Programming Assignment 3

Chenjie Luo (UIN: 324007289)
Yuwen Chen (UIN: 227009499)

a. Description

This project is roughly divided into two parts. Chenjie Luo is responsible for building up the framework and local file input and output while Yuwen Chen is responsible for the timoutouts, multiple clients handling and we verified server in different test cases together.

For the server's architecture, when target file is executed, users will have to enter two inputs which are: server's IP address as well as port number. I designed a struct to store named TFTP_MSG to store messages read from buffer with the given form in recitation. The struct comes with four attributes: opcode, block_num, filename and mode. The opcode is an integer indicates which type the message belong to. The filename stores the object file name we want to transmit. Block_num is used for ACK message and it indicates the acknowledgement packet number.

To handle multiple clients, we used fork() to create child process in order to take care of each client's transfer. When the socket is crated and bind successfully, we will listen to RRQ message. If RRQ is found in TFTP_MSG, we will start a new child process using fork() and firstly close the socket with port number specified when opening the server. We then created a new socket and bind with the ephemeral port.

For I/O operation, FILE* f is defined to implement I/O operations. If the f is failed to be open, an error message (opcode = 5) is generated and send back to the client. Else FILE* f can be open successfully and file size is first recorded number of packets needed is calculated. For the I/O part, at the beginning I used EOF to detect the end of file but it seemed sometimes getc() could not find EOF when we used large random generated files. I think reason lies in MACRO of "EOF" is different on different machines. Then I switched to calculate how many remaining bytes needed to be transmitted since I recorded total file size. All the packet except the last should have 512 bytes. And this turns out to work correctly. All the packet will be wrapped and send to the client. When client received an ACK message should be echoed back. We updated the block_num and transmit next packet then. When we came across extremely large file (> 32 MB), the default TFTP client will terminate after ACK 65536th packet. In this case we will need to reset the ACK number to make it always less than 65536. We used another variable named block_cnt to count total number of packets right now and used it for detecting last packet as well.

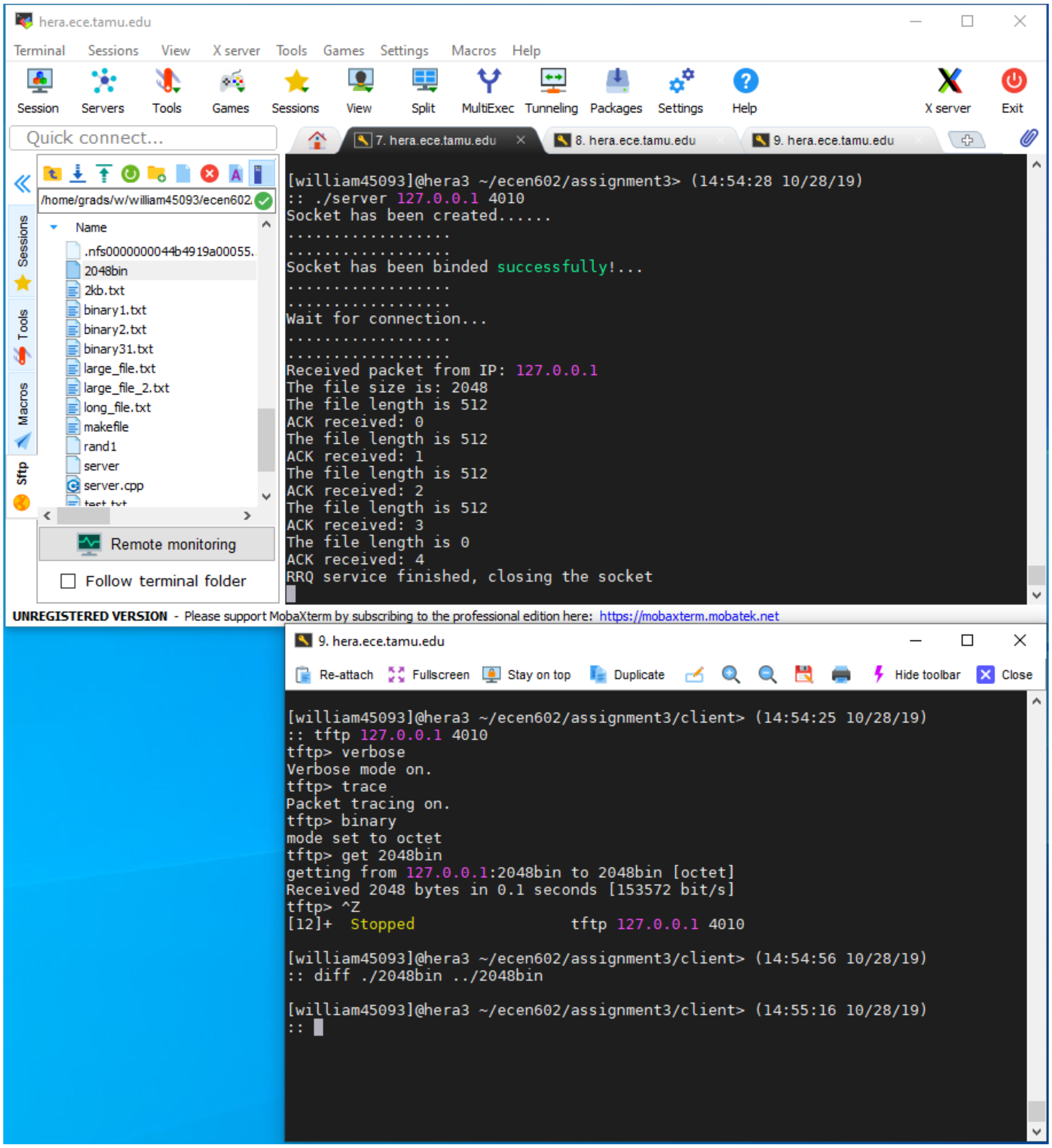
For the timeout function, after the server send a packet to the client, a select function is used in the server side to wait for the acknowledgement message from the client indicating that the packet is received by the client. If the server does not receive the acknowledgement message in 1 second after sending the packet, the select function will be timeout and return value 0. In this case, a for loop is used to retransmit the last packet again until the acknowledgement message is received (select function return value larger than 0). The for loop will only retransmit a certain packet for 10 times, and if a packet is retransmitted over 10 times, and the server still not receive the corresponding acknowledgement message, the following transmission to the client will be shutdown, and both the socket bound to the client and the child process to handle this transmission will be closed.

