Adrian Gamez-Rodriguez

CSCE 313-515

UIN:126009409

12/6/19

Report: MP5: The Data Server Moved Out

The purpose of this programming assignment is to build upon the previous programming assignment and change Request Channels to Network Request Channels, everything else stayed the same except for the dataserver.cpp file. By doing this, the client and server communicate across the network, using a single TCP connection. The method of implementation is socket programing. Essentially, the Network Request Channels class is a socket descriptor used to establish connection ends depending on which side the constructor is called. For the Client side the constructor parameters were a host name and a port number, for the data server the parameters were the port number and connection handler function. It is important to note that I changed the data server's network request channel constructor to include the backlog parameter of the listen() call. I could not have called the listen() call in the network request channel constructor without the backlog parameter. Also, note that I completed MP5 with the MP4's event driven worker thread. Below are my results for the experiment of varying the number of clients and sizes of the backlog buffer on the server.

NOTE: **To run client: type "**make all**" then type command** "./client -n <number of request per patient> -b <size of buffer> -w <number of network request channels> -h <hostname> -p <port number>**"**

NOTE: **To run dataserver: type command "**make all**", then type command**
"./datasever -p <port number> -b <backlog>**"**

NOTE: **I have default parameter in both ./client and ./dataserver**
n-10, -b50, -w 10, -h localhost -p 10101
and
-p 10101 -b 10

Varying Number of Client Terminals:

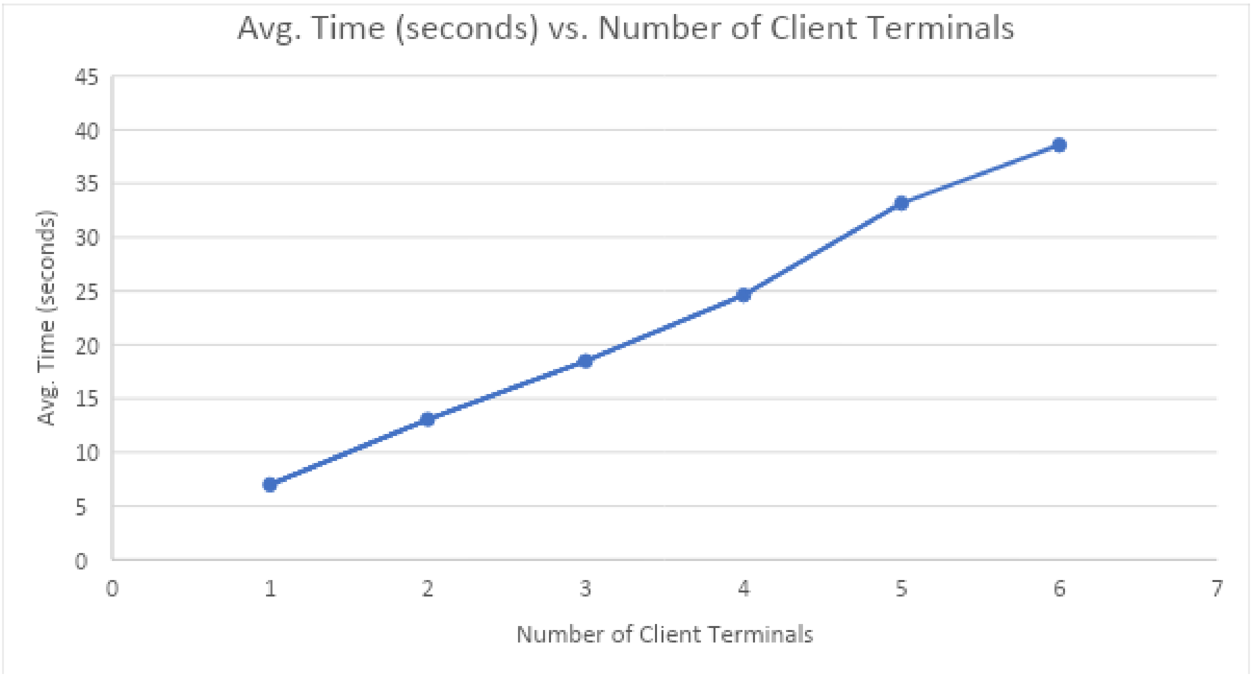
Constant: Backlog = 10

Constants per Client: (Each client has the same parameters)

