Arturo Villalobos

CSCE 313-503

Professor Ahmed

4-30-21

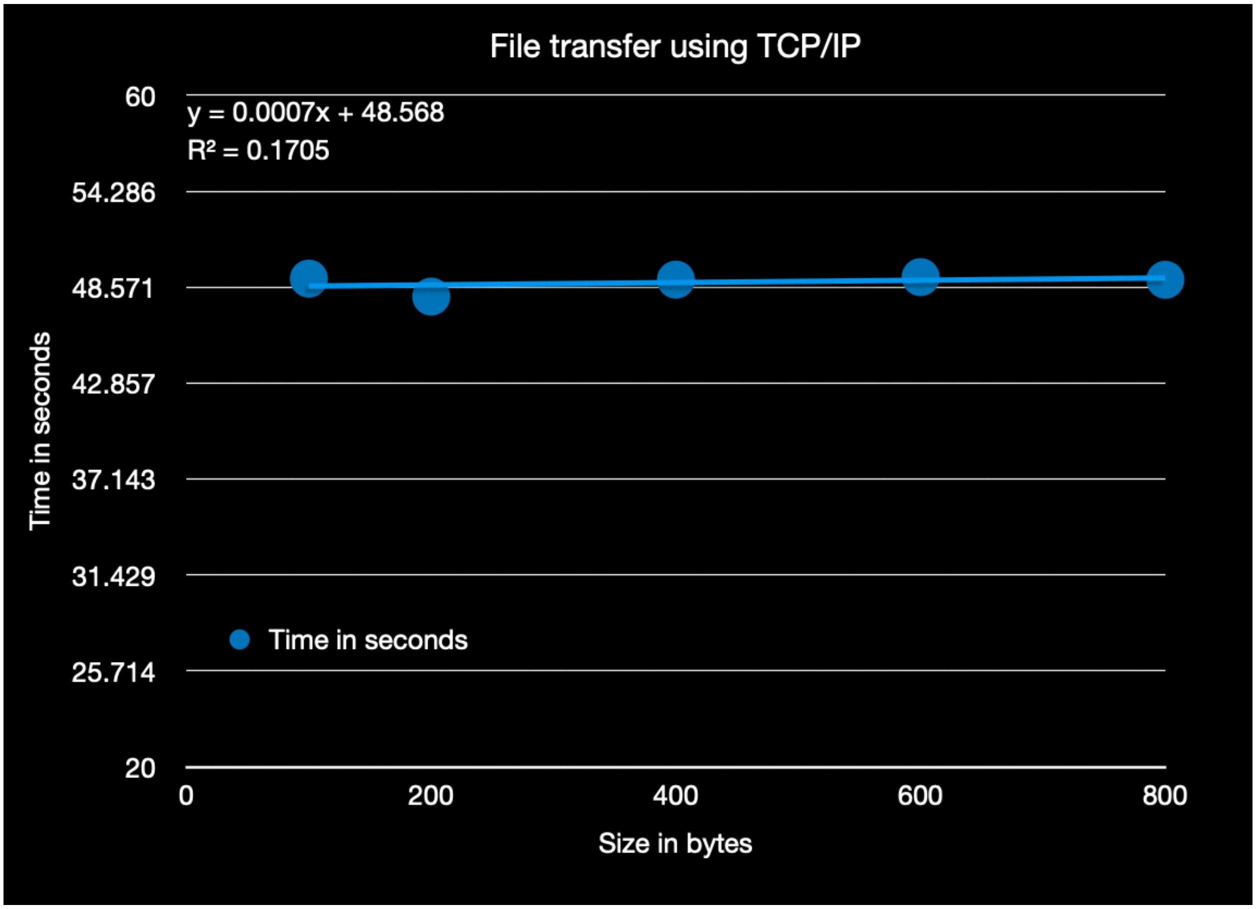
PA6

Video/drive link: *https://drive.google.com/drive/folders/12n5WB-5D6uJJYzBePec-agor4I-NGZQF?usp=sharing*