b. Instruction to run our code

1. After downloading the file from github, type “make” in the command line to generate the execution file: server
2. Type “./server [server IP address] [port number]” to execute the server
3. Now, users can execute the default TFTP client by typing “tftp [server IP address] [port number]” to connect to our server. After the connection is created successfully, users can send a RRQ request to our server to download files from our server.

c. Test result

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

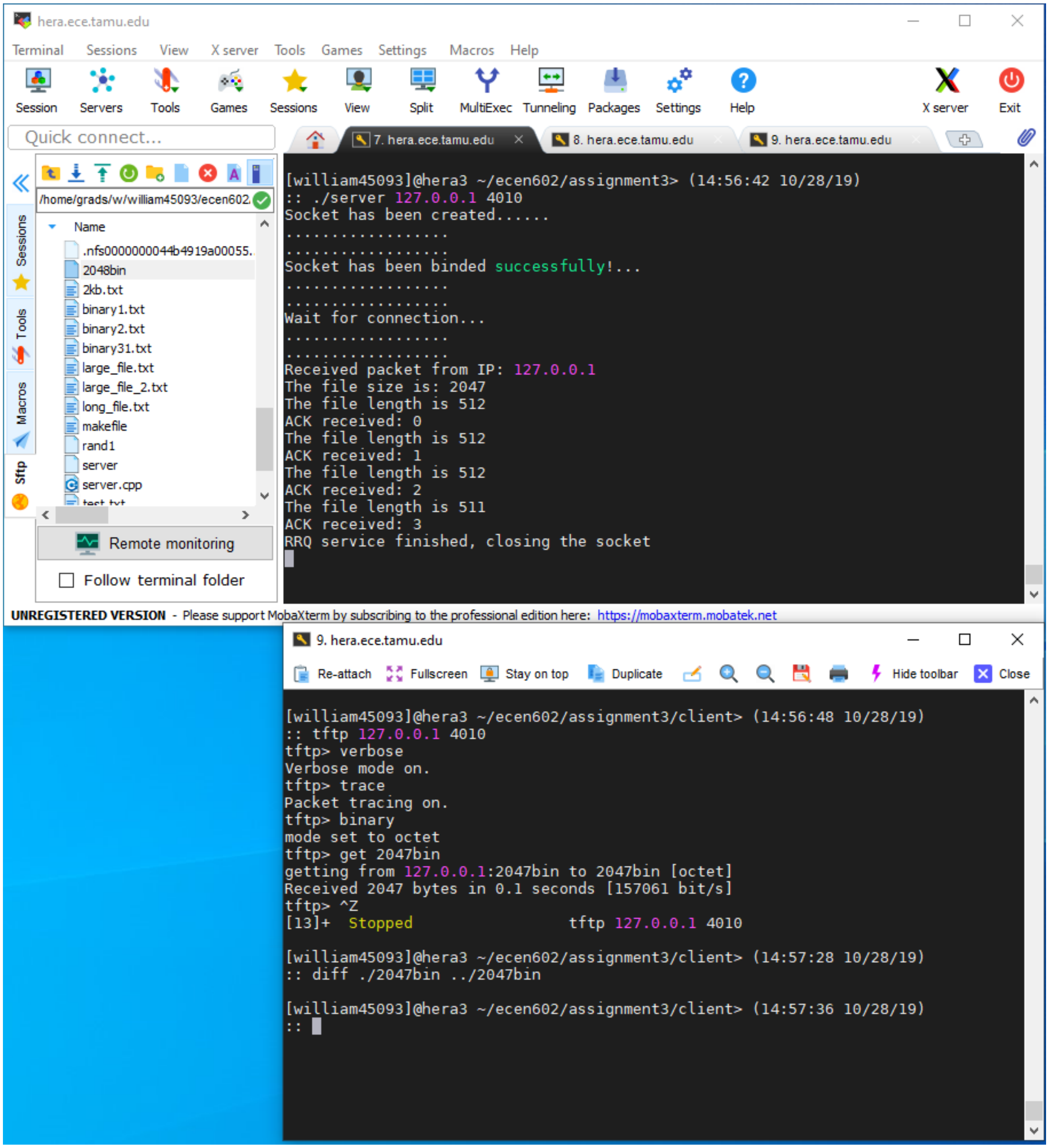


hera.ece.tamu.edu
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
7. hera.ece.tamu.edu
8. hera.ece.tamu.edu
9. hera.ece.tamu.edu
/home/grads/w/william45093/ecen602
Name
.nfs0000000044b4919a00055.
2048bin
2kb.txt
binary1.txt
binary2.txt
binary31.txt
large_file.txt
large_file_2.txt
long_file.txt
makefile
rand1
server
server.cpp
test.txt
Sessions
Tools
Macros
Sftp
Remote monitoring
☐ Follow terminal folder
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net
9. hera.ece.tamu.edu
Re-attach Fullscreen Stay on top Duplicate Hide toolbar Close

```
[william45093]@hera3 ~/ecen602/assignment3> (14:54:28 10/28/19)
:: ./server 127.0.0.1 4010
Socket has been created......
..................
..................
Socket has been binded successfully!...
..................
..................
Wait for connection...
..................
..................
Received packet from IP: 127.0.0.1
The file size is: 2048
The file length is 512
ACK received: 0
The file length is 512
ACK received: 1
The file length is 512
ACK received: 2
The file length is 512
ACK received: 3
The file length is 0
ACK received: 4
RRQ service finished, closing the socket
```

```
[william45093]@hera3 ~/ecen602/assignment3/client> (14:54:25 10/28/19)
:: tftp 127.0.0.1 4010
tftp> verbose
Verbose mode on.
tftp> trace
Packet tracing on.
tftp> binary
mode set to octet
tftp> get 2048bin
getting from 127.0.0.1:2048bin to 2048bin [octet]
Received 2048 bytes in 0.1 seconds [153572 bit/s]
tftp> ^Z
[12]+  Stopped                 tftp 127.0.0.1 4010

[william45093]@hera3 ~/ecen602/assignment3/client> (14:54:56 10/28/19)
:: diff ./2048bin ../2048bin

[william45093]@hera3 ~/ecen602/assignment3/client> (14:55:16 10/28/19)
::
```

