Project 2 Report

By

Eren Ali Aslangiray, Mahmut Sami Özmen

At first, we developed 4 py file for sending and receiving in both udp and tcp protocols. Tcp was easy because it has handshake protocol which is good for reliability and easy to develop. All the receiver files must work first to get sent files.

**For part A**; we successfully do GUI 1:1 from looking to project description file that you sent. And also, we successfully transferred the picture in each protocol.

**-receiver.py** (which is for tcp) consists a function called connectTCP(path). So basically, it gets a path value for writing the file. After that it establishes a socket and read the files as they come with 1000 bytes per iteration and counts the packets with a counter. After the process done, it gets the performance measures with respect to time and prints them on terminal and closes the socket.

**-sender.py** (which is for tcp) consists a function called TCPsender(host,path). So basically, it gets a path value for writing the file and the ip value of destination as host. After that it establishes a socket and read the file it sends them with 1000 bytes per iteration and counts the packets with a counter. After the process done, it gets the performance measures with respect to time and prints them on terminal and closes the socket.

**-receiverudp** consists a function called receiverudp(path). So basically, it gets a path value for writing the file but this time it establish a udp connection. It don’t have a handshake protocol so receiver have one shot to getting the files. After that it establishes a socket and read the files as they come with 1000 bytes per iteration and counts the packets with a counter. After the process done, it gets the performance measures with respect to time and prints them on terminal and closes the socket.

**-senderudp** consists a function called UdpSender(host,path). It will shoot the file like a bullet to receiverudp. It doesn’t care about wheter there is a connection or receiver gets them. It sends the file with 1000 bytes per iteration and counts the packets with a counter. After the process done, it gets the performance measures with respect to time and prints them on terminal and closes the socket.

**For part B**; we did split the file into packages that is big enough for only 1000 bytes by giving transfer buffer value 1000. Also, we give a variable named counter to track number of packages that is send. Then at the receiver part we reconstructed the file with a while loop. For time and speed calculation we used import time to set up timer to calculate the process. For udp, it needed a special attention. We measured the initial file before sending, and after the process ended we compared packet numbers to get the idea of number of loss packets and corrupted packages. Also, as we mentioned, we reported performance measures at the end of each run.

**For part C**; we wrote this beautiful report. ☺