VoCe: VOICE CONFERENCE

28th of March 2016

GROUP 8 (E/12/083, E/12/103)
Department of Computer Engineering, University of Peradeniya

Table of Content

1. 1st Iteration

1.1 Introduction

This project is to design a voice over IP programme which could be handled group chats. As basic peer to peer voice conferencing application similar to Skype in two iterations.

In this iteration implemented voice communication between two parties.

1.2 Message format

We used UDP protocol. Simply a packet containing long byte of 508 bytes array in application layer. UDP allowed to multicasting. That is why we chosen UDP to transferring message. In 1st iteration we used 8 bytes as header. It is included the id of the message. We can use it to re-arrange or count the packet loss.

1.3 Designing

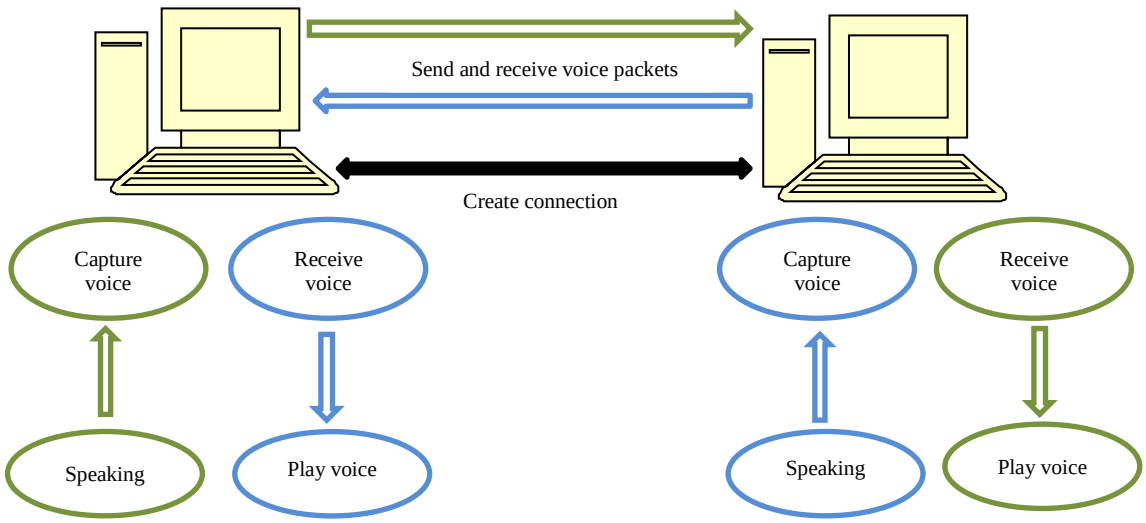

FIGURE 1

1.4 Application Design

Here we used 7 classes to implement the VoCe application. Below we mentioned the names of classes, what is the method used for it and short description about the classes.

1. RecordPlayback.java
   i. void captureAudio()
      In this method we connect with pc's hardware and initialize it.
   ii. void capture()
      Here we capture sound and store it into 500 bytes array with 2.4 kHz rate.
   iii. void play()
      At this stage we play the byte array which captured from capture method.
2. send.java
   There is thread runs always and it sends captured byte array to the other party.
3. receve.java
   Here is also a thread runs always to get byte array from the other party.

   (both send.java and receve.java use 2 sockets)
4. convert.java
   iv. byte[] serialize()
      This is used to serialized addid to byte array and return new byte array
   v. long getid()
      This method use to get id of received packet return id.
   vi. byte[] getdata()
      In this method we return decrypt byte array of received data.
5. counter.java
   vii. int id()
      Here we generate ids until 1200 and begining with 0.
   viii. int count()
      This method is used to count packet loss. We send 1200 packets waiting 1100 packets get unde 1000 packets loss.
6. peer1.java
   This is a peer. To use this first compile and give other peer's IP as argument.
7. peer2.java
   This is also a peer. To run this first compile and give other peer's IP as argument.

2. 2nd Iteration

2.1 Introduction

This project is to design a voice over IP programme which could be handled group chats. As basic peer to peer voice conferencing application similar to Skype in two iterations.

In iteration 2 we implemented multi-party conferencing using UDP multicast. Here only one party can speak at a time.

The enhanced application takes a multicast group address as a command line argument. Here all participants are directly reachable by their IP addresses (no NAT) and, that multicast functionality is available. And, not all participants need be on the same (wireless) LAN or subnet.

2.2 Sending Multicast Datagrams

Hosts can be in three different levels of conformance with the Multicast specification, but only level 2 supports for both send and receive IP multicasting. In principle, an application just needs to open a UDP socket and fill with a class D multicast address the destination address where it wants to send data to. There are some operations that a sending process must be able to control.

