Report of Otago Contributions to Telecom LifeLink Project

Nathan Lewis
Hailing Situ
Melanie Middlemiss

The Information Science Discussion Paper Series

Number 2008/02
July 2008
ISSN 1177-455X

UNIVERSITY of OTAGO

Te Whare Wānanga o Otāgo

Report of Otago contributions to Telecom LifeLink Project

Nathan D. Lewis
Hailing Situ
Melanie Middlemiss

Global Network Interconnectivity Project

Friday, 11 July 2008

1. Introduction

Gartner has for some time been reporting the potential for virtual world technology to become the next wave of the Internet, delivering what is known as the Web3.D environment. This is characterised by a high level of user participation through immersion in the virtual world. Gartner has predicted that by 2011, 80% of internet users will be regular users of Web3.D technology [1].

Project LifeLink was initiated to discover what opportunities for Telecom might exist in the growth of business and consumer interest in virtual worlds. This has focused on a number of technologies, in particular Second Life [2], OpenSimulator (OpenSIM) [3] and JAIN SLEE [4]. The project has been run by Telecom with coordination and support from MediaLab, and with researchers at Canterbury and Otago Universities. This report describes the work undertaken at Otago University to implement a gateway to enable demonstration of communications between an object in Second Life and the JAIN SLEE environment in order to interoperate with external network services.

The report is structured as follows: Section 2 gives a brief overview of the technologies used in this project; Section 3 describes the whiteboard application that has been developed; and Section 4 describes the HTTP-SIP gateway that has also been developed. To conclude, a copy of a paper entitled "*Using JAIN SLEE to Extend Second Life Services for Flexible Social Networking*" which has been submitted to the Hawaii International Conference on System Sciences (HICSS) is provided in Appendix A.

2. Technologies

2.1. Virtual Worlds

Currently there are numerous web-based computer-mediated social networking tools available that provide communities or networks of users who share similar interests and activities methods for chatting (via voice or text), file sharing, etc. Some examples of these tools are MySpace, Facebook, Bebo, and Flickr [5-8]. As well as these 2D web-based tools, virtual worlds, such as Second Life, are becoming increasingly popular as 3D platforms for collaborating and social networking.

Second Life is slightly different in that it is not Web based; rather client software is used to connect to the application-specific Linden Lab Second Life servers. Second Life allows users to easily interact with other users who are also in Second Life. It also offers some limited ability to render an image of a web page on an object in the Second Life virtual world. However, Second Life is a proprietary and closed platform and is restrictive in enabling service extensions or interaction with external services.

It provides only two protocols for communicating between objects inside Second Life and external servers – HTTP and XML-RPC protocols.

The HTTP protocol is a standard protocol used for Web-based applications. In Second Life, HTTP requests can be sent to external Web servers. Second Life can only process the response to a HTTP request; it cannot accept incoming HTTP requests. XML-RPC is a standard messaging protocol for invoking operations on remote machines. XML data is sent via HTTP to the remote system to be handled. Second Life can only receive external communications via XML-RPC. The communication must be initiated by an external server.

In order to make Second Life more accessible to outside communication, some mechanism is needed to facilitate interaction with external services. This can be achieved using JAIN SLEE.

2.2. JAIN SLEE

JAIN SLEE, sometimes referred to simply as JSLEE, gets its name from Java APIs for Integrated Networks (JAIN) Service Logic Execution Environment (SLEE). The SLEE provides a standard integration environment for multiple network resources and protocols, refer to Figure 1.

JAIN SLEE is a common middleware platform. With different resource adaptors, JAIN SLEE application servers can receive different kinds of requests, process the requests and send responses back.

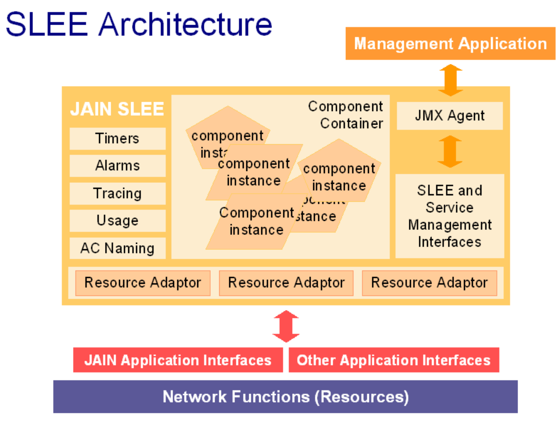


Figure 1: SLEE Architecture [9]

The benefits of using JAIN SLEE are as follows:

- Using existing resource adaptors can provide an easy start to application development. In this project, we use the HTTP and SIP resource adaptors.
- Developers can combine different servers (e.g. HTTP server and SIP server) in the same application.
- Developers do not need to learn different server programming models to develop applications. They only need to know the JAIN SLEE programming model and are able to develop different kinds of applications according to this model.

In the JAIN SLEE architecture, a SLEE defines how an application can be composed of components. These components are known as Service Building Block (SBB) components. Each SBB component defines the event types that it subscribes to and has event handler methods that contain application code that processes events of these event types. The SBB component also declares the SBB local interface of the SBB component. The SBB local interface specifies the methods of the SBB component that may be invoked synchronously. The SBB component may have zero or more child SBB components. The SBB component specifies its child SBB component relations [10, pp. 23].