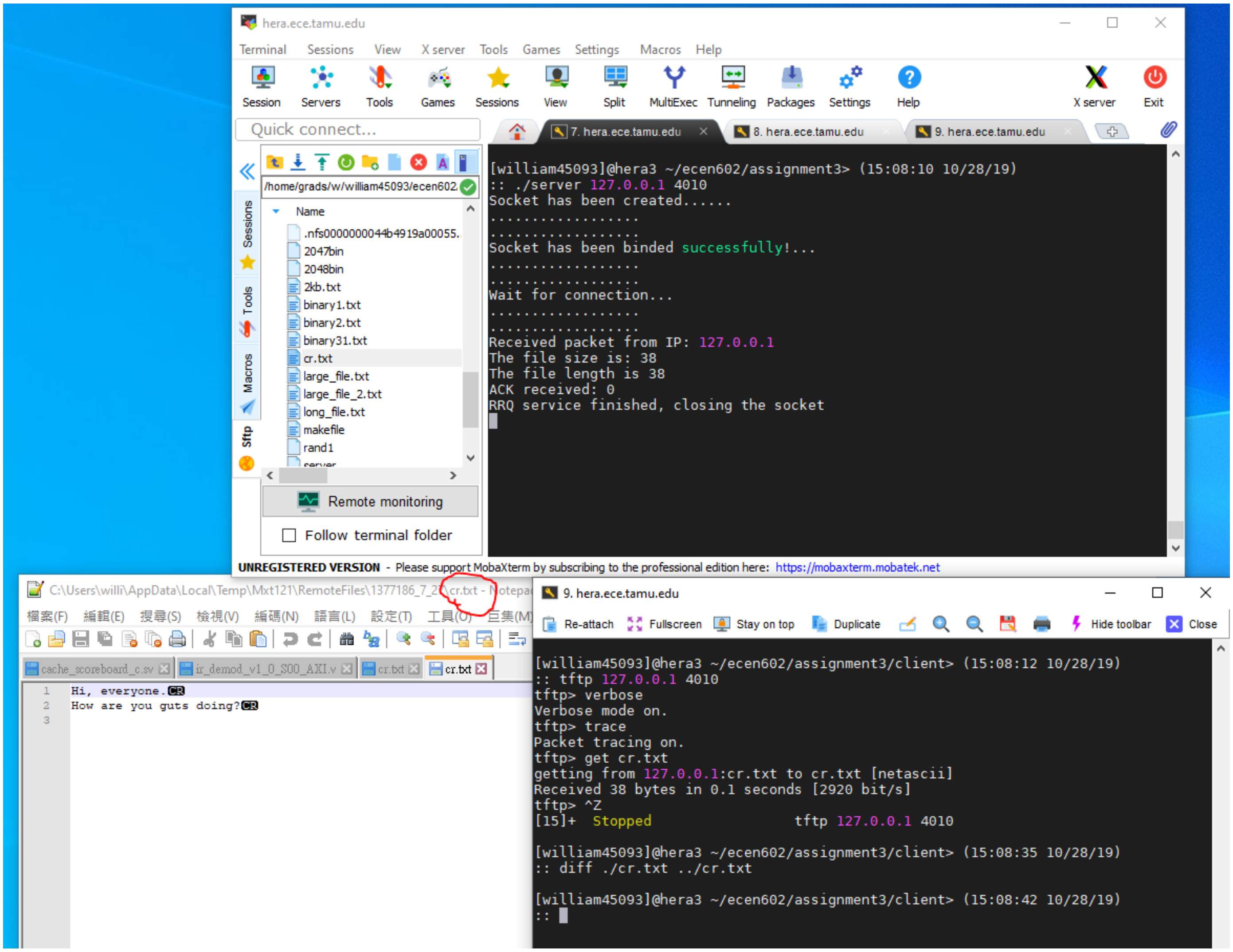
2. transfer a binary file of 2047 bytes and check that it matches the source file


hera.ece.tamu.edu
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
7. hera.ece.tamu.edu
8. hera.ece.tamu.edu
9. hera.ece.tamu.edu
/home/grads/w/william45093/ecen602
Name
.nfs0000000044b4919a00055.
2048bin
2kb.txt
binary1.txt
binary2.txt
binary31.txt
large_file.txt
large_file_2.txt
long_file.txt
makefile
rand1
server
server.cpp
test.txt
Sessions
Tools
Macros
Sftp
Remote monitoring
Follow terminal folder
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net
9. hera.ece.tamu.edu
Re-attach Fullscreen Stay on top Duplicate Hide toolbar Close

```
[william45093]@hera3 ~/ecen602/assignment3> (14:56:42 10/28/19)
:: ./server 127.0.0.1 4010
Socket has been created......
.................
.................
Socket has been binded successfully!...
.................
.................
Wait for connection...
.................
.................
Received packet from IP: 127.0.0.1
The file size is: 2047
The file length is 512
ACK received: 0
The file length is 512
ACK received: 1
The file length is 512
ACK received: 2
The file length is 511
ACK received: 3
RRQ service finished, closing the socket
```

```
[william45093]@hera3 ~/ecen602/assignment3/client> (14:56:48 10/28/19)
:: tftp 127.0.0.1 4010
tftp> verbose
Verbose mode on.
tftp> trace
Packet tracing on.
tftp> binary
mode set to octet
tftp> get 2047bin
getting from 127.0.0.1:2047bin to 2047bin [octet]
Received 2047 bytes in 0.1 seconds [157061 bit/s]
tftp> ^Z
[13]+  Stopped                 tftp 127.0.0.1 4010

[william45093]@hera3 ~/ecen602/assignment3/client> (14:57:28 10/28/19)
:: diff ./2047bin ../2047bin

[william45093]@hera3 ~/ecen602/assignment3/client> (14:57:36 10/28/19)
::
```

