Academic Instant Messaging System

Deploying Instant Messaging Over an Existing Session Initiation Protocol and LDAP Service Infrastructure Using the Message Session Relay Protocol

João Antunes

Author Affiliation(s)

Instituto Superior Técnico - Taguspark

Avenida Professor Cavaco Silva, 2780-990 Porto Salvo, Portugal

E-Mail: joao.antunes@tagus.ist.utl.pt

Abstract. In the last years, proliferation of the Instant Messaging (IM) systems occurred. This growth is propelled by the widespread usage of Internet and people's innate needs to communicate virtually anywhere and at anytime. Instituto Superior Técnico (IST) is a leading Engineer faculty in Portugal, with a total population of 10,000 users. Network access at IST is available almost anywhere, both in wireless and wired modes. In this and similar academic environments, IM is a powerful collaborating tool, providing remote and instantaneous communication between users. This work consists on planning, developing and deploying an IM system solution for IST's community, adapted to its current service infrastructure. IM addresses and accounts are associated with the central Identity Provider (IdP) service. This association enables the fast deployment of the optional IM service and its widespread use. The IM system is supported by the current Session Initiation Protocol (SIP) infrastructures using the SIP Instant Messaging and Presence Leveraging Extensions (SIMPLE) standards.
This IM service is based on the development of a Java generic purpose open source Message Session Relay Protocol (MSRP) peer library, and its integration with a popular Java multiprotocol IM open source client, Sip-Communicator. This work results in a contribution to the open source and IM worlds by means of software development, and deployment of an IM beta-stage system. The carried implementations as well as evaluations and results are presented. Tests were devised and executed to draw conclusions about the correct provisioning of resources for future versions of the IM system.
Keywords: Instant Messaging, SIP, SIMPLE, MSRP, IdP, LDAP.

1. Introduction

In the last years, proliferation of the Instant Messaging (IM) systems occurred. This growth is propelled by the widespread usage of Internet and people innate needs to communicate virtually anywhere and at anytime. IM technologies long gone left the ambit of providing text only communication. Currently, it is common to find that most IM technologies offer video and voice calls; folder sharing; whiteboard shared sessions; desktop sharing; picture sharing; voice clips communication; etc. This new kind of IM services offer users new, useful and joyful possibilities to collaborate and communicate over the Internet. IM is one of technologies that contributes more to enhance social usefulness of the Internet.
Along with the joyful aspect of IM, it can be also an invaluable tool. It can be used to do collaborative problem solving to people that are physically distant. It can also provide ways for people to attend and participate in events that are physically distant to them.
This work consists on planning, developing and possibly deploying an IM system solution for IST's academic community adapted to its current service infrastructure, in which all network users will automatically have an IM address associated with the central Identity Provider (IdP) service. While usage of the IM service will be obviously optional, this scheme enables the fast deployment of the service and it is widespread use.
Instituto Superior Técnico (IST) is a leading Engineer faculty in Portugal, with a total population of 12,000 users, including students, teachers, researchers and support staff. Network access at IST is available almost anywhere, both in wireless and wired modes. In this and similar academic environments contexts, IM is a powerful collaborating tool, providing virtual and instantaneous access between all academic users.
A wide-spread use of an IM system on a student community like this can prove to be a very important tool. It can aid students by facilitating collaboration. It can provide new and more efficient ways of problem solving, especially in IT engineering communities of students like the ones IST has. Given the communication features that modern IM systems have, like voice and video calls, desktop sharing and whiteboard sharing, students can work together in

group projects without the need to be in the same physical space. Professors assistance can be facilitated as they can be more easily contacted and have ways to provide help remotely, especially regarding IT.
With thorough testing and architecture details disclosure, a successful implementation and deployment of the IM system can be used and applied to other similar communities. As part of this effort, the contribution given by this work will hopefully aid in giving a successful alternative to the Open-Source community to the use of Skype and other popular commercial IM applications and protocols. Therefore contributing to the IM and open source worlds.
Section 2 of this document details the choices regarding IM protocols and does an overview of the currently available IM clients that may be involved in a solution for the IM system. Section 3 presents the reasons that motivated the defined and presented solution, and gives an implementation overview of the carried work. Following, section 4 evaluates the implementation work made and the IM system deployed by defining tests in several domains, and presenting the more relevant results along with important discussions about them. Picking up with the conclusions drawn from section 4, this document ends with section 5 where a detailed list of conclusions as well as future work is presented.

