VoIP Application Development using SIP protocol

Dee Milic
Dong Zhou
Hailing Situ

The Information Science Discussion Paper Series

Number 2008/01
March 2008
ISSN 1177-455X

UNIVERSITY of OTAGO

Te Whare Wānanga o Otāgo

VoIP Application Development using SIP protocol

Dee Milic
Dong Zhou
Hailing Situ (Supervisor)

07/08 Summer Work for
Global Network Interconnectivity Project

Tuesday, 4 March 2008

1. Project Objectives

This aim of this project was to the explore JAIN SLEE standard 1.1 programming model and SIMPLE (Sip for Instant Messaging and Presence Leveraging Extensions) protocols, developing a Voice over Internet Protocol (VoIP) application with functions that include making a phone call, instant messaging to peers, and at the same time providing users with buddy list information of their peers. The JAIN SLEE platform RhinoSDK 2.0 (developed by OpenCloud) was to be used and an example application that is included with RhinoSDK 2.0 was to be extended. During the project the phone call functionality was scoped out of the project and the focus was set on implementing the instant messaging and presence functionality. This report describes the functions that have been implemented on the server side and client side of this VoIP application.

2. VoIP

Voice over Internet Protocol (VoIP) is an internet technology that allows for transmission of voice conversations using a broadband internet connection instead of a regular (or analog) phone line. It is becoming increasingly popular because of the cost advantages to consumers over regular telephone networks.

In order for a computer to find a remote device and negotiate the means by which media can flow between the two devices, a number of protocols must be employed. This process involves protocols that are central to the system and are called call-signalling protocols. The most popular call-signalling protocols include Session Initiation Protocol (SIP), H.323, Registration Admission Status (RAS), Telephony Routing over IP (TRIP), DNS, Telephone Number Mapping (ENUM) and others.

Session Initiation Protocol (SIP) is an application-layer signalling protocol for creating, modifying and terminating sessions with one or more clients. It is a request-response protocol that can use both TCP and UDP protocols to connect to SIP servers and other SIP endpoints. SIP protocol advantages include simplicity and extensibility, mobility, bi-directional authentication and capability negotiation.

3. JAIN SLEE

A Service Logic Execution Environment (SLEE) is a well known concept in the telecommunications industry. A SLEE is a high throughput, low latency event processing application environment. JAIN (Java APIs for Integrated Networks) SLEE is the Java standard for SLEE. JAIN SLEE is designed to allow implementations of the standard to meet the stringent requirements of communications applications, such as network signalling applications[1]. The JAIN SLEE programming model is protocol agnostic, component based that supports software engineering best practises such object orientated programming and transactions. The JAIN SLEE programming model has been designed to simplify the work of the application developer, eliminate common programmer errors and ensure that robust services can be developed rapidly[2].

[1] http://www.opencloud.com/technologies/jain-slee.html

[2] http://www.opencloud.com/products/rhino-core-platform/jain-slee-container.html

The Otago Next Generation Networks and Services (ONGENS)[3] Test Bed has access to two of the leading JAIN SLEE platforms: Rhino (www.opencloud.co.nz) and Mobicents (www.mobicents.org). For this project Rhino, the JAIN SLEE software platform developed by the OpenCloud, has been used.

4. VoIP Server

In this project, the source code for the SIP example application that is included with RhinoSDK 2.0-Developer-preview-win-beta version has been utilised and modified for the VoIP server. This SIP example includes proxy, registrar and presence services which support client subscribe functionality for communications applications. Currently this provides functionality whereby users can subscribe to their friends and see whether they are on or offline. In this project the objective was to extend the presence service to include publish functionality for users to present their current status, such as “Busy” and “Gone for lunch”. Also server provides status updates to friends who have subscribed them.

In this project two functions have been added to the SIP example application– publish with notification and Instant Messaging. The existing registrar function has been modified. The “Registration” object which used to record each registered user’s information has been modified to add more entries, such as current states, expiry time of these states and entity tags for the publishers. The existing subscription function has also been modified to accommodate these new added functions. The “Subscription” function processes requests from subscribers who want to subscribe to others’ states and sends the states of the clients subscribed to the subscriber. The “Publish” function processes requests from a publisher who wants to notify the server of the new state it has, and sends notifications to other clients who are interested in receiving this publisher’s state. Instant Messaging is used to send instant messages between clients.

4.1. Subscription

Once the server receives a subscribe request from the client, it first validates this request. It checks whether this request is the right type for the subscription function and then checks its expiry time. If this request is not for this function or the expiry time is too short, it sends a response back to the client to indicate this request is a bad request and skips the remaining steps.

After validation, the server sets up a new dialog and stores this dialog into the memory. It also gets the Activity Context Interface (ACI) for the client being subscribed in this request and stores the subscriber’s information into the ACI. It sends an OK message back to the subscriber to indicate its request has been accepted and a notification request to it with the state of the client it subscribed.

If the expire time is 0, it means the subscriber does not want to subscribe this client anymore, so the server sends an OK response and a notification to the subscriber with the state of the client it subscribed and deletes the dialog and the information of this subscriber stored in the ACI.

[3] http://www.ongens.otago.ac.nz

4.2. Publish

Once the server receives a publish request from the client, it first validates this request. It checks whether this request is the right type for the publish function and then checks its expiry time. If this request is not for this function or the expiry time is too short, it sends a response back to the client to indicate this request is a bad request and skips the remaining steps.

