A Converged IMS Client for the IP Multimedia Subsystem

Richard Spiers and Neco Ventura
University of Cape Town, Rondebosch, South Africa
021 650 5296
Email: {rspiers,neco}@crg.ee.uct.ac.za

***Abstract*—The IP Multimedia Subsystem (IMS) has been developed to provide a single platform that offers many different converged services, over multiple access technologies, with several different end user terminals. It aims to unify the current disjoint point solutions offered by network operators, with the goal of reducing CAP-EX (Capital Expenditure), OP-EX (Operating Expenditure) as well as reducing the time taken to develop a new service. These benefits are realised by the move to a single, IP-based packet switched network. However, the IMS framework has not been adopted as widely as first envisaged, due to a lack of a "killer" application i.e. a service that will see wide use and will drive IMS adoption. There is a need to improve the service creation environment and encourage more developers to begin looking at IMS services if the IMS is to reach its full potential.**

This paper will discusses the design and implementation of a new IMS client for converged services. It explains the motivation behind the work, the design decisions made, the client architecture as well as the surrounding framework necessary to implement the proposed functionality. More specifically, it describes an implementation of an IMS client that supports instant messaging, status mechanisms for handling presence and voice and video calling, with the aim of providing a fully featured example client that can be used as a starting point for further research and to develop new services.

***Index Terms*—IMS, open source, presence, SIP, software design, XCAP, XDMS**

I. Introduction

Over the past couple of years it has become apparent that the next generation of telecommunication services will be provided by a single IP-based packet switched network, over a variety of access technologies and terminal devices. The IP Multimedia Subsystem (IMS) is a framework standardised by the 3GPP, which in theory, allows the rapid creation of prototypes of new services, by providing functionality such as Authentication and Quality of Service (QoS) that can be re-used by each service. This does increase the speed of development of new services, but this development still requires a high degree of technical knowledge with knowledge of multiple protocols (e.g. SIP, RTP, Diameter, etc) necessary as well as an understanding of the various elements and interfaces of the IMS.

Currently, there is no "killer" service - i.e. one that will attract wide spread usage and rapidly drive IMS adoption. The Evolved Packet Core (EPC) is gaining traction, with the IMS being placed as only one of several options that will run over the EPC. The EPC offers network operators a way of running alternative services such as currently existing web services, and as such is an attractive option when compared to the cost and resources needed to deploy an IMS solution. However, this would mean that the network operators would lose the tight integration and fine grained control of their services that the IMS provides as well as the rich Quality of Experience (QoE) that it has been designed to offer.

Thus there is a need for more experimentation and development in the IMS service area in order to provide the network operators with examples of services that will justify the heavier investment required to deploy an IMS based framework. This has lead to an investigation into existing IMS clients and services with the aim of reducing the number of problems new entries to the field of IMS service creation face. This will hopefully lead to the creation of new and innovative services, of which one might be significant enough to drive adoption of the IMS platform. The client proposed in this paper is not such a service, but it is hoped that it will aid the creation of such a service.

The rest of this paper is structured as follows. Section II discusses the related work in this area and describes the currently available IMS clients. This is followed by section III which contains a discussion of the identified requirements for an IMS client that will encourage service development, and section IV contains information on the various technologies deployed in the development of our proposed client. Section V contains a description of the proposed client architecture, followed by Section VI describing the components of the testbed that was used to develop, test and evaluate our client. This is followed by Section VII which concludes the paper and recommends areas for further work.

II. Related Work

An IMS client can be considered a key element in any IMS related services. The relevant features of each client will be highlighted in the following sections. A more detailed comparison between IMS clients can be found in [1] and [2]. While the proposed client has one of the first open source implementations of a Network Stored Address Book (NSAB), the NSAB is not the main focus of this paper and as such

This work was supported in part by Telkom SA, Nokia Siemens, the National Research Foundation and the Department of Trade and Industry

the discussed work will focus on existing IMS clients. Further details of a NSAB can be found in [3].

A. Mobile client for NGN

The first client discussed here was developed by a collaboration between the French Institut National Des Telecommunications and the South African Tshwane University of Technology [4], [5]. It was specifically targeted towards mobile devices and the low memory and processing capabilities of such devices. As such it was developed for the Java 2 Micro-Edition (J2ME) platform, with a focus on the Connected Limited Device Configuration (CLDC). This client is severely limited in functionality, and only supports registration and instant messaging. Neither it, nor its source code are available online.

