David Nguyen
Jonathan Nguy

Project 1

For *client.c*,
1st, we initialized the variables using the arguments the user inputted. Since there can be more than just one set of [count | pong packet size | filling character], we used arrays for those values, with each index holding the corresponding set of values.

Next, we got the socket number and outputted an error if the socket was not assigned. We used the inputted address to set up the server info. We checked to see if we were successful at connecting to the server.

Once we knew we knew we were connected, we went through each set of [count | pong packet size | filling character] and sent them to the server. Once we got the responses, we printed out the buffer and finally, closed the socket.

For *server.c*,
1st, we got the port number from the input, and if there was no argument, we set a default port. We then created a listening socket and bound the socket addresses to the listening socket, then started to listen.

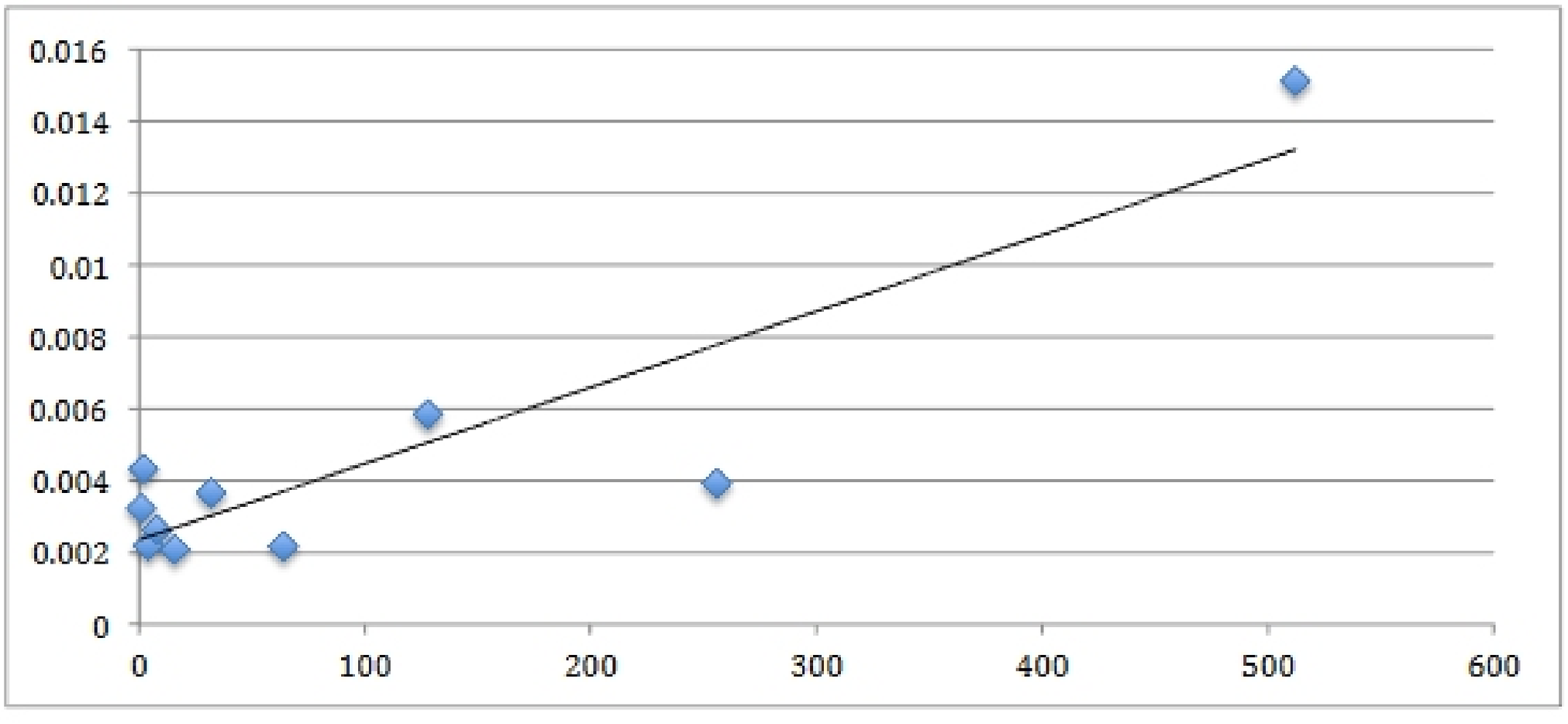
We continued to listen until we got a connection and then accepted it. If the version number was 1, we used the regular input, and if the version was 2, we responded with different characters using the formula in the project specs. We closed the connection and continued to wait for new connections indefinitely.

Evaluation question:

To test the RTT for our project, we found the time before we went through the for loop using `gettimeofday ()`. We collected the data with a count of 4 and then divided the final time difference by 4. We did this 5 times per payload size, and calculated the average of the 5.3

Payload Byte Size	Seconds Elapsed	Average	Payload Byte Size	Seconds Elapsed	Average	Payload Byte Size	Seconds Elapsed	Average
1	0.002507	0.00324	16	0.002539	0.0020928	256	0.005237	0.0039244
1	0.002081		16	0.001891		256	0.007819	
1	0.004272		16	0.002199		256	0.002315	
1	0.005677		16	0.001793		256	0.002361	
1	0.001663		16	0.002042		256	0.00189	
2	0.006265	0.0043282	32	0.002229	0.0036618	512	0.004649	0.015109
2	0.009601		32	0.001689		512	0.059816	
2	0.001801		32	0.002149		512	0.002519	
2	0.002185		32	0.003899		512	0.002234	
2	0.001789		32	0.008343		512	0.006327	
4	0.002672	0.002193	64	0.001869	0.0021666			
4	0.001945		64	0.00181				
4	0.002082		64	0.003057				
4	0.001665		64	0.0021				
4	0.002601		64	0.001997				
8	0.001731	0.0026214	128	0.005681	0.0058532			
8	0.005167		128	0.00276				
8	0.001531		128	0.00198				
8	0.00219		128	0.010724				
8	0.002488		128	0.008121				

Graphing the average values with the payload size in bytes, we get the following graph:



As we expected, the more bytes we needed to send, the longer it took for the server to respond.

It was not possible to split the time into the 4 types of delay. The processing delay can't be determined because after send the code, we cannot tell what the router does to process the code. In our case, we send the packets and wait for a response. Similarly, we cannot determine the queuing or transmission delays. We do not control these delays, but instead the router or network card do. Lastly, the propagation delay isn't something we control, either, and we can't determine when the actual packets are in the propagation phase.

References

http://cs.baylor.edu/~donahoo/practical/CSockets/textcode.html

http://rabbit.eng.miami.edu/info/functions/time.html
(for `gettimeofday ()`)