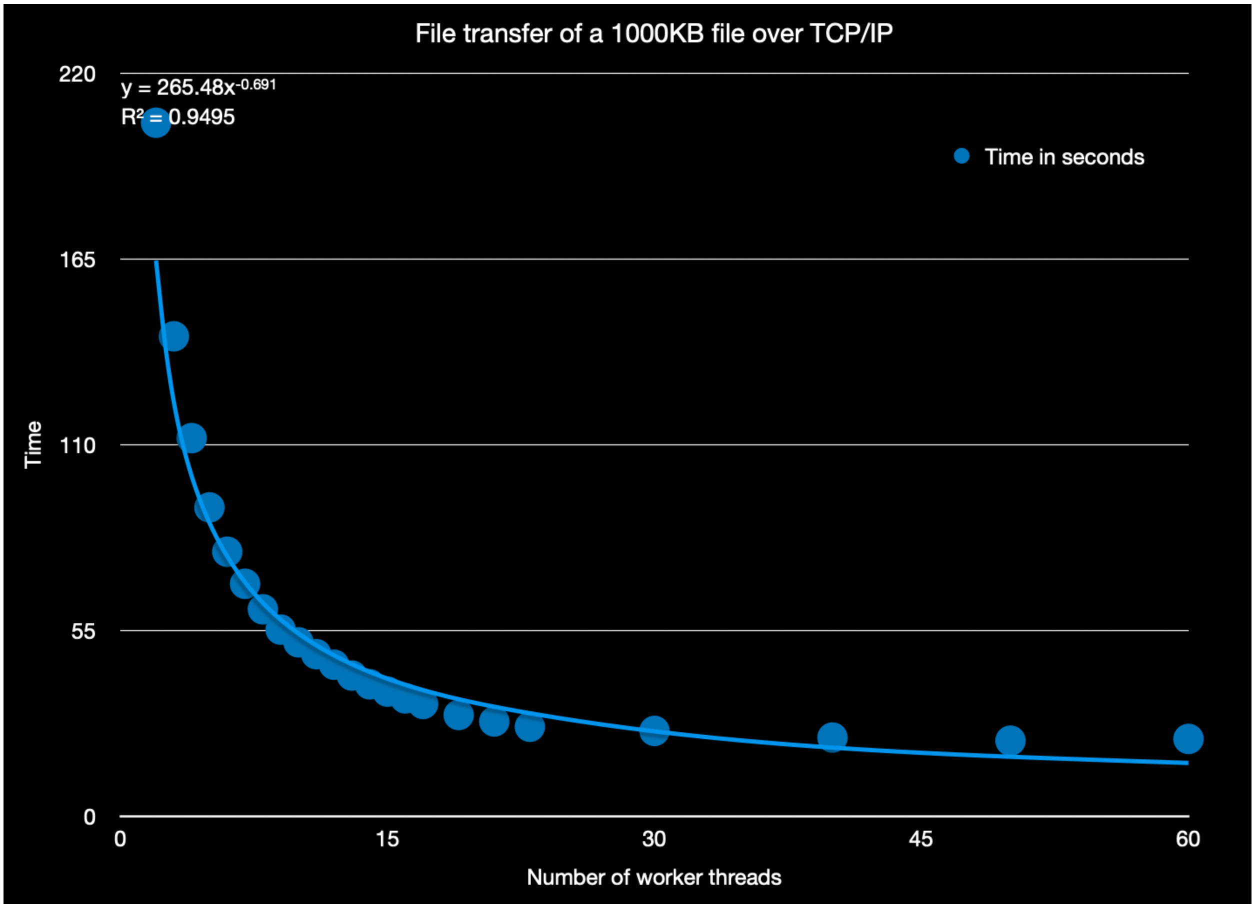
The important sections of this assignment to where is the TCPrecchanel.cpp file and in the client.cpp. the TCPrecchanel constructor was the section that gave me the most issues and before I figured it out , gave me a bad file descriptor error. This was due to the fact that I was re-declaring the socket file descriptor within the constructor every time the client would access it , instead of accessing the private attribute I had declared in the header. This leads into the second most important section and that is within the TCPrecchnnel header and in the client. Within the client we had declared a control Chanel 'chan'. This channel is needed for a lot of functions within the client.cpp and so deleting is wasn't an option , redefining it in the header was critical in this PA. the actual code that handles creating the socket , accepting , binding , and listening came from the TCPexample code provided and was added to the TCPrecchannel constructor.