3. Whiteboard application

3.1. Introduction

The whiteboard application was developed as an initial application to demonstrate the communications between an object in Second Life and the JAIN SLEE environment. The whiteboard was designed to allow several Second Life users to add, edit and view text on a commonly accessible Second Life object. It allows users to interact with it in a similar way to a real whiteboard. It is updated in real-time and has a persistent state so that it can be retrieved at a later date.

A brainstorming session is an example of the use of the whiteboard. Several people could be communicating together (with text or voice) and the ideas they produce can be recorded for everyone who is present to see. Additions and alterations can be made during the session, perhaps by more than one of the participants.

Second Life functionality does not allow arbitrary text to be displayed on an object. Textures containing text can be uploaded to Second Life, but required a fee to do so, and the process is not real-time and cannot be easily automated. Other solutions involve building an array of objects, each textured with a single letter of the text that is to be displayed, but are slow, require complex scripts to operate and are limited to the amount of text that can be displayed. The whiteboard overcomes these limitations by replacing a texture, which is on the surface of a single Second Life object, with the contents of a webpage.

3.2. What it does

The whiteboard listens to a chat channel for commands which are used to change the content or adjust the configuration.   The Whiteboard listens for the following commands:

- ADD – Add a line of text to the end of the whiteboard.
- REPLACE – Replace the specified line of text with a new one.
- GET – Retrieve a line of text.
- DELETE – Delete a line of text.
- MOVEUP – Move the specified line of text to a higher position on the whiteboard.

- MOVEDOWN – Move the specified line of text to a lower position on the whiteboard.
- RESET – Remove all text from the whiteboard.
- CHANNEL – Set the channel number that the whiteboard listens to.
- KEY – Change the key of the whiteboard that is being displayed.

The GET command is used to retrieve a line of text so that it can be edited. When a line number is specified, the person who issued this command receives a private chat message containing the line of text specified. This text can then be copied and pasted as required.

The KEY command is used to select a specific instance of the whiteboard. By specifying a key that was used during a previous session, the contents of the whiteboard that were present at the end of that session can be retrieved.

Further functionality could be implemented. Ideas for further functionality include:

- Controlling how the text is displayed on the whiteboard. The font, size, colours etc could be change. While some settings could be controlled by the user, a setting such as the size may be adjusted automatically by the program to make sure that all the text fits onto the object.
- Controlling who has permission to edit the whiteboard. It could also record who makes what changes.
- An undo command.

3.3. How it works

The whiteboard is built using a Second Life script and a JAIN SLEE SBB. When the whiteboard script is started, it sets the Land Parcel URL to the address that the SBB is listening on. When this URL is accessed, the SBB returns a webpage containing the current contents of the whiteboard.

A Second Life user can issue commands which are interpreted by the Second Life script. It sends the commands as parameters to another URL that the SBB is listening to. The SBB maintains a database of the contents of the whiteboards and makes changes as instructed. Once a change is made, the SBB informs the Second Life script that the change was successful by sending an appropriate HTTP response. If a command resulted in a change to the contents of the whiteboard, the Second Life script updates the Land Parcel URL, causing Second Life to retrieve a fresh copy of the webpage containing the updated contents.

User instructions for the whiteboard are provided with the Second Life object and given here in Appendix B.

4. HTTP-SIP Gateway

4.1. Introduction

Second Life is a proprietary and closed platform and is restrictive in enabling service extensions or interaction with external services. It provides only two protocols for communicating between objects inside Second Life and external servers – HTTP and XML-RPC protocols. In order to make Second Life more accessible to outside communication, some mechanism is needed to facilitate interaction with external services. In this project, the HTTP-SIP Gateway (or "gateway") was designed to allow users in Second Life to communicate with people outside Second Life, and vice-versa, by using the Session Initiation Protocol (SIP) and taking advantage of the JAIN SLEE platform.

SIP allows two people to negotiate a connection. The type of connection made is independent of the protocol, and is most often a voice or video stream. SIP has many features which including the ability to register presence, discover the status of other SIP users and can be used as transportation for sending instant messages.

Second Life users have the ability to type messages to people within hearing distance, send instant messages to each other and speak to nearby Second Life users. The ability for Second Life users to communicate with people who are not logged in to Second Life is limited (or non-existent).

In this project we implemented a gateway that allows Second Life users to send instant messages to people who are outside Second Life and are using a SIP software client or SIP enabled device. A person with such a device or software can also send an instant message to a user who is logged in to Second Life.

4.2. What it does

In Second Life, a user "wears" a text client. The text client object contains a *Settings* notecard, where the user can set their SIP address, and a *Friends* notecard, where they can list the SIP addresses of people they wish to communicate with.

When the *connect* button is touched, the text client informs a SIP registrar server, via the gateway, that the user is online. Once registered, a SIP user outside Second Life can send the Second Life user an instant message by specifying the Second Life user's SIP address. The instant message is routed through the gateway, the text client receives the instant message and then presents it to the Second Life user.

The Second Life user can click the *Friends* button to select a SIP address from those listed in the *Friends* notecard. The text client will then listen for any message written to it on the chat channel specified in the *Settings* notecard. When such a message is detected, the text client sends the message and the most recently select SIP address to the gateway. The gateway sends the message to the specified SIP address.

