Kent Mark, Christian Williams

Cpre 489 – Lab 2

9/26/21

Lab 2 Report

Lab Summary:

In this lab we were tasked with coming up with a program that would use the UDP protocol to forward packets from a source to a destination. The medium we chose to use was VLC Media Player. We were also tasked with implementing code that would allow users to add some amount of packet loss to the stream so that they could observe it's effects. We wrote a C file named Server.c that took five values from users. These values were: source IP, source port, destination IP, destination port, and loss rate. The program works by listening in on the port number that the user inputs as the source port parameter and then captures the packets provided by the source IP, then the program will forward the packets it received based on the loss rate the user specifies.

Lab Exercises:

1. Increasing the loss rate had very negative effects on the video stream. The higher we increased the loss rate the more frames we lost in the process. With an increased loss rate, we saw that the video itself became shorter since not all of the packets are being sent, which also caused the video to skip around a lot and the audio becoming choppy.

2. TCP is definitely viable for video streaming, but that viability depends on how one intends to use it. If someone wants to stream an existing file, then TCP would be a better option because then someone can pre-load a video and make use of re-transmission to fix any data loss that occurs. However, if someone tried to do the same thing for a live stream this would cause a huge delay because any lost packets would have to be re-sent before any more of the stream can be loaded. This would cause a lot of starting and stopping because the packets are being lost. Stemming from this UDP is a lot better for this because it is much faster, and since the live stream wouldn't be saved to storage it would be impossible to pre-load any information.

Copy of Code:

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdlib.h>

struct sockaddr_in srcIPAddr, dstIPAddr;
```

```
int main(int argc, char **args){
        int srcIP = inet_addr(args[0]);
        int srcPO = atoi(args[1]);
        int dstIP = inet_addr(args[2]);
        int dstPO = atoi(args[3]);
        int lossRate = atoi(args[4]);
        srand(time(NULL));

        int fd, n, sd;
        const char* buff[61440];

        srcIPAddr.sin_family = AF_INET;
        srcIPAddr.sin_port = srcPO;
        srcIPAddr.sin_addr.s_addr= srcIP;

        dstIPAddr.sin_family = PF_INET;
        dstIPAddr.sin_port = dstPO;
        dstIPAddr.sin_addr.s_addr = dstIP;

        #recieving socket
        fd = socket(PF_INET, SOCK_DGRAM,0);
         if(fd < 0){
          perror("Socket Error");
        }

        bind(fd, &srcIPAddr, sizeof(srcIPAddr));

        #transmitting socket
        sd = socket(PF_INET, SOCK_DGRAM,0);
         if(sd < 0){
          perror("Socket Error");
        }

        bind(sd, &dstIPAddr, sizeof(dstIPAddr));

        int randLoss = 0;



        for(;;){

          randLoss = rand()%1001;
          recvfrom(fd, buff, sizeof(buff), 0 , (struct sockaddr
*)&srcIPAddr, sizeof(srcIPAddr));




          if(randLoss < (1001-lossRate)){
            sendto(sd,buff, sizeof(buff),0,(struct sockaddr* )&dstIPAddr,
sizeof(dstIPAddr));
          }
```

```
			}
		}
```

Effort Levels:

Kent: 50%

Christian: 50%