Mobicents as a Service Creation and Deployment Environment for the Open IMS Core

Mosiuoa Tsietsi, Alfredo Terzoli and George Wells
Department of Computer Science
Rhodes University, P. O. Box 94, Grahamstown 6140
Tel: +27 46 6228642, Fax: +27 46 636 1915
email: m.tsietsi@rucus.net | {a.terzoli, g.wells}@ru.ac.za

Abstract – The Open IMS Core is a testbed environment that allows researchers to experiment with components of a next generation network, providing open source implementations of the IP Multimedia Subsystem call session control functions, as well as a Home Subscriber Server. As a control platform for IP communications, IMS behaves as a docking station for multimedia services which can be developed using various programming platforms. Open source implementations of certain multimedia servers do currently exist which are based on different platforms and have been used by developers to enhance their testbeds. However, there is a practical downside to a heterogeneous service deployment strategy that employs disparate communication services in an expanding testbed. In order to reduce time-to-market, a consistent service creation and deployment platform is needed, which is comprehensive enough to deliver most of the services currently available, but in a unified manner. The Mobicents platform is a suitable candidate for this requirement, bundling a number of multimedia servers and possessing interfaces to several popular IP protocols, all within the context of a common JAIN SLEE standard-compliant infrastructure, allowing a more consistent service creation and deployment experience for developers. This paper compares the various components of Mobicents with a few popular open source servers that have been used by the researchers in conjunction with the Open IMS Core, and shows the benefits of its use as a service creation and deployment platform.

Index Terms – IP Multimedia Subsystem, service creation, service deployment

I. INTRODUCTION

THE IP Multimedia Subsystem (IMS) is an all-IP framework that has been standardised by major bodies such as the 3GPP, ETSI TISPAN and PacketCable as a component of their Next Generation Network (NGN) specifications. IMS is a complex environment, enabling rich multimedia services in a converged fixed/mobile ecosystem across different access networks. One way of addressing this complexity is through promoting experimentation by providing an open sandbox environment that stakeholders can use to educate themselves on NGN principles [13]. An example of such an environment is the Open IMS Core [7].

The Open IMS Core was developed by Fraunhofer FOKUS in 2006. The system is largely based on the SIP Express Router (SER) project, which is used to provide the Proxy, Interrogating and Serving Call Session Control functions (CSCFs) of an IMS. A lightweight implementation of a Home Subscriber Server (HSS) is also provided which acts as repository for application server settings as well as service and user profiles. These four elements form the core of an IMS testbed, and whilst they are responsible for session control which is essential for supporting services, they do not provide the services themselves. Rather, services must be provisioned by the developer on the application plane.

The three main service provisioning platforms for IMS are Open Service Access (OSA) / Parlay, IP Multimedia Service Switching Function (IM SSF) and Session Initiation Protocol (SIP) applications. However, the latter is proving to be a popular platform for developing services due to the rapid proliferation of SIP protocol stacks which are readily available in open source on the Internet.

In this paper, we will describe some of our earlier efforts to integrate multimedia services, based largely on the SIP protocol, into an Open IMS Core testbed. The platforms described do not represent an exhaustive list of all possible open source platforms, but merely those that were of interest to us and seemed to have a large developer community. In section III we will highlight some of the drawbacks we have experienced while using these disparate platforms. Section IV introduces JAIN SLEE as an emerging service engineering specification and Section V describes Mobicents as an implementation of the JAIN SLEE standard, which has become our unified framework for service creation in the IMS. Finally, the new testbed that uses Mobicents is described and briefly analysed in section VI.

II. RELATED WORK

Open source software often poses as much of an opportunity as it does a challenge, and the Open IMS Core is no different. To understand how service integration is performed, and what platforms have proven to work well with Open IMS Core, much guidance is needed. Resources such as the Open IMS Core users' mailing list and the accounts of developers, who have integrated their services such as in [25], are invaluable to any developer working in this area. Our exposure to these resources has led to the identification of a set of open source tools that do work adequately.

A. OpenSER / Kamailio

OpenSER is a fork project of SER, allegedly arising out of dissatisfaction with the increased commercialisation of SER and doubts about the direction of the parent project

[8]. OpenSER was subsequently renamed to Kamailio for trademark reasons. As an offshoot of SER, Kamailio aims to deliver a similar level of flexibility and high performance. Like SER, it is also written in the C programming language and uses similar configuration syntax to fine-tune its behaviour. Kamailio ships with a number of optional add-on modules which can be compiled and built into an installation for added functionality. These additional modules are what can empower Kamailio to perform as an IMS Application Server (AS) in the ways described in the sections that follow.