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


hera.ece.tamu.edu
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
7. hera.ece.tamu.edu
8. hera.ece.tamu.edu
9. hera.ece.tamu.edu
/home/grads/w/william45093/ecen602
Name
.nfs0000000044b4919a00055.
2047bin
2048bin
2kb.txt
binary1.txt
binary2.txt
binary31.txt
cr.txt
large_file.txt
large_file_2.txt
long_file.txt
makefile
rand1
server
Sessions
Tools
Macros
Sftp
Remote monitoring
□ Follow terminal folder
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

```
[william45093]@hera3 ~/ecen602/assignment3> (15:08:10 10/28/19)
:: ./server 127.0.0.1 4010
Socket has been created......
.................
.................
Socket has been binded successfully!...
.................
.................
Wait for connection...
.................
.................
Received packet from IP: 127.0.0.1
The file size is: 38
The file length is 38
ACK received: 0
RRQ service finished, closing the socket
```

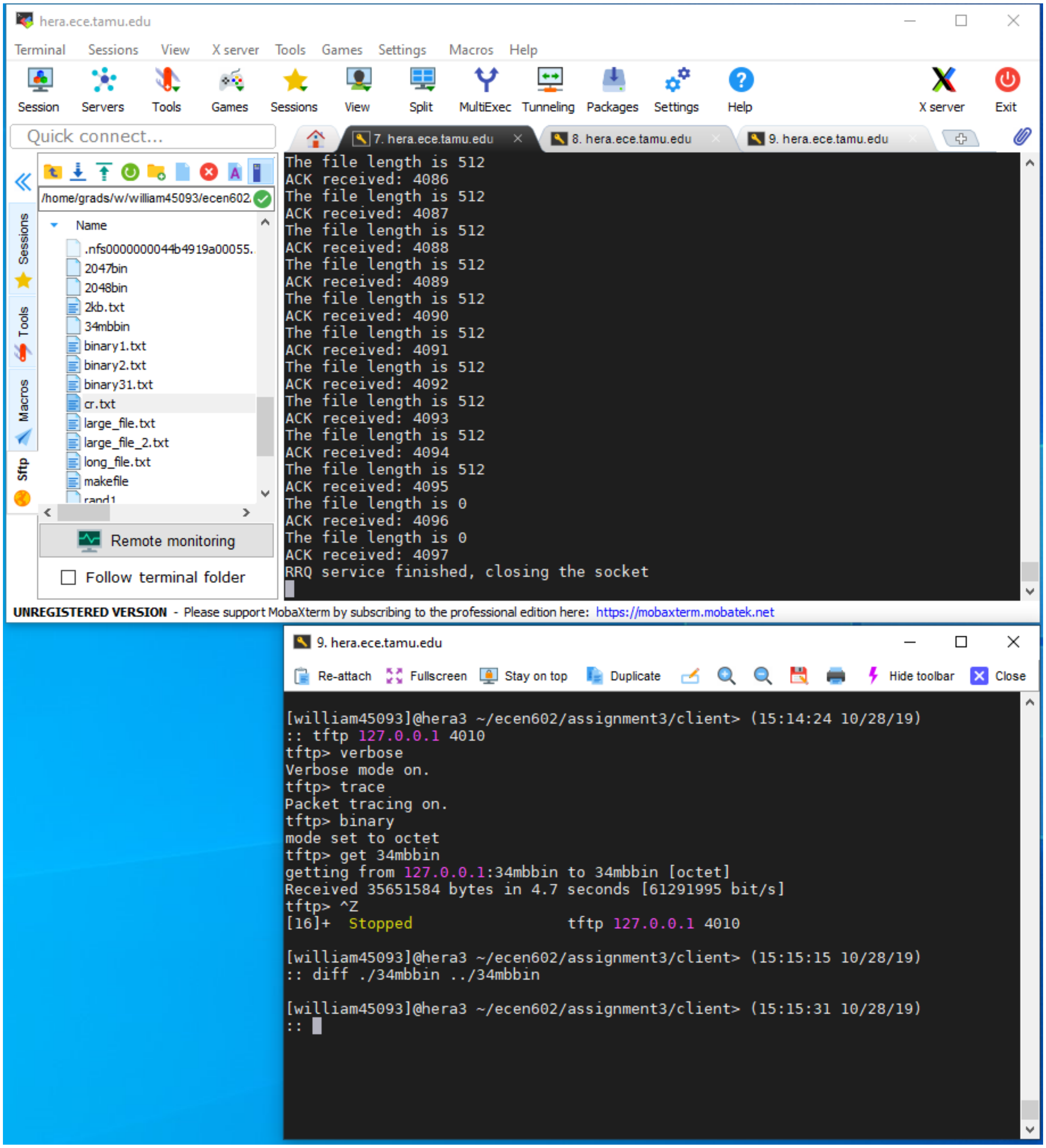
C:\Users\willi\AppData\Local\Temp\Mxt121\RemoteFiles\1377186_7_2\cr.txt - Notepa
檔案(F) 編輯(E) 搜尋(S) 檢視(V) 編碼(N) 語言(L) 設定(T) 工具(O) 巨集(M)
cache_scoreboard_c.sv
ir_demod_v1_0_S00_AXI.v
cr.txt
cr.txt

```
Hi, everyone.CR
How are you guts doing?CR

