Exploring Voice Over IP with Java in an Upper Division Networking Course

Jeannie R. Albrecht, s412270@gettysburg.edu
Rodney S. Tosten, rtosten@gettysburg.edu
Gettysburg College & CNAV Systems, Inc.
Gettysburg, PA 17325
(717) 337-6630

ABSTRACT

This paper introduces the concepts needed to implement Voice over IP applications with Java. Topics covered include Session Initiation Protocol, Real Time Protocol, Java Media Framework, and the development of an Internet-based netphone. It also discusses techniques for incorporating these subjects into an upper division Networking course in an undergraduate curriculum.

Keywords: *Voice over IP, Session Initiation Protocol, Real Time Protocol, Networking, Java Media Framework*

1. INTRODUCTION

Communication has been a fundamental aspect of society since the birth of the first civilizations. People have always had a need to send and receive information in various forms. As technology continues to spread to all corners of the world, the demand for quick and reliable communication, anywhere, anytime, and with any device is constantly increasing.

The invention of the Internet, or perhaps more importantly, the growing popularity of the World Wide Web, has already redefined the way people communicate and keep in touch with one another. Internet applications such as email and AOL's Instant Messenger offer simple, reliable, and inexpensive ways to interact with friends and family members from any personal computer. These technologies provide users with text messaging capabilities, which essentially are alternatives to letter writing and mailing items via the postal service.

The next logical step is to redefine voice communication. In the past, voice messaging over the Internet has been unreliable and of poor quality due to limitations in hardware. However in recent years, personal computer hardware specifications and network bandwidth capacities have increased significantly, making high quality audio and video streaming over the Internet possible.

One very popular area of development on the Internet today is a specific type of audio streaming, commonly called Voice over IP (VoIP). Applications implementing VoIP allow users to physically talk to each other over the Internet using only their computer's microphone and speakers. Many other protocols have aided in the development of these applications, including Session Initiation Protocol (SIP) and Real Time Protocol (RTP). VoIP based technologies, like Internet telephones called netphones, will continue to play a large role in the future of computing, and therefore also in computer science education. VoIP, SIP, and RTP incorporate concepts taught in an undergraduate Networking course. Thus, we believe that VoIP should be featured in a computer science curriculum.

2. WHAT IS SIP?

Jonathan Rosenberg and Henning Schulzrinne first developed SIP, which is formally defined in RFC 2543, at Columbia University in the late 1990s [2]. SIP is an application-layer control protocol for creating, modifying, and terminating sessions. These sessions include Internet multimedia conferences, Internet telephone calls, and multimedia distribution. Designed to be independent of lower layer protocols, it can be implemented using either TCP or UDP at the transport layer.

SIP essentially handles the initial "hand shaking" that must occur between two parties prior to engaging in any kind of communication. The benefit of using SIP in VoIP applications is that it keeps track of all registered users' locations and current contact information, including email addresses, telephone numbers, and cellular phone numbers. Thus the VoIP application only needs to know *who* to call instead of *where* to locate them. SIP's basic philosophy is "Find me, follow me." This means that as a user's location changes, or as a particular method for communication becomes unavailable for various reasons, SIP records the new information. Therefore, when one user needs to contact another, SIP knows exactly where and how to direct the call.

3. USING SIP

Although many companies now offer SIP enhanced hardware and software solutions, one of the leading software providers is a New Jersey based company called dynamicsoft (http://www.dynamicsoft.com). The chief scientist at dynamicsoft is Jonathan Rosenberg, and during the past few years the company has been very active in the production of commercially available SIP software development packages.

Three components involved in dynamicsoft's software are the User Agent, Location Server, and Proxy Server. Each element has a specific role in the implementation of SIP. The details of each of these parts are described in the following paragraphs.

