PROJECT REPORT

Design of the System

We use five java classes to implement this application. Those are,

1. InitResources.java
2. ReceiptionAndplay.java
3. RecordingAndTransmission.java
4. Statistics.java
5. VoicePacket.java

01) InitResources.java

This class is used to initialize the all resources like mic's and speakers which are exist in the computer. And also this is used to initialize the mixers. This is the parent class of ReceiptionAndplay.java and RecordingAndTransmission.java

02) ReceiptionAndplay.java

This class is used to get the receiving data packet and send those packet data to the speaker to play.

03) RecordingAndTransmission.java

This class is used to record the voice and send the data to the other listener's.

04) Statistics.java

This class is used to get the number of unordered packets and count the lost packets when transmission.

05) VoicePacket.java

This class is used to create a voice packet. All the packet formatting like adding sequence number things happening in this class.

When the client starts this application, he should give multicasting IP address to the application via terminal. If another person wants to join with the same multicast group he should give the same multicast IP address.

And any client using this application can see how many packets are lost within the network and how many packets receiving are unordered within every one minute.

(The solution for the packet lost is valid only if at most two clients are engaged)

Message Format

Following things are added in to the packet.

- Sequence number.
- Voice Data.

Data Serialization and Deserialization

Before sending the voice packet, the voice packet is serialized. On receipt data is deserialized to get the data packet instance.

Handling loss and reordering

We measured number of lost and out of order packets.

(The solution for the packet lost is valid only if at most two clients are engaged)

Concurrency

Two separate threads are used to handle receive and play, record and sent.

Test

A unit test was done on VoicePacket class.

Performance metrics using netem

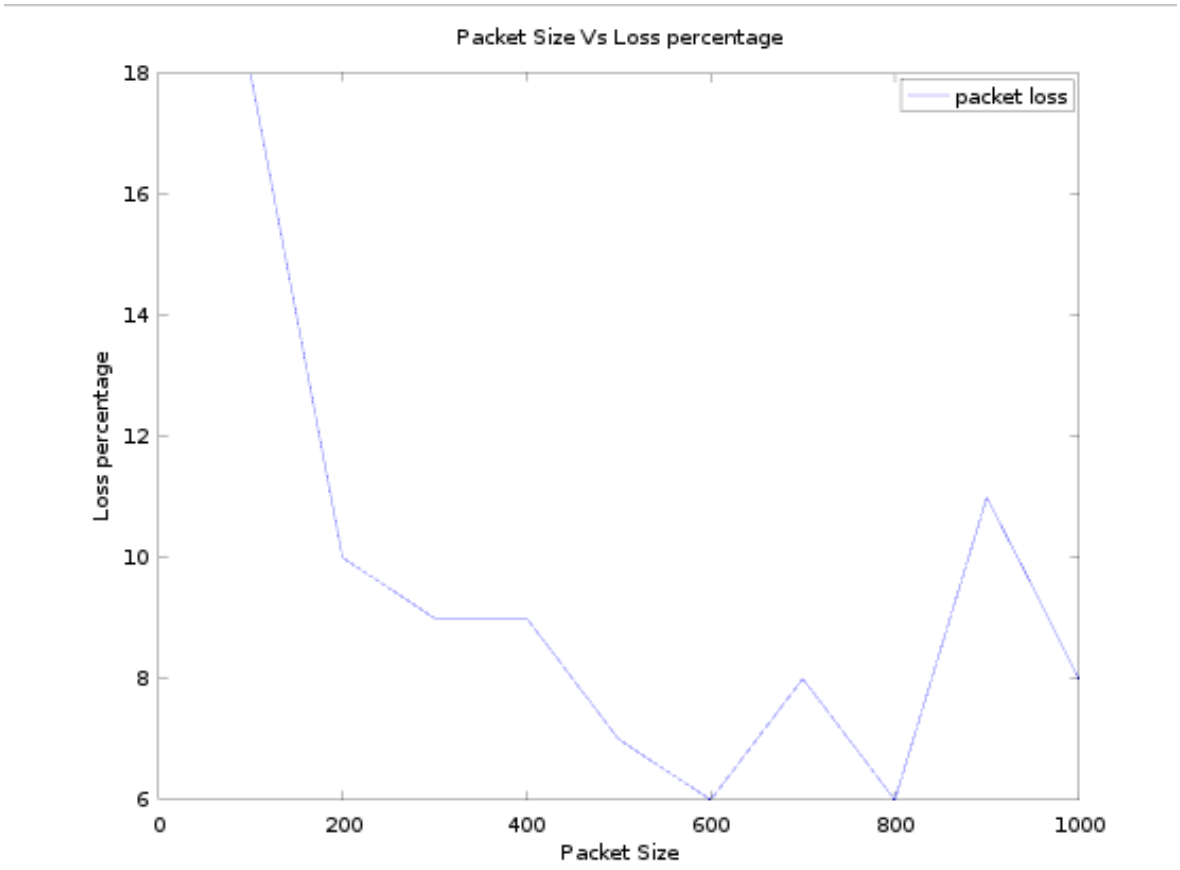
The below command was used to apply packet delay and lost.