2. Related work / State-of-the art

There are a considerable number of Instant Messaging (IM) technologies developed and available to use for this work. In this section, an overview of possible contributions by each of the IM protocols and clients considered is given.

2.1 Choice of the protocol to be used by the IM system

Pre-requisites

For the IM system, due to the academic nature of the developed work, the only IM protocols that were considered to be part of the solution are the ones that have open standards.
Therefore, the subset found appropriate is listed on table 1.

Name of IM protocol	Protocol's Official Website (or protocol's working group official website)
CSpace	http://cspace.in/
Gale	http://www.gale.org/
IRC[1]	http://www.irc.org/
PSYC[2]	http://www.psyc.eu/
Retroshare	http://retroshare.sourceforge.net/
SIP/SIMPLE[3]	http://www.ietf.org/html.charters/sip-charter.html
XMPP[4]	http://www.xmpp.org/
Zephyr Notification Service	Not Available / Not Found

Table 1. List of IM protocols considered for the solution.

The possible contributions of the protocols listed in table 1 are detailed in the next subsection.

SIP/SIMPLE vs XMPP – the only two real possible choices

After doing my research on the available IM protocols and clients it became very clear to me that the choice of which protocol to be used would be either SIP/SIMPLE or XMPP. The main reasons behind this are:

- Both SIP/SIMPLE and XMPP have reliable and considerable size organizations (SIP/SIMPLE has IETF[5], and XMPP the XSF[6]) that produce and provide oversight to the specifications. These organizations are open for contributions from everyone. None of the other considered IM protocols has such kind of organizations behind them. In both organizations, several contributions have been made from industry leading companies that in most cases also implement the protocols in their products.
- SIP/SIMPLE and XMPP are in practice more deployed and enrooted than others. They count with numerous server and client implementations that the other considered protocols don't have.
- Both SIP/SIMPLE and XMPP protocols have more features already specified and being specified than any of the other protocols.

[1] **I**nternet **R**elay **C**hat (IRC) [2]
[2] **P**rotocol for **SY**nchronous **C**onferencing (PSYC) [1]
[3] **S**ession **I**nitiation **P**rotocol for **I**nstant **M**essaging and **P**resence **L**everaging **E**xtensions (SIP/SIMPLE) [3]

[4] **E**xtensible **M**essaging and **P**resence **P**rotocol (XMPP) [4]
[5] More information on IETF can be found at http://www.ietf.org
[6] http://xmpp.org/xsf/

Therefore I consider that both these protocols are in a healthy state, because they are actively being developed and maintained. These reasons make me believe in the prediction that they represent the future of openly specified IM protocols.

PSYC [1] is also a very promising protocol. However, in my opinion, its development is actively maintained by a too small number of people. Therefore I believe that it has a dubious future ahead of it. With current development efforts, I find it very improbable that PSYC [1] gains sufficient critical development mass to have an important role in the IM open specifications scenario. Especially given the XMPP and SIP/SIMPLE alternatives.

Some of the features provided by Zephyr, are to be considered of great value in implementing IM in an academic community. More specifically:

- The notification system, that could for instance allow notification for mail arrival/printing job status/network, workstation logins, etc.

But the protocol itself wasn't found to be a viable choice.

2.2 Overview of the available IM clients suitable

Pre-requisites

The number of IM clients and protocols proliferated over the last years along with the number of Internet users [5]. Success and usefulness of the IM system deployment, can be measured by the number of users that adhere to it and use it on a daily basis. Choosing and promoting an IM client as part of the system, is a needed and critical task, because this will be the primary interface for the user with the rest of the IM system.
Developing an IM client for the system from scratch would be too much resource consuming and ultimately a waste of resources given the already available IM clients.
There is a large spectrum of possibilities to choose from for the IM client. In order to make this list smaller, I devised the following list of requirements:

- *Multiprotocol IM client* – I assume that the large majority of the most probable potential users of IST's IM system are users that already use IM and have their own IM accounts in the most popular protocols. Therefore, having an IM client application that provides compatibility with the IST's IM system and other IM systems, can function as an unifying application. Liberating the user from the hassles associated with using different applications for the same purpose;
- *Free to use application* – There are lots of good and free IM clients. It makes no sense to promote/choose an IM client for which users or IST needs to pay for.
- *Support for the major features (file-transfer, instant messaging) of the major IM protocols*:
  - MSN Messenger;
  - XMPP/Google Talk;
  - SIP/SIMPLE;
  - ICQ;
- *Support for the following operating systems* (a must, even if the client is open source, due to the issues involved in porting a software to another operating system (OS).)*:*
  - Microsoft Windows NT family – A must, as this is the most common OS used;
  - Max OS X – Not required, but very valued;
  - Linux – Not required, but very valued;
  - Other OSs – Not required, valued;

