3rd Group Homework Report

TCP Simple Broadcast Chat Server and Client

ECEN 602 Network Programming Assignment 3

Mainly, we worked together, but here is the role what we did.

Minhwan Oh : Developed client programming, tested for integration

Sanghyeon Lee : Developed server programming, debugged for integration

File info

===============

For this program, you can see how trivial File Transfer Protocol (TFTP) server. There is one main file.

1. tftp\_server.c

Path: cd Assignment\_3/tftp\_server.c

Main feature: Server application for TFTP server.

2. makefile

Main feature: Compile setting description

Build info

===============

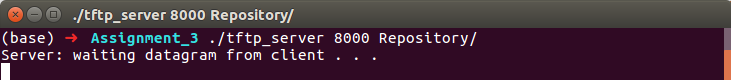
Command $make

Program scenario

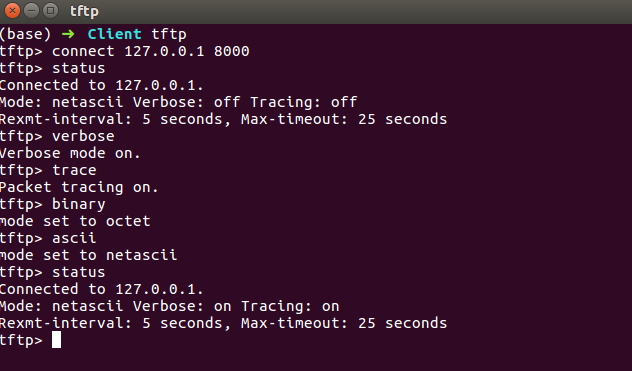
===============

This program includes only server for Trivial File Transfer Protocol (TFTP) because we are going to use a standard TFTP client, which does the following:

1. Start the server first with the commanding lien: $tftp\_server <Port> <Repository>, where “tftp\_server” is the name of the server program, Port is the port number on which the tftp server is listening, and repository is the folder that server can store or transfer files.



2. Assume that TFTP client is installed in linux environment, after server programming runs, tftp client also needs to run. Since we put port number as 8000 in server program, client also needs to set same port name. To test properly, please go cd Assignment\_3/Client then run TFTP client



3. Current our TFTP program can transfer files in both directions because we implemented Read Request Function (RRQ) and Write Request Function (WRQ) as well. The server allows retrieval of files in the assigned local directory.

4. There are several test cases as below, and we comply with all test cases.

(1) Transfer a binary file of 2048 bytes and check that it matches the source file

(2) Transfer a binary file of 2047 bytes and check that it matches the source file

(3) Transfer a netascii file that includes two CR’s and check that the resulting file matches the input file

(4) Transfer a binary file of 34 MB and see if block number wrap-around works

(5) check that you receive an error message if you try to transfer a file that does not exist and that your server cleans up and the child process exits

(6) Connect to the TFTP server with three clients simultaneously and test that the transfers work correctly (you will probably need a big file to have them all running at the same time)

(7) Terminate the TFTP client in the middle of a transfer and see if your TFTP server recognizes after 10 timeouts that the client is no longer there (you will need a big file)

(8) Transfer a binary file of 2048 bytes to server and check that it matches the source file ( Bonus Feature )

(9) Transfer a binary file of 2047 bytes to server and check that it matches the source file ( Bonus Feature )

