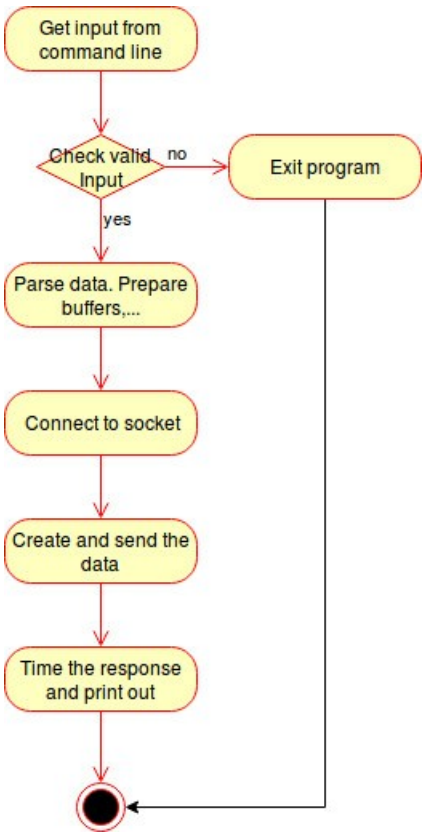
HW1 – Intro to Network Programming

Overview

This homework serves as a beginning exercise for the CSS 432 class. It covers topics like clients and servers, using sockets, how to do read and write between clients and servers.
This assignment involves creating a multi-threaded client-server program and to evaluate the reads and write made by the system.

Client program

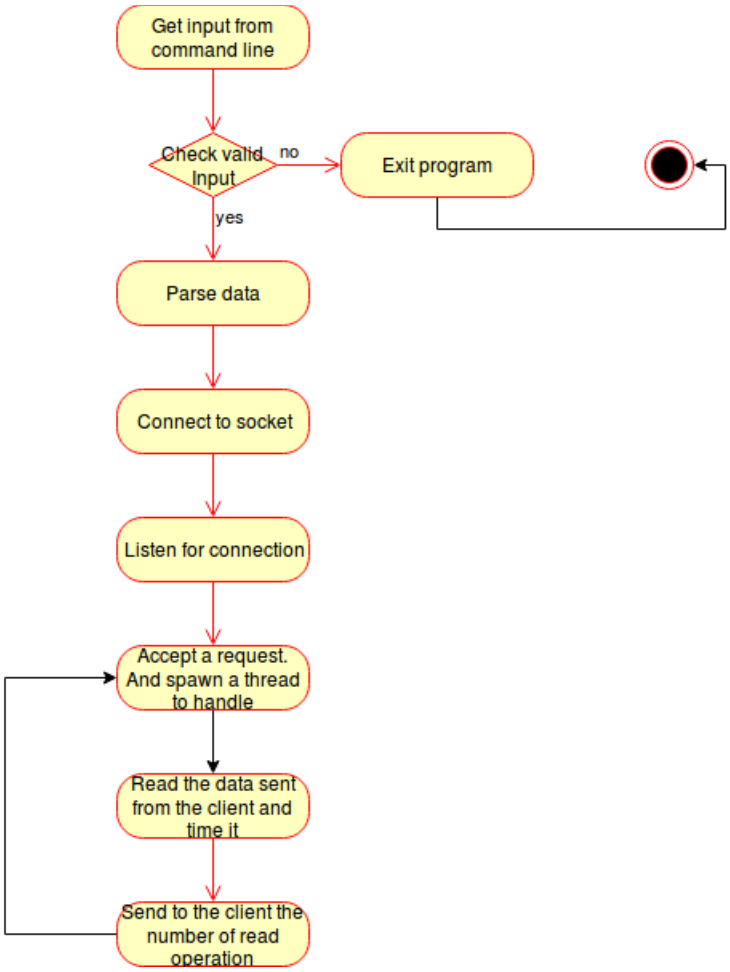
The client program is responsible for connecting to the server and depending on the input from the command line; send to the server data and receive back data and time them



