Comp 4985 Computer Systems Technology January 2014

Data Communication Option

Assignment #4 Design Doc
Comm Audio

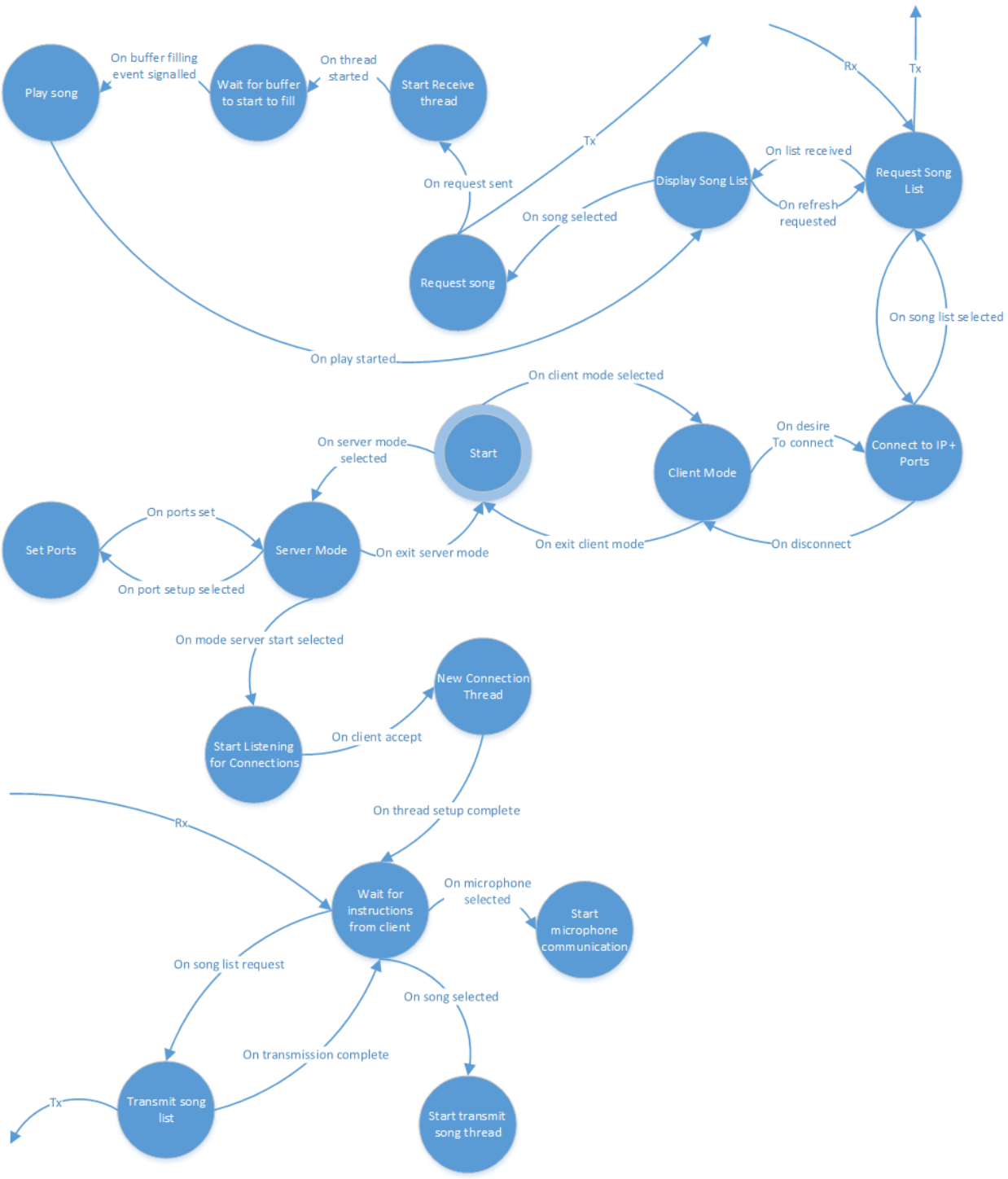
Ian Davidson, Josh Campbell, Clark Allenby

