EECS302002 計算機網路概論
Lab 2

1. Description

Write a server program and a client program. The client can download a video file from the server using **stop-and-wait** mechanism through a UDP socket. The partial code of server and client programs are provided. You must start with the code we provide and complete the client and server programs.

Extra bonus: You can get an extra bonus if you implement the **selective-repeat** mechanism in addition to the stop-and-wait mechanism.

2. Basic Requirements (100%)

a. For server program:

Create a UDP socket of port 9999 and then wait for a request from a client
The request should be "download fileName", and the server should make a response to the client.
If the desired file exists, the server will start to send the video to the client using stop-and-wait mechanism.
In the meantime, the server should keep receiving ACKs from the client.

1. If the ACK is not received in 100 milliseconds (#define TIMEOUT 100), the respective sequence number should be retransmitted.
2. (Hint: use clock()*1000/CLOCKS_PER_SEC+TIMEOUT to record the expired time in milliseconds)

In the partial code of server.c, you need to complete (see the comments in server.c):

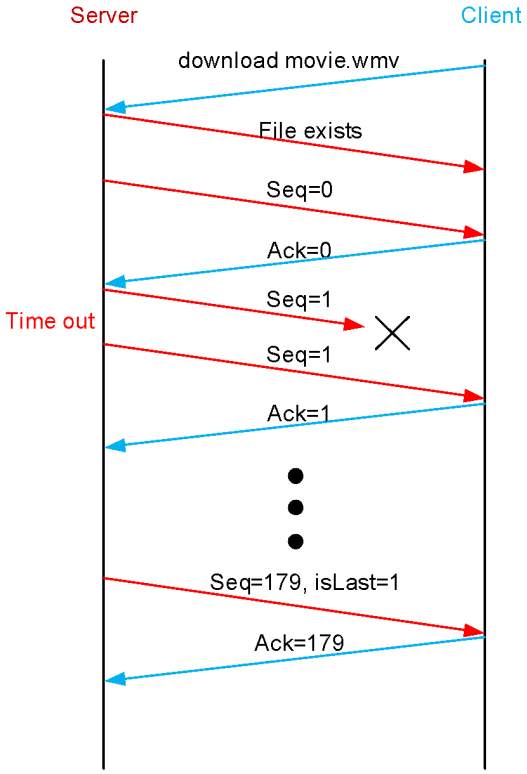
- //complete sendFile() function

b. For client program:

Create an UDP socket and assign the server address
The user can make a command "download fileName" to the server
If the response shows the file exists, then the client can start to receive the file
Upon receiving each sequence number, the client should reply an ACK and write the receiving data to the file if the sequence number is valid
To simulate packet loss, the client will ignore each packet with probability 0.5
In the partial code of client.c, you need to complete (see the comments in client.c)

- // complete recvFile() function

c. A diagram showing the message flow for stop-and-wait



3. Bonus (20%)

Use selective-repeat instead of stop-and-wait, and you should define the window size in the code (ex. #define WND_SIZE 4). For selective-repeat mechanisms, the synchronization between threads may become critical. You can use pthread mutex to organize critical sections such that no more than one critical section can be executed at the same time. For example, you can make three critical sections for sending packets, checking ACK timeout, and handling an ACK packet respectively.

The most challenging part is to maintain multiple ACK timers. When one ACK timeout occurs, only the respective sequence number needs to be present. The other important part is to maintain the sliding window. At the server side, the sliding window will move when the ACKs of the sequence number in the window is received. At the client side, the sliding window will move correspondingly and the data should be written to the file in order.

4. Examples

When you finish coding server.c and client.c, you have to use makefile to make an executable file.

Using makefile for Basic part :

1. Use the "cd" command, change the current path to where you put the server.c and client.c file.
2. Type "sudo apt install make", install make.
3. Type "make", make an executable file of server.c and client.c.
4. If you want to delete server.out and client.out, type "make clean".

Learn more about makefile:

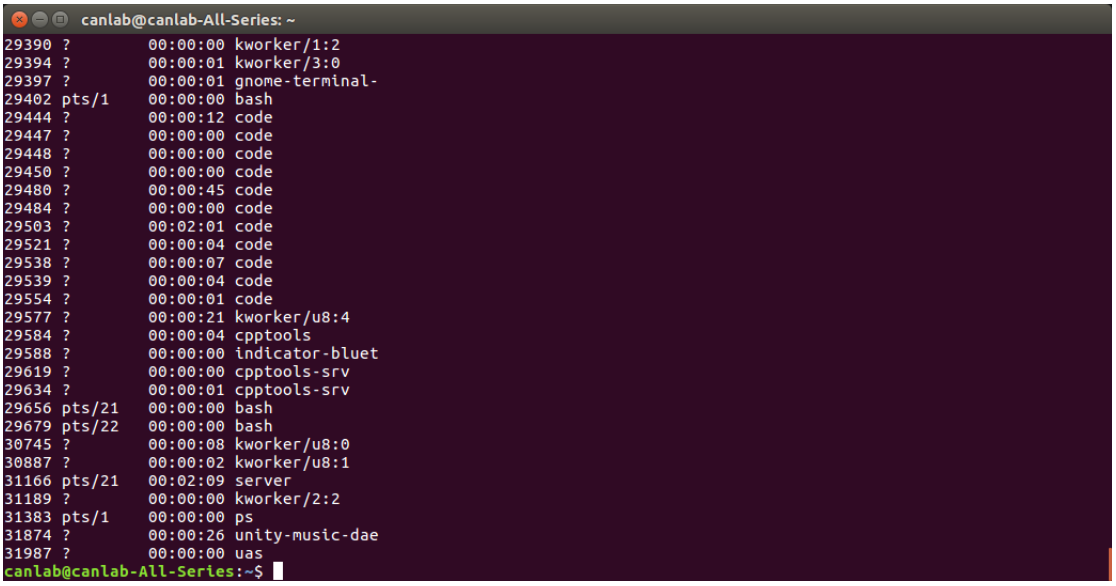
https://mropengate.blogspot.com/2018/01/makefile.html

Learn more about basic linux file management:

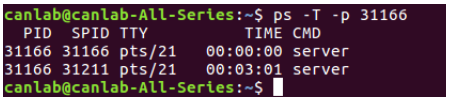
http://linux.vbird.org/linux_basic/0220filemanager/0220filemanager.php

Maintaining thread for Bonus part:

1. Type "ps -A" to watch process id.



2. Type "ps -T -p {pid eg. 31166}" to see # of thread running on a process.



Learn more about pthread:

https://blog.gtwang.org/programming/pthread-multithreading-programming-in-c-tutorial/

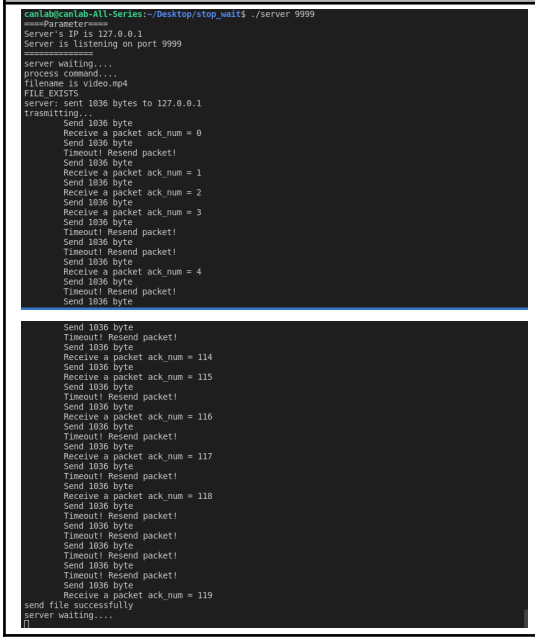
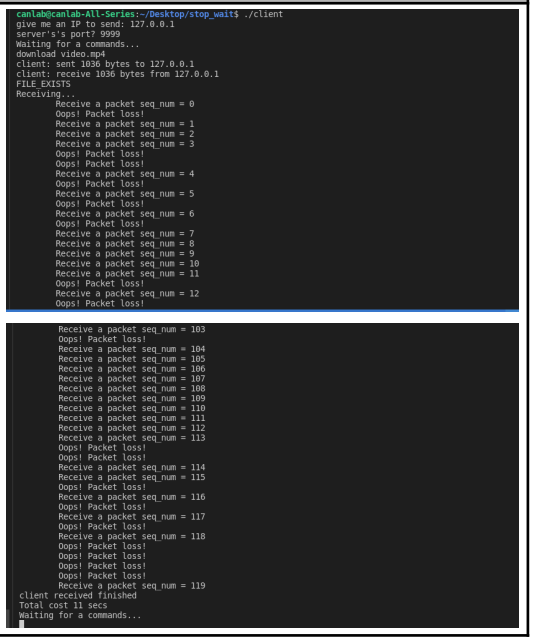
a. Stop-and-wait

Server:

1. Type "./server 9999", argv[1] is for setting the server's port.
   (In some cases, you might need to add "sudo" at the beginning of the command, like "sudo ./server 9999".)