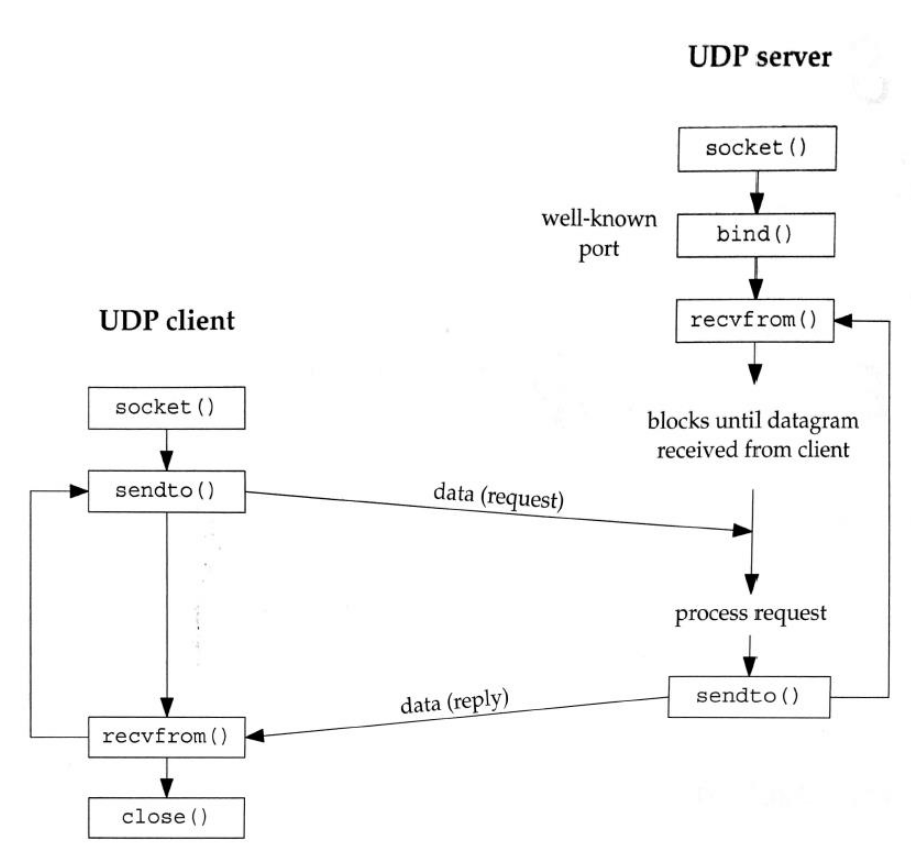
(10) Connect to the TFTP server with three clients simultaneously and test that the transfers work correctly – transmitting file to server simultaneously ( Bonus Feature )

(11) Terminate the TFTP server in the middle of a transfer and see if your TFTP client recognizes after 10 timeouts that the client is no longer there ( Bonus Feature )

Program details & File architecture

===============

1. User Datagram Protocol (UDP) Programming

- Recall that UDP is aconnectionless, unreliable, datagram protocol. In contrast, TCP is connection- oriented and provides a reliable byte stream.

- The UDP server does not issue a listen() or an accept(). Instead, the UDP server just calls the recvfrom() function, which blocks until data arrives from some client. The recvfrom() function returns the address of the client (IP address and port number), as well as the datagram, so the server can respond to the client.

2. Multiple simultaneous connections

- To support multiple simultaneous connections, used fork() function in server which a child process to handle each client transfer, similar to TCP Echo Server.

3. Timeout

- The sender of a TFTP DATA packet (the TFTP Server in the case of a RRQ request or the TFTP Client in the case of a WRQ request) needs to (1) send a DATA packet, (2) set a timeout timer, and (3) wait for an ACK. If the ACK arrives before the timeout timer expires, the timer is cancelled and you then process the next packet. If the timeout timer expires, you must retransmit the last DATA packet.

- We implemented timeout in select() function : Call select() to wait for either of the following two events: (1) data ready for reading on the socket descriptor created by the child server process and(2) a timeout of say one second from when select() is called.

4. File transmission

- For octet transmission, we used the standard Unix/Linux open(), read() (RRQ), and write() (Bonus WRQ) system calls.

- For netascii transmission, however, we used the Unix/Linux fopen(), getc() and putc() system calls.