Further functionality that could be implemented includes:

- Automatically refreshing the user's registration (currently times out after 3600 seconds).
- Retrieving the list of friends from an online source.
- Displaying the status of friends.
- Displaying instant messages in a chat session (requires additional features in the virtual world client and servers).
- Initiating voice communications (requires additional features in the virtual world client and servers).

Where the virtual world client and servers need to be changed to implement additional features, those changes may be possible to implement within OpenSIM.

4.3. How it works

The gateway is implemented as a JAIN SLEE SBB. It communicates with a Second Life script which is contained within a text client object.

The text client has buttons to allow the user to register and unregister their SIP address with their SIP registrar via the gateway. Another button lets the user select the SIP address which will be the recipient of instant messages. It sends HTTP requests to the gateway and listens for XML-RPCs.

The gateway listens for HTTP requests from the text client and sends XML-RPCs to it. It also sends and receives SIP requests to and from SIP servers. The states of the text clients are recorded in a MySQL database.

1. Registering

A Second Life user begins by wearing their text client and clicking the Register button. The text client establishes a XML-RPC channel for incoming communications. It constructs a HTTP request which contains parameters including the SIP address of the user, the XML-RPC channel identifier and the address where XML-RPC communications are to be sent. The request is sent to the `httpsipgateway_register` path on the gateway and the values are recorded in the database, using the XML-RPC channel identifier as the primary key. A `200 OK` response is sent back to the text client. The gateway then sends a SIP REGISTER request to the SIP registrar. The REGISTER request contains the SIP address of the Second Life user and the address of the gateway as the destination where communications with this user should be sent. The registrar should respond with a `200 OK` SIP response. When the gateway receives this response, it generates a XML-RPC containing the XML-RPC channel identifier and a message indicating that the registration was successful. The text client informs the user.

2. *Sending an instant message*

A Second Life user can send an instant message to any SIP address by first clicking on the Friends button. A dialog box is displayed containing a button for each SIP address in their Friends notecard. The user clicks on a SIP address to select it as the active recipient. The text client listens on the chat channel that the user specified in their Settings notecard. When the text client receives a message on that channel, it sends a HTTP request to the `httpsipgateway_message` path on the gateway containing the XML-RPC channel identifier, the SIP address of the selected recipient and the contents of the instant message. The gateway receives the requests and sends a `200 OK` response. It constructs a MESSAGE SIP request containing the From and To SIP addresses and the instant message contents. It sends the request to the SIP proxy server. When the server responds with a `200 OK` SIP response, the gateway sends RPC to the text client in Second Life an XML-RPC containing the XML-RPC channel and a message that indicates that the message was sent successfully. The text client sends a `200 OK` response and informs the user that the message was successfully sent.

3. *Receiving an instant message*

Second Life text client users can receive instant messages from other SIP users. When a SIP proxy server receives a MESSAGE SIP request addressed to the SIP address of a registered Second Life text client user, it will pass the request to the gateway. The proxy server is not aware that the destination user is logged in to Second Life, only that requests to that SIP address should be forwarded to the gateway. The gateway receives the request and attempts to retrieve the corresponding record from the database using the SIP address of the To header field. If found, it returns a `200 OK` SIP response. The gateway then creates an XML-RPC containing the XML-RPC channel identifier, the SIP address of the sender and the instant message contents and sends it to Second Life. The text client responds with a `200 OK` response and displays the sender's name and contents of the instant message to the user.

4. *Going offline*

When a Second Life user clicks the Unregister button, a HTTP request is sent to the `httpsipgateway_unregister` path on the gateway and contains the XML-RPC channel identifier.

The gateway retrieves the matching record from the database and sends a `200 OK` response back. It then sends a REGISTER request to the SIP registrar with the SIP address that was retrieved and an Expires header field with the value of `0`. When the gateway receives the `200 OK` SIP response from the SIP registrar it sends an XML-RPC to the Second Life text client telling it that the user is now offline. The text client responds with a `200 OK` response.

User instructions for the SIP text client are provided with the Second Life object and given here in Appendix C.

4.4. Use of gateway with other virtual worlds

The gateway was built with other virtual worlds (such as OpenSIM) in mind, but there are still several issues that need to be address to accommodate other virtual worlds.

The text client must be implemented in each virtual world. The implementation may differ from that of the Second Life text client, but at minimum it must:

- Send correctly formatted HTTP requests to the gateway and receive and interpret HTTP responses.
- Create a XML-RPC channel, receive and interpret messages on that channel and send correctly formatted responses.

From the gateway's perspective, the address to which XML-RPCs are sent is dependent on the virtual world being used. The virtual world text client is already required to send a parameter specifying the address to which XML-RPCs are to be sent. The gateway records this URL and uses it when sending XML-RPCs.

One problem that needs to be addressed is the format of the XML-RPCs. The gateway currently only sends XML-RPCs to Second Life and is using a Second Life specific format. If another format was required, it could be added to the gateway and a requirement could be added that the virtual world text clients specify which format is to be used.

5. Acknowledgements

The authors would like to acknowledge the financial support for this work provided by Telecom New Zealand Ltd. and the Global Network Interconnectivity Project (GIPI UOO KSE ICT 5023 TEC).

