Computer Networks, Fall 2023

Instructors: Shashi Prabh

Lab 4: Sending/receiving audio/video files over UDP

Due: March 17 (section 2), 20 (section 1)

1 Objective

In this lab you will gain familiarity with sending music and video at a suitable data rate over UDP socket. As you know, UDP is a connectionless protocol. The protocol does not have provision for the receiving node to send acknowledgments back to the sender. Accordingly, the sender does not automatically retransmit lost packets. Hence sending data at too high a rate can lead to packet loss. On the other hand, sending data at too low a rate will cause non-smooth playback if the client plays the audio/video file while receiving it. You will also evaluate the performance of your implementation. Use `client_udp.c` and `server_udp.c` available at the course website to get started. Have fun!

2 Client

After creating socket, the client sends `GET` in its first (and only) message to the server to get the audio or video feed. You can redirect the incoming data in a file and play from the file using ALSA, SOX, VLC or some other player. For example, the command `cat out.wav | vlc -` causes VLC to play the stream redirected to `out.wav`. We will be running servers which you can use to test your client.

3 Server

The server keeps listening for requests in an infinite loop. It sends a given file if it receives `GET` from the client. Sending any other string will have no effect. The server signals the end of message by simply terminating the connection. For this lab, you can send the same file to all clients. You can not, however, assume the file size to be either known or small. Your server should keep reading a block of data and sending that block to the client until the end of file is encountered or the server is terminated. Note that UDP has a payload limit of 65.5 KB.

4 Performance

Send a large file (10 MB or larger) from a server to a client. Both could be running on the same machine, or preferably connected over wireless link (try both!). Compare the size of sent and received files. Explain any discrepancy. One (easy) way to regulate the sending data rate is to insert delay between consecutive `sendto()`s using `nanosleep()` call (for more information do `man nanosleep`). Extend the server code so that it takes the UDP payload size and `nanosleep()` parameters as arguments. Calculate the data rates for a few settings and plot the received file size as a function of data rate.

Sending audio/video at too low a rate can lead to bad user experience. For a few audio/video files of your choice, determine appropriate data rate. Vary (decrease as well as increase) the data rate and observe its effect by playing the received data in real time. Report the attributes of data (type, bit rate(s) and encoding(s)) along-with your settings, that is, payload size and sleep delays, and the resulting data rate needed for playing the data smoothly in real time.

5 Extra credit 1 - 20%

Make your server serve multiple clients simultaneously.

6 Extra credit 2 - 20%

Make your server determine appropriate data rate automatically from the audio/video file. On the client side, stream the data directly without storing it in a file. You'll need to buffer some data before starting to play. For this part, you need to demonstrate delivery of live audio, preferably live video, to client.

Evaluation

- ☐ Server can send some specified file to any number of clients. [3 pts, latest by March 10/12] TA: _____
- ☐ Client can store the received file. Received music or video files can be played back. [4 pts, latest by March 10/12] TA: _____
- ☐ The server correctly paces the datagrams: the received file can be played concurrently. [3 pts] TA: _____
- ☐ Multithreaded server. [2 pts] TA: _____
- ☐ The server correctly paces the datagrams automatically. [3 pts] TA: _____