File Function Explanation

===============

1. tftp\_server

a) Architecture

- Scenario :

(1) Setup server\_info/bind port/broadcast/listen functions

(2) Then it is going to infinite loop until the exit

- Main function :

(1) int main (int argc, char\*\* argv) : Including main scenario

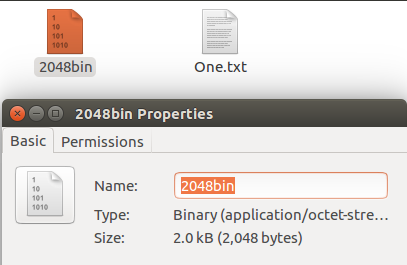
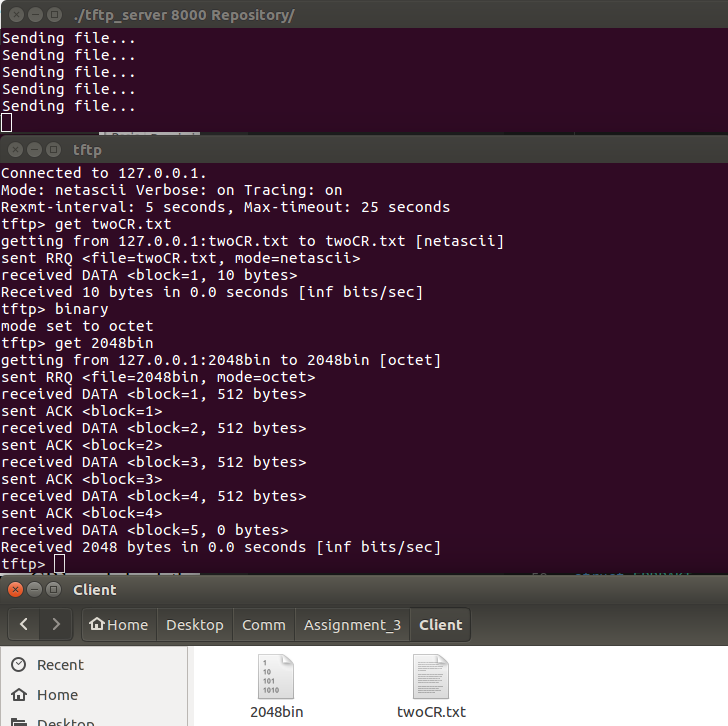
(2) void ErrSend (int errNum, struct sockaddr\_in client\_Addr) : Send an error message

(3) int DataSend (char\* fileDir, char\* mode\_Str, struct sockaddr\_in client\_Addr) : Send the file to client

(4) int recvData (char\* fileDir, char\* mode\_Str, struct sockaddr\_in client\_Addr) : Receive file from client

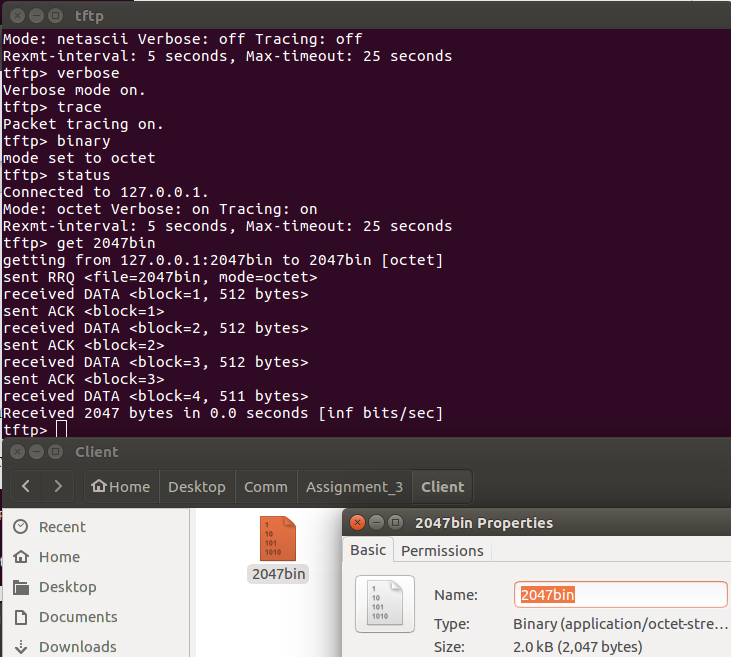
Test cases

===============

1.  Transfer a binary file of 2048 bytes and check that it matches the source file

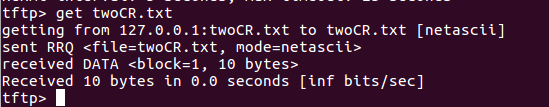
- After setting binary in client, we can see 2048 bin file to transfer from server to client properly.