Netem command:

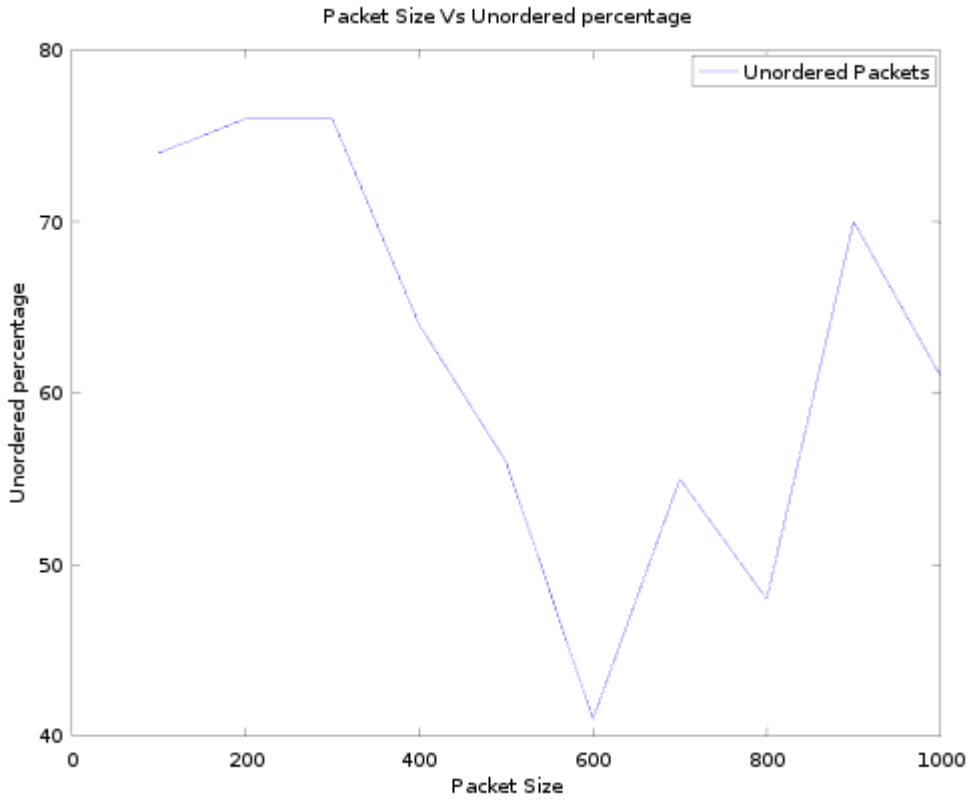
“ Iface = eth0
Loss = 10%
Delay = "400ms 800ms"

sudo tc qdisc add dev "$iface" root netem delay $delay loss "$loss" ”

Packet Size Vs Loss Percentage:



Packet Size Vs Unordered Percentage



Loss Packet Vs Received packets (Packet size = 500 bytes)