Developing a Cross Platform IMS Client using the JAIN SIP Applet Phone

Walter T. Muswera¹, Alfredo Terzoli²
Department of Computer Science
Rhodes University, P.O. Box 94, Grahamstown 6140
Tel: +27 46 6038642, Fax: +27 46 6361915
email: {g09m3278@campus.ru.ac.za; A.Terzoli@ru.ac.za}

Abstract—The interest in the adoption of the IP Multimedia Subsystem (IMS) by industry and research institutes has resulted in the establishment of IMS testbeds for developing and deploying IMS services. The client applications used for testing and consuming services within these testbeds are important for the overall success of the developed services. Currently, there is no single free client that provides researchers with all the required functionality needed to test their applications. For example, several open and closed source IMS clients are used within the Rhodes University Convergence Research Group (RUCRG) in order to utilise specific features to test developed applications.

In this paper we present an overview of three popular IMS clients currently in use within the RUCRG. We then discuss the features and the design architecture of the JSAP (JAIN SIP Applet Phone) which is used as the foundation for developing a new IMS compliant client. An analysis is then given regarding the functionality that JSAP lacks, features that have been added and the results of the various compatibility tests that have been performed between the new IMS client, SIP (Session Initiation Protocol) application servers and other freely available IMS clients. Finally, we discuss the features that are still to be added to the new client.

Index Terms—SIP/IMS client, IMS Compliant.

I. INTRODUCTION

The Rhodes University Convergence Research Group (RUCRG) is mainly concerned with current trends in the move towards converged service platforms for Next Generation Networks (NGN) and the Internet. Research in these areas covers complex service orchestration, policy frameworks for service development, development of toolkits for services such as IP television (IPTV), Location Based Services (LBS) and Video on Demand (VoD) using open standards. These applications are built on platforms such as the Mobicents Application Server, Opensource IMS Core, Kamailio and Asterisk. As the interest in IP Multimedia Subsystem (IMS) grows, applications being developed within the RUCRG use IMS as a deployment platform. This framework enables the deployment/or delivery of integrated services and uses open standards [1].

IMS is a service delivery framework specified by the 3rd Generation Partnership Project (3GPP) and other standards development organisations. This framework enables the development and/or delivery of IP based multimedia applications, such as Presence, Video Sharing and Instant Messaging (IM) [2]. IMS is the catalyst for convergence and enabler for service driven development and delivery of new communication applications.

IMS uses several Internet technologies and Internet protocols standardised by the Internet Engineering Task Force (IETF) such as Session Initiation Protocol (SIP), Session Description Protocol (SDP), Realtime transport Protocol (RTP) and Real Time streaming Protocol (RTSP). Most if not all applications that are developed within the RUCRG depend on establishment and termination of sessions between servers and clients. These sessions are established using SIP. Fortunately, SIP is the chosen session control protocol for IMS[3]. SIP is an application layer protocol for initiating, modifying, or terminating communication and collaborative sessions over Internet Protocol (IP) networks [1]. It allows endpoints on the Internet to discover one another and to negotiate variables for the session they would like to share. Essentially, SIP finds the best way for users to communicate given their preferences and the capabilities of the devices they have at their disposal.

Despite the fact that IMS was proposed several years ago and that there has been widespread interest in the technology worldwide there are relatively few IMS clients available that have all the required features to test the applications being developed for the RUCRG testbed. Although some core IMS developers have released SIP-to-IMS gateways to allow vanilla SIP clients to interact with the IMS, the true power of the IMS can never be realised with an ordinary SIP client. This means that applications developed for IMS rely on the availability of a fully IMS compliant client for proper testing and evaluation.

Unfortunately, the RUCRG has not been spared from the lack of a one stop IMS client and this has prompted the development of a pure IMS client within the group for their own research. The JAIN SIP Applet Phone (JSAP) is used as the foundation of the new client. Ideally the client should not be limited to any particular operating system platform and should be backward compatible with legacy SIP servers and applications.