Note that within the process, there are also check code like unable to connect to socket,..etc. In addition, instead of using gethostbyname(), I decided to use getaddrinfo() instead to “catch up to modern time”

Server

The server is responsible for creating a socket with a valid port number so that the client can connect to it. It is responsible for reading the data that the client send and return back how many read operations it performed



Note that the server also use similar code to connect to socket like client. And the server does not end at all. It just keep accepting request

How to compile the program and run it

Server:

Compile: g++ -pthread -std=c++11 Server.cpp

Run: ./a.out [port number] [repetition]

Client:

Compile: g++ -std=c++11 Client.cpp
Run: ./a.out [port number] [repetition] [nbufs] [buffsize] [serverIp] [type]

IMPORTANT: Run the server first then client later

Output

```
Data-receiving time for thread 1 = 256912 usec
Finish with thread 1
Creating new thread with count: 2
Data-receiving time for thread 2 = 256594 usec
Finish with thread 2
Creating new thread with count: 3
Data-receiving time for thread 3 = 257465 usec
Finish with thread 3
Creating new thread with count: 4
Data-receiving time for thread 4 = 257176 usec
Finish with thread 4
Creating new thread with count: 5
Data-receiving time for thread 5 = 257020 usec
Finish with thread 5
Creating new thread with count: 6
Data-receiving time for thread 6 = 257075 usec
Finish with thread 6
Creating new thread with count: 7
Data-receiving time for thread 7 = 259133 usec
Finish with thread 7
Creating new thread with count: 8
Data-receiving time for thread 8 = 257721 usec
Finish with thread 8
Creating new thread with count: 9
Data-receiving time for thread 9 = 257054 usec
Finish with thread 9
Creating new thread with count: 10
Data-receiving time for thread 10 = 257102 usec
Finish with thread 10
Creating new thread with count: 11
Data-receiving time for thread 11 = 258914 usec
Finish with thread 11
Creating new thread with count: 12
Data-receiving time for thread 12 = 257683 usec
Finish with thread 12
Creating new thread with count: 13
Data-receiving time for thread 13 = 256873 usec
Finish with thread 13
Creating new thread with count: 14
Data-receiving time for thread 14 = 257034 usec
Finish with thread 14
Creating new thread with count: 15
Data-receiving time for thread 15 = 261184 usec
Finish with thread 15
Creating new thread with count: 16
Data-receiving time for thread 16 = 266218 usec
Finish with thread 16
```