Table 2, provides a list of all the IM clients that:

- Already fulfilled or has good perspectives of fulfilling the requirements in 2009.
- If a client doesn't fulfil the requirements, they can still figure on the list if they are Open Source (so that they could be altered in order to fulfil the requirements).

The clients that figure on table 2 are compared in the next subsection.

IM client name	Available for download at:
AYTTM	http://ayttm.sourceforge.net/
BitlBee (IM Gateway)	http://www.bitlbee.org/main.php/news.r.html
CenterICQ	http://thekonst.net/centericq/
Climm	http://www.climm.org/
InstantBird	http://instantbird.com/
Miranda	http://www.miranda-im.org/
Naim	http://naim.n.ml.org/about
QuteCom	http://www.qutecom.org/
Pidgin	http://www.pidgin.im/

Sim-IM	http://sim-im.org/wiki/Main_Page
Sip-Communicator	http://www.sip-communicator.org/

Table 2. List of considered IM clients/gateways and the websites were they can be found.

Comparison of the clients, revealed that from all the clients listed in table 2, the more viable choices are:

- QuteCom;
- Pidging;
- Sip-Communicator;

Other clients weren't found suitable because they:

- Were either abandoned by the developers (SIM-IM);
- Didn't provided a level of implemented features as the ones stated above (Miranda, BitlBee);
- Don't have a modern looking GUI or don't have a GUI at all (Climm, Naim, CenterICQ, AYTTM);
- Have plans to be very good clients, and development seems steady, but just aren't there yet (InstantBird);

3. Implementation

3.1 Solution architecture

3.1.1 Choices and motivation

Reasons for the protocol and client choices are given below.

IM protocol – SIP/SIMPLE was chosen. ***Reasons:***

- Facilitates adoption of the IM system by the IST's users that already are using the SIP infrastructure. – Most SIP softphones already support IMing. The current VoIP domain (voip.ist.utl.pt) can both be used for the VoIP and IM services;
- Choosing a XMPP based system would probably prove to be less troublesome to implement, as more software is already implemented for this protocol. On the other hand, choosing SIP/SIMPLE and the required software developments, allowed me to give a more valuable contribution to the IM world and to the open source community. Seen that choosing SIP/SIMPLE, without developing other pieces of software could have lead to:
    - Worse interoperability with other IM protocols – As SIP/SIMPLE doesn't has the gateways (transports) software that XMPP has, it becomes a requirement to use a multiprotocol IM client;
    - An IM system that couldn't even provide file transfer capabilities – There were no clients at the time of start of the work, that had implemented file transfers over SIP/SIMPLE;
    - Less functionalities, as generally, there are more clients with more features supported for XMPP than for SIP/SIMPLE;

  In sum, an IM system with less features. Seen that his work aims to serve IST's community, this solution wasn't tolerable. To mitigate this problem the following decisions were made:

  - Choose a multiprotocol IM client – mitigates the interoperability problem of choosing SIP/SIMPLE;
  - Create the MSRP peer library and use it to enhance the chosen IM client – solves the file transfer problem as well as provides feature enhancement possibilities for the chosen IM client;

  IM client – Sip-Communicator was chosen. ***Reasons:***

- Fairly numerous and valuable set of features that are currently implemented, namely:
    - Secure call support through the use of zRTP (one of the first clients that support this protocol).
    - Support for some of the most popular protocols:
    - Supports the Portuguese language for the interface – although not listed in the Roadmap, this feature is implemented, check SC's SVN repository[7];
    - Video and Voice calls through SIP signalling;
    - Support for the most basic functionalities of the protocols listed in the requirements available in the *'Pre-requisites'* subsection of section 2.2;
    - Appealing, modern-looking minimalist and very functional GUI;
- Very good features expected to be implemented in the not so far future;
- Out of the box interoperability support between IM protocols at the client side;
    - Interoperability at the client side is preferred over server side. Client side interoperability by definition represents a more distributed effort. Server-side interoperability imposes extra resource requirements from the server;

[7] Information about accessing the SVN repository is available at: http://www.sip-communicator.org/index.php/Development/VersionControl

- Has an active and dedicated group of developers[8]. Developer efforts have a tendency to grow as SC is listed in the Free Software Foundation (FSF) high priority list[9];

Also, as stated above, development from the ground up of a Message Session Relay Protocol (MSRP) library was carried. This library allows to develop important features to the Sip-Communicator, as well as contribute to the SIMPLE IM and open source worlds in general.