3.1 User Agent

The User Agent is the part of the software that the user actually sees. A sample version of a User Agent is available for downloading from dynamicsoft in either Java or C++. All of the typical actions associated with a phone call are defined within methods in the User Agent. These include functions like call, hang-up, and answer. The User Agent is also the section that gives developers the most freedom to make their software as flexible and robust as desired. There are three levels of development available from dynamicsoft, namely High, Mid, and Low, allowing programmers to choose how much control they have. High level coding hides a lot of the technical connection details involved with SIP applications, however it has somewhat limited functionality overall. Low level coding forces programmers to handle most technical details specifically, but offers more control and power. Mid level coding is a compromise between the other two levels.

3.2 Location Server

The Location Server's role is to keep track of users' locations at all times. Location in this sense is not so much a physical place, but rather a communication address, such as an email address, telephone number, or cellular phone number. It basically is a table in a database that contains all valid users, their locations at any given time, and how long they will be at the specified location. As a user moves from place to place, or more accurately, from one location to another, the table is updated accordingly. If a user attends an important meeting, for example, and can no longer be reached via a telephone, the location server would be informed of the change, and an email address would become the new location for that user. When the meeting ends, the user would reregister their telephone number with the Location Server, updating their location once again.

3.3 Proxy Server

The Proxy Server is the component that acts as the connection between the Location Server and the User Agent. It handles requests from the User Agent, such as registrations or call initiations, and retrieves the needed information from the Location Server. To perform these functions, SQL queries and responses are used to execute the appropriate actions in the database. When initiating a call, all stages of the connection hand shaking also go through the Proxy Server. In some sense, the Proxy Server acts as a liaison between the two parties in the call until the call is completed.

3.4 Registration

Before placing or receiving any calls, users must register with the Location Server. To complete a registration, the User Agent notifies the Proxy Server that it needs to register. The Proxy Server sends a registration request (Figure 1) to the Location Server, who sends a message back to the Proxy Server when the action is complete. The Proxy Server then gives the User Agent the permission to participate in calls.

```
*** SENDING *** - to: /138.161.234.11/5060 via UDP
REGISTER sip:rod@gettysburg.edu:5060 SIP/2.0
Via: SIP/2.0/UDP 138.161.234.11:7300
From: <sip:rod@138.161.234.11>;
        tag=c0-a8-2-42-b2411169-fc00
To: sip:rod@gettysburg.edu
Call-ID: 953072942080@138.161.234.11
CSeq: 15608 REGISTER
Contact: <sip:rod-pc@gateway.com>; expires=30000
```

Figure 1 – Registration.

3.4 Typical Call Flow

Once a user is registered with the Location Server, that user has the ability to make and receive calls to and from other registered users. To place a call, the caller simply specifies the name of the callee (user being called) in the User Agent. The caller's User Agent contacts the Proxy Server, and the Proxy Server attempts to find the callee in the Location Server. If the callee is found, an invite message generated by the User Agent is sent to the callee. Once the callee accepts the incoming call, a "200 OK" response is sent back to the caller. The SIP software's job is complete (Figure 2) [2].

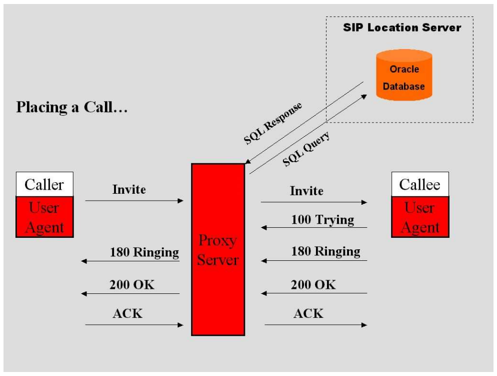


Figure 2 - SIP Call Flow.

4. WHAT IS RTP?

RTP is a protocol that provides end-to-end delivery services for data with real-time characteristics. Formally defined in RFC 1889, RTP is most often used when working with interactive audio and video on the Internet. One important feature of the protocol is that it does not guarantee quality-of-service to any degree. Reliability and quality are beyond the realm of the protocol's specifications. Applications that implement RTP rely on lower layer services, such as TCP or UDP, to handle these issues.