B. Mobicome IMS client framework

The work presented in [6] and [7] takes a different approach. Instead of building a client using the traditional layering approach, they propose a framework with a dumb terminal with the intelligence hosted on a server in the telecommunication operators network. This has the advantage of allowing relatively simple clients to look as if they have much more advanced capabilities, but latency and bandwidth use are concerns as now much more traffic flows between the client and the network. They propose using a browser plugin which interacts with an IMS GUI server to control the client's functionality. This also leads to complexity in integrating any new service as it now needs specific interfaces to interact with this framework.

C. UCT IMS client

The UCT IMS client is a free open source implementation of a 3GPP compliant IMS client [2], [1]. It is currently used worldwide by multiple research facilities as the source code is freely available online, together with tutorials on how to setup an IMS testbed utilising this client. This allows researchers to quickly experiment with service creation cost free. It is capable of registration, voice and video calling, presence support, instant messaging using either pager mode or session-based mode, streaming video and Video on Demand (VoD) as well as having built-in XCAP support. This makes it the most fully featured open source client currently available. While the proposed client described in this paper has been written from a clean start in a new language, several lessons learnt during the development of the UCT IMS client have been carried over.

D. IMS communicator

Another open source client is the IMS communicator. It is written in Java, and was originally developed by PT Inovacao as part of its Service Handling on IP Networks project. It is now a community driven open source project, but the last release was in July 2008, with the last change to its documentation made in April 2007. It has some support for presence, video calling and instant messaging, and can handle a list of contact entries. It seems that the project is no longer under active development, with open bugs regarding its operation with the latest version of the OpenIMS core remaining unresolved for almost two years.

E. Mecuro

The Mercuro IMS client is one of the most fully featured IMS clients currently available. It is available in several different editions, with each successive edition offering more functionality. It has a address book with built in presence information about each contact, as well as instant messaging and voice/video calling. There is a free limited edition of Mercuro available, which is useful for testing standard compliant existing services. It is also the only client discussed here that has enhanced contact management built in. However the lack of freely available source code means that any developers are limited to using its existing interfaces and cannot change the client at all.

F. Discussion

Out of all the clients discussed, only the IMS communicator and the UCT IMS client provide their source code freely available online. Both of these clients lack a enhanced form of contact management. A form of contact management is a default paradigm when using any existing form of communication, whether it is instant messaging or sending an email, and our proposed client includes a NSAB to handle this functionality.

III. Requirements

We have identified several requirements for a new IMS client that will be beneficial to the IMS service creation environment, furthering research in this area. Besides the requirements mentioned below, it should support the basic IMS services such as registration, video calling and presence. These requirements are obvious, and will not be discussed further.

A. Open Source

For developers to create new and innovative services, they need to have full control over the client code to implement the handling of this new service, adding it to the existing client functionality. This need can be mitigated to some extent by utilising existing standards and building a robust API into the client, but this limits the flexibility and innovative capacity of the developers. The ability to modify the client source code has proved to be invaluable in our research, and as a result we propose that to drive innovative service creation, a successful IMS client must be open source until enough services have been created. Once there are plentiful services in place, a closed source IMS client with a standardised API would be acceptable.

B. Low Entry Barrier

The easier it is to program or modify an IMS client the more likely it is that developers will become involved in creating new services. The difficulty of modifying an existing IMS client is not only related to the language it is created in, but also to the structure of the code and how well separated the existing functionality is.

C. Encourage new and innovative services

The existing client functionality should be very flexible, with the ability to make changes without affecting existing functionality, as well as allowing existing functionality to be used in new and innovative ways. It should be easy to extend with little connection between the different modules.

D. Wide target audience

The development environment should be chosen to ensure that the largest segment of developers is familiar enough with the tools and languages to pick up IMS service development without needing to retrain or spend lengthy time learning new tool chains.

E. Debug information

As the client proposed is mainly a research tool, it should provide additional information that is normally hidden from the end user. As an example, this information could contain the contents of the SIP messages, which is of minimal interest to the end user, but highly valuable to researchers.

IV. Utilised Technologies

There are several protocols and enablers used in our prototype client. The following section details the more important technologies used.

