by calling methods like transfer, which will transfer the session to another UE. The SIP message communications required for session transfer will be handled by the toolkit.

The VoDMCF application will then call the createMedia(url) method using the required URL parameter. The createMedia method will also check the compatibility of the media to the UE's capability and will provide error message if they are not compatible. When media setup is ready, the VoDMCF will use the startMedia method to start playing the media. The developer can also use the VoDMCF's modifyMedia(list, listType) method by supplying the two parameters. The id parameter gives the type of modification required. ListType 1 is a single file, 2 is a simple playlist and 3 a PSC list. The list will contain the list of media items to be played in an XML format. As such these methods can be used to develop a video mail service or a PSC service.

Another interesting feature of the VoDMCF API is that it provides the SIP session and the media session as one lifting the burden from the developer.

5. How to setup the protocol stack
The service developer has to provide configuration information in the configuration file of the stack. The toolkit has a parser to obtain the configuration information to initialize the stack.

D. Current Implementation
1. IPTV SCF
We have implemented a basic IPTV AS using Mobicents SBB. This will be expanded to include all aspects of the IPTV AS specified in Section II. As Mobicents uses the JAIN SIP API, we will use the code to re-create the AS so as to avoid its dependability on the SLEE environment.

2. RTSP Enabled Media Function Controller
In a similar way to the SCF, the MCF is created as an SBB and has the capability of negotiating media capabilities. As mentioned in [16], we have developed an RTSP proxy to use streaming servers as a MDF unit. The proxy has an RTSP client and server unit implemented using the Apache Mina Framework. This will be included in the toolkit to provide an IPTV service. The proxy also includes media switching and media packet handling functionalities.

VII. Conclusion
The promises of IPTV can be realized easily if an appropriate service development environment or toolkit exists. The development should be fast and easy so as to tap the ever existing high demand for video-oriented services. In this paper we have shown how complex it is to develop a service in Mobicents, a platform that supports most of the IPTV protocols, and how a toolkit can ease the development effort of IPTV service developers. With the basic functionalities of the IPTV service provided, developers can then concentrate on including other innovative aspects such as integrating the service with features like a calendar service, to provide a service that sends a birthday clip to the user on his TV on his birthday. Another example could be integrating the service with an email system.

References

Mr. Zelalem S. Shibeshi holds an MSc in Information Science, Diploma in Computer Science and BSc in Physics, all from Addis Ababa University, Ethiopia, and is currently working toward his PhD in the Computer Science Department at Rhodes University.