For most real time streaming, UDP is the transport protocol that produces the best results. Although UDP is unreliable in that it does not ensure the successful delivery of packets, it is relatively quick. This is very important when streaming media over the Internet because there needs to be a constant stream of data being transmitted. With more reliable but slower transport protocols, like TCP, lost packets are retransmitted, which leads to congestion and pauses in transmission. Using UDP, an occasional lost packet may cause a short break in the sound. However, with TCP, waiting for packets to be retransmitted every time one is lost creates long pauses that severely decrease the quality of the transmission. UDP is clearly a better choice in this situation.

There are many issues that developers must consider when working with RTP based programs. Currently, the Internet cannot yet fully support the high demand for real-time applications. High-bandwidth services using RTP, such as audio and video conferencing, have the potential to seriously degrade the quality-of-service of other network services if used excessively. Severe congestion and other problems result when all of the available bandwidth on a given network is being used for media streaming. In these cases there is no bandwidth left to fulfill the simpler requests, such as email retrieval or web surfing. Programmers must take precautions to limit excessive and accidental bandwidth usage.

5. JAVA MEDIA FRAMEWORK

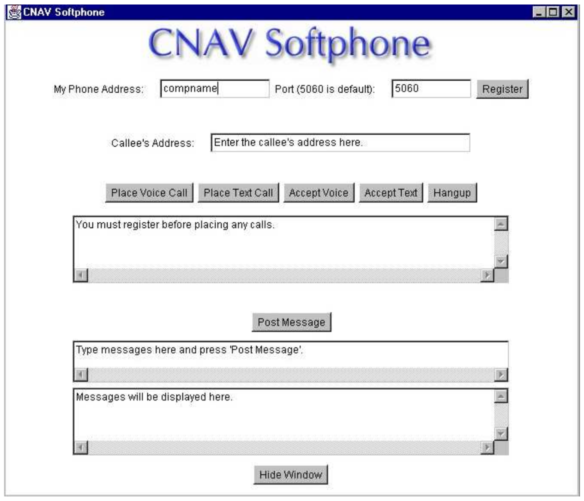
In response to the growing popularity of RTP and VoIP applications, Sun released a new technology that enables audio, video, and other time-based media to be added to Java applets and applications [4]. This new package is called Java Media Framework (JMF), and it acts as an extension to the standard Java Development Kit. JMF has the ability to capture, playback, transmit, encode, and decode multiple media formats. It makes the implementation of RTP much simpler, hiding many of the technical details from Java developers.

When dealing with audio and video streaming, it is necessary to compress and decompress the data in multiple formats. Any technology that performs this type of compression and decompression is called a codec. In VoIP telephony, G.711, which is also called ULAW, is the most popular and widely accepted codec. Another popular codec that is more frequently seen in the compression of sound and music files is MPEG. Fortunately, JMF offers support for both of these formats. Various code samples demonstrating the more common uses of JMF can be found on Sun's website.

6. CNAV SOFTPHONE

After learning about SIP, RTP, and JMF, which are all topics that fit easily into a Java based Networks course, students have the knowledge needed to begin creating their own VoIP applets and applications. The netphone project described in the following paragraphs incorporates all of these technologies into one Java applet. However due to its complexity, it probably would be more appropriate as a final project rather than a weekly assignment. The netphone, dubbed the "CNAV Softphone," is a project that started as a summer internship at a software development company called CNAV Systems (http://www.cnavsystems.com). The Softphone was designed to be part of an Internet based college portal system. Its purpose is to give campus members the ability to contact one another anytime, anywhere, and from any computer or telephone.

The CNAV Softphone (Figure 3) is a Java applet that runs within a web-based portal. Upon logging into the portal, the Softphone automatically registers with the Location Server and updates the contact information for that user. At this point, if the user wishes to place a call, they simply type the callee's name into the appropriate text field on the applet, choose either text or voice call, and wait for the callee to accept their invitation. If a voice call is chosen, a JMF process is started that captures and transmits the voices of the caller and callee. For text messaging, Java TCP Sockets and ServerSockets are used to allow instant messaging capabilities between the two parties. Once the call is complete, the user clicks hang-up, and all streams and sockets are closed. The user is now free to participate in another call.



