VoCe: Voice Conference

CO324 Assignment

Problem description

Peer to peer communication is a strategy used in applications such as BitTorrent, and Skype. In this project you will design and code a basic peer to peer voice conferencing application similar to Skype. First, you will implement voice communication between two parties. Then you will extend the application to support multi-party call conferencing. Finally, you will measure the performance of your application under real-world conditions using a network emulator.

Requirements

The sample code provided demonstrates the use of Java audio APIs to record and playback audio on the same machine. Extend this code to transmit and receive full-duplex audio over a network between two parties. You will need to use separate threads for recording and transmission, and reception and playback.

Your application should take the peer's hostname or IP as a command-line argument. Assume the peer is directly reachable by their IP addresses (no NAT is involved.) You are not required to implement a GUI.

Then you will implement multi-party conferencing using UDP multicast. Assume that only one party is speaking at a time, so there is no need to handle mixing multiple audio streams. Your enhanced application should take a multicast group address as a command line argument. Assume all participants are directly reachable by their IP addresses (no NAT) and, that multicast functionality is available. However, not all participants need to be on the same (wireless) LAN or subnet.

Design and testing

Before writing any code, plan the structure of your application for ease of modification and testing.

Begin by defining your application message framing atop UDP. You must have appropriate headers for handling packet loss and reordering. Separate logic that handles packet loss and reordering and message serialisation/deserialisation code into individual classes.

You are required to write unit tests for at least your serialisation/deserialisation code. Additional unit tests will earn you bonus marks. We suggest you to use a **Linux based** environment for development and testing.

Hint: When doing multicasting you might need to design a separate protocol to manage states of the users.

Performance measurement

To test the performance of your application under packet loss and reordering conditions use the *netem* Linux network emulator introduced in the course labs. First, modify your code to print statistics on the number of packets lost or received out of order every minute.

Collect the statistics by varying UDP packet size, and the *netem* loss and delay parameters. Include test results, observations and results in your project report.

Marking criteria

The project will be marked under 2 criteria as,

1. Demonstration + Viva - 60%
2. Submission (code + report) - 40%

Demonstration

You will have a demonstration for the project evaluation. All the members of the group will be evaluated individually for their contribution to the project.

Submission

Along with the code submission, you must write a report which describes the message format, application design, testing and performance measurements. Limit your report to 8 pages (11pt - Times New Roman). Your submissions will be marked according to the following criteria.

1. Correct implementation of each requirement: 16%.
2. Performance measurements and analysis: 16%.
3. Good design and unit tests: 8%.

Bonus marks will be awarded for creative features except for a GUI. However, first ensure that the basic functionality works exactly as defined before attempting to add bonus features!
Late penalties will apply as per course and department policy. **Remember that plagiarism gets you zero!**

References

- Java Sound tutorial
- Getting started with JUnit
- Multicast howto
- IP Multicast Technology
- Multicast example in Java
- Introduction to network emulation
- How to define netem rules