VoCe: Voice Conference

E/15/330

E/15/366

E/15/373

ABSTRACT

Voice Conference is taking analogue audio signals, and turning them into digital data that can be transmitted over a network. VoCe has become an important factor in network communication. It has a lower operational cost, greater flexibility, and a variety of enhanced applications. VoCe is time – based. To ensure real-time transmission, Real-Time Transmission Protocol (RTP) is used on top of User Datagram Protocol (UDP). RTP provides end to-end network transport functions suitable for applications transmitting real-time data, such as audio, video or simulation data, over multicast or unicast network services. The assignment aims at designing a system that will allow users to communicate over a data network. That is to be able to make voice call transfer over a network.

INTRODUCTION

VoCe is a standard for converting analog audio signals into digital data that can be transmitted over the Internet (or simply a data network) rather than traditional Public Switched Telephone Network (PSTN). VoCe has become an important development in network communication. It is a rapidly growing Internet service. It has lower operational costs, greater flexibility, and a variety of enhanced applications. VoCe technology uses the Internet's packet-switching capabilities to provide phone service. The internet is an Internet Protocol (IP)-based network. IP based networks are connectionless, packets switched networks. Transmission by the connectionless technology has no guarantee that the data is received by the destination. To avert this problem, VoCe communication should be connection oriented. In a connection-oriented communication, there is a guaranteed delivery of data: any data that is not received by the destination system is re-sent by the sending device.

Connection-oriented approach introduces a lot of delay in transmission. Since VoCe is time – based, using the connection –oriented approach is not the best. The known protocol that ensures connectionless-oriented transmission is User Datagram Protocol (UDP). Though UDP transmission is not secured, it is ideal protocol for transmitting time – based media. To ensure real-time transmission, Real-Time Transmission Protocol (RTP) is used on top of UDP. RTP provides end-to end network transport functions suitable for applications transmitting real-time data, such as audio, video or simulation data, over multicast or unicast network services.

Problem description

First, we implement voice communication between two parties. Then we extend the application to support multi-party call conferencing. Finally, we measure the performance of our application under real-world conditions using a network emulator.

Peer to peer communication

Peer to peer communication is a strategy used in applications such as BitTorrent, and Skype. In this project we design and code a basic peer to peer voice conferencing application similar to Skype..

Our application take the peer's IP as a command-line argument. The peer is directly reachable by their IP addresses. No NAT was used in this application.

Multicasting

Then we implement multi-party conferencing using UDP multicast. That only one party is speaking at a time, so there is no mixing multiple audio streams. Our application take a multicast group address as a command line argument. All participants are directly reachable by their IP addresses. IP multicast is a technique for one to many and many to many real-time communication over an IP infrastructure in a network. It scales to a larger receiver population by requiring neither prior knowledge of a receiver's identity nor prior knowledge of the number of receivers. Multicast uses network infrastructure efficiently by requiring the source to send a packet only once, even if it needs to be delivered to a large number of receivers. The nodes in the network (typically switches and routers) take care of replicating the packet to reach multiple receivers such that messages are sent over each link of the network only once.

Design

Peer to Peer

We designed the first peer to peer application to communicate between two parties, and created two threads for each party one for receive the packets from socket and write it to the mixer (source data line) and one for get the signals from mic read them and send them to the next party by the socket in our code Receive.java code will be receive the data from socket and write it to the audio

Output and the methods captureAudio( ), CaptureAndSend() methods are used to get the input from mic and send it to the socket. The ip address of next peer must be given as command line argument.

