Project 4: VoIP Chat program

Department of Mathematical Sciences
Computer Science Division
University of Stellenbosch
7600 Stellenbosch

July 2020

1 Introduction

Voice over IP (VoIP) is a very popular service. The ability of multiple users to have a voice conversation over the Internet is a valuable one. VoIP greatly increased the value of the Internet and gave rise to IP voice switching (a method used by new telephone companies to establish voice connections between users, using IP packets instead of hardware switching circuits). Using this technology, telephone companies' infrastructure costs and service costs can be reduced. Global telephone networks may now be set up, unbounded by country borders or political policy.

2 Goals

After successful completion of this assignment, students will have:

- A reasonable understanding of client/server interaction
- Understand basic inter-networking concepts, such as the use of IP addresses, hostnames etc.
- Understand how the User Datagram Protocol (UDP) works
- Know how to implement a multi-threaded client
- Have a reasonable understanding of the Java Sound API.

3 General Specifications

The objective of this assignment is to implement a working Voice over Internet Protocol (VoIP) system. The program will be used over the local LAN and should enable users to initiate voice conversations with other users. The program will consist of a server and multiple clients.

You can code this program in any language you like, although we recommend that you code it in Java. Keep in mind that the student assistants might not know your chosen language and therefore not be able to help you with all coding problems you might have. The practical has however been successfully implemented in Java and the marking scheme has been set up with Java socket programming and multi-threading difficulties in mind.

The project should be handed in together with a report. The report specifications may be found on this web-page. All projects and reports will be checked for plagiarism. If plagiarism is detected there will be serious consequences. You are allowed to work in groups of *no more* than *two*.

3.1 The Server

The role of the server is to coordinates all the activities of the clients in such a way that the clients interact as fast and efficient as possible. The Server should implement a minimal GUI to give feedback on client activities. The server will accept connections from multiple clients, but multi-threading is not necessary. When a client terminates its connection, it should be done in a clean fashion where the server operations are not disrupted.

With voice calls, the server's only role is to check and see if the requested user is still active, and then to initiate the session. All further communication between the two users are handled by the client programs. Users should also be able to send text messages and these should be correctly displayed by the receiving clients.

3.2 The Client

The role of the client is to interface with the user, translate requested operations, and interact with the server. It is essential for the client to have a GUI, to act as an interface for the user.

Clients may initiate voice calls to other clients by having the user give a 'call' command and a host name to call to. Voice communications, between clients are direct, without any server interference. The server should merely act as a proxy and look-up service, assisting clients in making the connections. Text communications should also be possible. Voice communications should occur in real-time

4 Project Specification

4.1 Server

A Server that coordinates all requests to and from clients and adheres to the following criteria:

1. Handle all the various requests from the clients.
2. Clients will have to get permission from the server to initiate a voice conversation.
3. Act as a mediator, proxy and directory for the clients.
4. A stable environment should be created where unusual actions/requests by the clients do not disrupt the server.
5. Multiple users must be able to connect simultaneously.
6. Handle connections/disconnections and and correctly update the user list.
7. A minimal Graphical User Interface (GUI) should be used containing:
   (a) A main window that outputs client activity.
   (b) A list of users currently active.

4.2 Client

A Client that interfaces with the user and adheres to the following criteria:

1. Clients must be able to disconnect/reconnect without incident.
2. A list of currently connected users must be shown, and updated as needed from the server.
3. Individual nicknames are not necessary, the use of hostnames are sufficient.
4. The ability to chat to each other during a call (similar to skype)
5. Conference calls should be implemented

6. Conference channels (chat channels) should be implemented.
7. The ability to send pre-recorded voice (similar to whatsapp)
8. A call command to call another user should be implemented.
9. Calls must use Real-time Voice transmission.
10. Voice quality should be as high as possible (at least 8000 samples per second, 16-bit sample size and 1 channel).
11. *Distortions, Echoes and Deadzones* in the Audio should be minimized
12. All voice communications should occur using the **UDP** protocol.
13. A GUI is essential for interfacing with the users and should contain:
    (a) A main window that outputs messages and descriptions of all actions performed by it's user.
    (b) A list of users currently active, sorted by host-name.
    (c) A "command" field, where messages and commands are entered. When an unrecognised command is received it should be ignored and an error message should be displayed in the main window.

5 Dates

The following dates are important for this project:

- **Project starts:** 23 September 2020.
- **Project deadline:** 14 October 2020 (13:00).

The report, Makefile and source codes must be submitted as a single tar or zip file whose name has the following form: `GROUP_NUMBER.tar.gz` where `GROUP` is your group name and `NUMBER` is the project number. Your project must be submitted to the GitHub repository `Computer-Science/rw354/2020/Project 4`.

6 References

Here follow some resources that may be helpful:

- Code example of how to record sound in Java:
  `http://www.developer.com/java/other/article.php/1565671`
- For basic Java tutorials look at the "Trails Covering the Basics" section on Sun's Java website:
  `http://java.sun.com/docs/books/tutorial`
- To get started with Java socket programming, the following tutorial is all you'll need:
  `http://java.sun.com/docs/books/tutorial/networking`
- Java Datagram tutorials:
  `http://java.sun.com/docs/books/tutorial/networking/datagrams/index.html`
- If you're a bit rusty with GUI's, then the Java Swing tutorial should help:
  `http://java.sun.com/docs/books/tutorial/ui`

7 Helpful Hints

1. Start early, do not leave this project to the last few days and think you will finish on time.
2. Look at the marking scheme and at what needs to be done to help you gauge your progress.
3. Do not think you are almost done after you have successfully had your client connect to your server.
4. It will make life easier if you run the recording, playback and the main client program as separate threads.
5. Look into Java Mixers instead before considering implementing your own mixing algorithm.
6. Find someone to test with, because you cannot use the same account to test your program in the labs.
7. If you use tokens to transmit commands, make sure all command characters have been added.
8. The Client is much more complex than the Server. Do not spend all your time perfecting the server and leave the client until the last minute. More time should be spent on the client than the server.
9. Keep in mind that the sound quality is dependent on the operating system as well as the Java JDK, so make sure to test your code in the lab where the demo will take place.

8 Marking Scheme

Housekeeping and style	3
Sending/receiving text messages from multiple clients, concurrently	3
Client GUI	4
Server GUI	2
User lists function correctly	6
Sending/Receiving voice concurrently	6
Voice Quality	9
Distortion	3
Echoes	3
Dead zones in the Audio	3
Program Stability	9
Making consecutive calls	3
Receiving consecutive calls	3
Conference calls (Manual initiation)	6
Conference channels	3
Sending pre-recorded voice	3
Real-time Voice transmission	9
Minimal (2 seconds max) delay between transmission and playback	3
Transmitting voice during playback of other user's voice	3
Sending/Receiving text messages during calls	3
Report	**15**
- Language, spelling and grammar	3
- Selection of experiments discussed	2
- Description of the experiments	5
- Conclusions drawn from the results	5
TOTAL	75