Iterative Socket Server

Chase Goodband, Alejandro Archuleta, Nick C

CNT4504 - Computer Networks & Distributed Processing

Professor: Scott Kelly

Introduction:

The purpose of the project is for us to implement a single threaded server and multi-threaded client to be used in a client server connection and then observe the effects of the turn-around times based on the number of clients we selected. The goal is to get accurate information read back for the turn-around times and to test and analyze different amounts of clients and see the results we are given back. We also need to plot that data and observe the changes in turn around times. As you read along you will come across the results of the project as well as analysis of that data and our conclusion of that data. Lastly you will come to find the lessons we have learned from this project and how we overcame problems we ran into.

Client-Server Setup and Configuration:

The client program is designed with two classes. First the client class which collects all the information from the user and creates the thread objects that connect to the server. Lastly, the client program prints out the total and average turn around time. The cthread class is extends the thread class so it inherits all the functionality that the thread class has. This class creates the client thread and puts it into start mode which automatically calls the run function. The run function has the socket object inside of it which creates the connection with the server and then sends the reques through a output stream and receives the data back from the server with an input stream. Lastly the cthread class also starts and stops the timer for each client and uses a get time function to send that information to the client class. The server program starts off by listening on the port that it is put into the main arguments. It then uses the accept function to create a connection with an incoming client. Using an input stream it reads in the request from the client and based off that request decides which information it needs send back to the client. After

sending the data to the client it flushes the the writer and looks for the next client connection. To run the Linux commands with the server program it creates a run command object based on the request sent from the client. The run command file execute the Linux commands and returns the data collected to the server program to then be sent to the client. One design decision was to collect some information in the main arguments. The reason we chose this was to have to ask the user for as little information as possible once the programs were running. Another design choose was to keep the classes in different files to make it easier to read and debug during implementation. The last major design decision made was to use a switch case function in the server program to decided what to request to run. This was an easy decision as we ask the user to input a number for the request and we just use the number from the user to be sent as the request to the server. The basic operation of the server is to start listening on the port that is chosen by the user. Then to connect to one client at a time and read the request to decide was information the server needs to execute and send back to the client. The clients basic operations is to collect the request and how many clients the user wants to send to the server. Once that is received it creates that many socket threads to connect to the server and receive the information requested by the user. Lastly, the client program starts a timer when the request is sent and ends the timer when the data is received then prints the time it took for that clients turn-around time. Once all the clients are finished the client program prints out the total and average turn around time from the clients.

Testing and Data Collection:

The way that we tested the iterative server was first by compiling and connecting the server to a port. Next, we would run the client application connecting to the same port as the

server and test each request with a varying amount of clients. We took the data returned for the total and average time from each test and entered it into a spreadsheet.

The table below is the data collected from testing the client-server connection for the total time. The top of the chart is the type of request with the number of clients going down the left side. The data collected is the total time in milliseconds for each request dependant on the number of clients.