My Phone Address: compname Port (5060 is default): 5060 Register

Callee's Address: Enter the callee's address here.

You must register before placing any calls.

Type messages here and press 'Post Message'.

Messages will be displayed here.

Figure 3 – CNAV Softphone.

6.1 Voice Messaging

JMF handles all of the voice messaging in the CNAV Softphone. Once SIP finds the callee's address and gets the call underway, JMF takes over to provide real-time voice streaming over the Internet. In order to transmit successfully, the caller and callee's IP addresses and two available ports on each machine are needed. One port will be used for sending data, while the other is used for receiving data. JMF captures the users' voices directly from their microphones, and using the codec specified, compresses the data in the appropriate format.

Several different formats have been explored for use in the Softphone, however MPEG consistently performed the best. G.711 also worked fairly well, although in terms of voice quality, MPEG was undoubtedly superior. JMF then transmits the compressed data to the callee's machine using their IP address and port number. Upon receiving the data on the callee's end of the connection, JMF decompresses it, and plays it through the computer's speakers.

6.2 Text Messaging

Test Messaging in the CNAV Softphone is implemented using the Java Socket and ServerSocket classes. When a caller initiates a text call, a ServerSocket is created that waits for clients to connect to it. If the callee accepts the invitation, a Socket connection is made to the caller. Two streams are needed to handle the data flow in both directions. To exchange messages, the users type in the text boxes within the Softphone and choose send. The typed text is put on the stream where it is immediately retrieved from the other party and displayed in the lower text box in the Softphone. This communication continues until one of the users end the call. Once the call is complete, all streams and Sockets are flushed and closed.

6.3 Voice-Text Negotiation

From a programmer's prospective, perhaps one of the most interesting aspects of this Softphone is the way in which voice-text negotiation takes place. Here the programmer has the freedom to create a new protocol defining the standards of communication in the Softphone. The main concern in this situation is that a conflict of interest arises when the caller and callee do not agree on the type of call. If the caller requests voice, for example, but the callee accepts only text, the Softphone must decide how to solve this problem before the call can be connected.

One solution is to always honor the caller's request. Since the caller was the one who initiated the call, they should get precedence over the callee. Therefore, if a caller requests a voice call, regardless of whether the callee accepts voice or text, the voice call will take place. Similarly, if the caller requests text only, as long as the callee accepts either a voice or text call, the text call will be completed. The problem with this solution though is that it assumes that a microphone and speakers are available to both the caller and callee at all times. It is important to realize the weaknesses of this particular solution. Consider the case where a callee does not have access to a microphone when a caller initiates a voice call. With this solution the call would supposedly take place anyway. There must be a better way to handle the problem.

A modified version of this solution involves assigning text messaging a higher priority than voice messaging. This makes more sense because text messaging does not require any extra hardware like microphones or speakers. The only needed hardware is a keyboard for typing. If this modified protocol for voice-text negotiation is used, the only way two users can engage in a voice call is if both parties agree to accept voice. This solution solves many of the problems that the first solution encountered, and is a better protocol for most situations.

6.4 Applet Security

When working with applets that capture from microphones connected to personal computers, there are several potential security loopholes that arise and must be taken into consideration during development. For example, consider the case where two users engage in a voice call. The conversation ends, and both parties walk away from their computers. However due to a flaw in the software, the connection is accidentally left open from the callee to the caller, and the microphone continues to capture and transmit to the caller's computer. Some time later, the caller returns to their computer and realizes that they can hear the conversation that is taking place in the callee's room. However the callee cannot hear the caller, and hence has no way of knowing what is happening. The caller's Softphone has essentially bugged the callee's room.

In order to allow applets to capture from microphones, each user must specify this privilege in their *java.properties* file. Every machine that runs Java has this file in their system. By allowing applets to capture, users are making themselves vulnerable to some serious invasions of their privacy. Thus as a developer of VoIP applets, it is essential that all connections are successfully closed.