A. Session Initiation Protocol (SIP)

A version of SIP with several standardised extensions is used as the control signalling protocol for the IMS. It provides the necessary control signalling that allows end users to register with the network, negotiate any form of session, as well as providing the necessary enhancements to request the appropriate form of QoS. It is also the recommended default way of communicating with various application servers, and IMS SIP support is the most basic requirement for any IMS client.

B. Presence

The concept of "presence" or "status" is one of the core building blocks for new IMS services. This service allows the end user to express his availability or willingness to communicate, together with details of their communication capabilities. This can be used to streamline the SIP session negotiate procedure or prevent incompatible sessions from being established before they reach the negotiation stage. The current presence framework used in the IMS [8] is based on several RFCs, most notably [9], [10], [11], [12], [13].

There are two main types of presence clients. The first form of client is known as a Presence Entity (PE) or Presentity for short. This client expresses presence information which supplies information regarding the client. The second form of client is known as a Watcher. This is an entity that requests information about the end user. It is possible for the watcher to subscribe to the status of the end user through the mechanism provided by the SIP event framework. The watcher sends a SIP SUBSCRIBE message to a Presentity Agent (PA) which builds a list of interested parties in the end user's status. When this status changes, the PA sends a SIP NOTIFY request to the watcher, allowing them to keep an up-to-date version of the end user's presence. Both the SUBSCRIBE and NOTIFY message contain a header known as the EVENT header. This field identifies the actual event that the subscribe or notify messages are related to, and allows different treatment of the messages depending on which event they are representing. For presence, the value of "presence" is contain in this header.

The PUBLISH / NOTIFY requests normally contain an XML document which has been formatted with special rules. The base XML document is known as the Presence Information Data Format (PIDF) [14], but there are several extensions that extend and enhance the information represented in these documents. For more details, see [15], [16] and [17].

C. XML Configuration Access Protocol (XCAP)

The XML Configuration Access Protocol has been designed to allow the upload, download and modification of XML documents. It is not a new protocol as such, but rather an enhancement to the Hyper Text Transfer Protocol (HTTP). It provides mechanisms detailing the access and modifications of XML documents via custom requests. It sits above HTTP in the protocol stack, mapping XML documents and their attributes to various custom HTTP Uniform Resource Identifier (URI). This gives the end user the ability to modify XML documents at various levels, from completely replacing the document to changing a single attribute by utilising regular HTTP methods with some custom headers and specific URIs. A specific example of an XCAP message is given further on. Two applications that use XCAP and are relevant to this paper are the Presence Enabler and the Network Stored Address Book (NSAB). For example, the presence service uses XCAP to control the amount of information exposed to the individual Watchers. It can also be used to perform "hard state manipulation" of presence information by controlling various XML documents. This is in contrast to the normal "soft state manipulation" which is provided by the PUBLISH mechanism discussed earlier.

D. XML Document Management (XDM)

XML Document Management refers to an architecture which is the logical extension of the XCAP protocol. It is a series of elements including document servers, clients and various proxies that provide document management services using XCAP. It contains several enhancements, such as an Aggregation Proxy which authenticates XCAP requests as well as combining XCAP requests. It allows the end user to issue one XCAP request, which will result in the Aggregation Proxy issuing several of its own XCAP requests. The responses to these requests will be combined and returned to the end user. Another improvement is the provisioning of a search capability to allow the end user to search for specific information in the XML documents. Note that the XDM specification is a functional one, and in practice several of the elements would be combined into one logical element. XDM also

provides a subscription/notification mechanism that provides a mechanism to learn of any changes to any XML documents stored within the system.

V. Proposed IMS Client Design

Several design decisions were made in order to satisfy the requirements listed previously. Firstly, our client is licenced under a permissive open source licence. Not only does this allow researchers the ability to modify the client code, it also enables the use of libraries that are only allowed to be used in open source projects.

Secondly, a high level language was chosen as it is more accessible than a low-level language. C#, together with its wide variety of "built-in" libraries allows for more rapid development than a low-level language like C.

Thirdly, the functionality was split into several different modules, with very little interlinking code. An event framework is used to pass messages from each module to the core of the program, allowing total separation between different sets of functionality. This allows new modules to be created without influencing existing modules. Generally speaking, the modules handle the particular implementation including the necessary SIP traffic for their functionality and generate events that other modules are then free to act upon. For example, the instant messaging window will be updated when the "Message_Received_Event" is generated by the Instant Messaging Module.