Total time for request varrying number of clients

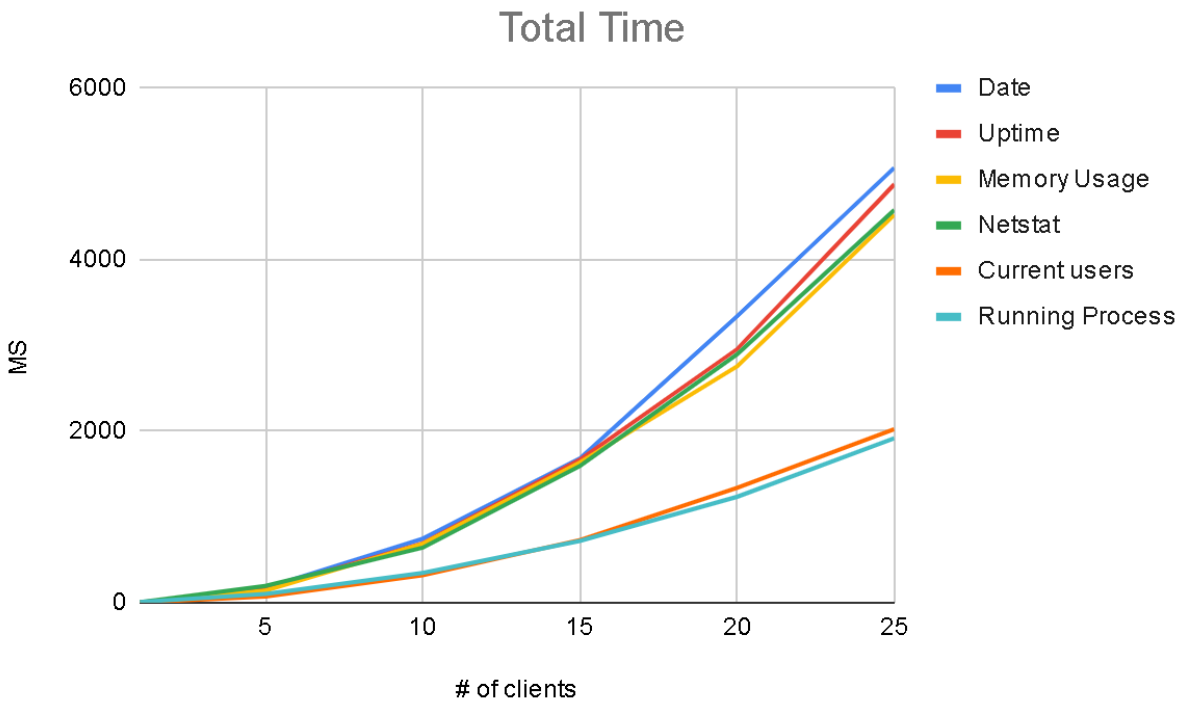
# of clients	Date	Uptime	Memory Usage	Netstat	Current users	Running Process
1	0	2	2	6	1	7
5	171	159	141	195	72	102
10	746	691	689	642	319	347
15	1682	1668	1636	1595	730	720
20	3341	2958	2758	2897	1339	1235
25	5072	4880	4525	4579	2025	1918

The table below is set up the same way as the previous table. With the data collected being the average turn-around time in milliseconds.

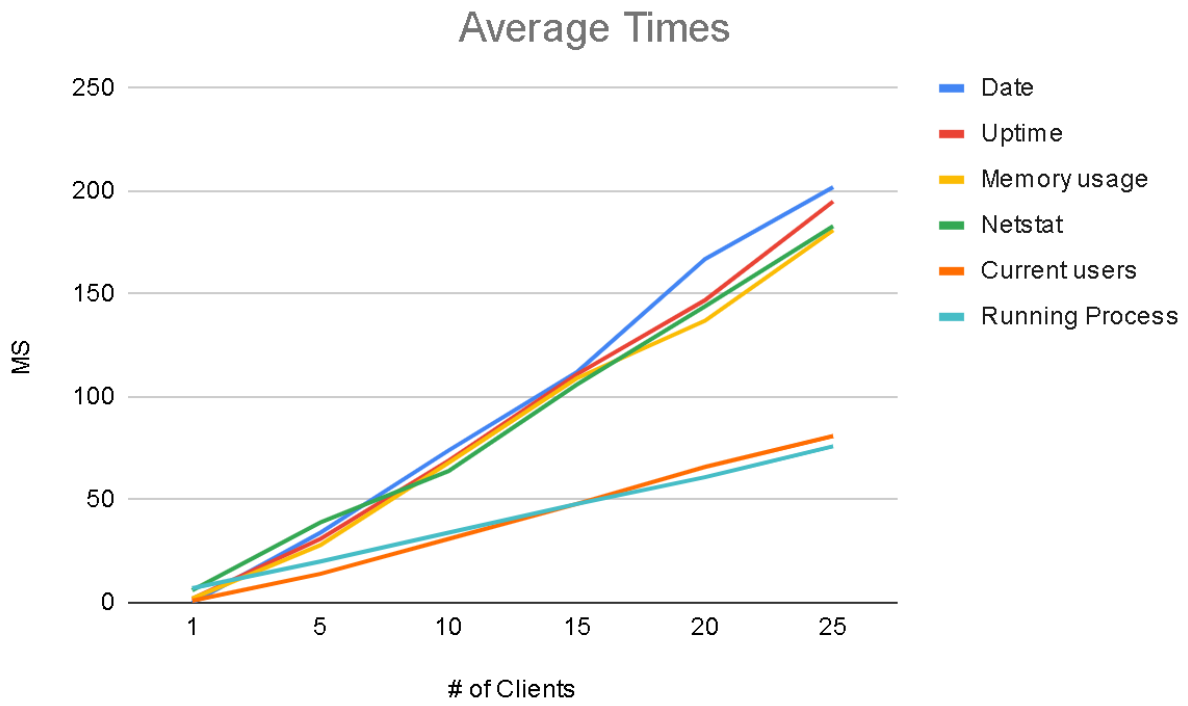
Average time for request varrying number of clients