```

9. hera.ece.tamu.edu
Re-attach Fullscreen Stay on top Duplicate Hide toolbar Close

```
[william45093]@hera3 ~/ecen602/assignment3/client> (15:08:12 10/28/19)
:: tftp 127.0.0.1 4010
tftp> verbose
Verbose mode on.
tftp> trace
Packet tracing on.
tftp> get cr.txt
getting from 127.0.0.1:cr.txt to cr.txt [netascii]
Received 38 bytes in 0.1 seconds [2920 bit/s]
tftp> ^Z
[15]+  Stopped                 tftp 127.0.0.1 4010

[william45093]@hera3 ~/ecen602/assignment3/client> (15:08:35 10/28/19)
:: diff ./cr.txt ../cr.txt

[william45093]@hera3 ~/ecen602/assignment3/client> (15:08:42 10/28/19)
::
```

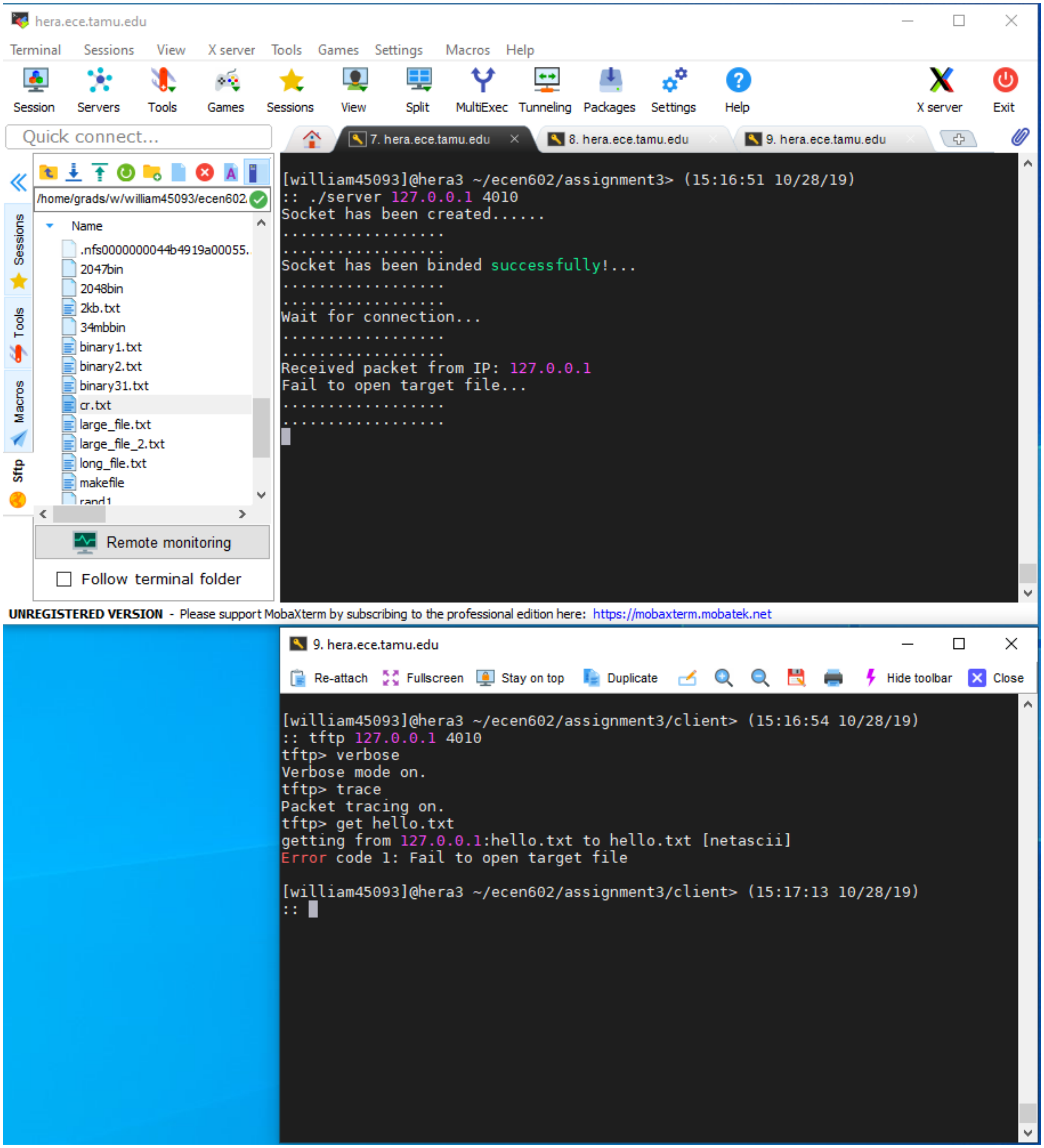
4. transfer a binary file of 34 MB and see if block number wrap-around works


hera.ece.tamu.edu
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
7. hera.ece.tamu.edu
8. hera.ece.tamu.edu
9. hera.ece.tamu.edu
Sessions Tools Macros Sftp
/home/grads/w/william45093/ecen602
Name
.nfs0000000044b4919a00055.
2047bin
2048bin
2kb.txt
34mbbin
binary1.txt
binary2.txt
binary31.txt
cr.txt
large_file.txt
large_file_2.txt
long_file.txt
makefile
rand1
Remote monitoring
Follow terminal folder
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net
9. hera.ece.tamu.edu
Re-attach Fullscreen Stay on top Duplicate Hide toolbar Close

```
The file length is 512
ACK received: 4086
The file length is 512
ACK received: 4087
The file length is 512
ACK received: 4088
The file length is 512
ACK received: 4089
The file length is 512
ACK received: 4090
The file length is 512
ACK received: 4091
The file length is 512
ACK received: 4092
The file length is 512
ACK received: 4093
The file length is 512
ACK received: 4094
The file length is 512
ACK received: 4095
The file length is 0
ACK received: 4096
The file length is 0
ACK received: 4097
RRQ service finished, closing the socket
```

```
[william45093]@hera3 ~/ecen602/assignment3/client> (15:14:24 10/28/19)
:: tftp 127.0.0.1 4010
tftp> verbose
Verbose mode on.
tftp> trace
Packet tracing on.
tftp> binary
mode set to octet
tftp> get 34mbbin
getting from 127.0.0.1:34mbbin to 34mbbin [octet]
Received 35651584 bytes in 4.7 seconds [61291995 bit/s]
tftp> ^Z
[16]+  Stopped                 tftp 127.0.0.1 4010

[william45093]@hera3 ~/ecen602/assignment3/client> (15:15:15 10/28/19)
:: diff ./34mbbin ../34mbbin

[william45093]@hera3 ~/ecen602/assignment3/client> (15:15:31 10/28/19)
::
```