Set 40

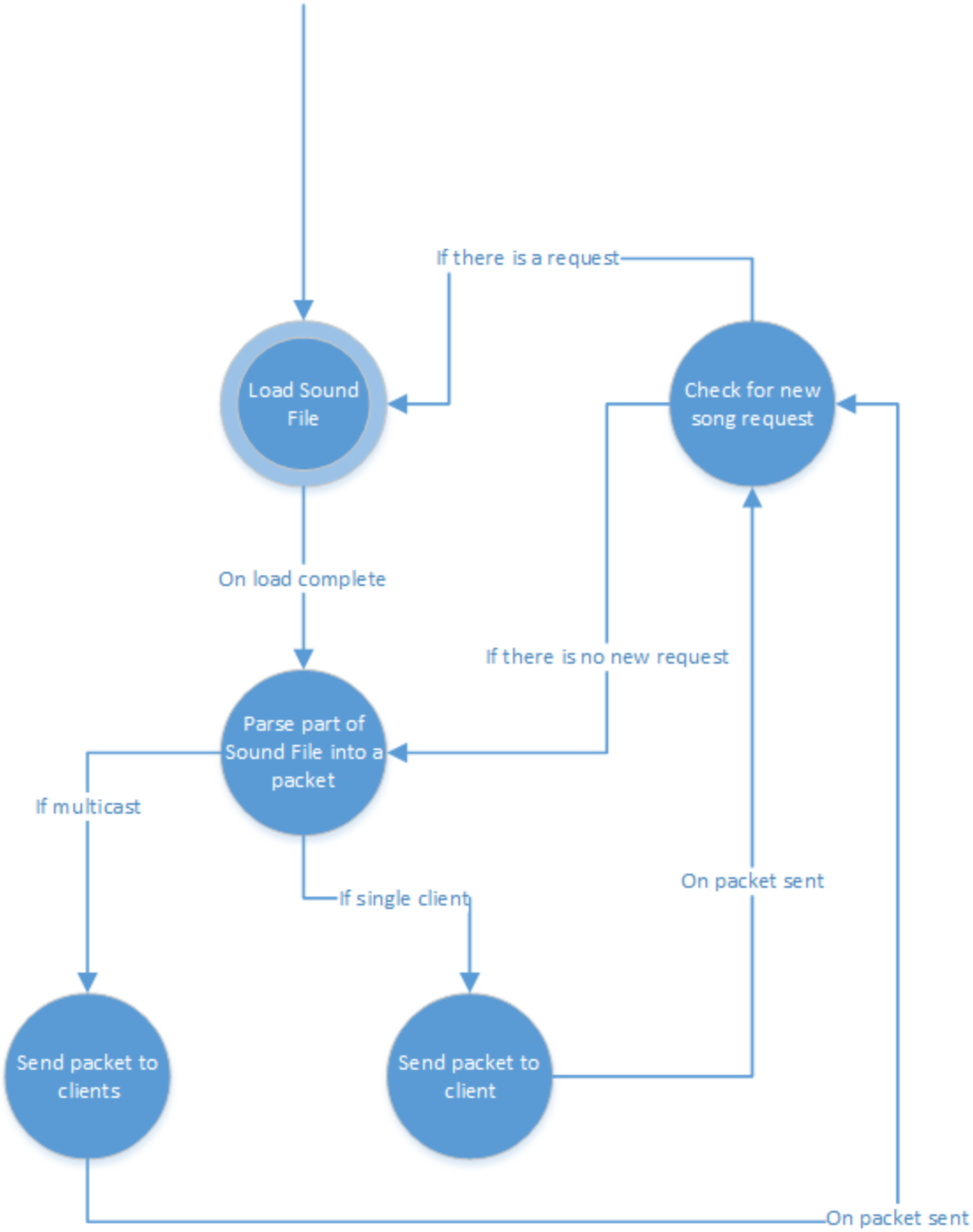
April 11th, 2014

Design

State Transition Diagram
Main program



Sound transmission



Pseudo Code

```
function Main()
      Start GUI
      CreateMenu (client mode, server mode)
```

```
      if client mode is selected
            start Client()
      end if

      if server mode is selected
            start Server()
      end if

end function

function Client()
      set port to default port
      Create Menu (set IP, set Ports, connect to server)

      if Set ports is selected
            display prompt for desired port
            set port to entered port
      end if

      if set IP is selected
            display prompt for desired IP address
            set ip to entered ip address
      end if

      if ip is set and connect is selected
            StartClientConnection(ip, port)
      end if

end function

function StartClientConnection(ip, port)

      create new socket connection using ip and port
      if connection is established
            create menu (request song list, disconnect)

            if request song list is selected
                  RetrieveSongList(socket)
            end if

            if disconnect is selected
                  return to client mode
            end if

      else
            return to Client Mode
      end if else
```

```
end function

function RequestSongList(socket)

      Create Song List buffer
      Send request for song list to server

      while receiving data
            store data in buffer
      end while

      Create array for list of songs
      parse buffer into new song list

      DisplaySongList(songlistarray)

end function

function DisplaySongList(songlist)

      create GUI of selectable list items using songlist

      if refresh button pressed
            RequestSongList(socket)
      end if

      if song selected and play button pressed
            RequestSong(song, socket)
      end if

end function

function RequestSong(song, socket)

      use socket to send request to play song to server

      Create buffer for song;

      start ReceiveSongThread(socket, songbuffer)

      PlaySong(songbuffer)

end function

function ReceiveSongThread(socket, songbuffer)

      While there is data to read
```

```
            try to read data from server into songbuffer

            if songbuffer has data read into it
                  send signal saying that the buffer has data
            end if
      end while

end function

function PlaySong(songbuffer)

      wait for song receiving signal
      while there is data in the song buffer

            load data from songbuffer into WAV structure

            play music

            if fastforward is pressed
                  increase the playrate
            end if

            if skip is pressed
                  see if the position in the song is available with the
                  buffered data

                  if not, wait for the buffer to get that data

                  end if
            end if

            if rewind is pressed
                  set the playrate to reverse through the song
            end if

            if pause is pressed
                  pause the song
            end if

            if play is pressed
                  if fastforwarding
                        slow down the playrate to normal
                  else if rewinding
                        reverse the playrate to normal
                  else
                        play the song if not currently playing
                  end else if
```

```
            end if

      end while

end function

function Server()

      set port to default port

      if exit is selected
            return to Main
      end if

      if set ports is selected
            display prompt for desired port
            set port to entered port
      end if

      if start server is selected
            StartServer()
      end if

end function

function StartServer()

      create new listening socket

      while the server is set to running

            wait for a new connection on the listening socket

            if there is a new connection
                  accept the connection into a new socket
                  ClientConnectionThread(acceptedSocket)
            end if

      end while

end function

function ClientConnectionThread(socket)

      while connection still exists
            Wait for for instructions from the client
```

```
            if instruction is transmit song list
                  TransmitSongList(socket)
            end if

            if instruction is to play song
                  TransmitSongThread()
            end if

            if instruction was to stop song
                  send signal to TransmitSongThread
            end if

      end while

end function

function TransmitSongList(socket)

      Scan Through Song List Directory
      Store songs into array list

      Send array of songs to client using socket

      return to ClientConnectionThread

end function

function TransmitSongThread()

      create udp send socket

      load song into memory

      while there is data to be sent
            pull data from file
            store data in song packet

            if sendtype is multicast
                  transmit song packet using multicast
            end if

            if sendtype is single client
                  transmit song packet using single address
            end if

            check if there was a new instruction from client
```

```
            if there was a new song request
                  load new song into song file
            end if

            if there was a stop command
                  exit loop
            end if

      end while

      return to ClientConnectionThread

end function
```