# of clients	Date	Uptime	Memory Usage	Netstat	Current users	Running Process
1	0	2	2	6	1	7
5	34	31	28	39	14	20
10	74	69	68	64	31	34
15	112	111	109	106	48	48
20	167	147	137	144	66	61
25	202	195	181	183	81	76

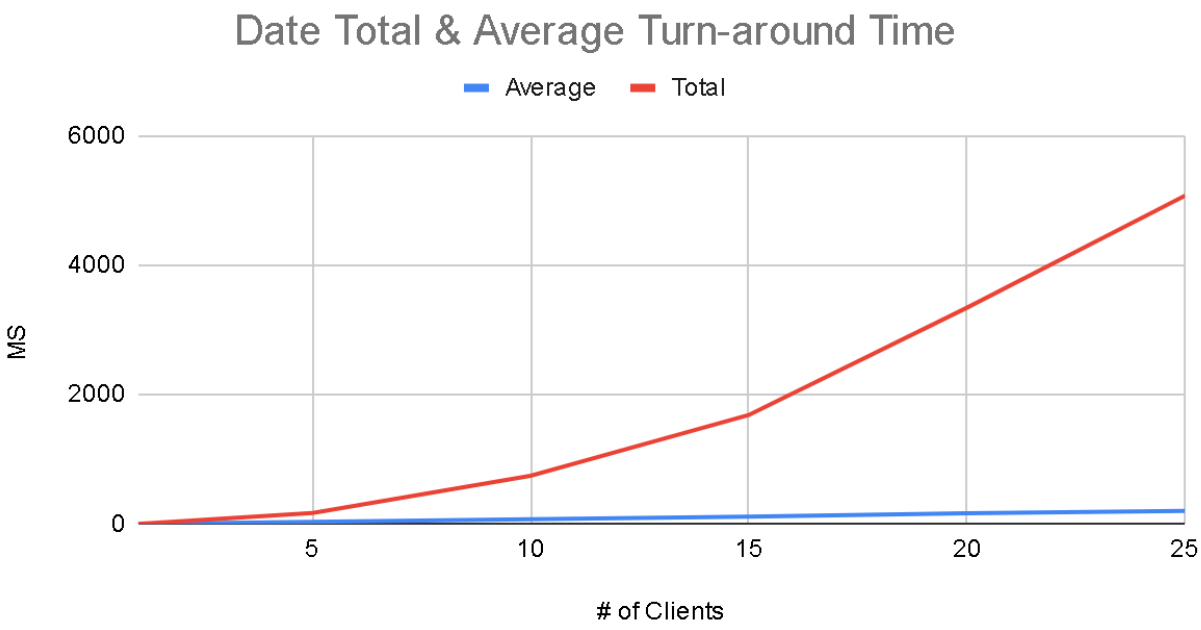
The graph below is the data from the total turn-around table.