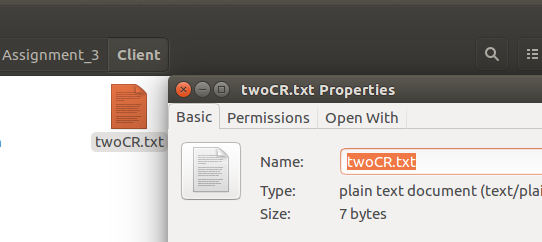
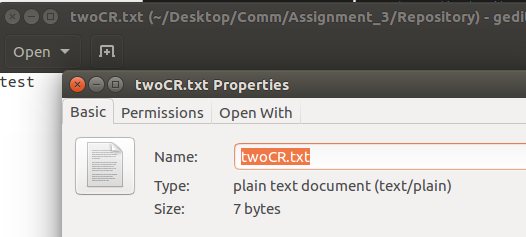
(2) transfer a binary file of 2047 bytes and check that it matches the source file



- It works properly as expected. The different thing from test 1 is last received data for block 4 is 511 bytes.

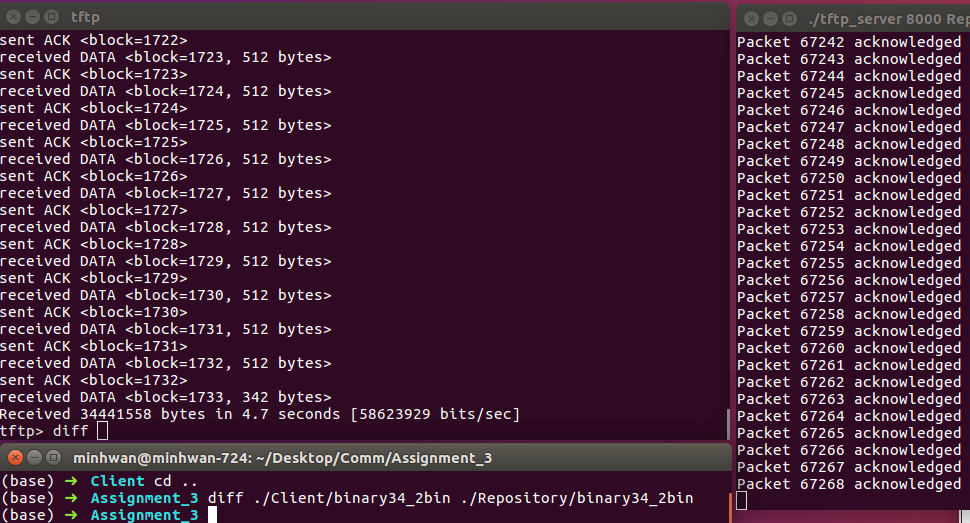
(3) transfer a netascii file that includes two CR’s and check that the resulting file matches the input file





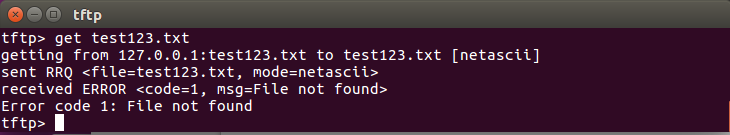
- As test, received data was same as original one which contained CR.

(4) transfer a binary file of 34 MB and see if block number wrap-around works



- As a result with diff function, we could see client could receive 34MB file properly. However, sender was marked 67268 packets as log and client was 1733 because the maximum packet client can log as 65535, so after that the packet number was reset to 0, then started again. The functional point of view, client worked fine.