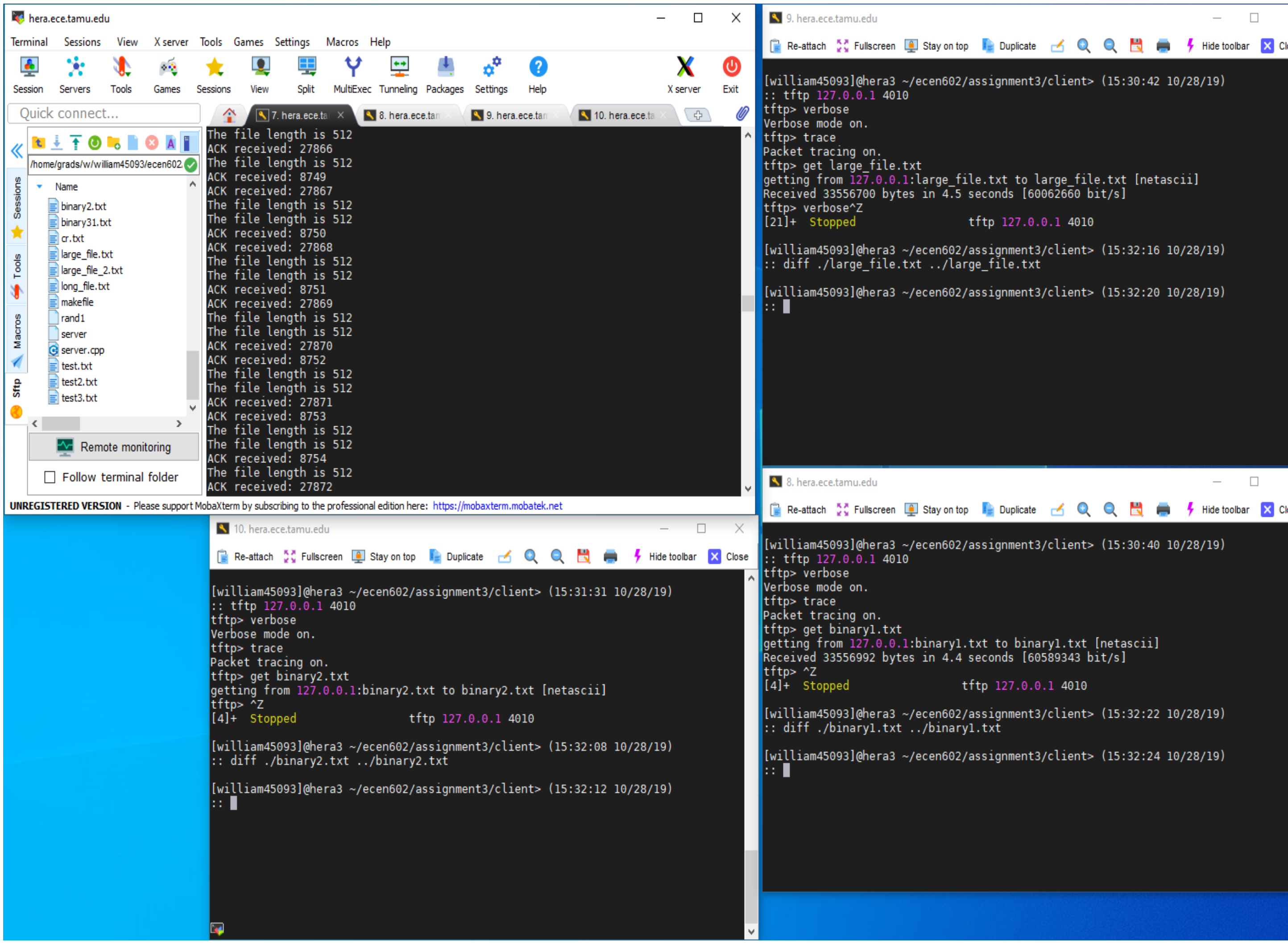
5. check that you receive an error message if you try to transfer a file that does not exist and that your server cleans up and the child process exits


hera.ece.tamu.edu
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
7. hera.ece.tamu.edu
8. hera.ece.tamu.edu
9. hera.ece.tamu.edu
/home/grads/w/william45093/ecen602
Name
.nfs0000000044b4919a00055.
2047bin
2048bin
2kb.txt
34mbbin
binary1.txt
binary2.txt
binary31.txt
cr.txt
large_file.txt
large_file_2.txt
long_file.txt
makefile
rand1
Sessions
Tools
Macros
Sftp
Remote monitoring
Follow terminal folder
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net
9. hera.ece.tamu.edu
Re-attach Fullscreen Stay on top Duplicate Hide toolbar Close

```
[william45093]@hera3 ~/ecen602/assignment3> (15:16:51 10/28/19)
:: ./server 127.0.0.1 4010
Socket has been created......
..................
..................
Socket has been binded successfully!...
..................
..................
Wait for connection...
..................
..................
Received packet from IP: 127.0.0.1
Fail to open target file...
..................
..................
```

```
[william45093]@hera3 ~/ecen602/assignment3/client> (15:16:54 10/28/19)
:: tftp 127.0.0.1 4010
tftp> verbose
Verbose mode on.
tftp> trace
Packet tracing on.
tftp> get hello.txt
getting from 127.0.0.1:hello.txt to hello.txt [netascii]
Error code 1: Fail to open target file

[william45093]@hera3 ~/ecen602/assignment3/client> (15:17:13 10/28/19)
::
```