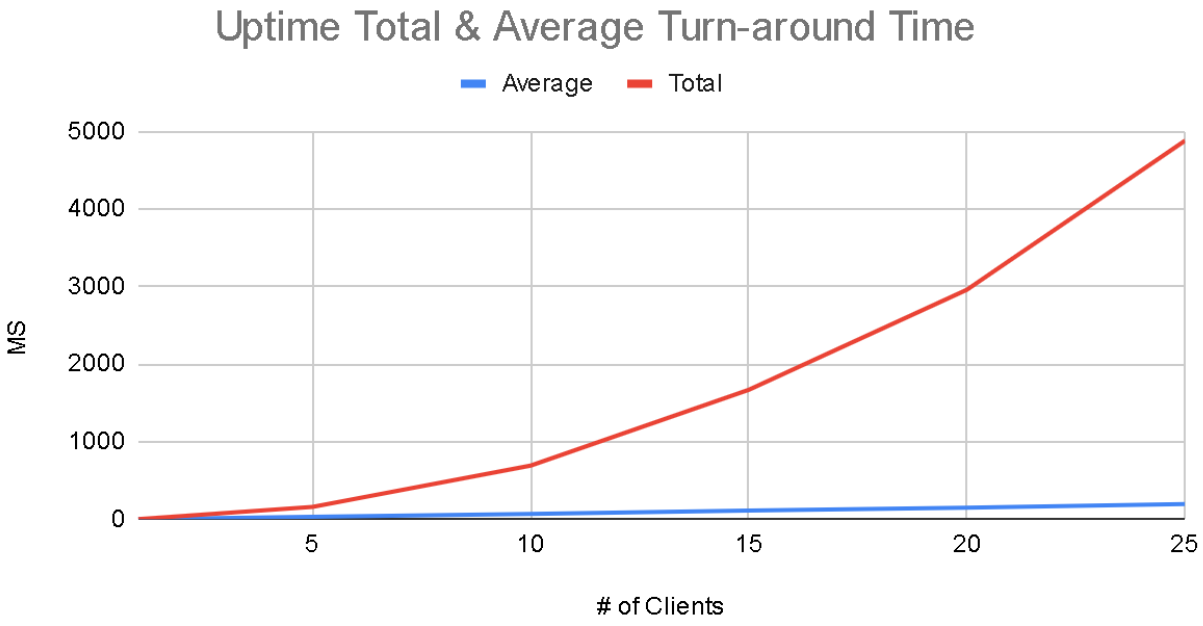
The graph below is the data from the average turn-around table.