6. References

[1] Gartner press release 24/04/07 http://www.gartner.com/it/page.jsp?id=503861; Last accessed 7/07/08

[2] Second Life http://secondlife.com/; Last accessed 7/07/08

[3] OpenSim – OpenSimulator Project http://opensimulator.org/wiki/Main_Page; Last accessed 7/07/08

[4] JAIN SLEE – JAIN SLEE portal http://jainslee.org/; Last accessed 7/07/08

[5] MySpace http://www.myspace.com/; Last accessed 7/07/08

[6] Facebook http://www.facebook.com; Last accessed 7/07/08

[7] Bebo http://www.bebo.com; Last accessed 7/07/08

[8] Flickr http://www.flickr.com; Last accessed 7/07/08

[9] S.B.Lim, D. Ferry, JAIN SLEE Tutorial: Introducing JAIN SLEE http://jainslee.org/downloads/jainslee-tutorial-04.pdf; Last accessed 7/07/08

[10] S.B.Lim, D. Ferry, JAIN SLEE 1.0 Specification, Final Release, 2004

Appendix A

Paper Submitted to HICSS-42 track on Internet and the Digital Economy (Minitrack: Social Networks and Virtual Worlds for Work, Learning, and Play)

http://www.hicss.hawaii.edu/hicss_42/minitracks/in-snv.htm

Using JAIN SLEE to Extend Second Life Services for Flexible Social Networking

Nathan D Lewis, Hailing Situ, Melanie J Middlemiss and Martin Purvis
Department of Information Science
University of Otago
PO Box 56
Dunedin, NZ
ndlewis,hsitu, mmiddlemiss, mpruvis @infoscience.otago.ac.nz

Abstract

Computer mediated social networking tools provide a platform to allow users to communicate and interact with each other. Currently there are numerous web-based social networking tools, such as MySpace, Facebook, Bebo and Flickr, which provide a platform for online social networking. Virtual worlds, such as Second Life, are becoming increasingly popular as 3D platforms for social networking. To provide the most benefits to users, these platforms must be flexible and easily extendible through the addition and integration of value-added services. In this paper we take the example of Second Life, which is rather restrictive in enabling service extensions, and investigate how JAIN SLEE can be used to extend the services provided by Second Life - in particular to allow users to communicate from within Second Life with other users outside Second Life.

1. Introduction

Currently there are numerous web-based computer-mediated social networking tools available that provide communities or networks of users who share similar interests and activities methods for chatting (via voice or text), file sharing, etc. Some examples of these tools are MySpace, Facebook, Bebo, and Flickr [1-4]. As well as these 2D web-based tools, virtual worlds, such as Second Life, are becoming increasingly popular as 3D platforms for social networking.

As these social networking tools become more popular, users are becoming increasingly interested in being able to integrate their different social networks from the various tools, and to be able to perform multiple types of tasks within each social networking tool. For example, Flickr initially began as tool for users interested in sharing their photographs with users who shared a similar interest. Now it has been extended to provide users with a service to share video as well as still photographs. As another example, Facebook recently extended its services from a message board type functionality, to provide users with an interactive text chat function. Some of these web based tools also allow users to easily integrate their various social networks. For example, a user may have a twitter account with a number of people following them, and a blog with a different network of people following this. Using RSS feeds, the twitter feeds can be easily integrated into their blog.

Second Life is slightly different in that it is not Web based; rather client software is used to connect to the application-specific Linden Lab Second Life servers. Second Life allows users to easily interact with other users who are also in Second Life. It also offers some limited ability to render an image of a web page on an object in the Second Life virtual world. However, Second Life is a proprietary and closed platform and is restrictive in enabling service extensions or interaction with external services. It provides only two protocols for communicating between objects inside Second Life and external servers – HTTP and XML-RPC protocols. In order to make Second Life more accessible to outside communication, some mechanism is needed to facilitate interaction with external services.

This is where JAIN SLEE comes in. JAIN SLEE, sometimes referred to simply as JSLEE, gets its name from Java APIs for Integrated Networks (JAIN) Service Logic Execution Environment (SLEE). The SLEE provides a standard integration environment for multiple network resources and protocols, refer to Figure 1.

In this paper we describe how JAIN SLEE can be used to extend the services provided by Second Life – in particular to allow users to communicate from

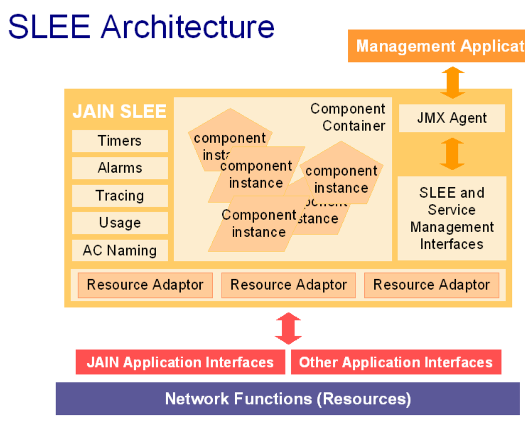


Figure 1: SLEE Architecture [5]

within Second Life with other users outside Second Life. Section 2 discusses how JAIN SLEE can be used to extend Second Life with new value-added services and the relevant issues involved. The detailed implementation is presented in Section 3. Section 4 describes the deployment of the application and demonstration results. Concluding remarks and proposed future work are given in Section 5.