3.1.2 Architecture overview

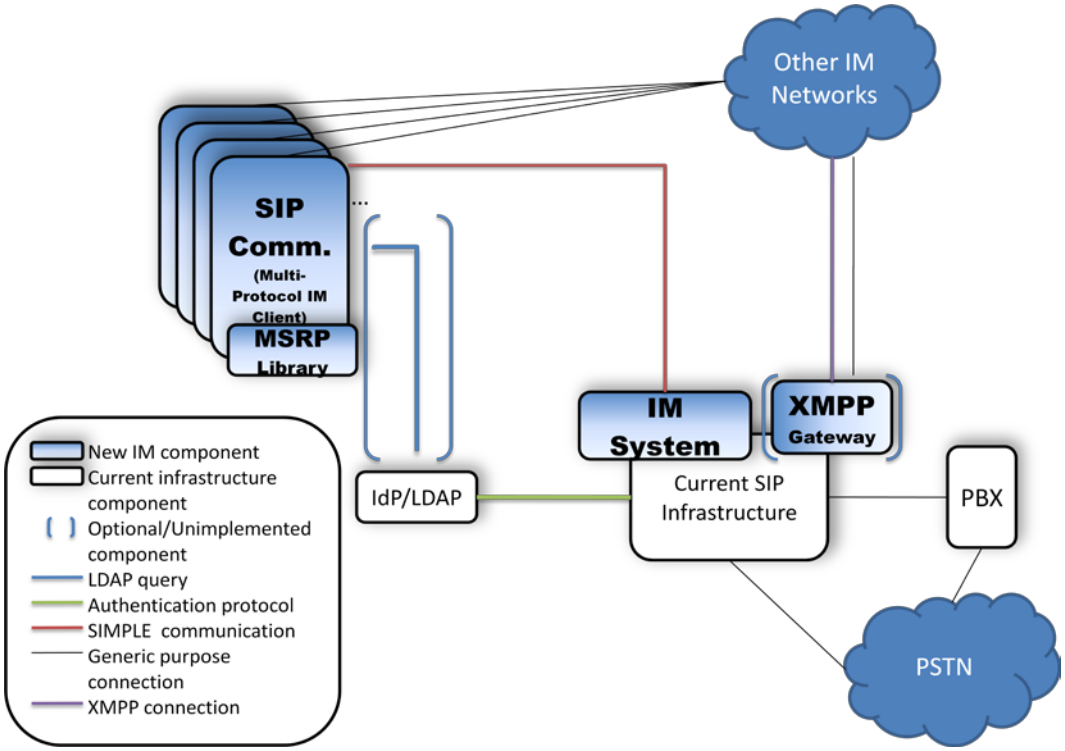

Figure 1. IM system solution architecture.

Figure 1 also gives general details on how the IM system and current SIP infrastructure will be integrated. To note that there are two optional components, the XMPP gateway, and the direct LDAP connection between SC and IST's LDAP. This is because:

- The direct LDAP connection is probable to be a reality in the near future;
- The XMPP gateway is always an option and can be deployed at anytime if seen as required;

The solution's architecture is expected to be fully integrated with the current IST's VoIP services. Seen that the VoIP services were already at use, to avoid VoIP service disruption, a testing system was devised and implemented. This testing system is a transitory one that will converge to the final production system that is depicted in Fig. 1.

3.2 IM System details

Details on the chosen solution software components as well as the reasons of the choices are given in this section. Also, the reasons and changes on the previously existing SIP infrastructure are exposed.