Illustration 1: Server

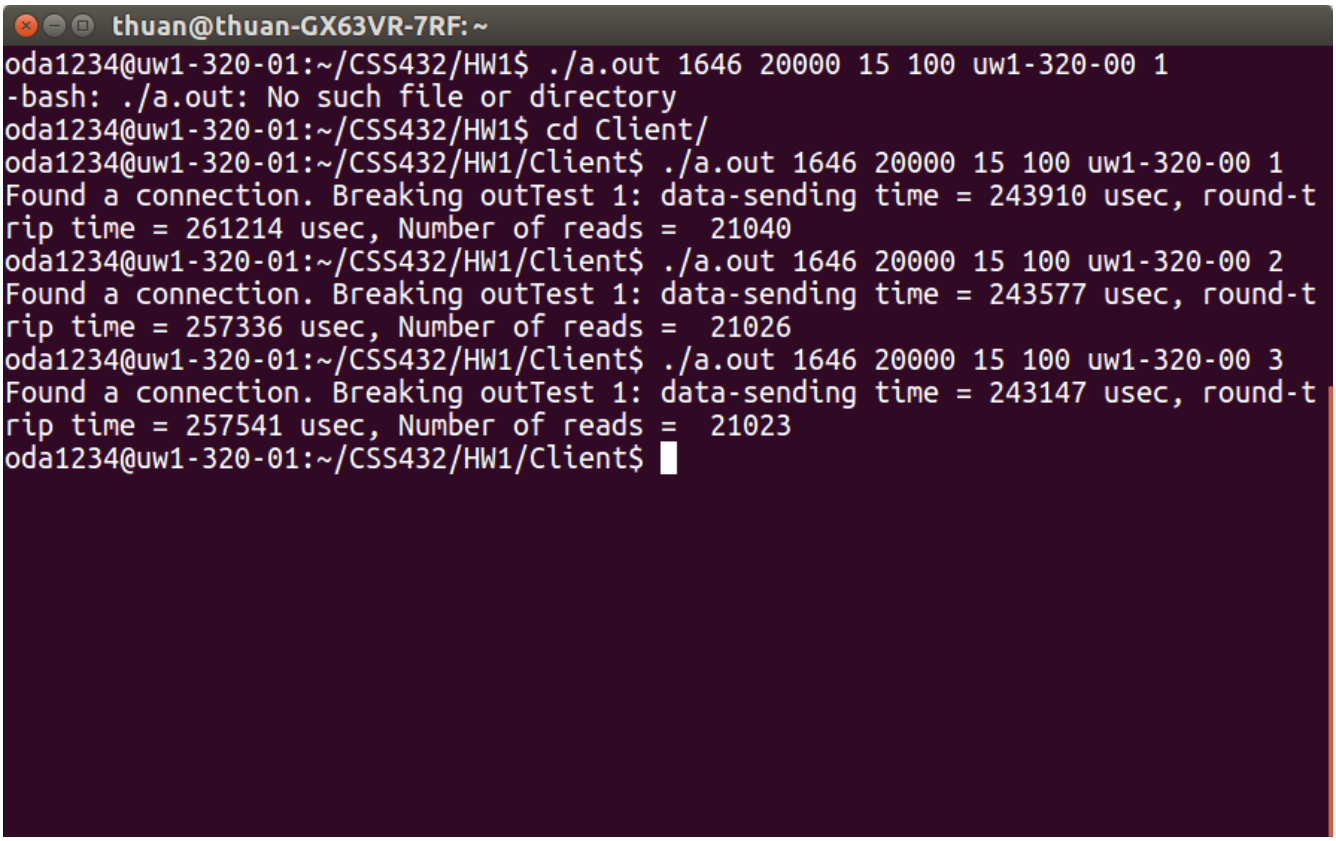


Illustration 2: Client

Performance Evaluation

To evaluate the performance of different write method, I created some scripts and generated data along with it. I will attach the data within this submission. The data in the table below are based on the data attached with the submission that I calculated by hand and input them here

Number Buffer	Buffer Size	Type	Average sending time	Average round-trip time	Average number of read
15	100	1	242488	258176	21327
15	100	2	242309	258183	21397
15	100	3	241095	257518	21584
30	50	1	245000	259828	21453
30	50	2	242251	257533	21602
30	50	3	242467	258307	21325
60	25	1	307480	308362	22280
60	25	2	243869	259908	21277
60	25	3	241417	257557	21447

Type 1: Multiple write

Time

Looking at the graph below. There are not much difference between when using different type of buffer. 30 Buffers with size 50 each may take more time than others. But those are considerable.