2. Service extension of Second Life

This paper describes how JAIN SLEE can be used to extend the services available in Second Life. When using Second Life, users communicate with other users that belong to their Second Life social network. However, many people will belong to multiple social networks and may want to integrate these. For example, they may be in Second Life and want to communicate with friends in their "real world" social network. As an example of extending Second Life services, we consider the extension of Second Life communication services to allow Second Life avatars to communicate with users outside Second Life.

2.1 JAIN SLEE

JAIN SLEE is a common middleware platform. With different resource adaptors, JAIN SLEE application servers can receive different kinds of requests, process the requests and send responses back. The benefits of using JAIN SLEE are as follows:

- Using existing resource adaptors can provide an easy start to application development. In this project, we use the HTTP and SIP resource adaptors.
- Developers can combine different servers (e.g. HTTP server and SIP server) in the same application.
- Developers do not need to learn different server programming models to develop applications. They only need to know the JAIN SLEE programming model and are able to develop different kinds of applications according to this model.

2.2 Protocol issues for Second Life

Currently, Second Life provides HTTP and XML-RPC protocols to communicate with external systems.

The HTTP protocol is a standard protocol used for Web-based applications. In Second Life, HTTP requests can be sent to external Web servers. Second Life can only process the response to a HTTP request; it cannot accept incoming HTTP requests.

XML-RPC is a standard messaging protocol for invoking operations on remote machines. XML data is sent via HTTP to the remote system to be handled. Second Life can only receive external communications via XML-RPC. The communication must be initiated by an external server.

2.3 SIP protocol

SIP (Session Initiation Protocol) is an open standard signalling protocol used for establishing sessions in an IP based network, such as voice and video calls over the Internet. There are other signalling protocols for VoIP, such as H.323 [6]. While the SIP protocol has been standardised and governed primarily by the IETF (The Internet Engineering Task Force) [7], the H.323 protocol has been traditionally more associated with the ITU-T (International Telecommunication Union) [8]. However, the two organisations have endorsed both protocols in some fashion [9]. A number of VoIP applications have adopted SIP as the session setup protocol because it is an open standard and appears to be gaining a more global popularity.

SIP provides registrar and proxy functions for an application. Registrar services allow users to register their SIP address and their current locations to the registrar server. Proxy services can route requests to a user's current location by looking up their SIP address from the registrar service.

SIP also provides an extended standard for instant messaging and presence services. Users can enquire about a friend's status, and presence information is used as buddy status in IM clients. It is this messaging service that we use to allow Second Life and real world users to communicate.

2.4 Communications services for Second Life avatars

In Second Life, users can find and communicate (via text and chat) with other users who are also in Second Life. We want to extend the communication services to allow users in Second Life to communicate with users who are not in Second Life, but in the real world.

In the real world, VoIP is becoming more popular. Many people have a SIP phone, MSN or Skype installed in their computers. They want cheap and instant communications through the Internet to keep in touch with friends in their social networks. The current communications services in Second Life allows users to easily communicate with other users in Second Life, but is rather restricted when it comes to communicating between Second Life and users outside Second Life. For example, users are able to send instant messages to the email account of a user that is not currently logged in to Second Life. Also, there are several vendors who offer the ability to send text messages to cell phones of real world users who are outside of Second Life. However, in this work we investigate how communications can occur between users in Second Life and any other user with a SIP based application.

Note that the SIP protocol is essentially an open standard and not a custom protocol, which is why it is ideal for opening up platforms, such as Second Life. It is easily extended and there are a number of free SIP clients available for use [10].

This project will implement a service that allows Second Life avatars to register with the SIP registrar server and send instant messages to a SIP text client outside of the Second Life virtual world. At the same time, a SIP text client will also be able to send messages to a Second Life avatar. The HTTP/SIP gateway can convert the message between the HTTP protocol and the SIP protocol. In this application, if a user registers their SIP address to the SIP registrar server, no matter where they are (Second Life, real world, Opensimulator virtual world [11], etc.) the SIP proxy will be able to locate them and forward messages to them.

3. Implementation of the new service

The new service provides communication between Second Life avatars and SIP clients using HTTP and XML-RPC protocols on the Second Life server and using SIP protocols on a SIP server. The Second Life avatars can register their presence and send or receive instant messages to SIP clients which could, for example, be running on a computer, PDA or mobile phone.

A Second Life agent can only send HTTP requests to external servers and receive XML-RPC requests from external servers, and the SIP registrar and proxy services can only send and receive SIP requests, we must implement a HTTP/SIP gateway to convert between the two systems. The main architecture of this gateway service is shown in Figure 2.

Second Life clients and SIP clients can communicate with each other via the Second Life server, HTTP/SIP gateway and SIP server. This is discussed in more detail in the following subsections.