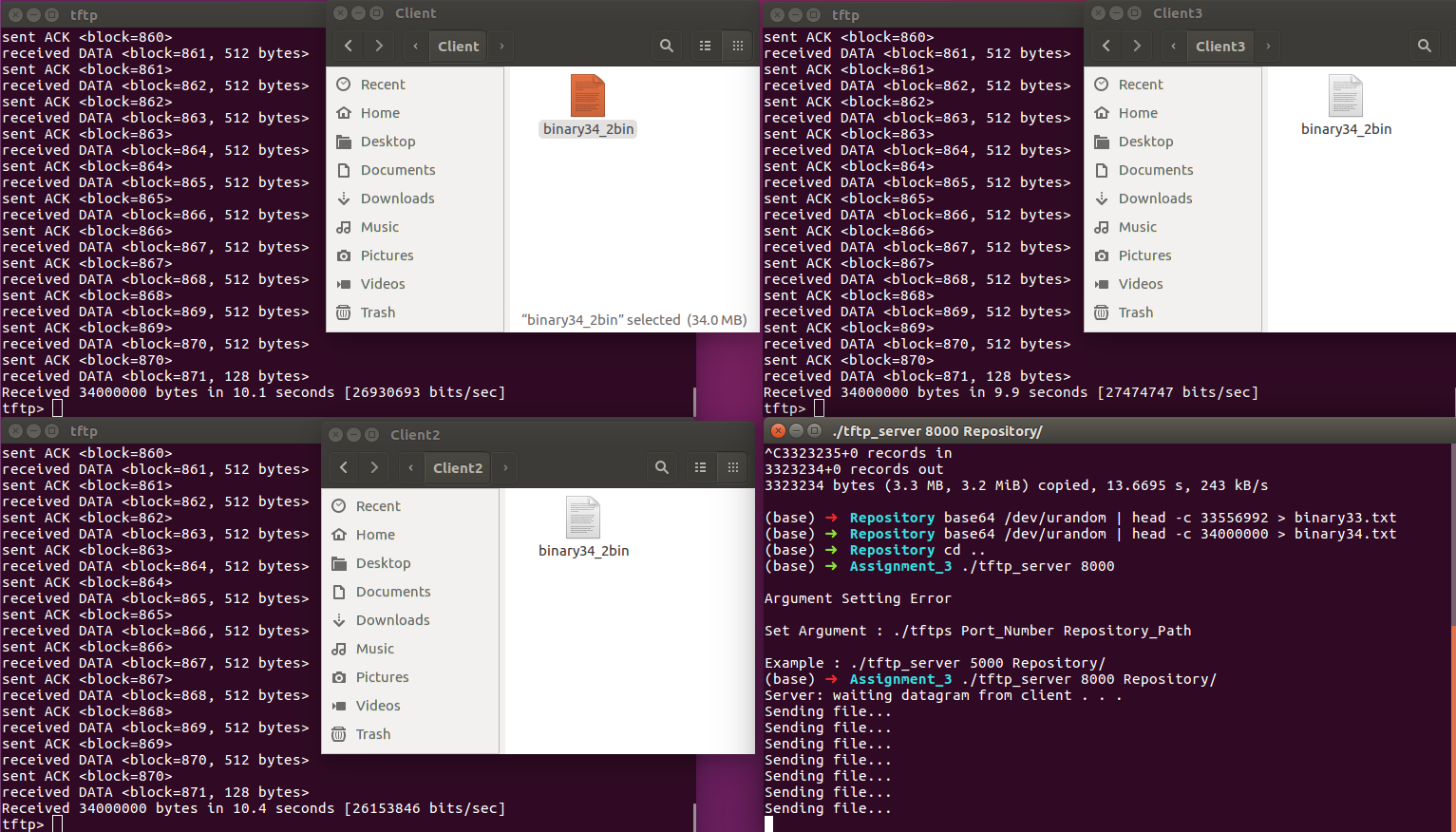
(5) check that you receive an error message if you try to transfer a file that does not exist and that your server cleans up and the child process exits

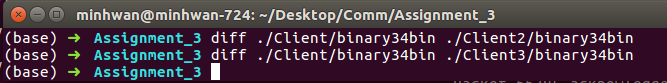


- If client requested some file that didn’t exist, we could check that server cleaned up and the child process existed.