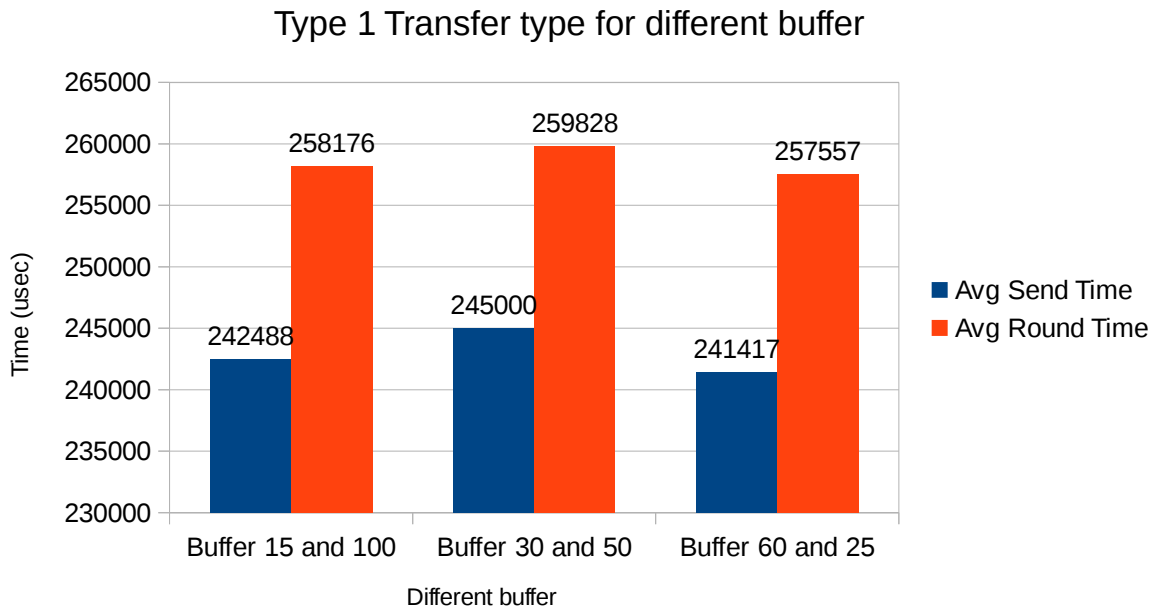


Illustration 3: Graph that represent time for type 1

Number of read

Looking at the last buffer, I can see that for 60 buffers with size of 25 each, it takes more operations to work on them. One reason that I can think of is that even though we have the same data, the overhead for each read is more since each buffer only have the size of 25. In my thinking from the CSS 422 class, each time a read happen, a bunch of data local to it was gather. This is useful because next time we don't need to read more data. But for this buffer, each read can only get 25 each. So more read operations is required because it could not get more data

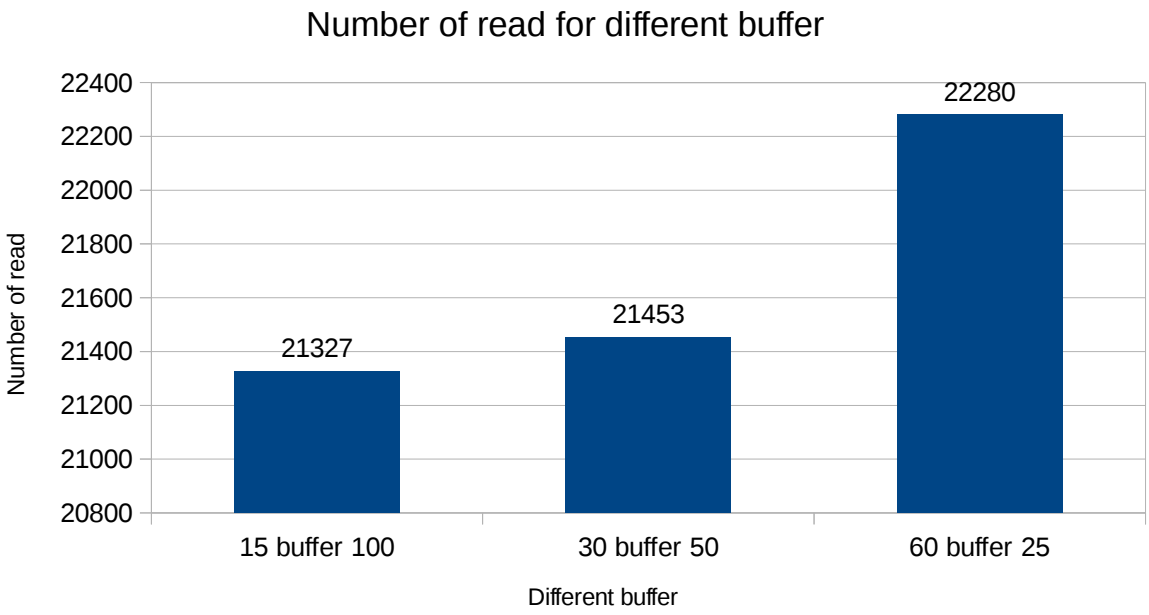


Illustration 4: Graph that represent read operations for type 1

Type 2: Writev

Time

Looking at the graph below. I also reached the same conclusion like type 1 in which there are not much difference between transfer time for different buffer

