Thigulla Vamsi Krishna
COE18B056

Objective:
A TCP and a UDP program to send an MP3 file and to use network commands to analyse the latency.

Approach:
Open the file on the client side and store the contents of the file into a buffer. Then send the buffer to the server.

- **Note:**
  - Since the file is a .mp3 file, the content is stored in binary format, thus opening the file for reading should be done with rb mode.
  - Take a buffer of sufficient size.
  - As the data is binary, '\0' is a valid character in the file and thus it should not be used as a termination for a loop while reading.
  - The data of the buffer is not a string.

On the server side, open a file in write binary mode (wb) and then store the contents of the received buffer into the file.

After copying the contents into the file, close the file and socket (for secure transfer of contents).

Use cmp to compare the output file with the input mp3 file to verify that the input file has been copied perfectly.

Source Code:

TCP:

Server:

```
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/time.h>
#include <string.h>
#include<time.h>

void die(char *message)
{
   printf("%s",message);
```

```
    exit(1);
}

int main()
{
    int socket_music,conn_socket_music,num_pack=0;
    struct timeval begin,end; //To measure latency
    double tmp=0.0,avg=0.0;
    char buffer[1024] = "",ack;
    FILE *track;
    track = fopen("CamelotCopy1.mp3","wb");

    if(!track)
        return(printf("Error! Unable to open file.\n"));

    socket_music = socket(AF_INET,SOCK_STREAM,0);
    if(socket_music==-1)
        die("Socket creation unsuccessful.\n");


    struct sockaddr_in Serve_Port,Client_Port;
    memset(&Serve_Port,0,sizeof(Serve_Port));

    Serve_Port.sin_family = AF_INET;
    Serve_Port.sin_port = htons(8080);
    Serve_Port.sin_addr.s_addr = inet_addr("0");

    if(bind(socket_music,(const struct sockaddr*)&Serve_Port,sizeof(Serve_Port)))
        die("Error! Binding unsuccessful.\n");

    if(listen(socket_music,5))
        die("Error! Connections queue exceeding the limit.\n");

    socklen_t cli_len = sizeof(Client_Port);
    conn_socket_music = accept(socket_music,(struct sockaddr*)&Client_Port,&cli_len);

    if(conn_socket_music < 1)
        die("Error! Cannot accept the connection.\n");

    int n;

    while(1)
    {
```

```
        gettimeofday(&begin,NULL);
        read(conn_socket_music,&buffer,sizeof(buffer));
        read(conn_socket_music,&n,sizeof(n));
        gettimeofday(&end,NULL);

        num_pack++;
        tmp = end.tv_usec - begin.tv_usec;
        if(tmp>0)
           avg = avg - avg/num_pack +tmp/num_pack;

        if(!strcmp(buffer,"END") && n==0)
           break;

        fwrite(buffer,1, n, track);
    }

    printf("Average latency:%lf\n",avg);

    close(socket_music);
    close(conn_socket_music);
    fclose(track);
}
```

Client:

```
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>
#include<time.h>

void die(char *message)
{
    printf("%s",message);
    exit(1);
}

int main()
{
    int socket_music,n,seconds;
```

```
   char buffer[1024] ="",ack;
   FILE *track;

   socket_music = socket(AF_INET,SOCK_STREAM,0);
   if(socket_music==-1)
      die("Error! Socket not opened.\n");

   struct sockaddr_in Serve_Port;
   Serve_Port.sin_family = AF_INET;
   Serve_Port.sin_port = htons(8080);
   Serve_Port.sin_addr.s_addr = inet_addr("0");

   if(connect(socket_music,(const struct sockaddr*)&Serve_Port,sizeof(Serve_Port)))
      die("Error! Unable to connect to server.\n");

   track=fopen("Camelot.mp3","rb"); //As the media file is stored in binary, thus the file is
opened in rb mode i.e. read binary mode

   while ((n = fread(buffer, 1, 1024, track)) > 0) //Store the file into the buffer
   {
      write(socket_music,&buffer,sizeof(buffer));
      write(socket_music,&n,sizeof(n));
      usleep(500); //To slow down transmittin speed from client as here, client is
transmitting faster than server
   }

   strcpy(buffer,"END"); //To signify the end of the file
   n=0;
   write(socket_music,&buffer,sizeof(buffer));
   write(socket_music,&n,sizeof(n));
   fclose(track);
   close(socket_music);
}

//sleep is used to prevent packet loss due to client transmitting faster than server can
transmit
```

UDP:

Server:

```
#include <unistd.h>
#include <stdio.h>
```

```
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <sys/time.h>
#include <arpa/inet.h>
#include <string.h>

void die(char *message)
{
   printf("%s",message);
   exit(1);
}

int main()
{
   int socket_music,n,num_pack=0;
        struct timeval begin,end;
   double tmp=0.0,avg=0.0;
   char buffer[1024] ="";
        socket_music = socket(AF_INET , SOCK_DGRAM , 0);
        struct sockaddr_in Serve_Addr,Client_Addr;
        FILE *track;
   track = fopen("CamelotCopy2.mp3","wb");

        if (socket_music == -1)
        {
                printf("Could not create socket");
        }

        Serve_Addr.sin_family = AF_INET;
        Serve_Addr.sin_port = htons(8080);
        Serve_Addr.sin_addr.s_addr = inet_addr("0");

        bind(socket_music, (const struct sockaddr *)&Serve_Addr,sizeof(Serve_Addr));

        int len = sizeof(Client_Addr);

        while(1)
        {       //Write the contents of the file, received through the client, into the buffer
                gettimeofday(&begin,NULL);
                recvfrom(socket_music, (char *)buffer, 1024, MSG_WAITALL, ( struct
sockaddr *) &Client_Addr, &len);
```

```
                recvfrom(socket_music, (int *)&n, sizeof(n), MSG_WAITALL, ( struct
sockaddr *) &Client_Addr, &len);
                gettimeofday(&end,NULL);

                num_pack++;
      tmp = end.tv_usec - begin.tv_usec;
      if(tmp>0)
         avg = avg - avg/num_pack +tmp/num_pack;

      if(!strcmp(buffer,"END") && n==0)
         break;

      fwrite(buffer,1, n, track); //Write from the buffer to the file
         }

         printf("Average latency:%lf\n",avg);

         close(socket_music);
         return 0;
}
```

Client:

```
#include <unistd.h>
#include <stdio.h>
#include <sys/socket.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <string.h>
#include <arpa/inet.h>

void die(char *message)
{
   printf("%s",message);
   exit(1);
}

int main()
{
   int socket_music,len,n;
   char buffer[1024]="Hello server!";
         socket_music = socket(AF_INET , SOCK_DGRAM , 0);
         struct sockaddr_in Serve_Addr;
```

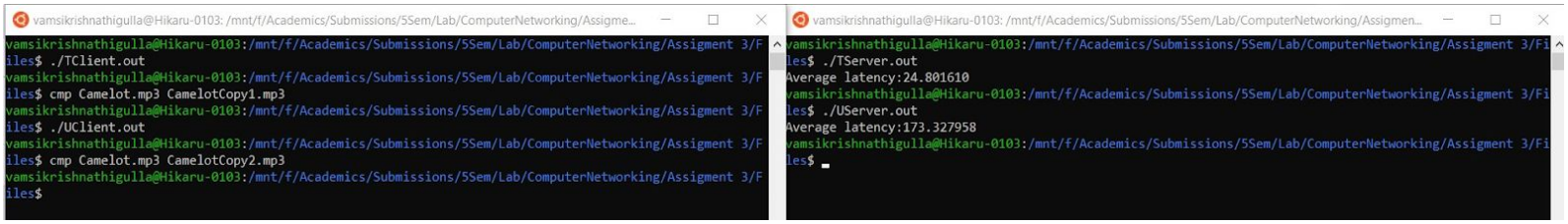
```
        FILE *track;

        track=fopen("Camelot.mp3","rb"); //Open file in read binary mode as the mp3 file
is stored in binary

        if (socket_music == -1)
        {
                printf("Socket could not be created.\n");
        }

        Serve_Addr.sin_family = AF_INET;
        Serve_Addr.sin_port = htons(8080);
        Serve_Addr.sin_addr.s_addr = inet_addr("127.0.0.9");


        while ((n = fread(buffer, 1, 1024, track)) > 0)
   {
                sendto(socket_music,buffer,sizeof(buffer),0,(const struct
sockaddr*)&Serve_Addr,sizeof(Serve_Addr));
                sendto(socket_music,&n,sizeof(n),0,(const struct
sockaddr*)&Serve_Addr,sizeof(Serve_Addr));
      usleep(100);
   }
   strcpy(buffer,"END"); //Signify end of file
   n=0;
        sendto(socket_music,buffer,sizeof(buffer),0,(const struct
sockaddr*)&Serve_Addr,sizeof(Serve_Addr));
        sendto(socket_music,&n,sizeof(n),0,(const struct
sockaddr*)&Serve_Addr,sizeof(Serve_Addr));

        close(socket_music);
        return 0;
}
```

While compiling, use makefile to make the executable .out files for server and client programs of the TCP and UDP protocols.

```
vamsikrishnathigulla@Hikaru-0103:/mnt/f/Academics/Submissions/5Sem/Lab/ComputerNetworking/Assigment 3/Files$ gcc -o UServer.out UDP_Server.c
vamsikrishnathigulla@Hikaru-0103:/mnt/f/Academics/Submissions/5Sem/Lab/ComputerNetworking/Assigment 3/Files$ gcc -o UClient.out UDP_CLient.c
vamsikrishnathigulla@Hikaru-0103:/mnt/f/Academics/Submissions/5Sem/Lab/ComputerNetworking/Assigment 3/Files$ gcc -o TServer.out TCP_Server.c
vamsikrishnathigulla@Hikaru-0103:/mnt/f/Academics/Submissions/5Sem/Lab/ComputerNetworking/Assigment 3/Files$ gcc -o TClient.out TCP_Client.c
vamsikrishnathigulla@Hikaru-0103:/mnt/f/Academics/Submissions/5Sem/Lab/ComputerNetworking/Assigment 3/Files$
```

Thigulla Vamsi Krishna
COE18B056

Output:



We use cmp function to check that the original file is copied perfectly into the output file without any differences.

Latency:

- TCP : 24.801610 ms
- UDP : 173.327958 ms

Compare this with latency found through usage of ping command.