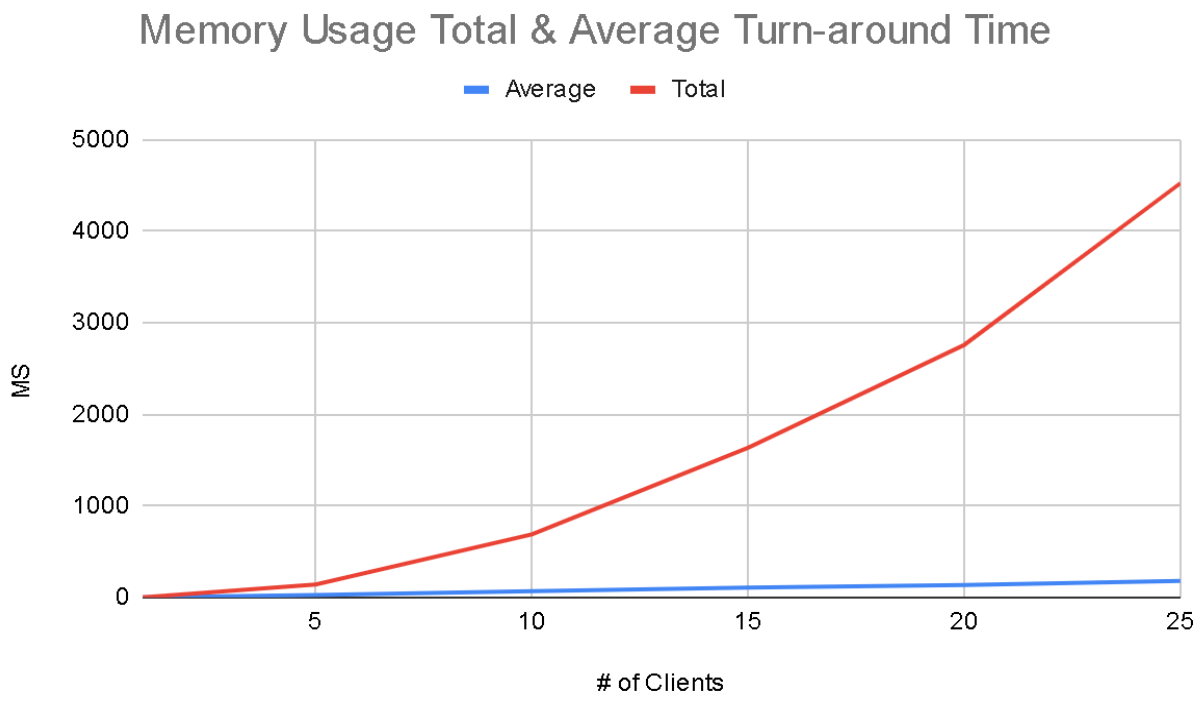
This graph is the total and average turn around time for the date request.



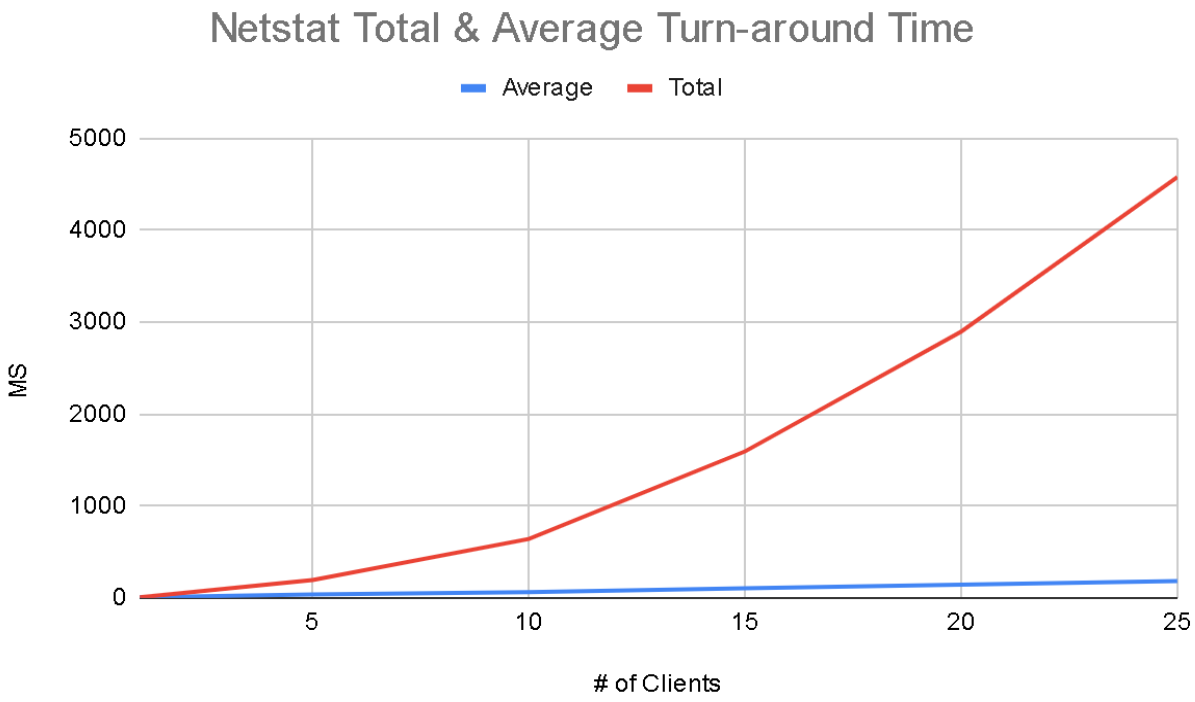
This graph is the total and average turn around time for the uptime request.