The requirement for a wide developer audience, together with the lack of a fully featured, open source IMS client for the Windows platform has lead to the client being developed for Windows. However, as the language the client has been programmed in is C# and the multimedia libraries used are also available for Linux, it could be ported to a Linux environment as well. C# is a portable language and could run under the Linux C# environment, Mono. However, the main goal of this work is to create a fully featured Windows based client and the Linux conversion has been left for further study.

Lastly, several techniques have been used to help further research using this client. The client provides a debug window which can be shown or hidden to display messages about the various aspects of client operation. Currently, it displays all the sent and received SIP messages, the sent XCAP messages and the server responses as well as the messages from the multimedia framework (GStreamer). This system has been designed in such a way as to be easily extended for new modules.

Dynamic visual interface techniques have been used in the creation and storing of the user's preferences, allowing new developers to extend the available configuration options without having to worry about presenting these options to the end user. The developer can add the option name and type to the settings module and it will handle the creation of the respective GUI components, as well as saving these new options to the configuration file.

The functionality of the client has been implemented in several modules, and the following sections describe the main modules.

A. Preferences Module

All user configurable options are stored in a XML document. This document is tied closely to the "Preferences" class, which is a code based representation of this document with additional functionality. When the end user wishes to view or change an option, the resulting GUI components are dynamically created depending on the options specified in this module. There are defined sections in this module that result in tabs so each set of options can be displayed together. Currently, the tabs in use are IMS, Presence, XDMS, VideoCall and AudioCall. Each of these tabs is used to show the options associated with that group. For example, the XDMS user name and password can be set on the XDMS tab. In addition, each option can have a specified type, such as "check-box" or "hidden" which determines which control is used to display that option. By default a text-box is generated which will save a string representation of the specified option. This was introduced because although the primitive types such as integers or booleans could be determined at run time, more advanced controls such as drop-boxes with multiple options need to be handled as well.

B. Presence Module

This module is used to handle the presence framework. It exposes functions to handle the subscription and publication of status information, and generates events when a change in presence has been detected - most notably on the receipt of a NOTIFY message. It takes advantage of two helper classes, "Contact" and "Status". These represent all the information about a particular contact as well as its current status, and are gathered together to represent an address book. When a change in presence is detected, it raises an event that is used to modify the current status. This in turn is detected by the status object, which generates its own events so that the GUI can be updated appropriately.

C. Multimedia Handler

All forms of multimedia transmission are handled by this module. It exposes the necessary function to send and receive audio and / or video, which is used for the voice / video calls. It also generates events based on feedback from the multimedia library (GStreamer). For example, an event is generated when the end-of-stream (EOS) is reached. These events are currently logged by the debug framework. The multimedia handler can be trivially extended to handle any form of multimedia that the GStreamer library can handle, such as the playback of mp3s or streaming video.

D. Instant Messaging Module

A basic standard compliant implementation of [18] is provided through this Module. It exposes the necessary function calls to send instant messages as well as to process the received replies. In both these cases events are generated which can be handled by other modules. There is also an implementation

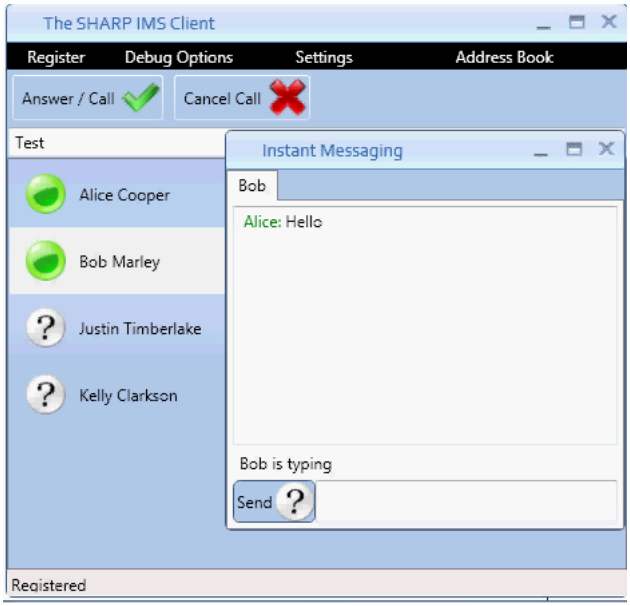

Figure 1. Instant Messaging Support with typing notifications

of [19], which generates events when the typing notification is received. Currently these events are only used to display the messages and notifications to the end user. However, it is trivial to change this to take other actions, providing a way to control the client via instant messages.