FEATURES:

- Two way microphone chat between the client and the server
- Song streamed from server to one client
- Song multicasted to multiple clients from the server
- Microphone multicasted to multiple clients from the server
- Songs can be chosen from a song list.
- Client and server are in a single application
- Data streamed over UDP
- Pause and play on the client side.

TESTING:

Test Number	Tools Used	Test Case	Expected Result	Actual Result
1	Wireshark and CommAudio application	Server streams to a single client.	Client received and played song	Pass, see fig 1 & 2.
2	Wireshark and CommAudio application	Server streams to multiple clients using multicast	Clients receive and play the song	Pass, see fig 2 & 3.
3	Wireshark and	Server and	Both client and	Pass, see fig 3

	CommAudio application	client can stream microphone data to each other.	server receive each others microphone chat	& 4.
4	CommAudio application	Songs can be chosen from a list.	Song is selected and sent out	Pass, see fig 5.
5	CommAudio application	Client can pause and play songs.	Song is paused and played	Pass, see fig 8 & 9.
6	Wireshark and CommAudio application	Server streams mic data using multicast.	Mic data is sent.	Pass, see fig 7.

FIGURES:

Figure 1 - Client Receiving Audio (Single Client):

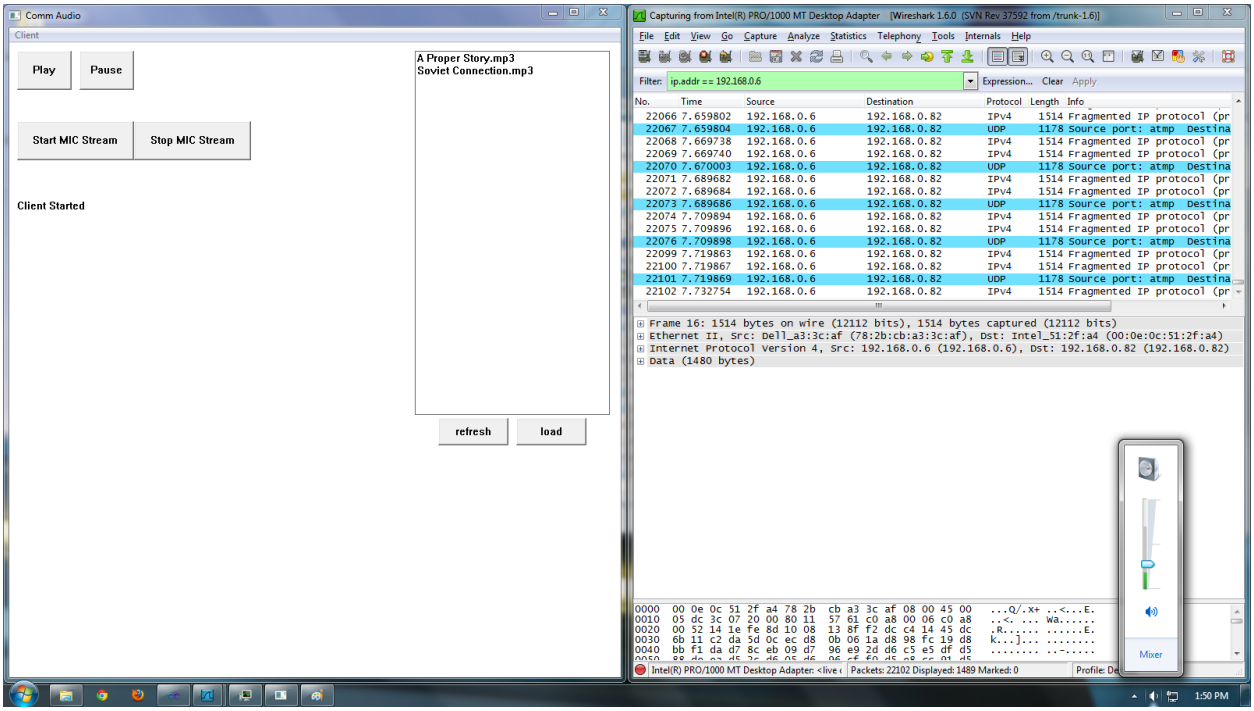


Figure 2 - Server Streaming to Single Client:

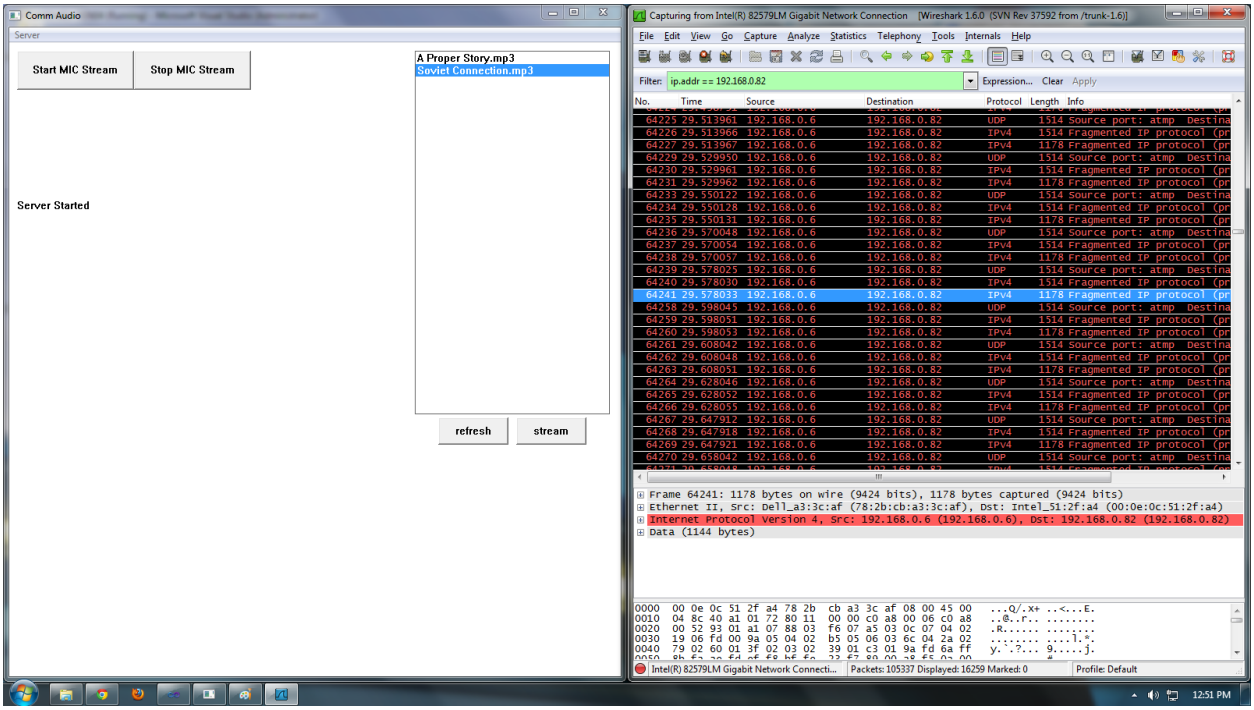


Figure 3 - Server Streams Audio to Multiple Clients Using Multicast:

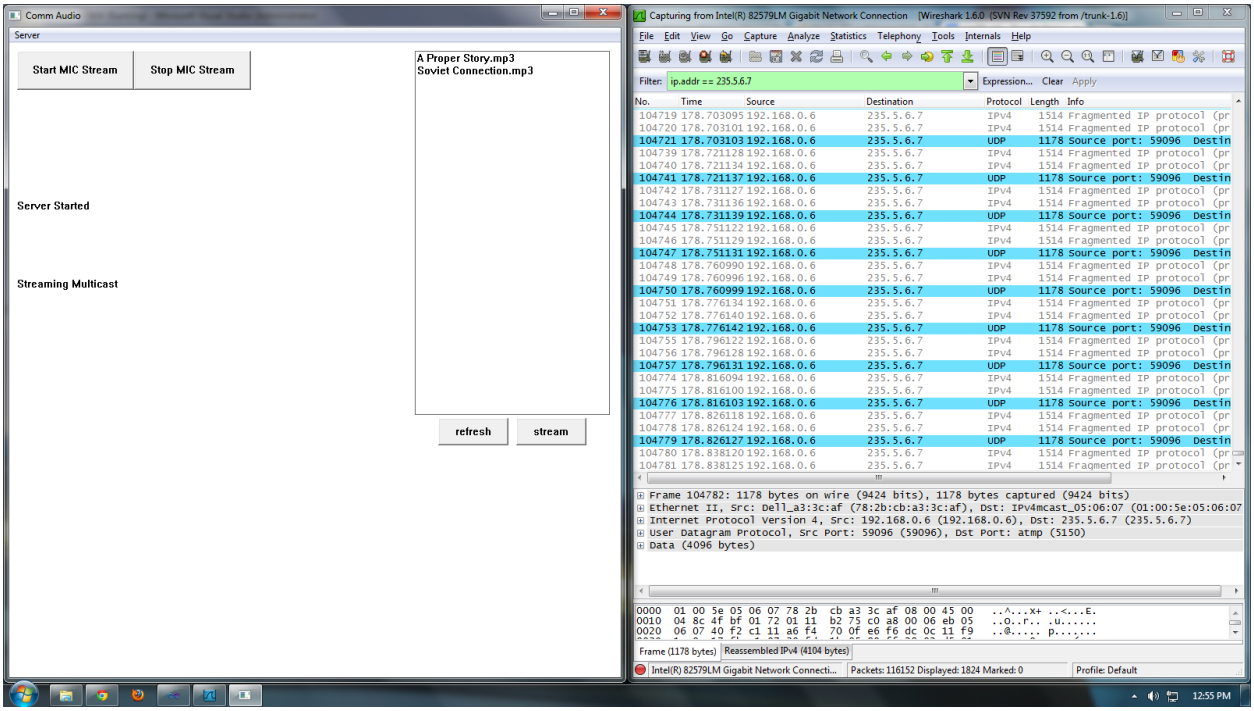


Figure 4 - Client Receives Audio From the Multicast Stream:

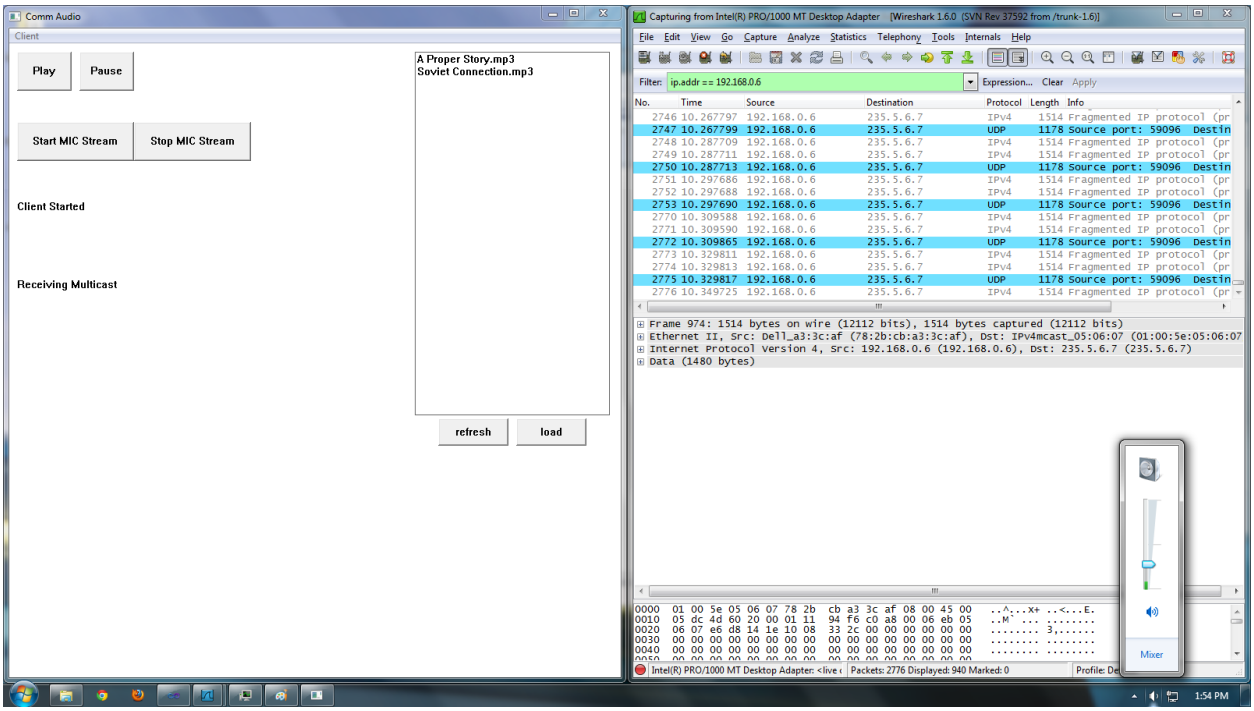


Figure 5 - Server Streaming and Receiving Microphone Data (one to one):

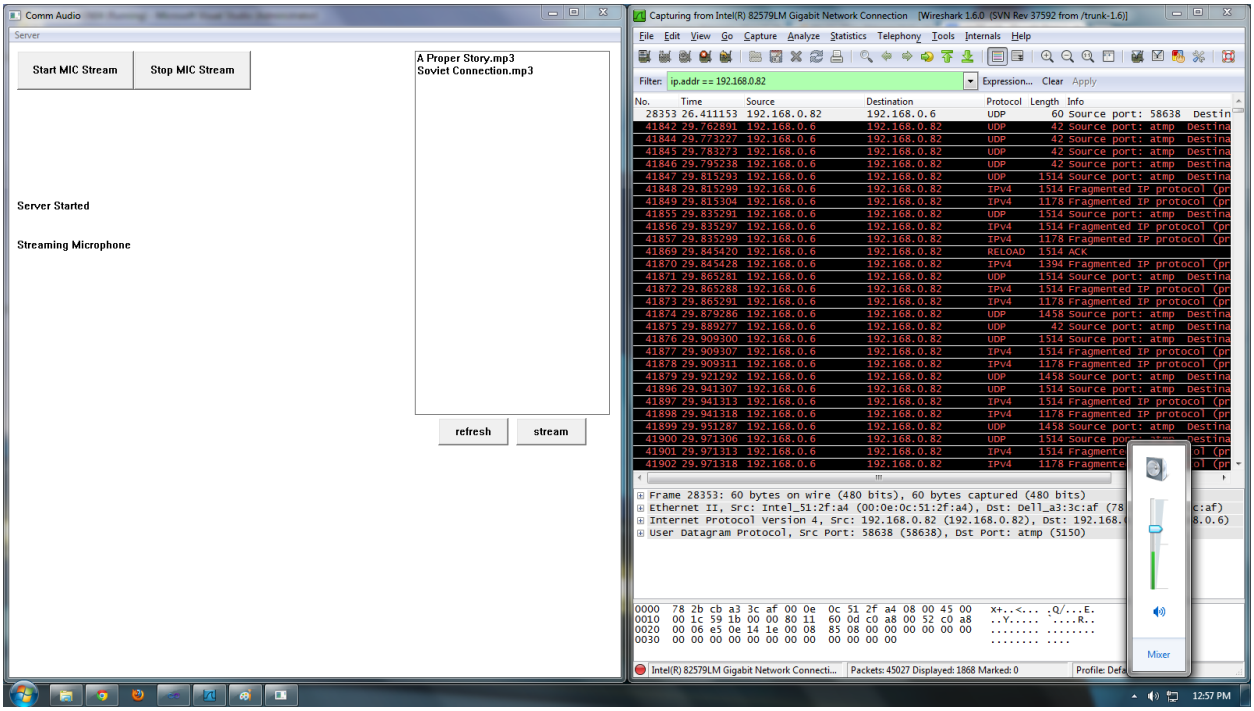


Figure 6 - Client Receiving and Streaming Microphone Data (one to one):