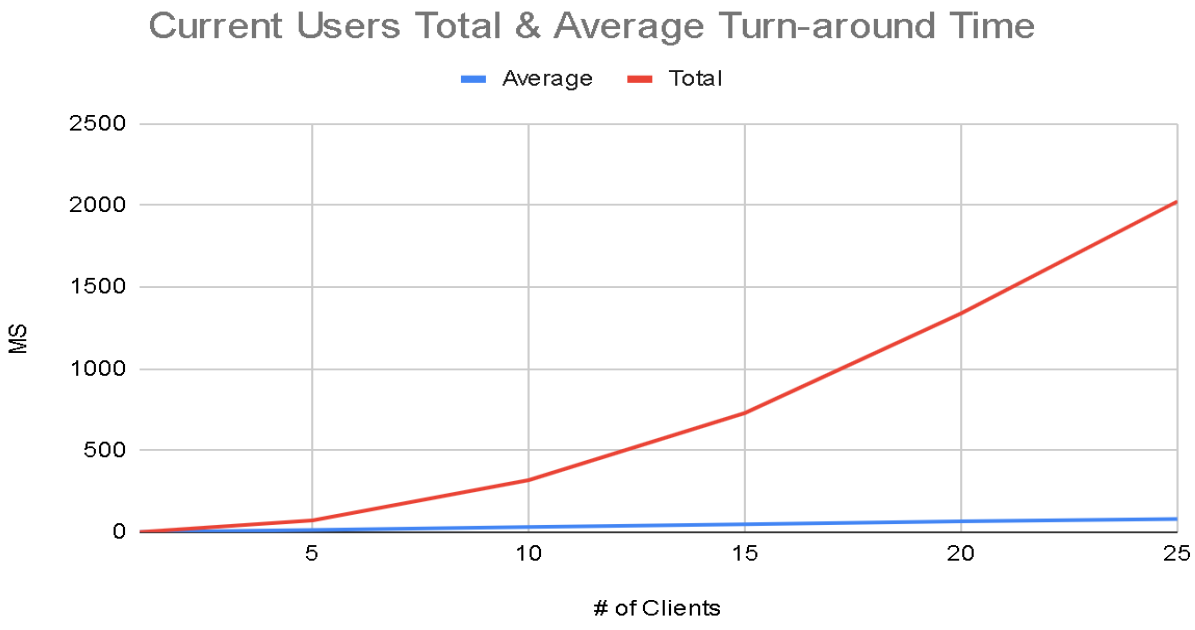
This graph is the total and average turn around time for the memory usage request.