It also checks whether the request has an entity tag header. If the value of the entity tag in the entity tag header does not match with the one it stored for this publisher, it sends a response back to the client to indicate this request is a bad request and skips the remaining steps.

After examining the request, it sends an OK response with a new entity tag value stored in the "Registration" object back to the client to indicate the request that has been accepted. It then checks whether the request has a body and stores the new states contained in the body into the "Registration" object. It also checks whether there are other clients subscribing this client and sends notifications to the subscribers with the new states of the publisher.

4.3. Instant Messaging

Once the server receives a message request from the client, it first checks whether the client who will receive this message has registered on the server. If no such client has been found, the server sends a response back to the sender to indicate that the destination has not been found and finishes processing the request.

After finding the client, the server looks up this client's physical IP address stored in the "Registration" object and forwards the message to the client according to this IP address. If receiving the receiver's response, the server sends a response to the sender to indicate the message has been successfully sent and received by the receiver, otherwise sends another response to indicate the message has been received and sent by the server.

4.4. Summary of VoIP server functionality

- Process registration request and send response
- Process subscribe request and send response and notification
- Process publish request, change the client's information and send response and notification
- Process messaging request, forward to the receiver and send response

5. VoIP Client

The objective of the client side part of this project was to develop a VoIP client that would use SIP protocol to connect to the server and send and receive data to and from this server. There were three main components involved – developing a graphical user interface (GUI), developing a SIP layer class that was to be controlled or used from the GUI, and developing an XML component that was to be used for storing the data to and retrieving the data from the server.

5.1. Graphical User Interface (GUI)

The client GUI includes a number of screens that the user will use to control this application. The login screen is the first screen the user will see and allows for this user to send the registration information to the server. Once the user logs in, the main menu with the contact list will become available. This main screen loads the contacts from the XML file and subscribes to them on the server through the SIP layer. Once the user subscribes to these contacts, their status (Offline/Online/Away etc) will become available and the user will then be able to send a message them. The main menu also allows the user to update their status, log out from the server, change server options, add or delete contacts and send a message to other users who are not on the contact list.

5.2. SIP Layer

The SIP protocol has been implemented in Java using JAIN SIP. JAIN SIP is the standard Java interface to the SIP signaling stack for both desktop and server applications and it supports the SIP functionality described in the RFC 3261. Advantages of JAIN SIP include service portability, network convergence and allowing abstracted services direct access to network resources and devices to carry out specific actions or functions.

The SIP layer class in the VoIP client uses the NIST SIP stack library JainSipApi1.1 in order to provide methods that allow the user to register on the server, subscribe to or unsubscribe from other users (telling the server whether this user wants or does not want to know the state of specified user(s)), publish this user’s status message (telling the server of changes in this user’s status) and unregistering from the server.

5.3. XML

The both GUI and SIP layer use XML for data storage and retrieval. The GUI uses XML for storage when the user enters information that should be saved, as well as retrieving information for display purposes. The SIP layer uses XML to store or retrieve data that is sent to or received from the server. The library used for manipulating and parsing XML is dom4j-1.6.1 (an open source XML framework that integrates with DOM and SAX with full XPath support).

5.4. Summary of VoIP client functionality

Graphical User Interface:

- Allows users to log in/log out by registering on server
- Allows users to add/delete contacts
- Allows users to change server options
- Allows users to change their status
- Allows users to send messages to other users

Client SIP Layer:

- Sends register and un-register information to the server
- Receives response/requests from the server
- Sends subscribe/unsubscribe information to the server

- Sends publish information to the server
- Sends/receives message requests to the server

6. Summary

During this project knowledge of Rhino 2.0 has been gained and SIP related protocols including register, proxy, publish, subscribe and message have been investigated. This has resulted in an application demonstrating VoIP presence and instant messaging. In this application users can configure their SIP URL and server address, register their SIP URL to the server, change their current state, display their friends states, and send instant messages to their friends. Basic functionalities have been implemented and demonstrated. This application can be extened in the future to enhance client side and server side functionality.

7. References

Understanding VoIP, http://www.packetizer.com/ipmc/papers/understanding_voip/

RFC 3261 – SIP: Session Initiation Protocol, http://tools.ietf.org/html/rfc3261#section-16

Java API for SIP signalling, https://jain-sip.dev.java.net/

An Introduction to the Jain SIP API, http://dev2dev.bea.com/pub/a/2007/10/introduction-jain-sip.html?page=1

JAIN SIP 1.1 API Specification. http://dpnm.postech.ac.kr/mcs/api/jain/overview-summary.html

Lim, S. and Ferry, D. (2004) **JAIN SLEE 1.0 Specification, Final Release**

Niemi, A. (2004) Network Working Group, **Session Initiation Protocol(SIP) Extension for Event State Publication**, http://tools.ietf.org/html/rfc3903

Rosenberg, J. *et al.* (2002) Network Working Group, **SIP: Session Initial Protocol**, http://tools.ietf.org/html/rfc3261

Roach, A. B., Campbell, B. and Rosenberg J. (2006) Network Working Group, **A Session Initiation Protocol (SIP) Event Notification Extension for Resource Lists**, http://tools.ietf.org/html/rfc4662