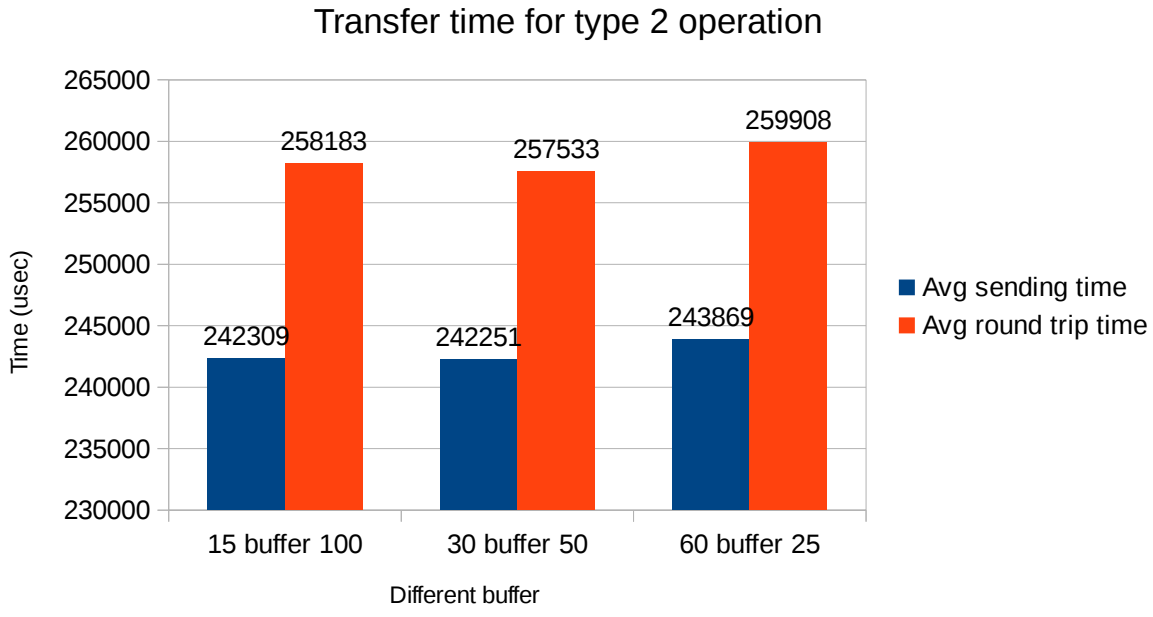


Illustration 5: Graph that represent transfer time for type 2

Number of read

Looking at the graph below. One significant different is the high amount of read for 30 buffer with 50 each and the low amount of read for 60 buffer with 25 each. I can say that there is a relationship between number of buffer and buffer size. At first, when the number buffer increase and buffer size decrease, the read operation increase. But when it reach to 60 and 25, it drops down again. So there must be a threshold in which it will reach the highest but then drop down