Multicast

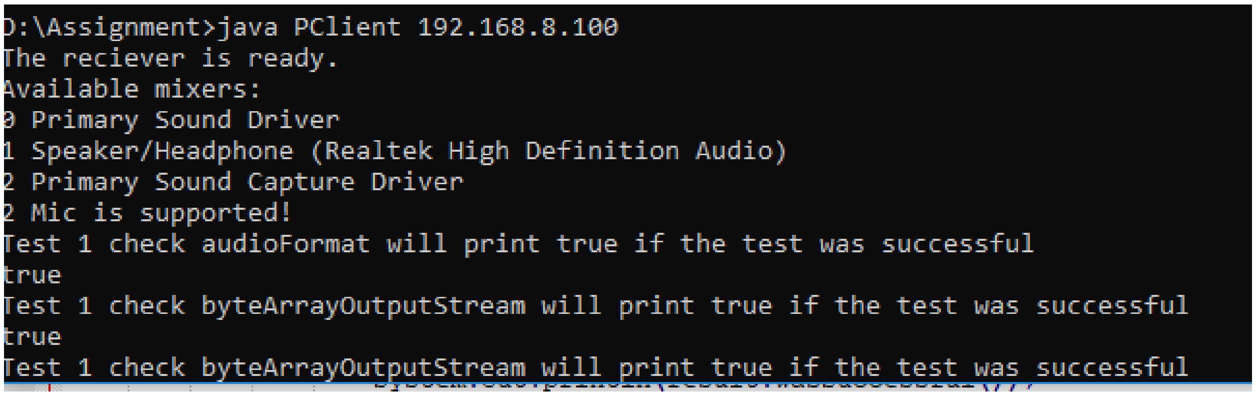
As same as the peer to peer connection but in Multicast the packets are replicated from one user and transmitted to other users by the network, This application is written using Internet Group Management Protocol on top of UDP. An IP multicast group address is used by sources and the receivers to send and receive multicast messages. Sources use the group address as the IP destination address in their data packets. Receivers use this group address to inform the network that they are interested in receiving packets sent to that group. For example, if some content is associated with group 239.1.1.1, the source will send data packets destined to 239.1.1.1. Receivers for that content will inform the network that they are interested in receiving data packets sent to the group 239.1.1.1. The receiver *joins* 239.1.1.1. The protocol typically used by receivers to join a group is called the (IGMP).

User should give the multicast ip address as command line argument and join the group conversation.

Unit testing

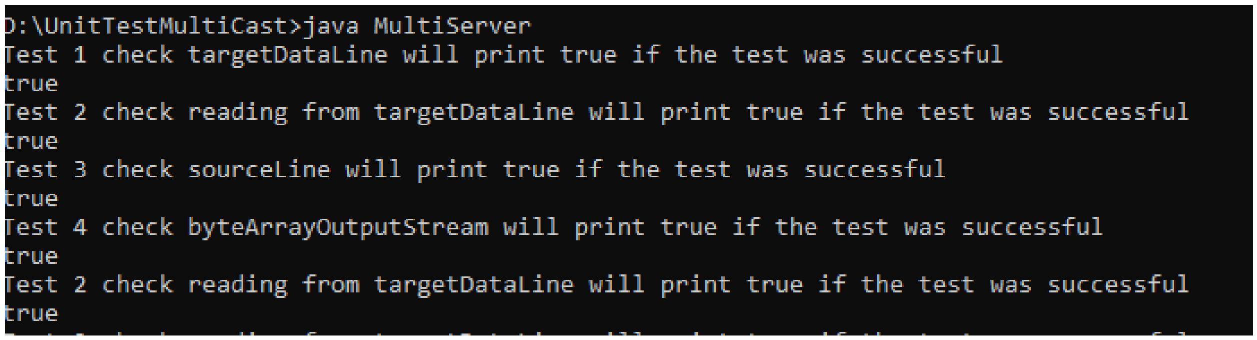
We tested both applications with Junit library, tests are done for the serialization of the signals and deserialization of the packets these test code is attached in the UnitTest folder.

Unit test results for peer to peer



```
D:\Assignment>java PClient 192.168.8.100
The reciever is ready.
Available mixers:
0 Primary Sound Driver
1 Speaker/Headphone (Realtek High Definition Audio)
2 Primary Sound Capture Driver
2 Mic is supported!
Test 1 check audioFormat will print true if the test was successful
true
Test 1 check byteArrayOutputStream will print true if the test was successful
true
Test 1 check byteArrayOutputStream will print true if the test was successful
```

Unit test results for multicast



```
D:\UnitTestMultiCast>java MultiServer
Test 1 check targetDataLine will print true if the test was successful
true
Test 2 check reading from targetDataLine will print true if the test was successful
true
Test 3 check sourceLine will print true if the test was successful
true
Test 4 check byteArrayOutputStream will print true if the test was successful
true
Test 2 check reading from targetDataLine will print true if the test was successful
true
```