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)


hera.ece.tamu.edu
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
7. hera.ece.ta
8. hera.ece.tan
9. hera.ece.tan
10. hera.ece.ta
/home/grads/w/william45093/ecen602
Name
binary2.txt
binary31.txt
cr.txt
large_file.txt
large_file_2.txt
long_file.txt
makefile
rand1
server
server.cpp
test.txt
test2.txt
test3.txt
Sessions
Tools
Macros
Sftp
Remote monitoring
□ Follow terminal folder
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

```
The file length is 512
ACK received: 27866
The file length is 512
ACK received: 8749
ACK received: 27867
The file length is 512
The file length is 512
ACK received: 8750
ACK received: 27868
The file length is 512
The file length is 512
ACK received: 8751
ACK received: 27869
The file length is 512
The file length is 512
ACK received: 27870
ACK received: 8752
The file length is 512
The file length is 512
ACK received: 27871
ACK received: 8753
The file length is 512
The file length is 512
ACK received: 8754
The file length is 512
ACK received: 27872
```

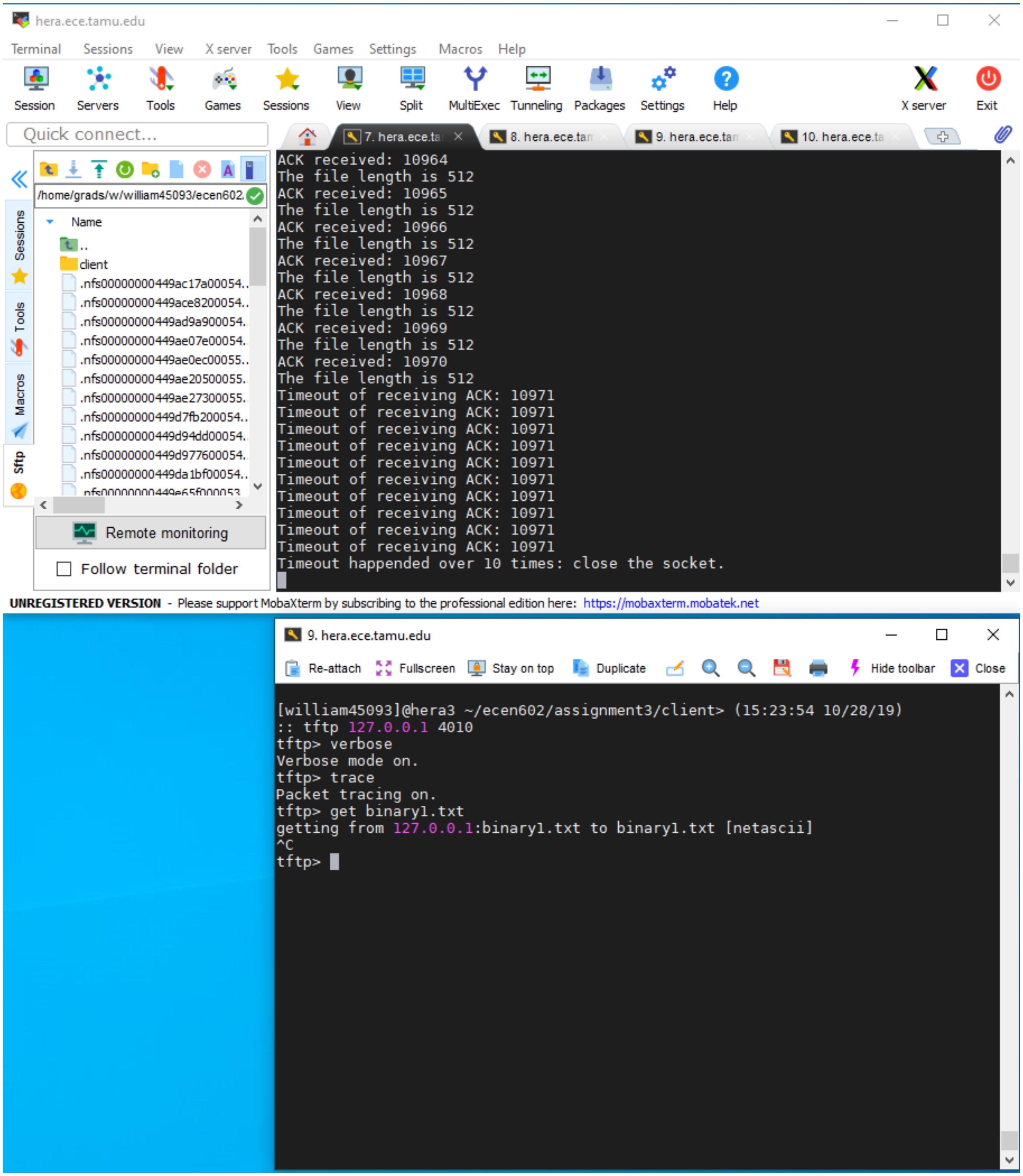
```
[william45093]@hera3 ~/ecen602/assignment3/client> (15:30:42 10/28/19)
:: tftp 127.0.0.1 4010
tftp> verbose
Verbose mode on.
tftp> trace
Packet tracing on.
tftp> get large_file.txt
getting from 127.0.0.1:large_file.txt to large_file.txt [netascii]
Received 33556700 bytes in 4.5 seconds [60062660 bit/s]
tftp> verbose^Z
[21]+  Stopped                 tftp 127.0.0.1 4010

[william45093]@hera3 ~/ecen602/assignment3/client> (15:32:16 10/28/19)
:: diff ./large_file.txt ../large_file.txt

[william45093]@hera3 ~/ecen602/assignment3/client> (15:32:20 10/28/19)
::
```