Since IMS is built inherently to support rapid service creation [1], it is necessary to develop the IMS client in manner that is easily extensible. Furthermore, the standardisation of IMS is an ongoing process thus IMS client development has to keep up in order to remain compliant with the IMS standards.

This paper describes the development of a Java based, IMS compliant client using the JSAP as the foundation of the new
client. A review of some existing SIP/IMS clients currently in use in the RUCRG is provided. A discussion is also provided of the current status and architecture of the JSAP, features that have been added, results of the various compatibility tests that have been performed between the new IMS client, SIP application servers and other freely available IMS clients and the possible extensions that still need to be implemented.

II. IMS CLIENTS

IMS client applications which are currently available to the RUCRG, lack features, that is, they support a subset of functions that are required to test the applications being developed [4, 5]. Furthermore, most SIP/IMS clients can only be used on specific platforms [6, 7, 8] and support a limited range of video and audio codecs [3, 9]. As it will be shown, clients exist that possess some of the required features needed for testing applications but there is no single client that allows researchers to perform testing without having to switch between clients or adjust their systems. An overview is given below of the feature sets of three freely available IMS clients that are currently in use in the RUCRG.

A. IMS Communicator

IMS Communicator is an IMS softphone based on the SIP Communicator Java project [10]. It is implemented on top of the JAIN-SIP stack [11] and the Java Media Framework (JMF) API [12]. The use of JMF as a media API presents a variety of challenges. Firstly, there have been significant improvements in video coding technologies over the last few years but JMF supports a limited set of these codecs. Secondly, Sun Microsystems ceased to support JMF in 2003. Lastly, JMF installation and configuration is complex especially for ordinary users. The code structure of IMS Communicator makes it difficult to debug, that is, classes are overloaded with various functions. For example, SIP messages are received and processed by the same class. Registration with the Opensource IMS Core fails (There are existing bugs that have not been fixed in a long time).

B. UCT IMS Client

The UCT IMS client is a free open source implementation of a 3GPP IMS client developed in ANSI C [8]. It supports a variety of IMS applications such as IM, Presence, VoD/IPTV, and the XCAP protocol among others. It was designed to be used on the Linux platform making it unavailable to other major platforms such as Windows and Mac OS. Notwithstanding its platform dependency the UCT IMS client is no longer under active development and the last release was published in Feb 2009. It was tested under a version of Ubuntu that is no longer supported (Ubuntu 8.10, which is not a Long Term Support (LTS) release) and mailing list discussions have also almost ceased.

C. Mercuro

Mercuro IMS client is closed source, proprietary and comes precompiled thus cannot be extended. It comes in various versions one of which is free and supports a limited set of functions [13]. Similar to the UCT IMS client, Mercuro is not cross platform. It is built only for the Windows environment. The Mercuro IMS client project has been stopped and the development team has been dissolved [7]. Given that the development and standardization of IMS and its associated services is an ongoing process this presents a big challenge. Client development needs to keep up to date with changes in the IMS to remain compatible with IMS standards.

D. Feature Comparison of Major IMS Clients

Table I shows a comparative assessment of the important features for IMS compliance that each of the discussed clients possess. Also included are the platforms that the clients are compatible with as well as the licencing of the clients.

III. JSAP: ARCHITECTURE AND LIMITATIONS

A. Architecture

JSAP is an open source project which possesses some of the basic features which are required in an SIP/IMS compliant client such as voice and text Instant Messaging (IM). It uses JMF as its media API, JAIN SIP for SIP signaling as well as JAIN SDP (Session Description Protocol) for session description. It was chosen as the foundation of the new client for the following reasons:

1) It uses JAIN SIP for SIP signaling which is a low level Java API thus allowing access to the full power of the SIP protocol.
2) It supports ordinary SIP signaling and has a mature code base.
3) JSAP project leadership is at Rhodes under one of the key developers of the initial project.
4) The client is part of the current RUCRG testbed.