E. Call Module

This is where the handling of SIP traffic for establishing calls (voice or voice and video sessions) is handled. These functions are generally called by another module or by event handlers for specific events (such as incoming call). The necessary SIP messages (e.g. INVITE, CANCEL, 200 OK etc) are generated here as well as a mechanism to control and update the call status (active, terminating etc.). The actual transmission of the media is dealt with by functionality exposed by the multimedia handler.

F. XDMS Module

This module uses XCAP to store or retrieve XML documents. It generates an event whenever it builds a request or receives a response. At the moment these events are tied into the debug log framework, but they could be handled by new modules if so desired. This functionality is used to store and retrieve the NSAB.

VI. Testbed Environment

The prototype client was developed and tested in a fully open source environment. This allows researchers free access to a complete IMS service development environment that is fully customisable, from the client to the services provided. A succinct description of the elements used is provided below.

A. OpenIMS Core

The core IMS components are the Home Subscriber Server (HSS), the Proxy Call Session Control Function (PCSCF), the Interrogating Call Session Control Function (ICSCF), and the Serving Call Session Control Function (SCSCF). These elements were provide by the FOKUS Open Source IMS Core [20], and perform all the IMS control functions, including handling the registration and routing of the SIP messages.

B. OpenXCAP

OpenXCAP was used to provide the XML document management support. It is a fully featured XCAP server developed and made available under a permissive open source licence. In our testbed, it is used to manage the XML documents relating to the NSAB as well as the policy for subscriptions to presence or other events. OpenXCAP is aligned with the OpenSIPS project, and is designed to work together with it. However, it can be deployed independently as an XCAP server without the integration for subscription events.

C. OpenSIPS

OpenSIPS is an open source implementation of a standard SIP server. Together with OpenXCAP, it performs watcher authorisation and controls access to presence information.

D. Discussion

Our prototype client was able to successfully register with the OpenIMS core. Multiple instances of the client were started and were able to successfully establish voice and video sessions, as well as instant messaging sessions. As each client came online or went offline, their presence status was updated successfully.

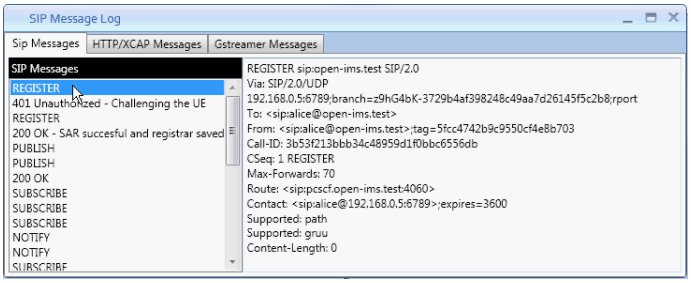

Figure 2. Example of SIP Signalling

VII. Conclusions and Recommendations

The client described in this paper successfully met the requirements discussed earlier. It is open source, modular in design, and can be extended easily. Additional debug information is provided through a debug window and modifying the options presented to the end user is trivial. The proposed client could successfully perform its functions in an open source test bed.

Currently each user has their own NSAB. Future work could involve the creation of a single address book that

multiple users have access to, such as a company wide address book. In the future, we will look at implementing additional authentication algorithms such as AKAv2 as well as IPSEC tunnels to allow secure services to be examined.

REFERENCES

[1] J. Carlin, "A survey of IMS clients for NGN service delivery," in *Communications and Information Technology, 2009. ISCIT 2009. 9th International Symposium on*, Sept. 2009, pp. 488 –494.

[2] D. Waiting, R. Good, R. Spiers, and N. Ventura, "The UCT IMS client," in *Testbeds and Research Infrastructures for the Development of Networks Communities and Workshops, 2009. TridentCom 2009. 5th International Conference on*, April 2009, pp. 1 –6.

[3] L. Prati, S. Lim, and N. Aschoff, "XDMS-Network Address Book enabler," in *IP Multimedia Subsystem Architecture and Applications, 2007 International Conference on*, Dec. 2007, pp. 1 –4.