3.2.1 Chosen software components

The current software application being used as a SIP Registrar and proxy server in the VoIP infrastructure Sip Express Router (SER). SER development seems abandoned, as the most recent version released of SER dates from January of 1996. Therefore, it has been losing some terrain in the number of implemented features, to other open source implementations as Kamailio[10] and OpenSips[11].
After inspecting Kamailio's history pages, it became clear that some core developers abandoned the SER project to join the OpenSer (that lately forked into OpenSIPS and Kamailio). Due to these reasons, I consider the SER

[8] Evidences of the communities activity can be found here: https://sip-communicator.dev.java.net/servlets/Search?scope=project&resultsPerPage=40&query=top+project&Button=Go
[9] Sip-communicator is listed under the '*Free software replacement for Skype'*. List available at: http://www.fsf.org/campaigns/priority.html
[10] Kamailio – Project website: http://www.kamailio.org/
[11] OpenSIPS – Project website: http://opensips.org/

application outdated. Having gathered these facts, it became clear to me that the solution was a choice between OpenSIPS or Kamailio.

OpenSips vs. Kamailio

There aren't many differences between these two pieces applications. Both originated from OpenSER. Apparently, they became two distinct projects due to fundamental divergences between groups of developers.
The chosen brand ended up to be OpenSips 1.5. The only reason to choose OpenSIPS over Kamailio is that OpenSIPS provides out of the box integration with MSRP Relay software[12].
To note that due to the similarities of the configuration syntax and modules, a migration to Kamailio can be easily attained.

Changes in the existing SIP infrastructure

The IM production system (see figure 1) can also be used to provide an update to the currently used software in the VoIP infrastructure. This update, if done, won't change any of the features of the current VoIP services. The changes would be:

- Removing the need to have a RADIUS service in the middle to provide authentication (as it is required and implemented in the current SIP infrastructure [6]). Therefore, support for direct LDAP authentication with credential cache to reduce the amount of LDAP queries;
- General software version upgrade – All of the software has a chance to be upgraded, with the typical inherent advantages associated (e.g. security improvements, optimization improvements, feature improvements, bug removal, etc.);
- Simpler infrastructure maintenance – The software upgrades weren't as simple to make in the existing SIP infrastructure. This happened due to a patch applied in the source code of some of SER's modules, to allow it to work with the intermediary RADIUS service;

3.3 MSRP Peer library

This section presents a brief overview of the current and future features of the open source peer library implemented in Java, of the Message Session Relay Protocol (MSRP) [7] that was developed from the ground up.

Currently available features:

- Basic, non-TLS, MSRP peer functionality support (RFC 4975 [7]);
- MSRP message level of abstraction (i.e. the application doesn't have to care with transactions but only with messages) which allows:
  - Accepting and denying MSRP messages;
  - Notification of aborted message events;
  - Notification of sent bytes;
  - Notification when REPORTs are received – provides a way for the API to notify the application when the receiving peer successfully acknowledges the receipt of message parts (Success REPORTs). Also notifies the application when a Failure REPORT is received;
- Provides full customisation of the granularity of the sent bytes notifications, as well as of the successfully received parts of message notifications (success reports);
- Data storage abstraction layer – Provides a common set of methods for reading and writing data independently of the used medium. Also provides an easy way to extend the support to other kind of mediums. Mediums currently supported:
  - Files;
  - Memory;
- Support for all the major Java logging systems[13];
- Well documented library (Javadoc and in-code comments) as well as some example classes[14]– some of them requested by fellow developers interested in using the library;

Features to be implemented:

- Support of TLS;
- Relay extensions [8] support;
- Enhancement of the library's robustness and specification compliance;

---

[12] Which might prove to be a valuable software component to add to the IM system in the future (seen that currently no free to use IM clients that I know of use this technology).
[13] This was possible through the use of the simple logging facade for java (SLF4J) more information and specific details of which logging systems are supported are available at: http://www.slf4j.org/
[14] Example classes are available under the msrp.examples Java package. Source code is available at the project's webpage (https://msrp.dev.java.net/).

- Transaction level interface – which allows the application that uses the library a control over each of the MSRP transactions and their responses (e.g. like the one that the JAIN-SIP stack provides);
- Prioritization and fair usage of connections/sessions support;
- MSRP message objects being extended beyond the session and outliving them – already semi implemented but tests are lacking because it isn't a critical feature for the current uses of the library;
- An even better performance – A lot of performance improvements have been made already. But still, like stated in the results section 4.3, some gains can still be achieved;
- Extending and testing the library to be able to be used by the Android operating system (to which Sip-Communicator has already developed and concluded a working prototype);
- Extending the library to be used by other programming languages – solutions may come from the use of sockets to communicate internally between the API written in another language, and the core written in Java;
- Make clearer to the library users which classes they should use and which are off-limits and should only be used internally – Refactoring the source code and implementing some packages with interfaces should be the way to attain this feature;
- Data content wrapper validator;
- Library level support for the Common Presence and Instant Messaging (CPIM) [9] message format wrapper;

Part of the features are planned to be implemented until February 2010[15].