(6) Connect to the TFTP server with three clients simultaneously and test that the transfers work correctly (you will probably need a big file to have them all running at the same time)

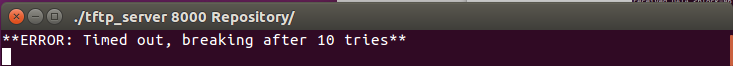
- It works properly. To have this test, three clients to receive 34MB file at same time. Comparing to running one client, running time is linearly increased because server technically can’t send data at same time. Instead, server sends data to clients in order.





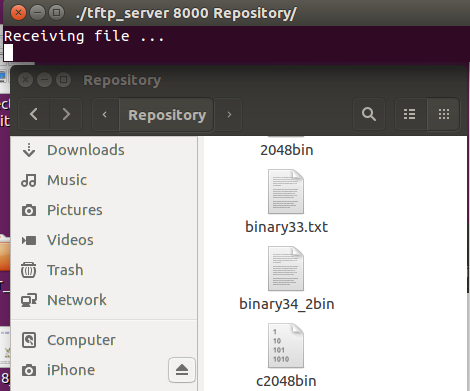
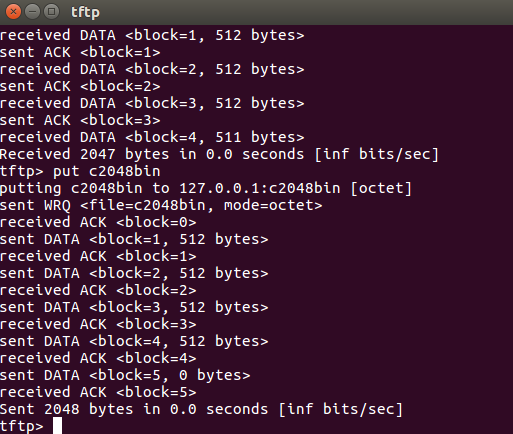
(7) Terminate the TFTP client in the middle of a transfer and see if your TFTP server recognizes after 10 timeouts that the client is no longer there (you will need a big file)

- After trying to transmit 10 times, server occurs time outs



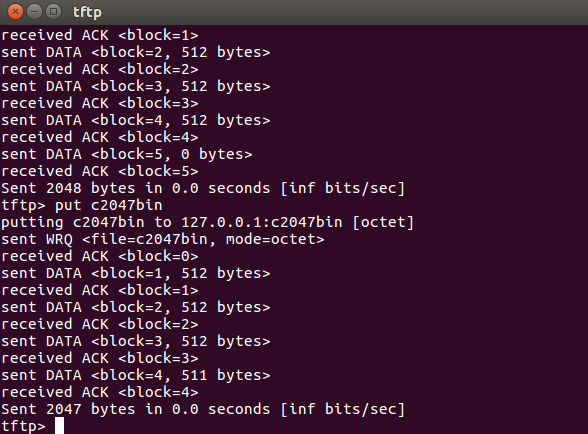
(8) Transfer a binary file of 2048 bytes to server and check that it matches the source file ( Bonus Feature )

- We can see 2048 bin file to transfer from client to server properly. Also, we can check WRQ in log as well.