Before any implementation of IMS support and replacement of the media library could be done there was need to perform an extensive assessment of the JSAP particularly because there was no documentation and it only supported basic SIP functionality. This assessment involved carrying out systematic experiments such as tracing of messages sent by JSAP using network analysis tools to check their sequencing and well formedness, reverse engineering to find out the relationship among the various classes, and discussing with some of the original developers of JSAP.

Having gone through the aforementioned processes a structure of JSAP was drawn up. Figure 1 gives an overview of the architecture of the JSAP that resulted from the experiments that were done. The dotted lines show various components of JSAP that needed to be added or modified to make it IMS compliant.

A brief overview of some of the components that were identified in the JSAP follows:

1) Invite/Session - provides high level management to call control. The call setup procedure for an IMS call is more complex as the SIP precondition and reliable provisional response mechanisms are used.
<table>
<thead>
<tr>
<th>Feature Summary of UCT IMS Client, IMS Communicator and Mercuro [8]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UCT IMS client</strong></td>
</tr>
<tr>
<td><strong>Registration</strong></td>
</tr>
<tr>
<td><strong>IMS Signalling</strong></td>
</tr>
<tr>
<td><strong>Media Support</strong></td>
</tr>
<tr>
<td><strong>Presence Support</strong></td>
</tr>
<tr>
<td><strong>Instant Messaging</strong></td>
</tr>
<tr>
<td><strong>XCAP Support</strong></td>
</tr>
<tr>
<td><strong>Platform</strong></td>
</tr>
<tr>
<td><strong>License</strong></td>
</tr>
</tbody>
</table>

---

### Figure 1. JSAP Architecture

2) Presence - provides functionality to manage presence information of the client and associated contacts. Client/Server mode needs to be added.

3) Registration - hides the complexity of the SIP registration process including dealing with multiple types of user identities. The registration procedure to the IMS is more complex as the Authentication and Key Agreement (AKA) algorithm is used together with md5.

4) Instant Messaging - enables sending and receiving of IMs to and from buddies. JSAP currently supports pager mode IM in which messages are sent in the body of SIP MESSAGE requests and no sessions are established.

5) Call manager - provides the mechanisms for controlling calls. It acts as an interface for controlling SIP related communications. This includes SIP based calls and registration. It interfaces with Presence and IM module to provide proper signaling for IM and Presence.

6) SIP stack - provides a low level API that provides full control over SIP communication between the client and IMS.

7) RTP/RTCP Stack - provides low level API to provide full control over real time data transport between the client and the application server or another client [2].

### B. JSAP Limitations

The assessment of the JSAP also brought to light a variety of limitations in the client: JSAP lacks support for IMS functionality, that is, it is an ordinary SIP client that cannot be used in an IMS setting. Presence in JSAP is implemented in a peer to peer manner but ideally should also support Client/Server and Video implementation in the JSAP is not fully functional and requires attention (JMF fails to initialize video capture devices in Linux but works under Windows. JMF also lacks support for some of the new high quality well compressed codecs as discussed in II-A.)

### IV. DESIGN AND IMPLEMENTATION

As previously indicated in III-B, JSAP only supports ordinary SIP signalling, SIP session setup and uses JMF as its media API. This section will outline how JSAP was modified to support IMS signalling and IMS session setup. A discussion is also given of how the JSAP has been modified to use gStreamer media library in place of JMF to receive and transmit video data. The main IMS User Equipment (UE) procedures were implemented and some of the IMS functionalities implemented include IMS Registration (Authorization, Security Agreement) and IMS session setup (PRACK and Precondition Mechanism). The client is now capable of interacting with the Opensource IMS Core for the setup, control and termination of IMS services. The client still supports ordinary SIP functions and can work with non-IMS sip proxy servers. Furthermore, the client can now receive and stream audio and video using gStreamer media library.