[4] M. Masonta, M. Girod-Genet, O. Oyedapo, and D. Chatelain, "Light-weight IP Multimedia Subsystem (IMS) client for mobile devices," in *Electrical and Computer Engineering, 2008. CCECE 2008. Canadian Conference on*, May 2008, pp. 000 453 –000 458.

[5] M. Masonta, O. Oyedapo, and A. Kurien, "Mobile Client for the Next Generation Networks," in *Broadband Communications, Information Technology Biomedical Applications, 2008 Third International Conference on*, Nov. 2008, pp. 274 –279.

[6] D. van Thanh, P. Engelstad, T. Jonvik, D. van Thuan, and I. Jorstad, "Towards a Uniform IMS Client on Heterogeneous Devices," in *Networking and Communications, 2008. WIMOB '08. IEEE International Conference on Wireless and Mobile Computing,*, Oct. 2008, pp. 196 –201.

[7] D. V. Thanh, P. Engelstad, D. V. Thuan, I. Jrstad, and T. Jonvik, "Personalised IMS client widget," in *Intelligence in Next Generation Networks, 2009. ICIN 2009. 13th International Conference on*, Oct. 2009, pp. 1 –6.

[8] Open Mobile Alliance, "Presence Simple Version 2.0. Draft Enabler Release," Mar. 2008.

[9] M. Day, J. Rosenberg, and H. Sugano, "A Model for Presence and Instant Messaging," RFC 2778 (Informational), Internet Engineering Task Force, Feb. 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2778.txt

[10] A. Niemi, "Session Initiation Protocol (SIP) Extension for Event State Publication," RFC 3903 (Proposed Standard), Internet Engineering Task Force, Oct. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3903.txt

[11] M. Lonnfors, E. Leppanen, H. Khartabil, and J. Urpalainen, "Presence Information Data Format (PIDF) Extension for Partial Presence," RFC 5262 (Proposed Standard), Internet Engineering Task Force, Sept. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5262.txt

[12] M. Lonnfors, J. Costa-Requena, E. Leppanen, and H. Khartabil, "Session Initiation Protocol (SIP) Extension for Partial Notification of Presence Information," RFC 5263 (Proposed Standard), Internet Engineering Task Force, Sept. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5263.txt

[13] A. Niemi, M. Lonnfors, and E. Leppanen, "Publication of Partial Presence Information," RFC 5264 (Proposed Standard), Internet Engineering Task Force, Sept. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5264.txt

[14] H. Sugano, S. Fujimoto, G. Klyne, A. Bateman, W. Carr, and J. Peterson, "Presence Information Data Format (PIDF)," RFC 3863 (Proposed Standard), Internet Engineering Task Force, Aug. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3863.txt

[15] H. Schulzrinne, V. Gurbani, P. Kyzivat, and J. Rosenberg, "RPID: Rich Presence Extensions to the Presence Information Data Format (PIDF)," RFC 4480 (Proposed Standard), Internet Engineering Task Force, July 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4480.txt

[16] H. Schulzrinne, "CIPID: Contact Information for the Presence Information Data Format," RFC 4482 (Proposed Standard), Internet Engineering Task Force, July 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4482.txt

[17] M. Lonnfors and K. Kiss, "Session Initiation Protocol (SIP) User Agent Capability Extension to Presence Information Data Format (PIDF)," RFC 5196 (Proposed Standard), Internet Engineering Task Force, Sept. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5196.txt

[18] B. Campbell, J. Rosenberg, H. Schulzrinne, C. Huitema, and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging," RFC 3428 (Proposed Standard), Internet Engineering Task Force, Dec. 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3428.txt

[19] H. Schulzrinne, "Indication of Message Composition for Instant Messaging," RFC 3994 (Proposed Standard), Internet Engineering Task Force, Jan. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc3994.txt

[20] P. Weik, D. Vingarzen, and T. Magedanz, "The FOKUS Open IMS Core - A Global Reference Implementation," *IMS Handbook - Concepts, Technologies and Services - Mohamed Ilyas and Syad Ahson*, pp. 113–132, Nov. 2008.

Richard Spiers is studying for his PhD at the University of Cape Town, having graduated with a B.Sc. in Electrical and Computer Engineering. His previous work in this area involves research on IMS-based IPTV systems, video conferencing systems using the IMS platform as well as dynamic service orchestration.