- TTL

  TTL (Time to live) has double tasks in multicasting. It controls the live time of the datagram to avoid it being looped forever due to routing errors. And it acts as threshold. It limits how long multicast traffic will expand across routers.

- Loopback
- Interface selection

  Hosts attached to more than one network should provide a way for applications to decide which network interface will be used to output the transmissions. If not specified, the kernel chooses a default one based on system administrator's configuration.

2.3 Receiving Multicast Datagrams

- Joining a multicast group

  First we have to inform kernel which groups we are interested in. Then we have to ask kernel to join with those groups. In multicasting we can join with pre-selected multicast groups only.

- Leaving a multicast group

  If we don’t interested in no longer, we can ask kernel to leave the group. But kernel still accepts multicast datagrams destined to that multicast group.

- Mapping of IP multicast addresses to Ethernet addresses

2.4 Procedure

6 classes are including in this iteration. Peer1.java is the main class. Send.java, receve.java, convert.java, counter.java and RecordPlayBack.java are other classes. Multicast address is hard corded on peer1.java. When compiling we must give initial identity number as an argument. It should be greater than 2000. Admin (first talker) must take 3000 as initial identity number. Admin gets the first chance to talk. If anyone wants to talk, he must press 'r' and hit 'enter'. 1st request sender takes the next chance to talk. Then the current talker press 'g' and hit 'enter' to give up talking. After request sender must press 's' and hit 'enter' to talk.

2.5 Application Design

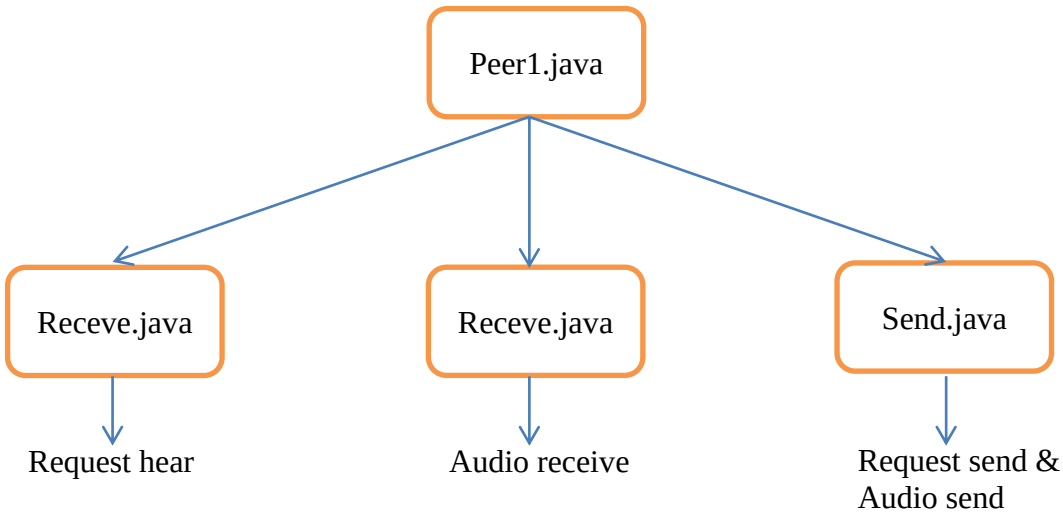

FIGURE 2

Peer1.java uses 3 threads. 1 thread use to send request and send audio and it is used by send.java. This is act as a client. And this can use 2 different sockets. Other 2 threads use to hear the request and receive audio separately. Those 2 threads use by receve.java. This use separate threads because it is difficult to pass a port and make a new socket. So these threads are developed for pause and resume. That is useful for multicasting else everyone collapse. Therefore in this method only one person can talk at a time and others hear it. This is like a token pass.

Convert use serialization and deserialization. In the serialization it adds a header to packets. At the deserialization it removes header and packets into two parts.

2.6 Handling loss and Reordering

A sequence number is attached to every packet. Initially the sequence number is '0'. For every 1000 packets receve.java counts the packets loss.

If there is a disordering we can reorder the packets using sequence number.

2.7 Testing

Our code works for any number of clients in localhost. And packet loss is zero for it.

And this multicasting worked for 3 clients in separate machines but only in SLT ZTE broadband router. We tested this for Dialog, 3com, CE4thfloor routers. But at every time it took huge packet lost. Greater than 80% packet loss had been occurred. Sometimes even there is not a 100% packet loss, there is not a packet sending between clients.

Therefore we couldn't test it for NetEM packet loss. Manual packet loss is calculate the code itself and it is displaying for every 1000 packets.