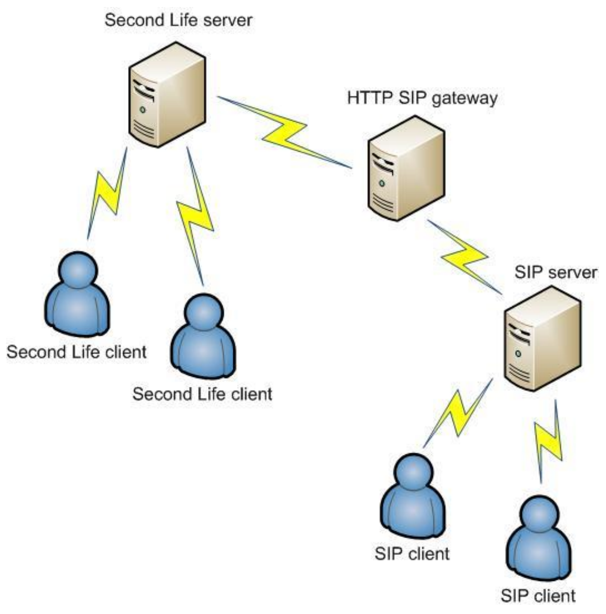


Figure 2: Architecture of New Service

3.1 HTTP/SIP Gateway

The main implementation is the HTTP/SIP gateway. This gateway is implemented based on Rhino—a JAIN SLEE platform developed by OpenCloud, Ltd [12]. Two resource adaptors (RAs) are used: HTTP RA and SIP RA. The architecture is shown in Figure 3.

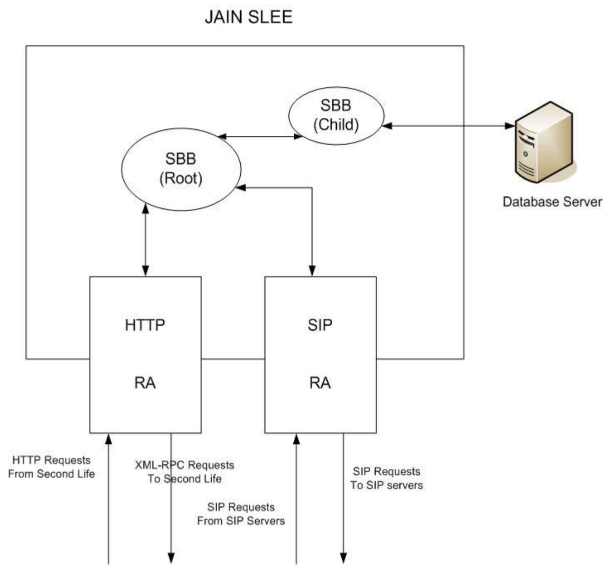

JAIN SLEE
SBB (Child)
SBB (Root)
Database Server
HTTP RA
SIP RA
HTTP Requests From Second Life
XML-RPC Requests To Second Life
SIP Requests From SIP Servers
SIP Requests To SIP servers

Figure 3: HTTP/SIP Gateway in JAIN SLEE

In JAIN SLEE, the purpose of a resource adaptor is to adapt particular resources to the requirements of the SLEE. Resources are entities that represent and interact with other systems outside the SLEE, such as network devices, protocol stacks, directories, databases, or as in this case – a SIP based service [13, pp. 253].

The HTTP RA provides a generic HTTP interface for SLEE services. The RA is bidirectional; applications can receive incoming HTTP requests and can also initiate outgoing HTTP requests. Request methods GET, HEAD and POST are supported [14].

The SIP RA supports RFC 3261 functionality and some SIP extensions, such as the INFO method (RFC 2976), the UPDATE method (RFC 3311), the MESSAGE method (RFC 3428), and so on (see [15] for a list of SIP specifications). It provides an interface based on JAIN SIP 1.1 with some proprietary extensions for SLEE applications [16].

In Second Life, an object uses the `llHTTPRequest` Linden scripting language function to send the object's information to the HTTP RA and creates an XML-RPC channel for receiving requests coming from the gateway. The information passed to the gateway from the Second Life object includes user information and the XML-RPC channel number which the object will be listening on.

When the HTTP RA receives an HTTP request, the SLEE will trigger the relevant event method in the appropriate SBB (Service Building Block). The SBB will process these requests, prepare relevant SIP requests, and send these requests (such as REGISTER or MESSAGE requests) to the SIP registrar server or SIP proxy server via the SIP resource adaptor. The SIP servers will then process the requests or send the requests to the SIP clients.

In the other direction, when a SIP client sends a request to the SIP server, the SIP server will send this to the SIP RA of the gateway. The SLEE then triggers the relevant request method in the appropriate SBB to process the request, prepare the XML-RPC request, retrieve the Second Life XML-RPC channel number from its database, and send the request via the HTTP resource adaptor. When the Second Life object receives the request from the gateway, it will display the relevant information such as the SIP response or instant message.

3.2 Service Building Blocks (SBBs)

In the JAIN SLEE architecture, a SLEE defines how an application can be composed of components. These components are known as Service Building Block (SBB) components. Each SBB component defines the event types that it subscribes to and has event handler methods that contain application code that processes events of these event types. The SBB component also declares the SBB local interface of the SBB component. The SBB local interface specifies the methods of the SBB component that may be invoked synchronously. The SBB component may have zero or more child SBB components. The SBB component specifies its child SBB component relations [13, pp. 23].

In Figure 3, the root SBB defines the following event handler methods:

- Process HTTP requests.
    - Get the information of the Second Life object including XML-RPC channel key and SIP URI;
    - Save object information to the database;
    - Create REGISTER or MESSAGE SIP requests;
    - Send SIP requests to the SIP server through SIP RA.
- Process SIP requests.
    - Retrieve the Second Life object information and XML-RPC channel key from the database;
    - Create the XML-RPC data and send it to the Second Life object;

The child SBB in Figure 3 is used for communicating with the database. The child SBB implements save and retrieve database functionality.