1) SIP Application Server:

A SIP Application Server (SIPAS) can host and execute services and has the ability to manipulate a SIP session by using the IMS Service Control Interface (ISC) in conjunction with the S-CSCF [2]. The ISC supports event notification between the SIP AS and the S-CSCF so that the SIP AS receives information about the registration state of the users and the capabilities of the user's SIP User Agent (SIP UA). The S-CSCF uses initial filter criteria (iFC) to relay a user request accurately for service handling. Kamailio can be used to play the role of a SIP AS in an IMS network by provisioning rules in the Kamailio configuration scriptlet so as to handle SIP messages as they are received from the S-CSCF, and performing some application logic.

2) SIP Presence Server

Possibly one of the most popular services in IMS is presence, which can be defined as the ability to convey information regarding availability or willingness to communicate with others. A SIP Presence Server (SIP PS) behaves as a specialised SIP AS, whereby iFC specify that the SIP presence events such as SUBSCRIBE, NOTIFY and PUBLISH are to be forwarded to it. Kamailio has a general purpose **presence** module, which is an event-package independent handler for SUBSCRIBE and PUBLISH methods as well as a NOTIFY generator in accordance with relevant SIP specifications [3, 20]. In addition, Kamailio possesses two presence-related client modules: **presence_xml** and **presence_mwi**. The former registers presence events while the latter registers message waiting events to be handled by the **presence** module according to [22], [15] and [14] respectively. Figure 1 depicts the interactions between a SIP client, the **presence** module and event processing with **presence_xml**.

3) OpenXCAP Document Server

It is uncommon for systems nowadays to allow direct access to the presence data of a presentity by a watcher without a policy framework in place that grants or withholds such access. Users also need some mechanism to manage the list of buddies or contacts with which they would like to communicate, or even to set up a single address that indexes a number of contacts for easier use. The terms presence rules, resource lists and resource list services are used to describe each of these scenarios respectively. These are just three possible use cases for the XCAP protocol. XCAP is an IETF protocol which allows a client software to read, write and modify application configuration data stored in XML format on a server using HTTP protocol methods such as GET, PUT and DELETE [23]. Each of the three mentioned uses of the XCAP protocol has its own XML schema for representing data. OpenXCAP is an example of a free, fully featured, open source, XML Document Management Server (XDMS) [19] that respects the relevant IETF standards already mentioned that relate to presence, as well as the Open Mobile Alliance (OMA) SIMPLE framework [4]. It is written in the Python programming language and allows XCAP clients to create and manipulate XCAP documents which it stores in a database such as MySQL. In Figure 1, we can substitute OpenXCAP for the XCAP server and it becomes clear to see, for example, how authorisation policies stored in the database can affect the SIP signalling between the SIP client and the SIP PS. Kamailio uses an XML RPC module to interface with OpenXCAP in order to retrieve XCAP documents and to receive updates made by users.

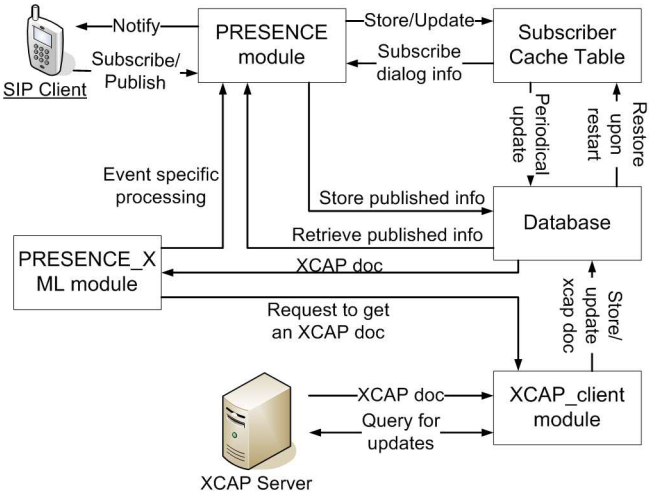

Figure 1: OpenSER/Kamailio's Presence Infrastructure. Adapted from [9]

4) Resource List Server

In order to implement the functionality needed to support the resource list services application of XCAP, a Resource List Server (RLS) is required. An RLS can receive a subscription request from a client that refers to a group of users, and the server will attempt to obtain the state of each of those users and relay that information back to the subscriber. The document structure for resource list services is defined in [24] and allows the definition of user list URIs and service URIs, the latter of which embeds entries of the former. In order to perform its duty, the RLS will often need to create subscriptions known as backend subscriptions to a SIP PS in order to learn the state of a resource for which it is not an authority [21]. The chief advantage that accrues from the use of an RLS is that it eases the burden of communication (with respect to bandwidth utilisation) on the side of the user equipment. Kamailio can operate as a RLS by building the **rls** module into the installation.