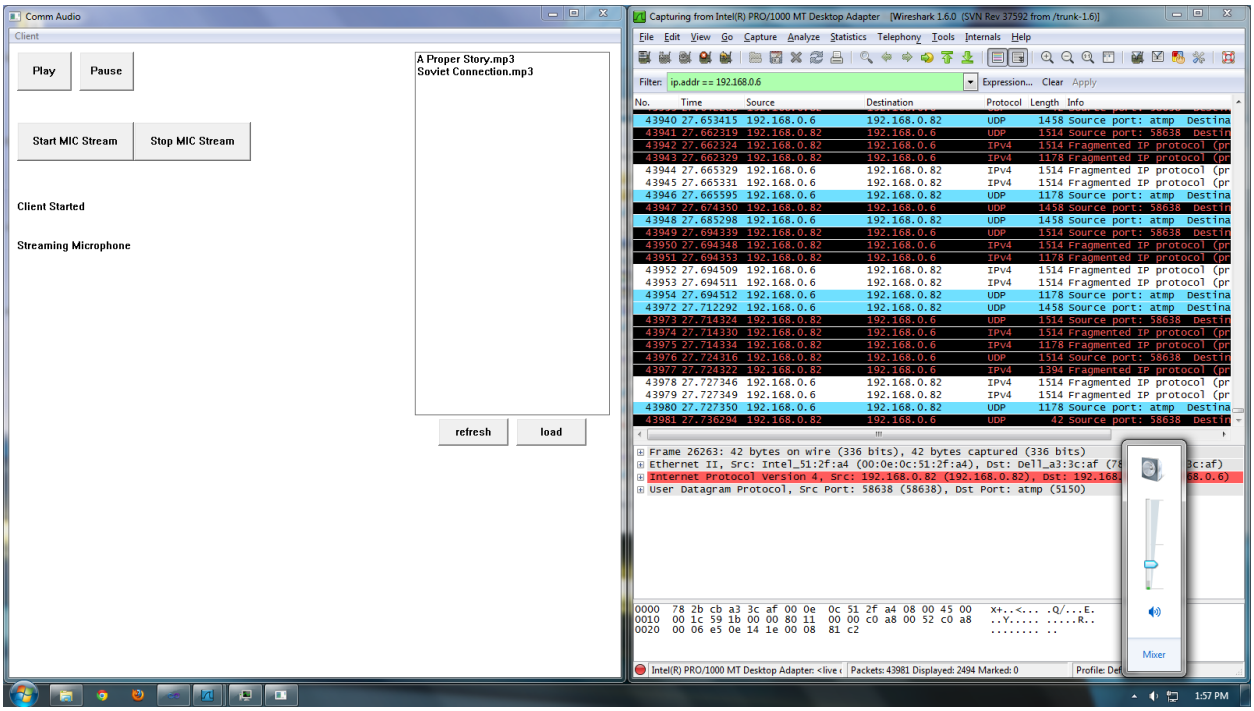


Figure 7 - Server Streaming Multicast Microphone Data (one to many):