The implementation of the new client was achieved by the use of free open source libraries: the JAIN SIP stack was replaced with a new one that contains support for the generation and parsing of IMS headers and the gStreamer framework replaced JMF for media coding, decoding and transport. This is particularly useful for research projects that aim to produce innovative services that require modifications to the client software. The JSAP client also lacked standards compliant IMS signalling and session setup. A full discussion of how extensions to the JSAP were made are as follows:
A. Choice of Technology

The need to produce a client that is portable across different platforms (operating systems) resulted in Java being the language of choice. Java is both a compiled and interpreted language. Any computer that has the Java Virtual machine (JVM) installed can interpret compiled Java byte code. Furthermore, the Java Community Process (JCP) through the Java APIs for Integrated Networks (JAIN) initiative, define Application Programming Interfaces (API) for using Java technologies to provide next generation telecommunications services [2].

The JAIN SIP stack was replaced with a later version that supports generation and parsing of IMS headers

B. Implementation: Enhancements to the JSAP

In order to make the necessary IMS specific changes a full analysis of JSAP classes was carried out and the classes which needed to be modified, removed or replaced were identified. This was done in light of the fact that IMS functionality is built on top of ordinary SIP functionality. Since the JSAP already supported ordinary SIP registration and SIP session setup, IMS specific headers needed to be added on top of SIP headers to make the client IMS compatible. In summary the process involved:

- Studying the structure of the JSAP and identifying the classes which needed to be modified to add IMS support
- Adding helper classes for populating IMS specific parameters
- Removing and or replacing some existing classes with optimised ones that allow the support of IMS
- Adding XML support to allow the populating of IMS/SIP attributes and to allow persistence

1) 3GPP IMS SIP signalling : For communication between the P-CSCF and the client, JSAP was modified to use IMS signalling as defined in the 3GPP specifications TS 24.229, TS 23.228 and others. This includes the establishment and teardown of an IMS session with the P-CSCF, as well as the negotiation of media delivery between the client and another IMS client. In addition to the IMS signalling, RTP and RTCP signalling were added to the client using the gStreamer library [14].

2) Decoding and Display of Multimedia Streams : The client was extended to make use of the gStreamer library for receiving, decoding and displaying multimedia content. The gStreamer library was chosen because it provides modules to capture audio and video, encode the voice and video and pack- etise the resultant bit stream into RTP packets for transmission across the network [8]. Several gStreamer elements can be linked together to form a pipeline which can easily be modified to accommodate new elements[14]. GStreamer also supports a range of coding formats [14]. The gStreamer library provides a modular framework for handling many types of multimedia. This extensible and modular architecture provides a great deal of flexibility to the program developer.

3) Video and Audio Support: As previously pointed out the video implementation in the JSAP was not fully functional. This prompted a complete replacement of the media library used in JSAP in light of the problems discussed in III-B about the JMF. The decision to completely replace the media stack was also motivated by the fact that synchronisation of audio and video sessions managed by different libraries would be a problem.

C. Testing

To evaluate basic interoperability, test cases were defined for IMS session initiation and media support. These scenarios were based on end-to-end systems testing, that is, higher level functionality, rather than on specific protocol layers or requirements.

1) Registration: To evaluate registration the client was registered with Kamailio (acting as the SIP proxy server) and the Opensource IMS Core (acting as the core IMS registration server).

- SIP Registration
  Since the JSAP client already supports ordinary SIP registration the aim of this test was to make sure the client could still successfully register with any ordinary SIP proxy server after modifications had been made to the code.

- IMS Registration
  This test was used to evaluate whether the client could register successfully with the Opensource IMS Core. The IMS registration flow is illustrated in Figure 2. The client sends an initial REGISTER request to the P-CSCF address that is manually configured into the client. This requires that the client perform a DNS look-up on the P-CSCF address before sending the request. The P-CSCF then forwards the request to the I-CSCF that, with the help of the HSS, relays the request to a suitable S-CSCF. In order to authenticate the user the S-CSCF challenges the client with a 401 Unauthorised response, which contains a nonce value. The client inputs the nonce value and the user credentials into the AKA algorithm and generates

![Figure 2. IMS Registration Message Flows](image-url)
a reply that is inserted into a second Register request. Upon receiving the request with a correct authorisation header the S-CSCF registers the user’s Public User Identity (IMPU) and associates it with the client’s IP address.