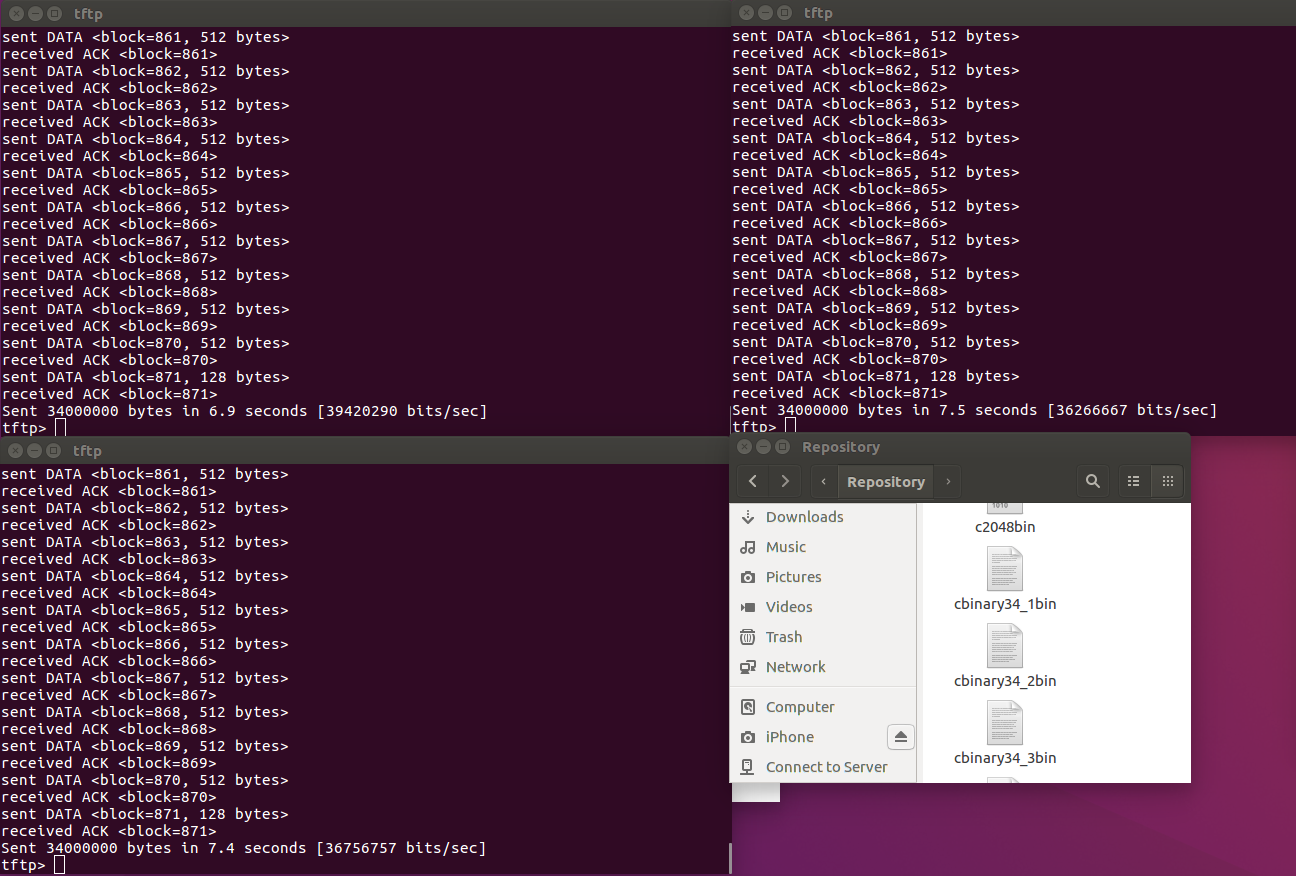
(9) Transfer a binary file of 2047 bytes to server and check that it matches the source file ( Bonus Feature )

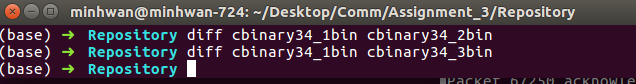
- We can see 2047 bin file to transfer from client to server properly. Also, we can check WRQ in log as well. Different thing from test case 8 is last block 4 transmits 511 bytes as expected



(10) Connect to the TFTP server with three clients simultaneously and test that the transfers work correctly – transmitting file to server simultaneously ( Bonus Feature )

- As you can see below picture, 3 clients can send their files to server at same time. And, checked all files were sent properly.





(11) Terminate the TFTP server in the middle of a transfer and see if your TFTP client recognizes it that the server is no longer there ( Bonus Feature )

- Of course, it could be not our project range because client is not developed from us. However, for make sure, we tested client side as well. As you can see, after trying 5 times, client occurs time out.