4. Demonstration of functionality

The gateway is deployed on Rhino 1.4.5-2 running on one machine. The Second Life client is installed on another machine, and a SIP based instant messaging client is deployed on a mobile phone. The SIP registrar and proxy servers, developed based on Rhino 1.4.5-2 by OpenCloud, are deployed on another machine. The resulting functionality is described in the scenario below.

Sally goes online in Second Life. She starts her Second Life text client which sends her SIP URI, `sip:sally@opencloud.com`, to the gateway using an HTTP request. The gateway records her text client's Second Life XML-RPC channel key and sends a REGISTER request to the SIP proxy server. The request contains Sally's SIP URI and indicates that any request for Sally should be sent to the gateway's address. The SIP proxy replies with a `200 OK` message. The gateway then sends an XML-RPC to Sally's text client to tell it that her SIP URI has been successfully registered.

Richard turns on his SIP-enabled phone. It registers his SIP URI, `sip:richard@realworld.com` with his SIP proxy server. To send a message to Sally, he needs only to know her SIP URI – he does not need to be aware of her current location. He addresses a text message to `sip:sally@opencloud.com` and presses the send button. The MESSAGE request is passed to his SIP proxy server. The proxy server discovers Sally's location (the address of the gateway) and forwards the request. The gateway reads the SIP URI from the request and retrieves Sally's Second Life object information from its database. It creates an XML-RPC with the contents of Richard's message and the XML-RPC channel key of Sally's text client and sends it to Second Life. Sally's Second Life text client shows her the contents of Richard's message.

In Second Life, Sally then types a message to `sip:richard@realworld.com` which is read by her Second Life text client. It creates an HTTP request with the address and message and sends it to the gateway. The gateway creates a SIP MESSAGE request with the address and message and sends it to the SIP proxy server. The SIP proxy server knows the address to send the message directly to Richard's SIP phone.

5. Conclusions

Creating a more flexible communication environment is useful for social networking. We can use existing communication paths of social network platforms and develop gateways to allow different communication clients to communicate with each other.

We have created a new service which allows SIP users to communicate with each other, no matter whether they are within Second Life or using real-world SIP applications. We have demonstrated the general nature of this approach, which means that we can extend our new service to provide other services, such as a presence service so that Second Life users can know their friends' current status.

JAIN SLEE is a good development platform for this type of service extension, because developers can use existing resource adapters to receive and send requests and responses, ignore the transactions and threading processes which are handled by the SLEE and focus on the logical development of the applications. In this way, developers can develop their applications faster and more easily.

6. Acknowledgments

The authors would like to acknowledge the financial support for this work provided by Telecom New Zealand Ltd. and the Global Network Interconnectivity Project (GIPI UOO KSE ICT 5023 TEC).

7. References

[1] MySpace http://www.myspace.com/, last accessed June 15, 2008

[2] Facebook http://www.facebook.com, last accessed June 15, 2008

[3] Bebo http://www.bebo.com, last accessed June 15, 2008

[4] Flickr http://www.flickr.com, last accessed June 15, 2008

[5] S.B.Lim, D. Ferry, *JAIN SLEE Tutorial: Introducing JAIN SLEE* http://jainslee.org/downloads/jainslee-tutorial-04.pdf, last accessed June 15, 2008

[6] International Engineering Consortium (IEC) http://www.iec.org/online/tutorials/h323/index.html, last accessed June 15, 2008

[7] The Internet Engineering Task Force (IETF) http://www.ietf.org/, last accessed June 15, 2008

[8] International Telecommunication Union, Telecommunication Standardization Sector (ITU-T) http://www.itu.int/ITU-T/, last accessed June 15, 2008

[9] Wikipedia Session Initiation Protocol entry http://en.wikipedia.org/wiki/Session_Initiation_Protocol, last accessed June 4, 2008

[10] Wikipedia entry with list of SIP software http://en.wikipedia.org/wiki/List_of_SIP_software, last accessed June 15, 2008

[11] OpenSimulator http://www.opensimulator.org, last accessed June 15, 2008

[12] OpenCloud Ltd http://www.opencloud.com, last accessed June 15, 2008

[13] S.B.Lim, D. Ferry, *JAIN SLEE 1.0 Specification, Final Release*, 2004

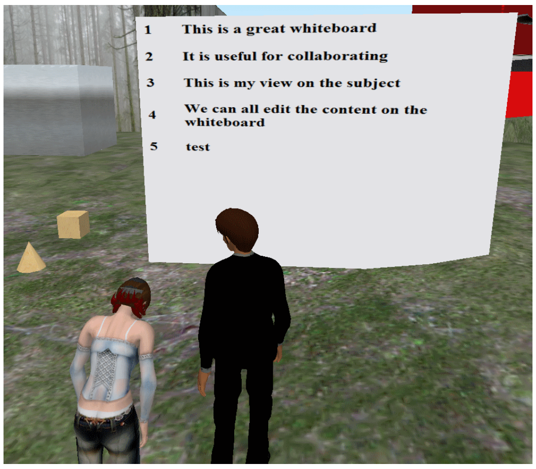
[14] OpenCloud HTTP RA document

[15] IETF Tools SIP Status page http://tools.ietf.org/wg/sip/, last accessed June 15, 2008

[16] OpenCloud SIP RA document

Appendix B

User instructions for the Second Life Whiteboard object.