B. Media Server

Thus far, we have only discussed services that are accomplished on the signalling plane, but a multimedia testbed will also have functionality on the media plane as well. Services such as an announcement player, an interactive voice response (IVR), a video mailbox server or a multiparty video conference server are also of interest, but often require the presence of a media server in the SIP network. A media server contains physical or virtual endpoints which can be referenced externally by a call agent

	Kamailio SIP AS	*Kamailio PS*	*Kamailio RLS*	*OpenXCAP*	*Asterisk*
Programming language	C	C	C	Python	C
Configuration	Kamailio scriptlet	Kamailio scriptlet	Kamailio scriptlet	OpenXCAP scriptlet	Dialplan
Pre-requisites	gcc, bison, flex, make	gcc, bison, flex, make	gcc, bison, flex, make, libxml2, libxmlrpc-c3, libxmlrpc-c3-dev	libxml2, python, python-twistd, pythonapplication, python-gnutls, python-lxml, python-twistdweb2, pythonzopeinterface, python-mysqldb	libc6, libcurl3, libgcc1, libncurses, libnewt, libssl, zlib1g

Table 1: Technologies used in the heterogeneous testbed

in order to instruct the server to establish media connections with other endpoints using a protocol such as SIP or MGCP (Media Gateway Control Protocol) [5].

Asterisk is an open source IP PBX that was developed by Mark Spencer of Digium [16]. Asterisk is often considered as middleware, neatly positioning itself between underlying telephony protocols (such as SIP, IAX, MGCP, and ISDN) and telephony applications (such as voicemail, IVR and conferencing). At the core, Asterisk is composed of five core modules: a PBX switching core, a dynamic module loader, a scheduler and I/O manager, an application launcher and a codec translator. Asterisk's media support is provided by a set of codec implementations and a codec translator API which allows it to perform media transcoding. Current audio and video codec support in Asterisk consists of G.711u/a, GSM, G.729, speex and lpc10 codecs for audio and H.261, H.263 and H.264 codecs for video. Asterisk can be controlled using SIP and can behave as a call agent in MGCP mode.

III. Disadvantages Of A Heterogeneous Testbed

The SIP application and media servers described in the previous section were integrated, one by one, into the IMS testbed at our institution. To validate the testbed, two IMS clients were used, the UCT IMS client [18] and Mercuro [6]. In the process of installing, configuring and deploying these services, we became aware of several shortcomings in regards to working with services which are built and configured using different techniques. To help justify this assertion, a breakdown of the software used is provided in Table I.

Firstly, prior to installation, the developer must be aware of the dependencies in the form of external libraries that are associated with a given service. Fortunately, with regard to the Kamailio-based services, most of those dependencies are common and shared, except for the XML RPC libraries which are needed by the RLS to communicate with OpenXCAP. However, OpenXCAP requires many new libraries that aren't required by Kamailio. The same is true with Asterisk.

Closely linked to this challenge of dependencies is the use of different programming languages by the developers of these services. Again, we are fortunate with Kamailio in that all the services are written in the C programming language, as also is Asterisk. However, OpenXCAP is written in Python and makes use of the Python event-driven network engine known as the twisted framework, which is known for its complexity. For any competent programmer, these hurdles need not be too wearisome, but it does mean that a company or institution using such a setup must posses a broad skills set. As even more services are added that are based on different development platforms, so the burden to administrate and extend the testbed becomes increasingly difficult and time consuming, particularly as software and libraries need to be upgraded, and APIs change and are extended.

Even when the services are correctly installed, their appropriate behaviour needs to be configured. Table I shows that the same scriptlet syntax can be used across Kamailio services. Unfortunately, those skills are not transferable to OpenXCAP or even to Asterisk. There are subtle similarities between the OpenXCAP configuration and Asterisk's dialplan. Though both use plain text files that organise the configuration into contexts that contain assignment statements, they are essentially very different. OpenXCAP uses static variable assignment statements whereas Asterisk's dialplan defines the sequence of steps to be followed upon receipt of a matching request.

