EE4204 Lab report

Submitted By:

Name: LUO ZIJIAN

Matric.No: A0224725H

Mobile: 98912483

Electrical and Computer Engineering, School of Engineering

14 April 2021

1 Introduction

In this lab, the requirement is to find a suitable data unit size to satisfy the UDP transport assignment. As for the objective function: to get more sending rate and more throughput in some conditions.

And then, I get idea how to apply the assigned problem from Ex3 source code, I searched the difference between UDP and TCP socket programming. The key is that TCP use recv() and send() functions to apply socket properties. But for UDP, we need to use recvfrom() and sendto() functions alternatively. So, it is interesting to rethink our program from the source code.

2 Description of input data

As for the object of process, there are these basic definitions I used in

Table 1: The definitions of data

name	**definition**
sending time	The time of client send all the packets and receive all acks
receiving time	The time of server receive all the packets and send all the acks
sending rate	The total file length divide sending time
throughput	The total file length divide receiving time
client socket	The client uses this socket to send short packet to the server.
server socket	The server uses this socket to send ack socket to the client

So, the performance of algorithm is judged by sending rate and throughput.

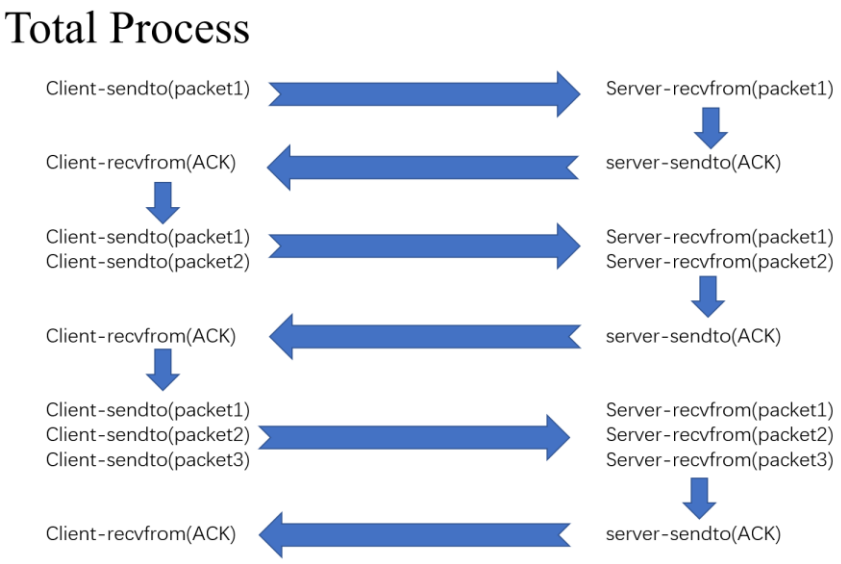
$$sending\_rate = \frac{file\_length}{sending\_time} = \frac{file\_length}{finish\_sending\_time - start\_sending\_time}$$

$$throughput = \frac{file\_length}{receiving\_time} = \frac{file\_length}{finish\_receiving\_time - start\_receiving\_time}$$

I have put these calculating functions into the main coding structure, so if you run the code, the result would be generated automatically.

3 Algorithm description

The first algorithm is that the sender sends one DU, waits for an acknowledgment (ACK); sends two DUs, waits for an ACK; sends three DUs, waits for an ACK; and repeats the above procedure until the entire file is sent. The receiver sends an ACK after receiving a DU; sends next ACK after receiving two DUs; sends next ACK after receiving three DUs; and repeats the above procedure.

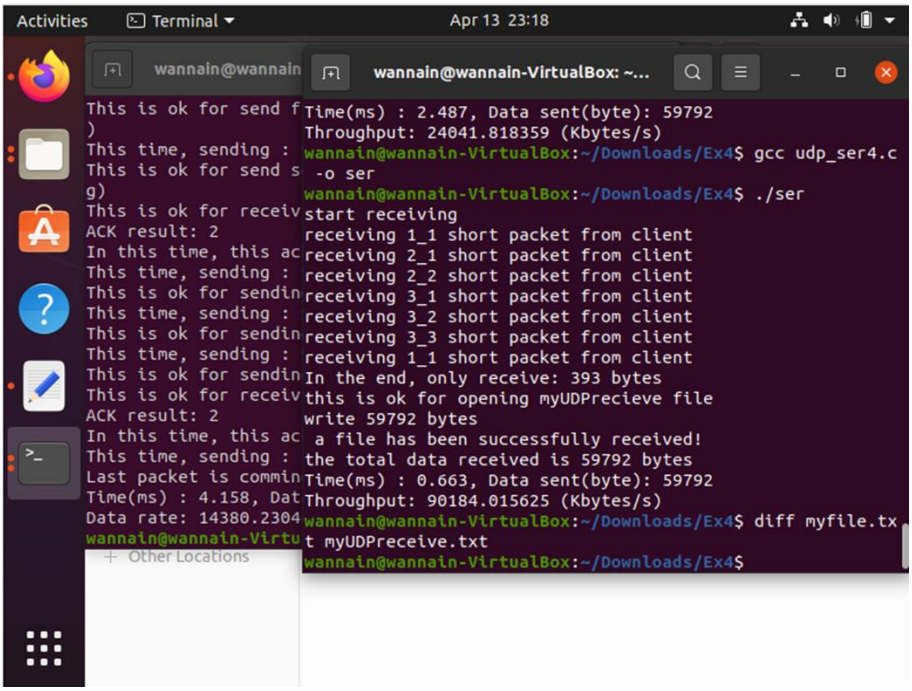


UDP-based progress

Second algorithm is that stop and wait algorithm. After rethinking, I got an idea that we just apply the batch size is always one, and then the most part is similar as our first algorithm.