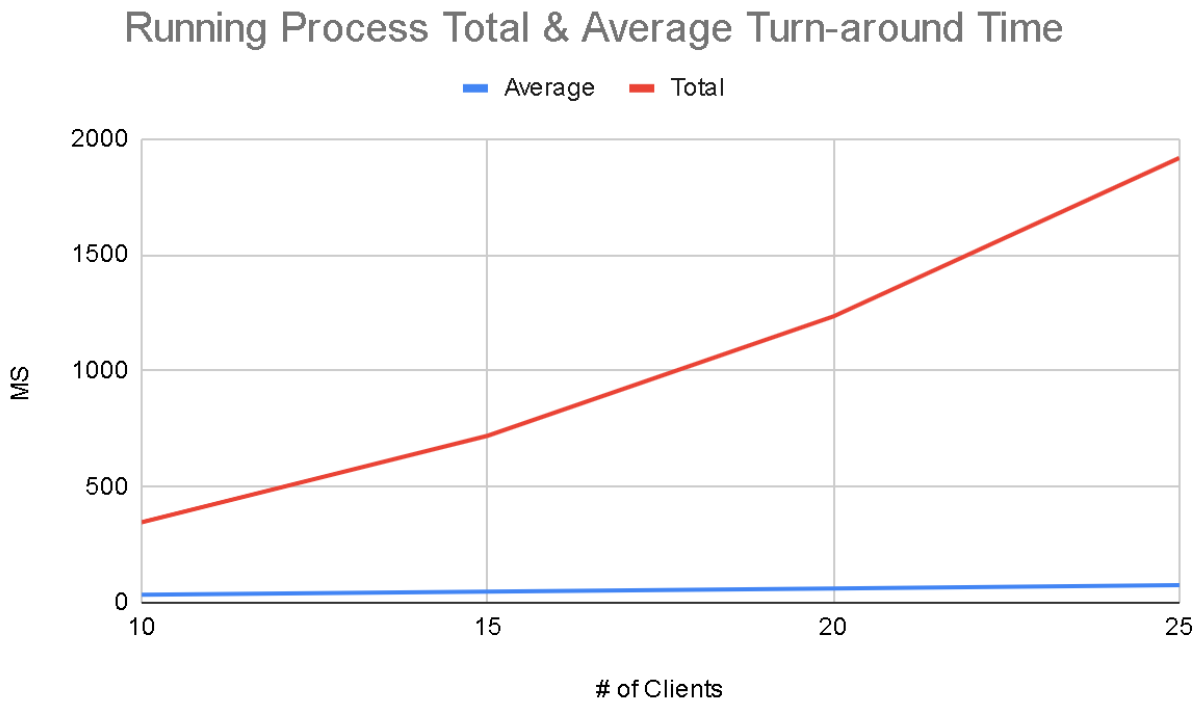
This graph is the total and average turn around time for the netstat request.



This graph is the total and average turn around time for the current users request.



This graph is the total and average turn around time for the running process request.



Data Analysis:

What affect, if any, does the increasing number of clients have on the Turn-around Time for individual clients?

Increasing the number of clients increases the individual turnaround time because each client is in a queue waiting their turn.

What affect, if any, does increasing the number of clients have on the Average Turnaround Time?

Increasing the number of clients also increases the average turnaround time due to the individual times increasing, it makes the average time increase as well.

What is the primary cause of the affect on individual client Turnaround Time and Average Turnaround Time?

Since the server is a singlethreaded server it can only handle one client at a time. The server puts each client in a queue. The remaining clients wait their turn to gain access to the server due to the clients in front going first.

Conclusion:

In conclusion, from the data we collected, the more clients there are on a serve, the longer the turnaround time and average turnaround time will be. Not every client can access the server at once and so they are put in a queue awaiting their turn until their is an opening for them to access the server.

Lessons Learned:

Our group has learned a lot of lessons during the implementation of the Iterative Server assignment. Overall, this assignment has solidified our understanding of the client-sever connection. This assignment was very different than any other programming assignment that we have done before. We have never used the Server Socket or Socket Java classes which ended up being very easy to work with. Using the extend thread in the cthread which allows us to inherit the thread class was also new. Using the resources provided and other resources online, there were plenty of examples showing a multithreaded server but not very many showing a multithreaded client. This was the most difficult aspect of the project and learning about the thread class and how to use it is what made it possible to finish the multithreaded client. In the past we had never thought about creating a java project that would execute Linux commands. Following the tutorial provided to learn how to do this opened us up to ideas in how this would be used. Anyone who has the responsibility to keep a server up and running could use this to

have automated emails sent to them to keep them updated on the performance of the server.

Overall, this assignment taught us a lot of new information.