In an effort to explore alternative approaches to deploying SIP ASs, the Java standard JAIN SLEE caught our attention. In the next section, we will introduce this environment that can provide the same types of services to an IMS network while overcoming some of the problems that have been described.

IV. JAIN SLEE: A Specification for Service Creation and Deployment

JAIN™, or Java APIs for Intelligent Networks, is an initiative within the Java™ world to provide programming interfaces to popular communication protocols. JAIN SLEE is a Java™ standard that is a product of the Java Community Process (JCP) which makes extensive use of JAIN™ technologies [12]. The specification takes advantage of the abstraction provided by JAIN™ by defining protocol adapters called Resource Adapters (RAs). These RAs are external to the SLEE itself but interact with other resources such as protocols stacks and databases, and can adapt the interfaces to those required by the SLEE. The SLEE also defines a component model for the creation of software entities know as Service Building Blocks (SBBs). These are atomic, 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. A logical event router in the SLEE forwards events to SBBs that have explicitly "registered" themselves for those events. After processing the event, the SBB can then pass the event to another SBB, based on a service interaction model that is based on either parent-child relationships or a prioritisation model.

As an example, an e-learning system that provides a virtual classroom to multiple participants and features audio, video and group chat facilities, can be developed using a combination of SBBs interacting with each other. One SBB can register for HTTP events which it receives from a website when a user logs on or performs some other action. In a converged SIP/HTTP container, a SIP proxy SBB can help route events that are SIP related, for example IM requests to a SIP instant messaging SBB or audio requests to a SIP Conferencing SBB. Figure 2 shows the architecture of JAIN SLEE.

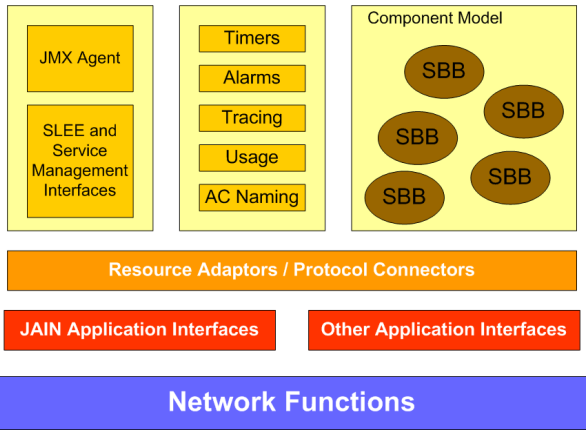

Figure 2: The JAIN SLEE Architecture. Adapted from [11]

V. Mobicents Platform: A JAIN SLEE Implementation

To complement the JAIN SLEE standard, the Mobicents project was created. Mobicents is currently the only JAIN SLEE 1.0 compliant application server for the Java™ platform [17] and was designed to be deployable over JBoss. For this purpose, Mobicents can be downloaded with JBoss pre-bundled. Resource adapter implementations for protocol stacks such as SIP, XMPP, HTTP, Diameter and MGCP are provided, as well as some example applications. The full suite comprises of the following services:

1. Mobicents Media Server - A media server that supports a variety of call control protocols such as SIP and MGCP and supports the G.711a/u, speex, G.729 and GSM codecs.

2. Mobicents Integrated Presence Server - A presence service that is adherent to the relevant IETF and OMA standards and comprises of a SIP PS, an XDMS and an RLS.

3. Mobicents JAIN SLEE Server - An application server that implements the JAIN SLEE 1.0 specification and bundles several RAs and simple demo services for proof of concept.

4. Mobicents SIP Servlets - The world's first certified implementation of the SIP Servlets 1.1 specification [1].

VI. Using Mobicents in the Open IMS Core

Figure 3 shows what the new testbed that utilises Mobicents looks like. To justify a move to Mobicents, it is important to ensure that two major requirements are met. Firstly, every effort must be made to ensure that little or no functionality is lost from the previous one. Secondly, we need to analyse Mobicents according to the criteria used in Table I and highlight the potential benefits that can be harnessed by such a move. With regards to the first requirement, by looking at the features of Mobicents as they are presented in section V, it is quite evident that we lose almost nothing of our current functionality. The SIP AS role can be provided either by a SIP servlet or an SBB using the SIP RA from the JAIN SLEE server. The presence support is provided by the Mobicents SIP PS and the Mobicents XDMS. The added option of being able to either separate the SIP PS from the XDMS or host the integrated service on a single host is also quite useful, affording the developer the freedom to choose between a self-contained or a distributed model for presence services.