The S-CSCF replies with a 200 OK message in order to inform the client that it is successfully registered on the network. This message also serves to inform the client of any other IMPUs that are available to be registered by the user and the Service-Route informing the client which route should be followed by all subsequent requests.

2) Session Setup: To evaluate session initiation, the client initiated SIP/IMS sessions with other SIP/IMS clients. The client also responded to sessions originating from other SIP/IMS clients.

- SIP Session Setup
  Similar to the registration process the client already supports ordinary SIP session setup. The aim of this test was to make sure the client could still successfully establish a call session with an ordinary SIP client after changes had been made to allow IMS call setup.

- IMS Session Setup
  This test was performed to verify that the client can successfully setup an IMS call session with another client. The IMS call setup flow is illustrated in Figure 3.

Figure 3. IMS Call Setup Procedure

IMS call setup starts with an initial INVITE request sent from the client to the P-CSCF containing a media offer. The request is relayed to the target client through the originating IMS core network and possibly through a terminating network if the two clients are located on different networks. The target client replies with a 183 Session Progress response with a media response. In SIP, provisional responses are usually not sent reliably [15], but since the provisional response is essential to the IMS call setup procedure it requires acknowledgement in the form of a PRACK message. Once the originating network has provisioned suitable resources for the call the originating client sends an PRACK message informing the target client that it is ready to start a call. If the terminating network has also provisioned sufficient resources for the call the target client replies with a 180 Ringing response and informs the caller that there is an incoming call. Again this provisional response elicits a PRACK from the originating client. If the callee accepts the call then it sends a 200 OK message followed by the originating client sending an ACK.

3) Video and Audio Support: To evaluate media support, audio and video sessions were established between the new client and another client implementation. The audio and video streams were analysed by a human user. The purpose of the test was to verify whether the client could establish a multimedia (audio and video) session with the other client. This test was also carried out with the client being the one responding to a session established remotely. The quality of the video and audio were not assessed.

The tests conducted to verify the ability of the client to perform the functions described in IV-C were successful. This means that the new client can now register with the Opensource IMS core as well as ordinary SIP proxy servers. The client is also capable of establishing an IMS session with another IMS client through the Opensource IMS Core as well as a SIP session with another SIP client. Furthermore the client can now establish a multimedia session with another IMS/SIP client, that is, it can transmit and receive video and audio using the gStreamer library.

V. Future Work

Notwithstanding the important modifications made to the client to support the core IMS functionality some work still needs to be done to enhance the client.

- Integration of SIP Extensions
  – Session update: UPDATE needs to be added to allow the client to provide/interpret updated session information before/after a final response to the initial INVITE request is generated. This will be used within a dialog to update session parameters (such as the set of media streams and their codecs) without impacting the dialog state itself [16].
  – Session Transfer: REFER SIP method also needs to be integrated into the client in order to provide session transfer functionality.

- Configuration of video and audio preferences: Options for frame-rate (video) and encoding format should be added as configurable preferences in the client. This will allow users to customise their video and audio experience.

VI. Conclusions

In this paper three IMS client applications that are currently being used in the RUCRG have been discussed and analysed and it has been established that they lack features that are required to test IMS applications being developed. This was
followed by a discussion on how the JSAP (which is the foundation of the new client) was analysed to come up with an overview of its structure and limitations. The JSAP client was extended and re-organised as reported in IV

REFERENCES


Walter Muswera received his honours degree in 2009 from Rhodes University and is presently studying towards his Master of Science Degree in Computer Science at the same institution. His research interests include converged telecommunication networks and mobility.

Alfredo Terzoli is Professor of Computer Science at Rhodes University, where he heads the Telkom Centre of Excellence in Distributed Multimedia. He is also Research Director of the Telkom Centre of Excellence in ICT for Development at the University of Fort Hare. His main areas of academic interest are converged telecommunication networks and ICT for development.

The work reported in this paper was undertaken in the Telkom Centre of Excellence in Distributed Multimedia at Rhodes University, with financial support from Telkom, Converse, Tellabs, Stortech, Bright Ideas 39 and THRIP.