1 This is a great whiteboard
2 It is useful for collaborating
3 This is my view on the subject
4 We can all edit the content on the whiteboard
5 test

To use this whiteboard you must be using *version 1.19.1(4)* or above of the Second Life client.

Click the Play button in the media control bar in your user interface.

The whiteboard must be deeded to the group that controls the land parcel it is placed on.

Once deeded, touch the whiteboard to restart it.

The whiteboard listens to a chat channel which can be set in the Settings notecard. To issue a command, type the command in local chat. If, for example, the whiteboard is listening on chat channel 1 and you wish to add some text to the bottom of the whiteboard then you would type:

```
/1 ADD Some text
```

The following commands are accepted by this whiteboard:

- `ADD <text>`: Adds <text> at the bottom of the whiteboard
- `REPLACE <linenumber> <text>`: Replaces the text at <linenumber> with <text>
- `DELETE <linenumber>`: Deletes the text at <linenumber>
- `MOVEUP <linenumber> <distance>`: Moves the text at <linenumber> up by <distance> lines
- `MOVEDOWN <linenumber> <distance>`: Moves the text at <linenumber> down by <distance> lines
- `RESET`: Wipes the whiteboard clean
- `GET <linenumber>`: The whiteboard will say the text from <linenumber> so that it can be copy and pasted
- `KEY <string>`: Tells the whiteboard to use <string> as its database key. By specifying <string> you can create a persistant whiteboard. Omit <string> to tell the whiteboard to use its object key as its database key (the object's key will change when the object is rezzed). The whiteboard uses its object key as its database key by default.
- `CHANNEL <channel number>`: Tells the whiteboard to start listening for commands on the new channel number.
- `SETTINGS`: Re-load the settings from the settings notecard.

Appendix C

User instructions for the Second Life SIP text client object.

To use this SIP text client:

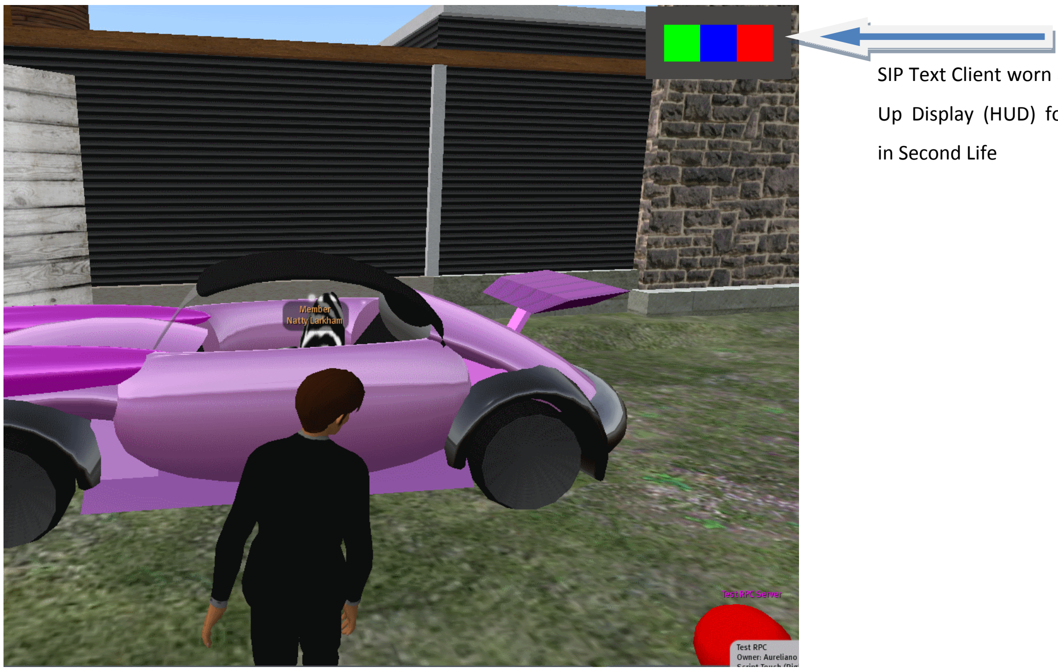
1. Wear the text client object.
2. Edit the Settings notecard and enter your SIP address and other details.
3. Edit the Friends notecard and enter names and SIP addresses of people you wish to send instant messages to.

To send or receive SIP instant messages, you must first register with your SIP Registrar. To do so, click the green button. The text client will tell you when you have successfully gone online.

Once online, any messages sent to your SIP address will be received by the text client and displayed to you in local chat.

To send a SIP instant message you must first select the recipient. Clicking the blue button will cause a dialog box to appear containing the names of the people in your Friends notecard. Click a name to mark it as the active recipient. Once selected, any messages you type on the chat channel specified in the Settings notecard will be sent to that recipient. Click the blue button again to select a new recipient.

To go offline, click the red button.



SIP Text Client worn as Heads Up Display (HUD) for avatar in Second Life