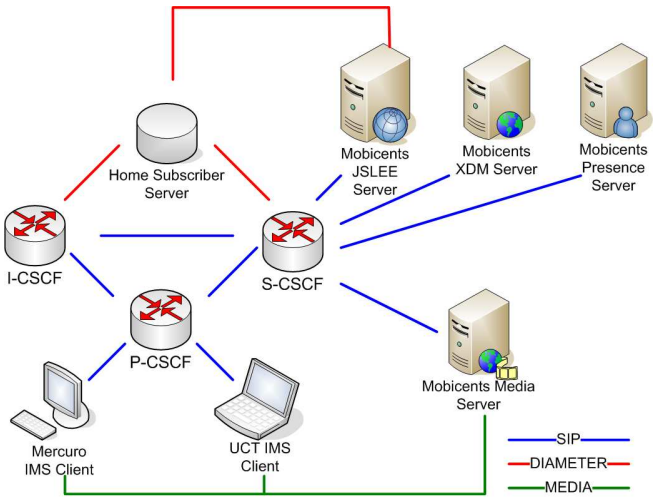

Figure 3: Status of the testbed using the Mobicents suite of services

The Mobicents Media Server also satisfies the requirements of a media server, albeit without support for the video capabilities that Asterisk supports natively, though video is part of the project roadmap. For the second requirement, we note that Mobicents has only one real pre-requisite for installation, which is that a Java™ runtime environment must exist on the host. In addition, to run the server, the appropriate environmental variable must be set that points to the user's JBoss installation directory. Since the entire platform is based on Java alone, it means that it is much easier to maintain and extend the testbed with a group of capable programmers. The unified nature of the platform also exhibits itself in a common configuration syntax. 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. As an example, a follow-me service was developed that would allow users to request that subsequent calls to their extension be re-routed

to an alternative extension. This service would be useful to support user mobility or for privacy reasons. In the configuration presented in Listing 1, the root SBB declared as **CallSBB** has two child nodes, **DtmfDemoSbb** (which processes DTMF events for retrieving the redirect extension) and **FollowMeDemoSbb** (which performs the application logic for a follow-me request). In order to do this, **CallSBB** must register for both SIP and media events as shown in the bottom half portion of the file. This XML-like POM configuration is consistent in Mobicents for configuring both the base servers described in section V and new services created by the developer that rely on those services, as in the follow-me example.

```
# Service registration details
<description>Represents the entire call</description>
<sbb-name>CallSbb</sbb-name>
<sbb-vendor>org.mobicents.mms-demo</sbb-vendor>
<sbb-version>1.0</sbb-version>
<sbb-alias>CallSbb</sbb-alias>
...
<sbb-ref>
<sbb-name>DtmfDemoSbb</sbb-name>
<sbb-vendor>org.mobicents.mms-demo</sbb-vendor>
<sbb-version>1.0</sbb-version>
<sbb-alias>DtmfDemoSbb</sbb-alias>
</sbb-ref>
<sbb-ref>
<sbb-name>FollowMeDemoSbb</sbb-name>
<sbb-vendor>org.mobicents.mms-demo</sbb-vendor>
<sbb-version>1.0</sbb-version>
<sbb-alias>FollowMeSbb</sbb-alias>
</sbb-ref>
...
# Event registration
<event event-direction='Receive' initial-event='True'>
<event-name>CallCreated</event-name>
<event-type-ref>
<event-type-
name>javax.sip.message.Request.INVITE</eventtype
-
name>
<event-type-vendor>javax.sip</event-type-vendor>
<event-type-version>1.1</event-type-version>
</event-type-ref>
<initial-event-select variable>'ActivityContext'/>
</event>
...
<event event-direction='Receive' initial-event='False'>
<event-name>ConnectionOpen</event-name>
<event-type-ref>
<event-type-name>org.mobicents.slee.media.
CONNECTION_OPEN</event-type-name>
<event-type-vendor>org.mobicents.media</event-
typevendor>
<event-type-version>1.0</event-type-version>
</event-type-ref>
</event>
```

Listing 1: POM file for example service

VII. Conclusion

This paper has described an initial investigation into service platforms that can be used to deploy multimedia services over an IMS testbed. Due to the complexity associated with managing a testbed that consists of heterogeneous entities, an alternative design was required that is based on a consistent service description model such as the one provided by JAIN SLEE. We have analysed this model and introduced a possible realisation of it, which is based on Mobicents. We have also shown that Mobicents has the potential to reduce development and deployment time, and may pose fewer challenges in service integration as the testbed is extended.