3.4 File transfer support via SIP in Sip-Communicator

The file transfer feature is available in every instant messaging protocol considered. The Sip-Communicator application, with the virtues already stated in section 3.1.1, is lacking support for file transfer over SIP. In my opinion, lack of such a feature makes it an unappealing choice to be used on the IM system. Efforts into building an open standard to allow SIP/SIMPLE systems to have file-transfer, had already been initiated by the IETF when I did my initial research to make the choices regarding the system. Therefore, it was possible for me to choose to implement file-transfer in SC guaranteeing interoperability. The standardisation efforts matured to its final RFC form [10] in May 2009.

4. Results and discussion

4.1 MSRP Peer library

I used a test-driven development (TDD) methodology to develop the library. Therefore, many of the corrected bugs and functionalities have an associated test.
MSRP peer library was evaluated in the following domains:

- Performance;
- Memory efficiency;
- Functionality;

The tools used to make such evaluations were essentially JUnit tests together with support classes and a Java profiler. The profiler used was the Java profiler Test and Performance Tools Platform (TPTP)[16].
Different mixes between the tests and tools where used to test each of the domains.
In this document, only part of the performance test results and conclusions are detailed. Memory efficiency tests attained very satisfactory results. Functionality tests where carried by the use of the above mentioned JUnit tests, and successfully confirmed correct implementation of the features listed in section 3.3. However, these tests don't fully guarantee that existing features work flawlessly, but certainly provide much more guarantees than if they didn't existed.

Performance results and discussion

File size (bytes)	TestSendingExistingFile (with success report)			
	Runs: (all values in ms)			Max-Min (ms)
	1st	2nd	3rd	
1219974	266	156	172	110
16742799	922	953	1047	125
243809310	46391	19188	13907	32484

Table 3. Performance test results. All tests ran on an Intel Core 2 Duo T7300 @ 2GHz with 2GB of RAM (LG e500 laptop).

[15] Work sponsored by the NINet foundation.

[16] More details on the TPTP software can be consulted on the project's website: http://www.eclipse.org/tptp/

Values presented in table 3 represent the time spent transferring files from two peers located in the same network endpoint. The big variation of time spent between runs, gathered when sending the file of 243809310bytes, probably corresponds to a bug. I consider that these test results, without the variation, are satisfactory, but further tests with the use of the profiler show that these times can be reduced to a maximum of ~43,89% by cutting in the ability for the stack to have custom granularity Success-REPORTs. Profiler tests also show possible performance gains without sacrificing functionalities, by making some refactoring of the code. However, due to the nature of the enhancement, reduction estimates are impossible to be given.

4.2 Sip-Communicator, adding the file transfer over SIP feature

To provide automatic testing of the implemented file transfer functionality, a SLICK test[17] was implemented. This SLICK test is based on the already available generic file transfer operation set SLICK test available in the package net.java.sip.communicator.slick.protocol.generic. The implemented test, validates correct behaviour in the following scenarios:

- A simple send and receive of a file;
- The sender of the file cancels the file transfer before the request is accepted by the receiver;
- The receiver declines the file transfer;
- The receiver cancels the file transfer while it was in progress;
- The sender cancels the file transfer while it was in progress;

As automated tests might not provide full guarantees on all of the functionalities behaviour, these test scenarios were also performed manually, by running two instances of SC, with success.
Regarding the specifications used to implement this feature [10], SC's implementation isn't fully compliant. These are mostly due to structural limitations of SC, and probably don't break compatibility with most clients that may have adopted the same specifications. More details on those limitations can be found in the full version of the dissertation [11].
The file-transfer feature has no support for MSRP relays. Meaning that the MSRP protocol can only be used if the sending endpoint can contact directly the receiving endpoint's IP, which can be a problem for endpoints behind NAT systems. The MSRP peer library doesn't yet support the MSRP Relays extension. Also, SC doesn't have the necessary GUI(s) for configuring the relays. This limitation is planned to be solved in the future after the implementation of the Relay extensions to the MSRP library.

