VoCe: Voice Conference
CO324 Assignment

E/15/119 HASANIKA D.L.D
E/15/202 LIYANAGE D.P
E/15/208 MADHUSHANEE G.G.R.D

CONTENT

1.Task

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

There are two parts of the implementation.

1. First, we implemented voice communication between two parties. (Peer to peer)
2. Then we extended the application to support multi-party call conferencing. (Multicast)

Finally, we measured the performance of our application under real-world conditions using netem Linux network emulator.

2.Peer to Peer communication

Introduction

Peer to peer communication enables any two devices to communicate directly with one another without the mediation of another central switching system.

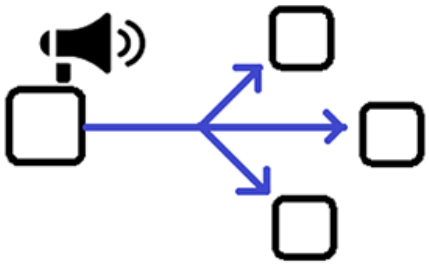
Implementation

1. Before starting the peer to peer communication, they should connect to the same wifi network.
2. Then after, both of them run the same code (Peer2Peer.java) they can communicate with each other.
3. In the code, captureAndSend() method captures the voice from the sender and it is transmitted through the UDP socket 5501.
4. Class Play will get those audio and play them at the receivers end.
5. Capturing the audio and sending will be done parallely with the playback so that communication can be done fast.

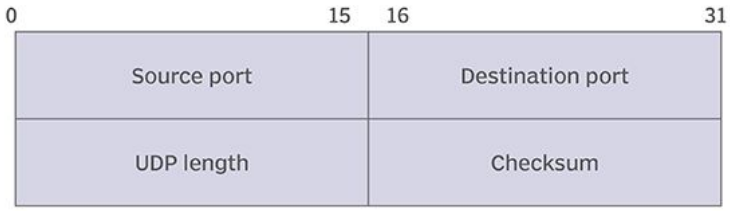
3. Multicast communication

Introduction

Multicast is group communication where data transmission is addressed to a group of destination computers simultaneously. Multicast can be one-to-many or many-to-many distribution.



4.Message Format



Datagram packets are used to implement a connectionless packet delivery service. We used java Datagrampacket class

DatagramPacket(byte[] buf, int length,InetAddress address, int port)

Constructs a datagram packet for sending packets of length “length” to the specified port number on the specified host.

Datagrampacket packet = new Datagrampacket(tempBuffer, tempBuffer.length, host, 55001)

Parameters according to above example,

buf - the packet data - tempBuffer
length - the packet length - tempBuffer.length
addr - the destination address - host
port - the destination port number- 55001

5.Performance Measurement with netem emulator

Introduction

Netem allows you to setup a delay profile on the outgoing packets on a network interface. NetEm is an enhancement of the Linux traffic control facilities that allow to add delay, packet loss, duplication and more other characteristics to packets outgoing from a selected network interface. NetEm is built using the existing Quality Of Service (QOS) and Differentiated Services (diffserv) facilities in the Linux kernel.

As Netem only affects outgoing packets, so, when testing, we run any commands on both of the switch's network interfaces to simulate problems in both directions of the connection.

netem OPTIONS :

netem has the following options:

- Limit packets
  Maximum number of packets the qdisc may hold queued at a time.

- Delay
  Adds the chosen delay to the packets outgoing to chosen network interface. The optional parameters allows to introduce a delay variation and a correlation. Delay and jitter values are expressed in ms while correlation is percentage.

- Reorder
  To use reordering, a delay option must be specified. There are two ways to use this option (assuming 'delay 10ms' in the options list).

Reorder 25% 50% gap 5

In this example, the first 4 (gap - 1) packets are delayed by 10ms and subsequent packets are sent immediately with a probability of 0.25 (with correlation of 50%) or delayed with a probability of 0.75. After a packet is reordered, the process restarts i.e. the next 4 packets are delayed and subsequent packets are sent immediately or delayed based on reordering probability. To cause a repeatable pattern where every 5th packet is reordered reliably, a reorder probability of 100% can be used.

Reorder 25% 50%

In this second example 25% of packets are sent immediately (with correlation of 50%) while the others are delayed by 10 ms.

The specific command to simulate reordering is tc qdisc add dev <dev> root netem delay <Xms> reorder <Y%>. This will delay Y% of packets by X ms, causing reordering if the packets are originally sent less than Xms apart.

- Packet Loss
  To simulate packet loss, the command to use is tc qdisc add dev <dev> root netem loss <X.Y%>. Have to Replace <dev> with the network interface we want to simulate loss on. Have to Replace <X.Y%> with a percentage of packet loss to test with.10% is reasonable for testing.

- Packet Duplication
  Receiving the same packet again and again is packet duplication. The specific command to simulate duplication is tc qdisc add dev <dev> root netem duplicate <Y%>. This will duplicate Y% of packets by X ms.

Testing

1.Peer to Peer communication

I. Varying packet size

- When packet size = 100 (100 bytes), peer to peer connection was established properly and the communication happened clearly.
- When packet size was increased(1000 bytes), peer to peer connection was established but the communication didn't happen clearly.
- When packet size was decreased(2 bytes), peer to peer connection didn't establish properly and an exception was encountered.

```
java.lang.IllegalArgumentException: illegal request to write non-integral number of frames (2 bytes, frameSize = 4 bytes)
        at java.desktop/com.sun.media.sound.DirectAudioDevice$DirectDL.write(DirectAudioDevice.java:727)
        at Play.run(Peer2Peer.java:48)
        at java.base/java.lang.Thread.run(Thread.java:834)
```

II. Varying netem loss

- Command : sudo tc qdisc add dev eno1 root netem loss 50%

This command will add 50% of packet loss during the communication. That means if there are only 6 packets, then 3 of them will be lost and only 3 of them will be received by the other end.

III. Varying netem delay

- Here, with a delay and without a delay both instances are tested.
- The result is an average of 240ms of latency per ping packet route trip. This is right around what we would expect after adding the 250 ms of latency.

- Command : `sudo tc qdisc add dev eth0 root netem delay 250ms`

6.Why we used UDP ?

UDP is not reliable as TCP. UDP is a best effort service. But UDP provides a better experience for users as it allows them to enjoy a real-time and uninterrupted call (conversation) without any delay. Errors such as packets losses have minor impact on the output audio and usually goes unnoticed. TCP errors on the other hand translated with streaming services manifest themselves differently. For example, a YouTube video that keeps freezing every now and then for few seconds while trying to deliver a high-quality image. This conversation users won't really appreciate such experience when they are making that all important phone call.