Assignment Report

Sanket Khajuria
Anuj Verma
Anvit
Dheeraj

Explanation:

For this project, we have to implement App8. In App8, we have to Measure the capability of the server to process data (also referred to as server throughput). The client in this application exchanges several large messages (actually files) with the server that are then processed by the server, and the result returned to the client.The result here is a packet sent to the client. The result will depend upon the file size and the nature of the processing, which in out case is, first writing into the file and then compressing the file. And then, we repeat the experiment with files that are 1 MB, 10 MB and even 100MB.
The language used for implementing this is Python3. We wrote two programs for this. One program handles the client side and one handles the Server side.
In client's program, we first store the servers's ip address and its port, which we will use to send the data. Then we will ask the user to input the number of files to send. Then we will store the name of all the files which user wants to send. Once we receive all the file names, we will try to connect to the server using the ip address and port stored. When the connection with the server is established, we are ready to send files to the server. For sending the files, we first need to break the files into small packets and then send those packets over to the designation. Our packet contains following information: Packet sequence number, packet acknowledge number, length of the file name, file name, actual data. Here our packet size is 1024 bytes or 1 KB. The protocol used to send the files over resembles alternating-bit protocol. Once we encode and send the packet over, we set a timeout of 2 seconds. This timeout is used to give client side time to wait for receiving the acknowledge of the packet that it just send. In case it does not receive the desired acknowledgement, the timer rans out and the packet is re-transmitted. If the desired acknowledgement is received, the next packet is constructed, encoded and sent to the server. When we reach the end of the file for a file, we switch to the next file and start this process from the start.
In server's program, we declare its port and socket which client uses to connect to the server. The server socket is started in listening mode. It will accept the listening request it receives from client's side. Then server will start receiving the data packet from the client. Upon receiving the data, we will decode the packet and extract the different parts of the

packet. We then save the file on server's side with same name as the name on client's side. After saving the file, we open this file in 'write' mode to write to this file. After modifying the file, we close the file. Now we again want to compress this file and note down the time taken to compress the file. For this we first start the timer. Then we again open the file in 'read' mode. Them we compress the file. After compressing the file, we stop the timer. The time difference between start and stop of the timer is the required processing time. We get this time. Now we construct a packet, with the result time and send it to the client. After receiving this packet client will figure out how much time it took to process the file.

Analysis:

For this project, we used different file sizes to determine the time taken by the server to process them. Following are the statistics for time taken(in seconds) with respect to file sizes:

- 1MB file took 0.029 seconds to process.
- 2MB file took 0.049 seconds to process.
- 3MB file took 0.071 seconds to process.
- 4MB file took 0.089 seconds to process.
- 5MB file took 0.111 seconds to process.
- 6MB file took 0.133 seconds to process.
- 8MB file took 0.178 seconds to process.
- 10MB file took 0.220 seconds to process.
- 12MB file took 0.269 seconds to process.
- 15MB file took 0.347 seconds to process.
- 20MB file took 0.443 seconds to process.
- 30MB file took 0.385 seconds to process.
- 50MB file took 0.892 seconds to process.