6.5 Possible Project Extensions

Once the student finishes the project, if they want to continue working with VoIP applications there are a variety of extensions to the Softphone that involve new and challenging technologies. One such possibility is making the Softphone work with an IP phone. There are a few vendors that make IP phones currently, and one of the more popular ones is a company called Pingtel (http://www.pingtel.com). Pingtel phones are essentially regular telephones that plug directly into an Ethernet connection. They have an IP address that can be used from the Softphone to complete calls. Pingtel phones can be programmed using Java, and they also come with a built-in SIP stack for performing SIP functions.

Yet another extension to the Softphone is to make it work with regular telephones connected to the Plain Old Telephone System (POTS) or Public Switched Telephone Network (PSTN). This requires the developer to purchase a router that plugs into an Ethernet connection, and allows regular telephones to plug into it. Several versions of this type of router are available from companies like Cisco, however they tend to be rather expensive. The router basically assigns every telephone connected to it an IP address, so that calls can be made to and from the Softphone.

7. VoIP IN THE CLASSROOM

Traditional networking courses cover popular topics such as TCP, HTTP, and Internet application construction. Topics such as multicasting, UDP, and databases are usually forgotten.

In the spring semester of 2001, we introduced the CNAV Softphone project in the networking course at Gettysburg College. Since the Softphone incorporates many different network technologies, it was an ideal case study to present various network concepts working together in one system.

The first concept is the use of UDP as the primary communication transport. With TCP dominating the Internet and Internet applications, the focus on UDP gives students a different perspective of communication methods. UDP is used as both a session medium and as a communication transport in the Softphone.

To demonstrate the use of UDP, students study the packets from a snoop session between two Internet phones. The packets first demonstrate the use of the SIP protocol using UDP. Students examine the packets in ASCII and hexadecimal format, and see the headers of the packets where handshaking information is stored. Once the session is initiated, students view the RTP packets containing the G.711 encoding of voice signals. This allows them to analyze the hexadecimal representation of voice packets.

The last concept to cover is the handshaking between two Softphones. More times than not, students only study existing protocols. In this project, a handshaking protocol has to be developed. Here students must address the issues revolving around the hardware and user preferences.

8. REFLECTIONS

As the main student programmer of the CNAV Softphone previously described, I found the project exciting, challenging, and sometimes frustrating. It was fun to work with these popular, innovative VoIP technologies. However, like most new technologies, the documentation available was somewhat lacking and inaccurate. Even some of the sample code I downloaded initially did not work properly. But once I got past these issues, I really enjoyed the programming. It was difficult at times, but never overwhelming. Also, it was very rewarding to successfully complete my first call with the Softphone. The words "I can hear you, can you hear me?" never sounded as good as they did that day! I would definitely recommend this project to other Networking students.

–Jeannie Albrecht

9. SUMMARY

As the hardware specifications of personal computers and bandwidth capacities of networks continue to increase in the upcoming years, there will be a greater demand for reliable VoIP applications. Protocols such as SIP and RTP, in addition to programming technologies like JMF, offer many new options for developers. Students enjoy learning about media-based applications such as the CNAV Softphone, and projects such as this offer them an opportunity to apply concepts taught in undergraduate Networking courses.

10. ACKNOWLEDGEMENTS

The authors express their appreciation to Gettysburg College and CNAV Systems for partially funding this project. In addition, we appreciate the review comments from David Athey at CNAV Systems.

11. REFERENCES

[1] DeCarmo, Linden. *Core Java Media Framework*. Prentice Hall, 1999.
[2] dynamicsoft SIP Software Documentation (included on CD with software). http://dynamicsoft.com
[3] Horstmann, Cay and Gary Cornell. *Core Java 2, Volume 1: Fundamentals*. Sun Microsystems Press. 1999.
[4] Java Media Framework API Documentation http://java.sun.com/products/java-media/jmf/index.html
[5] Java 2 Platform, Standard Edition, v 1.3 API Specification http://java.sun.com/j2se/1.3/docs/api/index.html