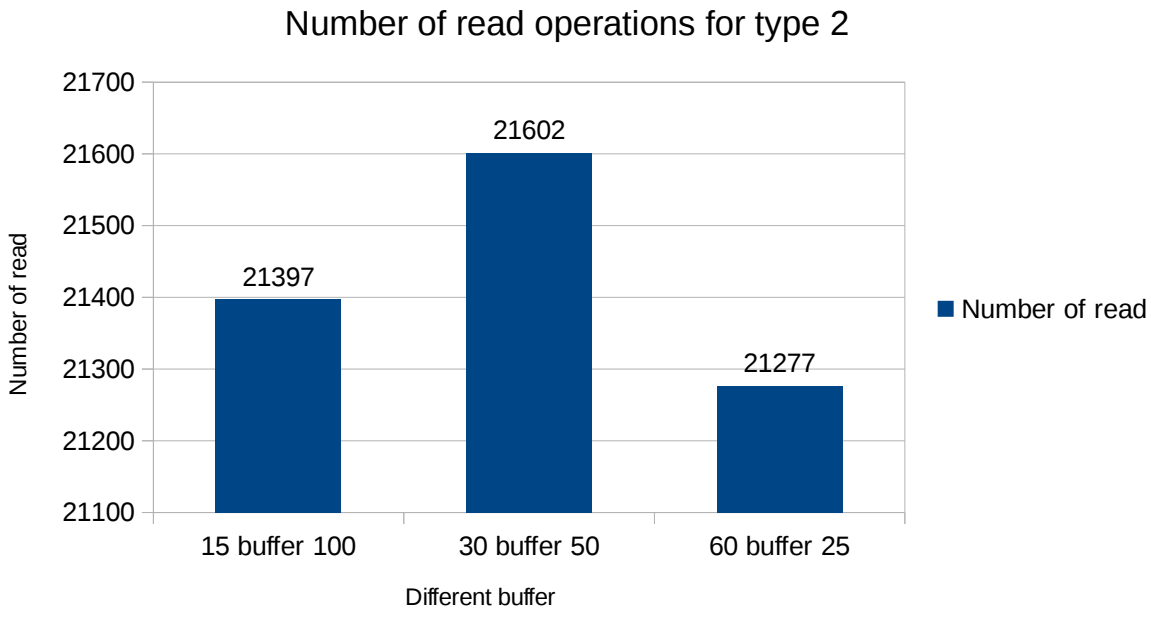


Illustration 6: Number of read for type 2

Type 3: Single write

Time

I also reached the same conclusion for type 3 just like what I