4.3 Instant Messaging system

IST's community has around 10 000 potential IM users. The tests that I carried on the IM system are divided into:

- Performance tests – To evaluate the testing system's capability to handle all the potential users;
- Feature tests – To ensure that all of the system's functionalities are working correctly.

I executed tests only to evaluate the IM features and VoIP functionalities that changed due to the new software components of the new architecture.
For the feature tests, very straightforward methodologies were applied. Whenever possible, clients that used the features where used. For the core IM system, Sip-Communicator and CounterPath's Xlite[18] were used to test the instant messaging and presence capabilities. For the RLS capabilities, the IM server's database was checked to see if the resource's persistence was assured.
All of the feature tests described above where concluded successfully. Due to the nature of the tests, no useful output from them are available, as they are simply pass or fail tests. To note that further tests could be made given more fully featured SIP/SIMPLE IM clients, that simply weren't available at the time of writing of document.

Performance tests

Sipp [12], in combination with Sysstat [13], were the chosen tools to carry the tests and gather the results.
Two test scenarios for use with the Sipp tool where written. One for testing the performance of the REGISTER SIP method, whose internals changed as described in section 3.2.1. The other to test the MESSAGE SIP method. The Sipp scenario files are available in the appendixes of the full dissertation [11]. Both tests where made with the credential cache turned on and off.

[17] SC has a package with the so called Service Leveraging Implementation Compatibility Kit (SLICK) tests. These SLICK tests are developed using JUnit. They are used to test the service classes found in the service Java packages (net.java.sip.communicator.service.*) of SC.
[18] A popular free to use SIP client that also supports IM functionalities. More information and download links can be obtained at: http://www.counterpath.net/

To note that the gathered tests on CPU, memory and bandwidth consumptions, indicate that fine tuning of the number of threads run by the IM server, or giving a bigger priority to the OpenSips processes, might increase the rates. Regarding the fine-tuning, the reported CPS rates ran with OpenSips configured to have 4 threads.
Results on register tests, with and without cache, have overall satisfying results.
The most important test and results come from the message test results with credential cache. Details are provided below:
As page mode[19] is the current mode employed by the IM system, at least until session mode isn't widely implemented in most existing SIP/SIMPLE IM clients, the capacity of the server to handle the MESSAGE requests is critical. Extending the recommendations found in RFC 3428 [14], as stated in chapter 3 section 3.2.1, message authentication is required by the implemented system. This adds an extra overhead to the transmission of the messages, but with important gains on security.

Message retransmission starts to grow at 1050 CPS target rate, and transmission failures (failed calls) at 1130 target CPS.

These results, assuming that no other parallel requests are made, support a scenario of about 3000 users that are actively engaged in IM, and therefore send a message every 3 seconds. To note that the fact that the rhythm of message generation usually depends on the rhythm of the conversation, when delay is introduced with the retransmissions, the message generation rate per user should also decrease (if the behaviour stated in RFC 3428 [14] is followed). However, waiting for more than 4s for a message to be delivered, doesn't promote a very instantaneous instant messaging system, and users may abandon such a system if it becomes that sluggish.

These results support the setup of an IM server where the MESSAGE requests should be routed to a server or a cluster of servers to allow higher CPS rates. These setup recommendations make the implementation of a production system more expensive. However, with the mass adoption of session mode IM by SIP/SIMPLE clients, page mode could be abandoned. If all the IM clients support session mode instant messaging, the dedicated server or cluster of servers to handle the MESSAGE requests is no longer needed[20].

5. Conclusions and future work

The SIMPLE extensions to the SIP protocol are still in an immature state. This is due to the underdevelopment of actual implementations of the already defined SIMPLE standards, which are relatively recent. Nevertheless, mostly due to the close connection with SIP and SIP related technologies, SIMPLE has a very big potential to promote IM services.
This work helped to mature the SIMPLE open standards by contributing with palpable open source implementations of relatively recent standards. It also contributed to the open source world, as all the implementations are open sourced. To the IST's community, a new IM service possibility has been opened and deployed in a beta stage. The IM service is in a beta stage mostly due to the immature state of the chosen protocol, that forced me to spend a great deal of effort in software implementations.
Mostly due to the tests made and consequently results and conclusions drawn from them, directions and obstacles are now more clear providing a good basis to develop a concise and good deployment plan for a mature IM service for IST and possibly other communities.

Future work

Due to the multiple work directions pursued, this work opens many doors to further developments and improvements that ultimately will continue to provide contributions to the IM, SIMPLE, and open source worlds.