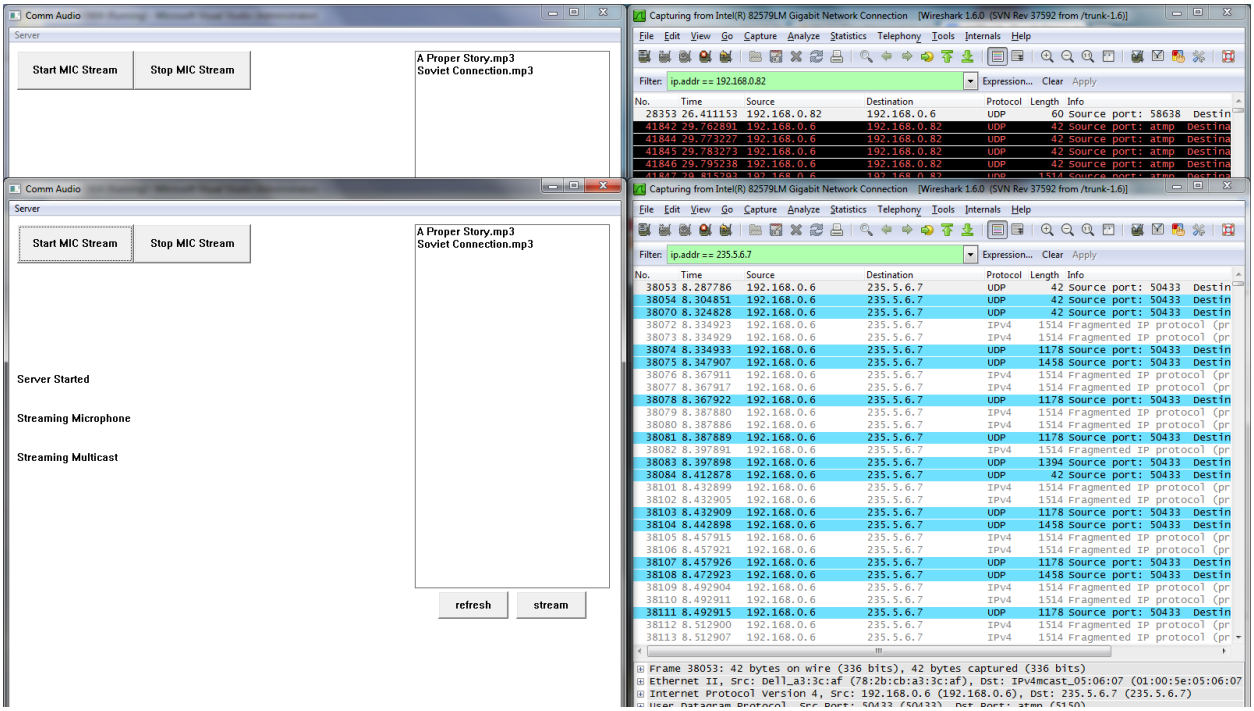


Figure 8 - Client Pausing Song:

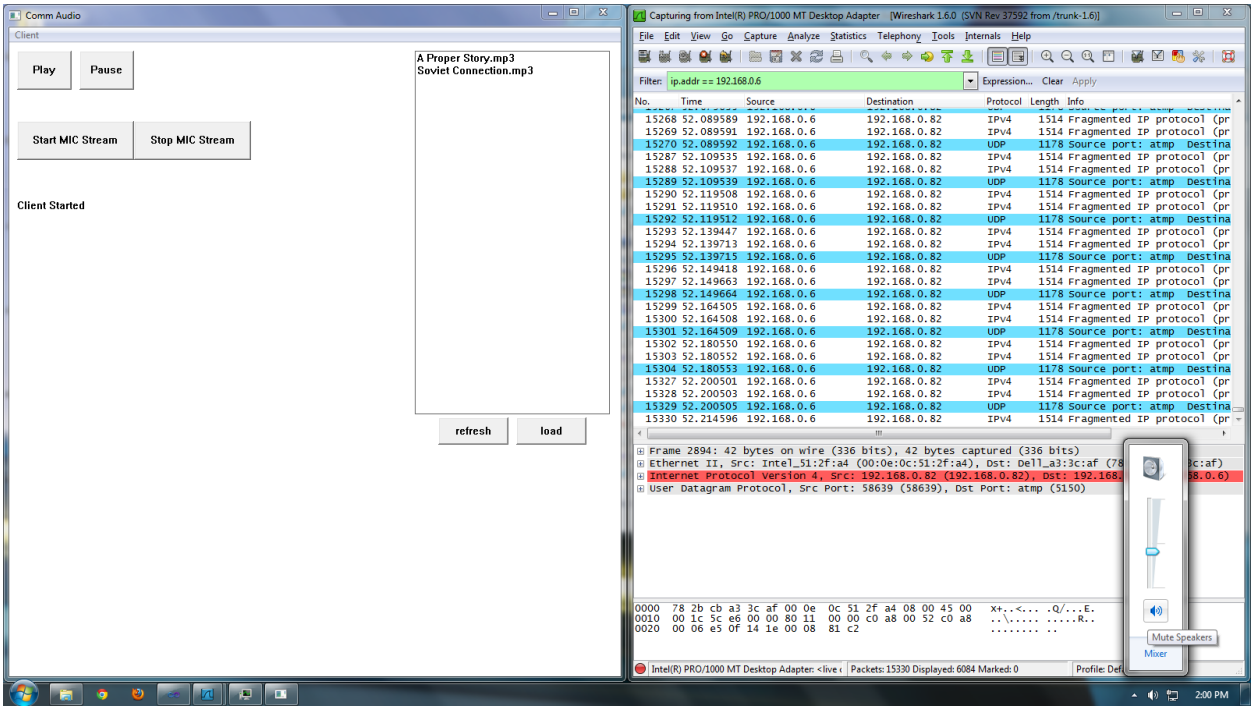
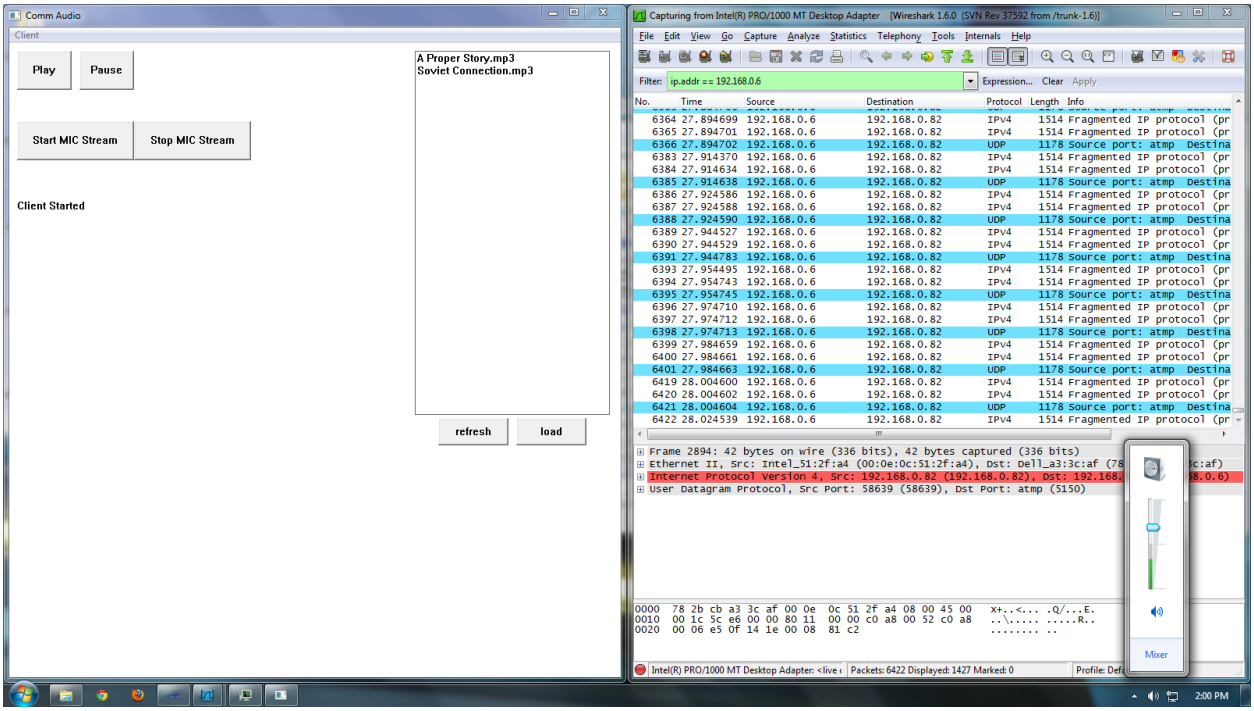