Client:

1. Type "./client"
2. Type "127.0.0.1", set IP address.
3. Type "9999", connect to port.
4. Type "download video.mp4", send download request to server.
5. Type "exit", disconnect to server.

Server	Client
	

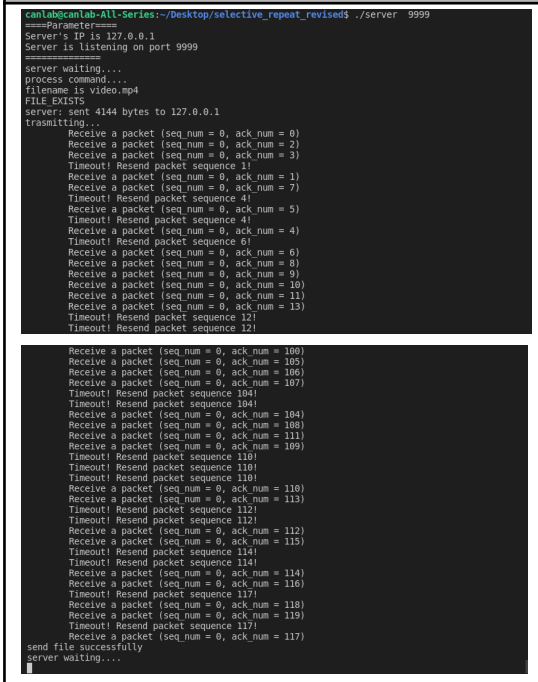
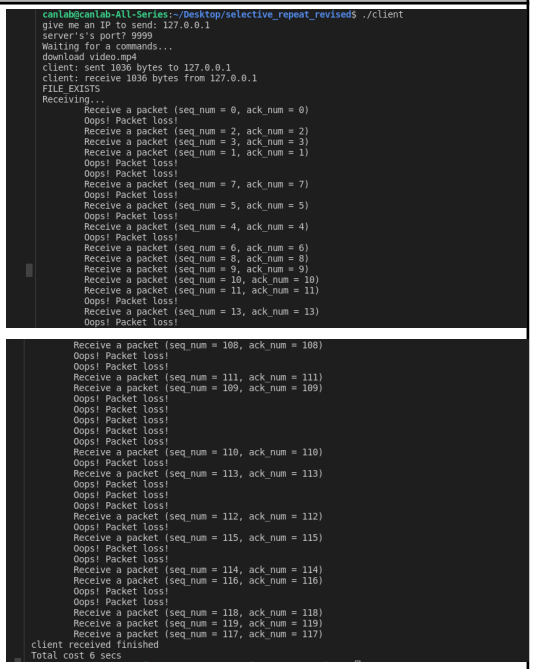
b. Selective-repeat (Window size = 4)

Server:

1. Type"./server 9999", argv[1] is for setting the server's port.
   (In some cases, you might need to add "sudo" at the beginning of the command, like "sudo ./server 9999".)

Client:

1. Type"127.0.0.1", set IP address.
2. Type"9999", connect to port.

Server	Client
	

As we can see, Selective-repeat is more efficient.

5. Hint
   a. How to maintain the ACK timer?
      You can set a timer using clock() of <time.h> in c library to calculate timeout or any other method which can achieve the goal.
      Eg. expiredTime = clock() * 1000 / CLOCKS_PER_SEC + TIME_OUT
      You should periodically check if a timeout occurred.
      Eg. if clock() * 1000 / CLOCKS_PER_SEC >= expiredTime
   b. How to know whether the packet is the last one or not?
      On the client side, check is_last flag in the packet header set by the server.
      On the server side, keep checking the remaining file size and set the is_last flag.
   c. How to simulate packet loss?
      The client can ignore each packet with probability 0.5 using isLoss(0.5).
   d. What if the client is shut down unexpectedly?
      Too many consecutive ACK timeouts.
      If the above event occurs, the server can stop the transmission.
      You don't need to handle this case. Just for your reference.

6. Submission
   a. Please provide a pdf file to show what functionalities your homework has.
      For example, is it able to be compiled by gcc? Does it meet all requirements?
      If you can run your C program, please provide a screenshot to show how it works just like the examples in this document.
   b. Compress the C source file(s) and related files (including readme.pdf) into 學號_作業_版本.zip (ex: 109062599_lab2_v1.zip).
   c. Discussion is encouraged. However, plagiarism is not allowed. We will use, e.g., "Moss" for similarity comparison and 0 points will be given if plagiarism.
   d. You should submit your assignment by the deadline, or your assignment will not be graded, meaning that you will receive zero points.