did for type 1 and 2: There are not much different between sending time for different buffer

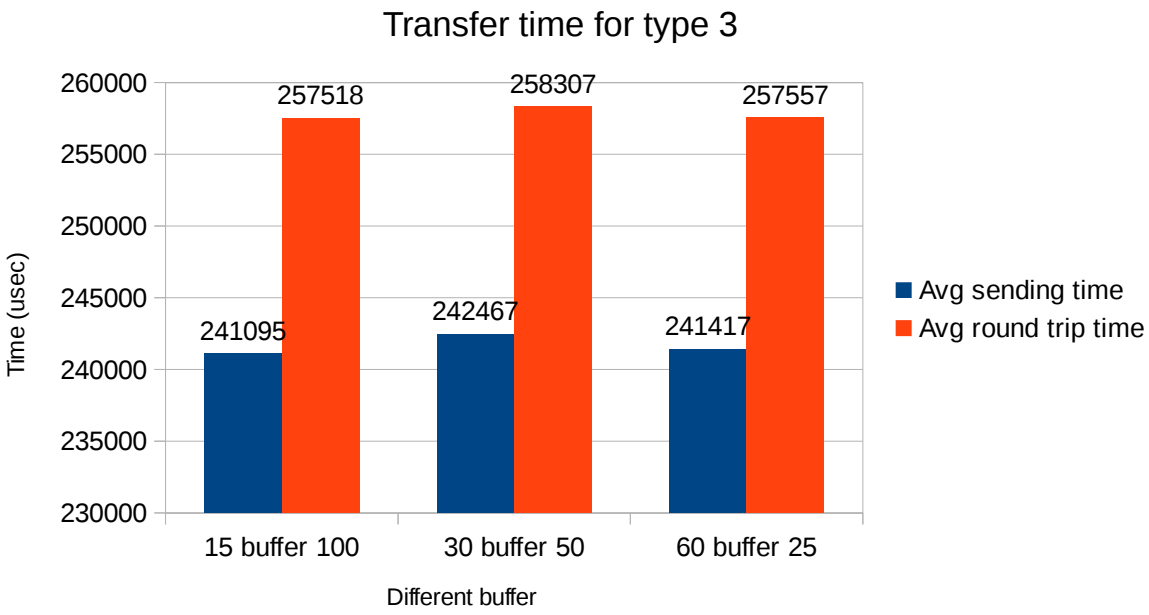
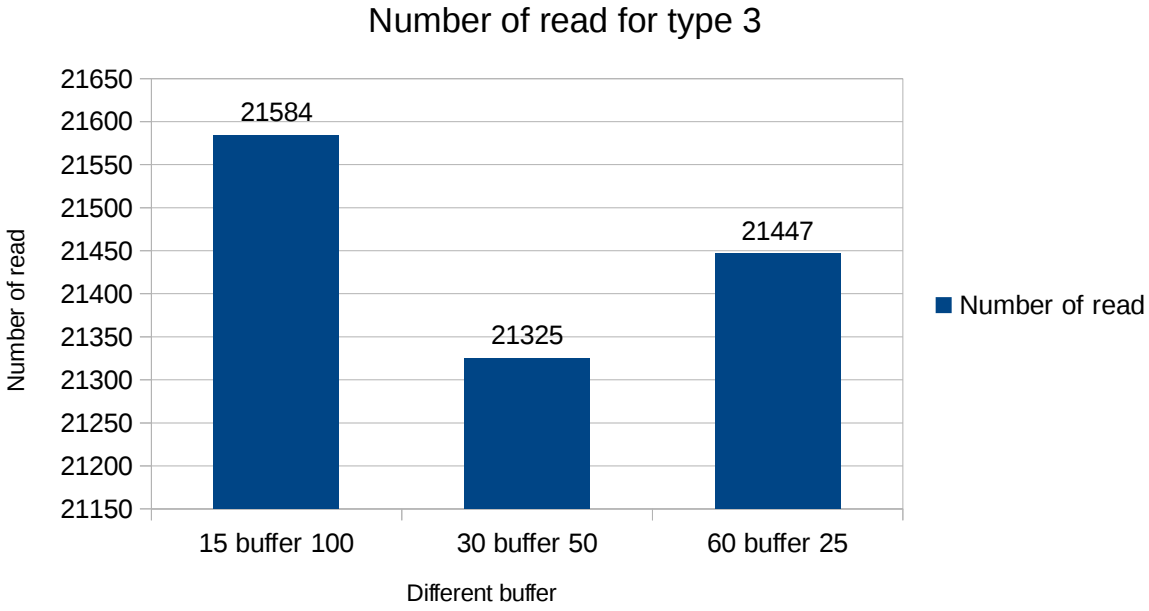