MSRP Peer Library

As stated in section 3.3, MSRP library development will continue to be done. In the same section, the features to be implemented are listed. Development of these features might be aided by contributions from fellow developers, as I continue to sporadically receive manifestations of interest regarding the library via e-mail. These contributions can add different features to the library.
The achieved results of the library state regarding performance, point that the there is room for improvement in the runtime performance of it.

---

[19] SIMPLE's page modem in a nutshell, defines that all of the transmitted messages should pass through the SIP proxy.
[20] Because the session mode is more resource efficient for long run IM sessions. Only the initial IM media session setup costs any resources to the IM server just like a regular voice call. After the initial setup, the connection is typically made directly between the participants (unless some sort of other mechanism is needed to enable connectivity, e.g. MSRP Relay for users behind NAT. But even these mechanisms should consume much less CPU and memory resources).

Sip-Communicator

Together with some MSRP library improvements, the MSRP library opened possibilities for implementation of the following features in Sip-Communicator:

- MSRP relays support – improving the connectivity of all of the features that depend on MSRP, including the already implemented file transfer functionality;
- SIMPLE's session mode IM support;
- Desktop sharing;
- Whiteboard sharing;

IM System

Further tests are yet to be done before migrating the system to a production phase, as these tests might impose new requirements to the IM system. Also, the deployment of the MSRP Relays and OpenXCAP components can prove to be valuable in the future. At the present moment, as currently there are no free IM clients that are able to use the features offered by these components, this implementations where be of doubtful value. To attain a fully integrated deployment of these two components, LDAP authentication support should be implemented in both of them. As these two software components are developed in Python (with a modular design) and seen that Python has a mature LDAP library, implementing LDAP authentication should be a very straightforward task.
Deployment of the production system should be accompanied by an implementation plan as well as promotion amongst the IST and possibly other communities.
NlNet foundation is aimed to stimulate open sourced implementations and documents that promote network research and development in the domain of Internet technology. Therefore, development and deployment of a publicly disclosed architecture of an IM system that uses SIP/SIMPLE open standards and open source solutions as this one is, is well within the scope of NlNet support activities. NlNet already agreed on funding part of the deployment of this IM system, therefore, conditions exist that make probable that the above stated future work will be carried.

References

[1] Carlo von Loesch, "PSYC – Protocol for Synchronous Conferencing", http://about.psyc.eu/
[2] C. Kalt, "Internet Relay Chat: Architecture", RFC 2810, April 2000
[3] J. Rosenberg, "SIMPLE made Simple: An Overview of the IETF Specifications for Instant Messaging and Presence using the Session Initiation Protocol (SIP)", Internet-Draft draft-ietf-simple-simple-04, 31 October 2008.
[4] P. Saint-Andre, "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 3920, October 2004
[5] Raymond B. Jennings III, Erich M. Nahum, David P. Olshefski, Debanjan Saha, Zon-Yin Shae and Chris Waters, "A Study of Internet Instant Messaging and Chat Protocols", IEEE Network, July/August 2006
[6] A. Regateiro, "Voice over IP System in an Academic Environment", Instituto Superior Técnico MsC thesis, September 2007
[7] B. Campbell, R. Mahy and C. Jennings, "The Message Session Relay Protocol (MSRP)", RFC 4975, September 2007
[8] C. Jennings, R. Mahy, "Relay Extensions for the Message Session Relay Protocol (MSRP)", RFC 4976, September 2007
[9] G. Klyne, D. Atkins, "Common Presence and Instant Messaging (CPIM): Message Format", RFC 3862, August 2004
[10] M. Garcia-Martin, M. Isomaki, G. Camarillo, S. Loreto and P. Kyzivat, "A Session Description Protocol (SDP) Offer/Answer Mechanism to Enable File Transfer", RFC 5547, May 2009
[11] J. Antunes, "Academic Instant Messaging System - Deploying Instant Messaging Over an Existing Session Initiation Protocol and LDAP Service Infrastructure Using the Message Session Relay Protocol", unpublished, October 2009
[12] SIPp project webpage: http://sipp.sourceforge.net/
[13] SYSSTAT utilities home page: http://pagesperso-orange.fr/sebastien.godard/
[14] B. Campbell, J. Rosenberg, H. Schulzrinne, "Session Initiation Protocol (SIP) Extension for Instant Messaging", RFC 3428, December 2002