Results:

To start I want to show the results of file transfers of varying size. Below is a graph of file transfers with varying sizes. With all other parameter kept the same what is strange is that the times where relatively similar, at first I thought I was because I used files that where too small but after trying a 1000k file and getting a similar time I knew that file size was not the variable impacting time taken to completion:

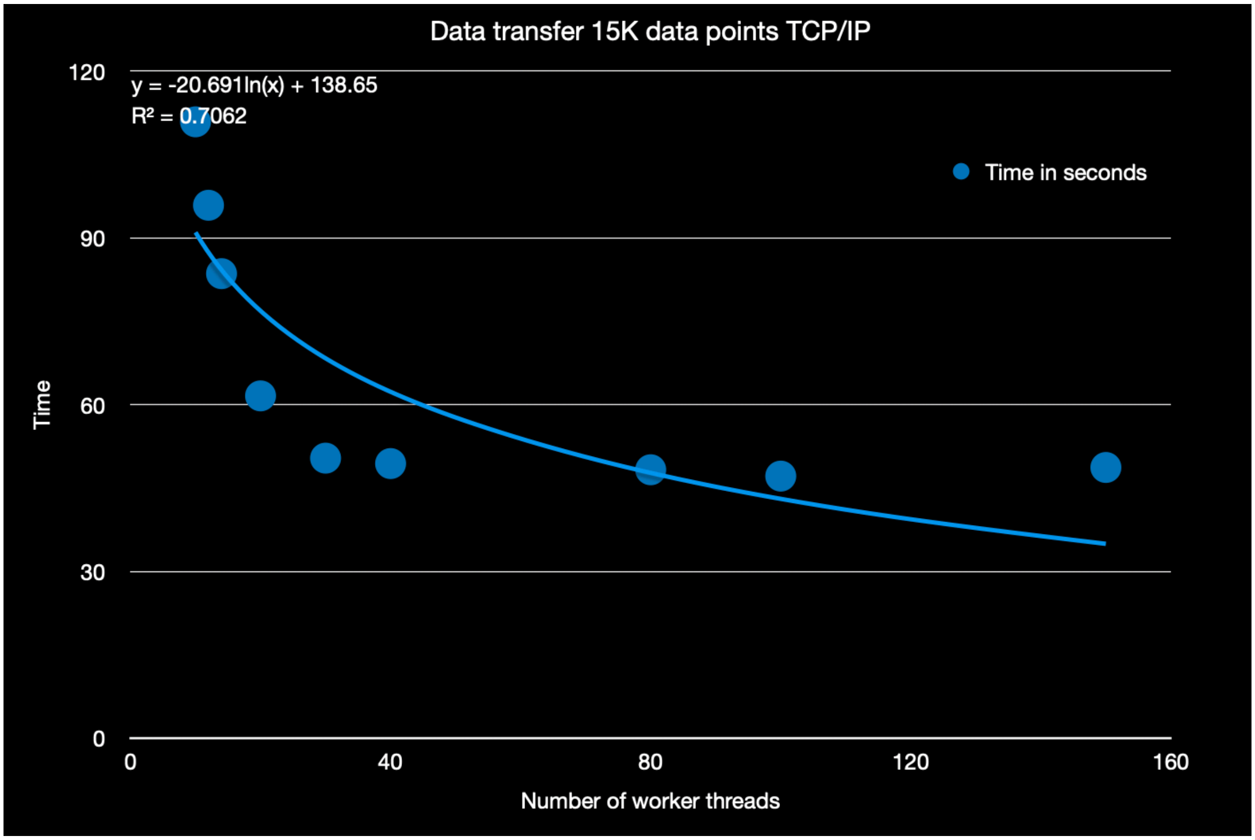


But even after varying the number of worker threads , I found that there was another problem , while my initial testing had worker thread increase increments of 50 threads each data point after the first request I already hit the point of diminishing return. I decided to increment the threads slowly and here is the resulting graph:



The point of diminishing returns happens actually around 50 worker threads, we see a massive improvement in performance the first few increments but slowly the context switch overhead overtakes the benefits . This is a similar shaped graph but the point at which this happens is much less than that of the result of PA5. Even with a small amount of threads , for file transfer , PA5 was able to preform much better than this PA as in PA5 we approach single digit time where as here we do not.

Going on to data transfers , I took the same approach in finding the point of diminishing return by starting the incrementation small. Graph below:



As you can see from the chart there is a point of diminishing return at around 100 worker threads and as compared to PA5, both the performance and point of diminishing return are different . The performance in PA5 is much better as it is able to do this task in under 15 seconds while this program doesn't go below 30 seconds. Secondly the point at which more threads doesn't yield better performance is somewhere around 130 worker threads .

TCP/IP vs FIFO :

It should probably be expected that the performance of FIFO on where both the server and client are on the same machine is much better, as this PA has to deal with establishing a connection but still with the worker thread parameter tuned, this PA can still preform reasonably well and be much more practical than the FIFO implementation.