Illustration 7: Graph that represent transfer time for type 3

Read

For the number of read for type 3 operation. Seems like the middle ground 30 buffer with size 50 is the most beneficial. This also have me related to what I discussed for number of read in type 1 that relate with CSS 422. In which the size 50 seems to be the appropriate size to do a read operation.



Conclusion

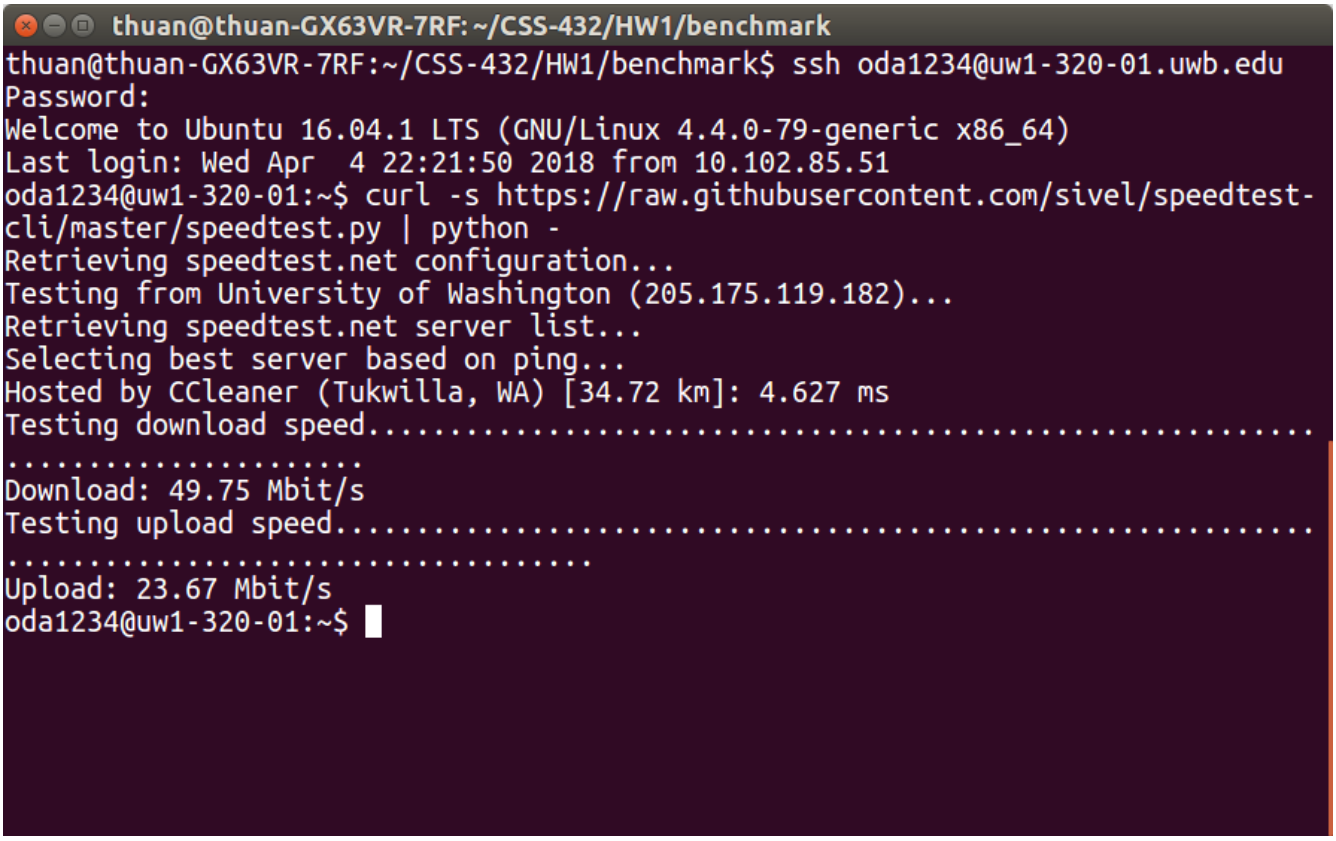
For transfer type using different buffer and type, there are no significant different. This maybe due to the ability to process the information of the machine

For the number of read, there are differences based on how the data is write and how it is stored based on the explanation above

What happen if this was run on a slower network

Below is the result for the speed test at the lab, from the picture, we can see that the upload speed is 23.6 Mbit/s. For every second, 23.6 Mbit can be transferred. If we only have 1 Mbit, surely the result would be slower. That is because for a package with total L bits, to send it completely, we will need L/R where R is the transmission rate , which in this case is 23.6. But if it were only 1, then the result would be 23.6 time slower. So instead of time like 250000 usec like above, multiply it by 23.6 , we will get 5900000 . Increase from 0.25 second to 5.9 seconds

The read operation will most likely stay the same since it runs on the machine. And the total data that we need to read does not change



Why would I want to use a thread ?

By using the thread, I will take full use of the computer and also allow the server to service more request. Computer nowadays have multiple cores that allows multiple threads to run within it. If I just use it on the main function only, then I will be missing out the potential of other threads/cores
In addition, if it were to run on the main function only, then the server will not able to accept multiple connections at the same time because it was too busy finishing the request of one connection (accept the request, perform the task, when it is done then return to accept another request). That will slow down the ability of the server. So a better approach will be to spawn a thread and let it do the task, while the server accept new request.

Reference

Largest port allowed:

https://stackoverflow.com/questions/113224/what-is-the-largest-tcp-ip-network-port-number-allowable-for-ipv4

How to use getaddrinfo

https://beej.us/guide/bgnet/html/multi/getaddrinfoman.html

https://github.com/angrave/SystemProgramming/wiki/Networking%2C-Part-4%3A-Building-a-simple-TCP-Server