Figure 9 - Client Unpausing Song:


Comm Audio
Client
Play
Pause
Start MIC Stream
Stop MIC Stream
Client Started
A Proper Story.mp3
Soviet Connection.mp3
refresh
load
Capturing from Intel(R) PRO/1000 MT Desktop Adapter [Wireshark 1.6.0 (SVN Rev 37592 from /trunk-1.6)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: ip.addr == 192.168.0.6
Expression... Clear Apply
No. Time Source Destination Protocol Length Info
6364 27.894699 192.168.0.6 192.168.0.82 IPv4 1514 Fragmented IP protocol (pr
6365 27.894701 192.168.0.6 192.168.0.82 IPv4 1514 Fragmented IP protocol (pr
6366 27.894702 192.168.0.6 192.168.0.82 UDP 1178 Source port: atmp Destina
6383 27.914370 192.168.0.6 192.168.0.82 IPv4 1514 Fragmented IP protocol (pr
6384 27.914634 192.168.0.6 192.168.0.82 IPv4 1514 Fragmented IP protocol (pr
6385 27.914638 192.168.0.6 192.168.0.82 UDP 1178 Source port: atmp Destina
6386 27.924586 192.168.0.6 192.168.0.82 IPv4 1514 Fragmented IP protocol (pr
6387 27.924588 192.168.0.6 192.168.0.82 IPv4 1514 Fragmented IP protocol (pr
6388 27.924590 192.168.0.6 192.168.0.82 UDP 1178 Source port: atmp Destina
6389 27.944527 192.168.0.6 192.168.0.82 IPv4 1514 Fragmented IP protocol (pr
6390 27.944529 192.168.0.6 192.168.0.82 IPv4 1514 Fragmented IP protocol (pr
6391 27.944783 192.168.0.6 192.168.0.82 UDP 1178 Source port: atmp Destina
6393 27.954495 192.168.0.6 192.168.0.82 IPv4 1514 Fragmented IP protocol (pr
6394 27.954743 192.168.0.6 192.168.0.82 IPv4 1514 Fragmented IP protocol (pr
6395 27.954745 192.168.0.6 192.168.0.82 UDP 1178 Source port: atmp Destina
6396 27.974710 192.168.0.6 192.168.0.82 IPv4 1514 Fragmented IP protocol (pr
6397 27.974712 192.168.0.6 192.168.0.82 IPv4 1514 Fragmented IP protocol (pr
6398 27.974713 192.168.0.6 192.168.0.82 UDP 1178 Source port: atmp Destina
6399 27.984659 192.168.0.6 192.168.0.82 IPv4 1514 Fragmented IP protocol (pr
6400 27.984661 192.168.0.6 192.168.0.82 IPv4 1514 Fragmented IP protocol (pr
6401 27.984663 192.168.0.6 192.168.0.82 UDP 1178 Source port: atmp Destina
6419 28.004600 192.168.0.6 192.168.0.82 IPv4 1514 Fragmented IP protocol (pr
6420 28.004602 192.168.0.6 192.168.0.82 IPv4 1514 Fragmented IP protocol (pr
6421 28.004604 192.168.0.6 192.168.0.82 UDP 1178 Source port: atmp Destina
6422 28.024539 192.168.0.6 192.168.0.82 IPv4 1514 Fragmented IP protocol (pr
Frame 2894: 42 bytes on wire (336 bits), 42 bytes captured (336 bits)
Ethernet II, Src: Intel_51:2f:a4 (00:0e:0c:51:2f:a4), Dst: Dell_a3:3c:af (78
Internet Protocol Version 4, Src: 192.168.0.82 (192.168.0.82), Dst: 192.168.
User Datagram Protocol, Src Port: 58639 (58639), Dst Port: atmp (5150)
0000 78 2b cb a3 3c af 00 0e 0c 51 2f a4 08 00 45 00 x+..<... .Q/...E.
0010 00 1c 5c e6 00 00 80 11 00 00 c0 a8 00 52 c0 a8 ..\..... .....R..
0020 00 06 e5 0f 14 1e 00 08 81 c2 ....
Intel(R) PRO/1000 MT Desktop Adapter: <live
Packets: 6422 Displayed: 1427 Marked: 0
Profile: Defa
Mixer
2:00 PM