4 Result and conclusions

Through the simulation, we found UDP-based algorithm has successfully send the file from client to server.

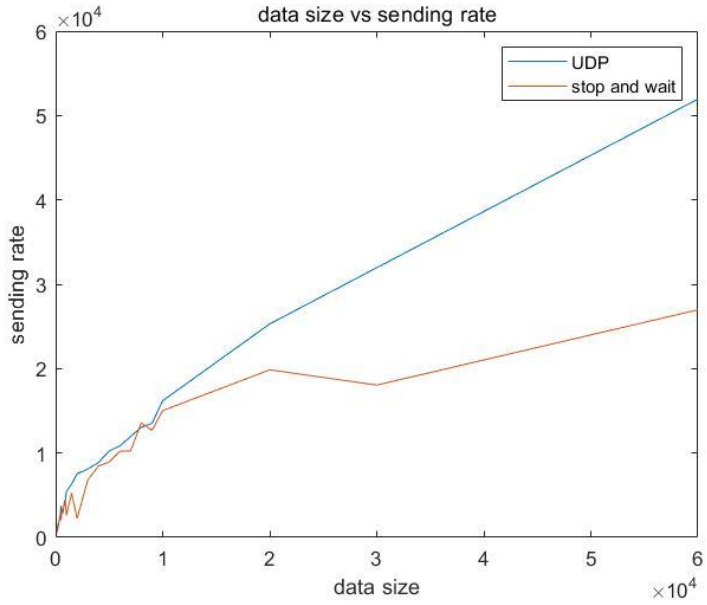


After many times, we record the result and plot the performance between UDP-based algorithm and (stop and wait) algorithm.

4.1 performance in sending rate

For UDP-based algorithm, we can easily find that with the increase of the data unit size, the sending rate also get larger. As for stop and wait algorithm, the performance is similar. The difference between them is that the sending rate of UDP-based algorithm is always larger than of (stop and wait) algorithm.

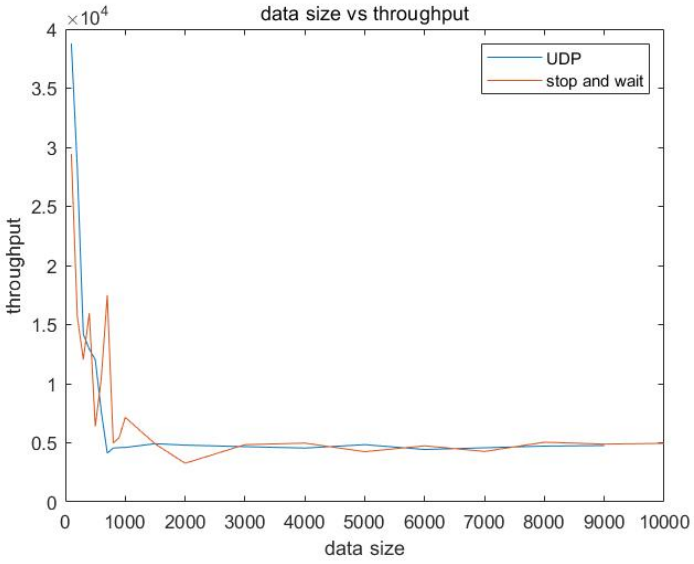
The reason is that, Our UDP-based algorithm reduces the use of ACK, so there is an improvement in sending time, which means the sending rate is increasing.



4.2 performance in throughput

For UDP-based algorithm, we can easily find that with the increase of the data unit size, the throughput gets less. As for (stop and wait) algorithm, the performance is similar. The difference between them is that the throughput of UDP-based algorithm is always larger than of (stop and wait) algorithm. But when the data size get enough large, the number of ACK is same, so the receiving time is similar.

The reason is that, Our UDP-based algorithm reduces the use of ACK, so there is an improvement in sending time, which means the throughput is increasing.



5 Conclusions

The result for the comparison of the two algorithms UDP-based algorithm and (stop and wait) algorithm shows that they perform differently depends on the data size. In a word, I believe the total performance of UDP-based algorithm is better than (stop and wait) algorithm owing to the improvement in sending rate and throughput. As for the disadvantages in our simulation, I suppose that printf function may cost some time in printing meaningless information, so the final result in calculating time maybe not very precise. In the future work, I should update these details in simulations.

Coding effort

In this total simulation, the coding effort is 100%. At the beginning of simulation, I get idea from the example source code from Ex3.

All the steps in this simulation are signed by myself. Here are two simulations, the first one is to compare the performance in sending rate between them. The second one is to compare the performance in throughput between them.