VIII. Acknowledgements

This work was undertaken in the Distributed Multimedia Centre of Excellence at Rhodes University, with financial support from Telkom, Comverse, Stortech, Tellabs, Amatole Telecom Services, Mars Technologies, Bright Ideas 39, and THRIP.

References

[1] JSR: SIP servlet v1.1. Available Online, 2006. URL: http://jcp.org/en/jsr/detail?id=289.

[2] 3GPP. TS 23.228: IP Multimedia Subsystem (IMS) - Stage 2. Technical Report TS23.228v8.4.0, 3GPP, April 2008.

[3] Ed A Niemi. RFC 3903: Session Initiation Protocol (SIP) Extension for Event State Publication, 2004.

[4] Open Mobile Alliance. OMA Presence SIMPLE v1.1. Available Online. URL: http://www.openmobilealliance.org/Technical/release_program/Presence_simple_v1_1.aspx.

[5] F Andreasen and B Foster. RFC 3435: Media Gateway Control Protocol (MGCP), 2003.

[6] Inexbee. Mercuro - IMS Client. Available Online. URL: http://www.mercuro.net/.

[7] Fraunhofer FOKUS Institute. Open IMS Core.org | The Open IMS Core Project. Available Online. URL: www.Open IMS Core.org/.

[8] Kamailio. History - Kamailio (Genuine OpenSER) SIP Server. Available Online. URL: http://www.kamailio.org/mos/view/History.

[9] Kamailio. Kamailio (OpenSER) Presence Module. Available online. URL: http://www.kamailio.org/dokuwiki/doku.php/presence:presencemodule.

[10] Vignesh Karthik and Shandangi Prateek. Building an IMS Client Test bed with Open Source Tools. In *International Conference on IP Multimedia Subsystem Architecture and Applications*, pages 1–5. IEEE, December 2007.

[11] Swee Lim, Phelim Doherty, David Ferry, and David Page. The JAIN SLEE tutorial. Available Online. URL: http://jainslee.org/downloads.html.

[12] Swee Boon Lim and David Ferry. JAIN SLEE 1.0 specification, final release. Technical report, Sun Microsystems, January 2004.

[13] T Magendanz, K Knuttel, and D Witaszek. Service Delivery Platform Options for Next Generation Networks within the National German 3G Beyond Testbed. In *SATNAC '04: Proceedings of the 7th South African Telecommunications Networks and Applications Conference*, September 2004.

[14] R Mahy. RFC 3842: A Message Summary and Message Waiting Indication Event Package for the Session Initiation Protocol (SIP), 2004.

[15] R Mahy. RFC 3857: A Watcher Information Event Template Package for the Session Initiation Protocol (SIP), 2004.

[16] Jim Van Meggelen, Jared Smith, and Leif Madsen. *Asterisk: The Future of Telephony*. O'Reilly, September 2005.

[17] Mobicents. Mobicents open source community white board. Available Online. URL: http://groups.google.co.za/group/mobicents-public.

[18] University of Cape Town. UCT IMS Client. Available Online. URL: http://uctimsclient.berlios.de/.

[19] AG Projects. OpenXCAP - free XCAP server for SIP SIMPLE. Available Online. URL: http://www.openxcap.org/.

[20] A B Roach. RFC 3265: Session Initiation Protocol (SIP) Specific Event Notification, 2002.

[21] AB Roach, B Campbell, and J Rosenberg. RFC 4662: A Session Initiation Protocol (SIP) Event Notification Extension for Resource Lists, 2006.

[22] J Rosenberg. RFC 3903: A Presence Event Package for the Session Initiation Protocol (SIP), 2004.

[23] J Rosenberg. RFC 4825: The Extensible Markup Language (XML) Configuration Access Protocol, 2007.

[24] J Rosenberg. RFC 4826: Extensible Markup Language (XML) Formats for Representing Resource Lists, 2007.

[25] David Waiting, Richard Good, Richard Spiers, and Neco Ventura. Open source development tools for IMS research. In *TridentCom '08: Proceedings of the 4th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities*, pages 1–10, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

Mr. Mosiuoa Tsietsi holds an MSc degree in Computer Science from Rhodes University and is currently reading towards his doctorate in the same institution.

Prof. Alfredo Terzoli is Director of the Center of Excellence in the department of Computer Science at Rhodes University.

Prof. George Wells is Head of department of Computer Science at Rhodes University.