Constant: Number of requests = 10000

Constant: Buffer size = 100;

Constant: Number of Network Request Channels = 124



Here we can have a single terminal that acts as the data server and various terminals acting as the clients. For this data collection, I varied the number of terminals, but kept the data requests the exact same. Each terminal is requesting 10,000 data per patient, using a buffer size of 100 and 124 worker threads. The graph shows the number of terminals compared to the average time taken for the data requests. We see that there is linear scaling for this data, and that as we add more terminals, the data server is receiving more data point requests. Clearly the more clients the slower the average time is for every client to finish.

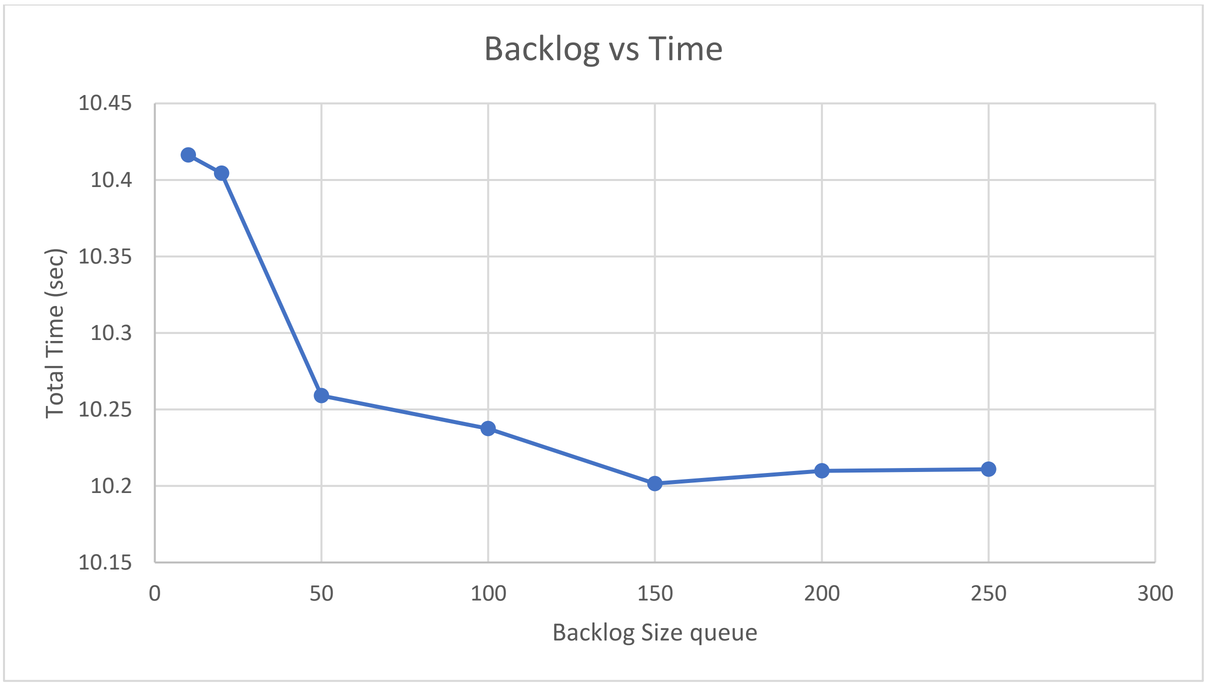
Varying Size of Back log:

Constant: Number of clients = 1

Constant: Number of requests = 10000

Constant: Buffer size = 100;

Constant: Number of Network Request Channels = 124



The graph above shows that increasing the backlog size queue has little to no effect on the total time of execution. It decreases time until it reaches a size of around 150, which makes sense because we only have 124 network request channels waiting to connect to the server. Thus, the size of the backlog has no affect after it exceeds the number of connections possible. This demonstrates that the most effective change in the server and client model is to increase the number of network request channels with a decent size buffer.