**CSCE 313-509**

**PA2 Report**

Vidhur Potluri  
UIN: 626007235

**Introduction**

The aim of this assignment is to establish a connection between the client and server terminals, and have the client request different ECG data and files from the server. The FIFOreqchannel .cpp and .h files were used to create this connection. The client requests to be sent to the server are given by the user through the getopt() function. The time taken by the server to process these requests of varying buffer capacities was also calculated using the gettimeofday() function. The buffer capacity of both server and client were the same for every request. The buffer capacity could be changed for both. However, the default capacity was 256 bytes.

**Requesting data points**

Individual data points were requested by the client by sending a signal which was written as a buffer of type datamsg. The variable returned by the server for each request was read using cread() and cast as a double and displayed. All of the ECG values for patient 1 (“BIMDC/1.csv) were requested and saved to a file (“received/x1.csv”). Since only 1 ECG value was processed in each request, the process took a relatively long time of 119.617 seconds.

The files were then compared using diff tool on linux and were found to contain the exact same values.

**Requesting files**

Files were requested by sending a character array signal. The character array contained a filemsg, the filename, and a NULL delimiter. The NULL delimiter allowed the server to recognize the end of the buffer. Because of the buffer capacity limit, the file had to be requested in smaller sizes. First, to establish the number of data requests to be sent, a filemsg with length of 0 and offset of 0 was sent to the server. The server then returned the filesize, and the file data was divided into segments with size of the buffer capacity or lower. This data was then copied to a file with the same name in the “received/” folder.

Similar to the individual data request process, the diff tool was used to compare the two files and found that the files were exactly the same. I tested the transfer of files of type .dat, .txt and .csv files. The transfer of data was perfect in all cases.

The transfer of 1.csv took only 0.407722 seconds (buffer capacity = 256 bytes) which is significantly lower (119.617 seconds) than requesting individual data points from the file. This difference in time occurred because of the significantly lower number of requests to the server. When the buffer capacity was increased to 10000 bytes, the process only took 0.370167 seconds.

I used this method to transfer a .dat file of 10 MB. The times that I recorded to transfer the files with different bottlenecks are as follows:

* 100 bytes: 32.0482 seconds
* 256 bytes: 12.5913 seconds
* 1000 bytes: 3.51269 seconds
* 10000 bytes: 0.487584 seconds

**Requesting a new channel**

To request a new channel, I sent a buffer containing a message of type NEWCHANNEL\_MSG using cwrite(). This requested the server to create a new channel. The channel was then connected to the server by using the same name the server had used (Default name = “data1\_”). The integer was incremented each time a new channel request was processed. The new channel behaved exactly as the original. A quit message for each new channel created is also sent to the server.