10. hera.ece.tamu.edu
Re-attach Fullscreen Stay on top Duplicate Hide toolbar Close

```
[william45093]@hera3 ~/ecen602/assignment3/client> (15:31:31 10/28/19)
:: tftp 127.0.0.1 4010
tftp> verbose
Verbose mode on.
tftp> trace
Packet tracing on.
tftp> get binary2.txt
getting from 127.0.0.1:binary2.txt to binary2.txt [netascii]
tftp> ^Z
[4]+  Stopped                 tftp 127.0.0.1 4010

[william45093]@hera3 ~/ecen602/assignment3/client> (15:32:08 10/28/19)
:: diff ./binary2.txt ../binary2.txt

[william45093]@hera3 ~/ecen602/assignment3/client> (15:32:12 10/28/19)
::
```

```
[william45093]@hera3 ~/ecen602/assignment3/client> (15:30:40 10/28/19)
:: tftp 127.0.0.1 4010
tftp> verbose
Verbose mode on.
tftp> trace
Packet tracing on.
tftp> get binary1.txt
getting from 127.0.0.1:binary1.txt to binary1.txt [netascii]
Received 33556992 bytes in 4.4 seconds [60589343 bit/s]
tftp> ^Z
[4]+  Stopped                 tftp 127.0.0.1 4010

[william45093]@hera3 ~/ecen602/assignment3/client> (15:32:22 10/28/19)
:: diff ./binary1.txt ../binary1.txt

[william45093]@hera3 ~/ecen602/assignment3/client> (15:32:24 10/28/19)
::
```

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)



hera.ece.tamu.edu
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
7. hera.ece.ta
8. hera.ece.tan
9. hera.ece.tan
10. hera.ece.ta
/home/grads/w/william45093/ecen602
Name
..
client
.nfs00000000449ac17a00054..
.nfs00000000449ace8200054..
.nfs00000000449ad9a900054.
.nfs00000000449ae07e00054.
.nfs00000000449ae0ec00055..
.nfs00000000449ae20500055.
.nfs00000000449ae27300055.
.nfs00000000449d7fb200054..
.nfs00000000449d94dd00054.
.nfs00000000449d977600054.
.nfs00000000449da1bf00054..
Sessions
Tools
Macros
Sftp
Remote monitoring
Follow terminal folder
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net
9. hera.ece.tamu.edu
Re-attach Fullscreen Stay on top Duplicate Hide toolbar Close

```
ACK received: 10964
The file length is 512
ACK received: 10965
The file length is 512
ACK received: 10966
The file length is 512
ACK received: 10967
The file length is 512
ACK received: 10968
The file length is 512
ACK received: 10969
The file length is 512
ACK received: 10970
The file length is 512
Timeout of receiving ACK: 10971
Timeout of receiving ACK: 10971
Timeout of receiving ACK: 10971
Timeout of receiving ACK: 10971
Timeout of receiving ACK: 10971
Timeout of receiving ACK: 10971
Timeout of receiving ACK: 10971
Timeout of receiving ACK: 10971
Timeout of receiving ACK: 10971
Timeout of receiving ACK: 10971
Timeout happended over 10 times: close the socket.
```

```
[william45093]@hera3 ~/ecen602/assignment3/client> (15:23:54 10/28/19)
:: tftp 127.0.0.1 4010
tftp> verbose
Verbose mode on.
tftp> trace
Packet tracing on.
tftp> get binary1.txt
getting from 127.0.0.1:binary1.txt to binary1.txt [netascii]
^C
tftp>
```

d. Code

server.cpp

```
#include <stdlib.h>
#include <stdio.h>
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <errno.h>
#include <unistd.h>
#include <signal.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <sys/wait.h>
#include <string.h>
#include <sys/time.h>
#include <sys/select.h>

using namespace std;

#define RRQ 1
#define WRQ 2
#define DATA 3
#define ACK 4
#define ERROR 5
#define ERROR_MSG_OPEN_FILE "Fail to open target file"


//  SELF DEFINED DATA STRUCTURE NAME TFTP_MSG, IT CONSISTS OF FOUR ATTRIBUTES
struct TFTP_MSG{
   int opcode;
   char *filename;
   int block_num;
   char *mode;
};


void print_status(std::string s){
   std::cout << s << "..." << std::endl;
```

```
    std::cout << "................." << std::endl;
    std::cout << "................." << std::endl;
}

void address_set(struct sockaddr_in &address, int &PORT){
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(PORT);
}

void handle_sigchld(int sig) {
    int saved_errno = errno;
    while (waitpid((pid_t)(-1), 0, WNOHANG) > 0) {}
    errno = saved_errno;
}

//  CONVERT UNSIGNED SHORT INTEGER TO HOST BYTE ORDER
unsigned short int convert_network_byte(char *buf){
    unsigned short int i;
    memcpy(&i, buf, 2);
    i = ntohs(i);
    return i;
}

//  CONVERT TO NETWORK BYTE ORDER
void convert(char *buf, unsigned short int i){
    i = htons(i);
    memcpy(buf,&i,2);
}

//  READ FROM BUFFER AND CONVERT INTO TFTP_MSG STRUCTURE
struct TFTP_MSG* readFile(char *buffer){
    struct TFTP_MSG* out = (struct TFTP_MSG*)malloc(sizeof(struct TFTP_MSG));
    out->opcode = convert_network_byte(buffer);
    if (out->opcode == ACK){
        out->block_num = convert_network_byte(buffer + 2);
    }
    if (out->opcode == RRQ){
        int i = 0;
        char temp[512];
        while (buffer[i + 2] != '\0'){
            temp[i] = buffer[i + 2];
            i += 1;
        }
        temp[i] = '\0';
        out->filename = (char*)malloc(strlen(temp)*sizeof(char));
        strcpy(out->filename, temp);
        while(buffer[i + 2]=='\0')
            i++;
        int j = 0;
        while (buffer[i + 2] != '\0'){
            temp[j] = buffer[i + 2];
            i += 1;
            j += 1;
        }
        temp[j] = '\0';
        out->mode = (char*)malloc(strlen(temp)*sizeof(char));
        strcpy(out->mode, temp);
    }
    return out;
}

//  WRAP THE MESSAGE FROM BUFFER INTO CHAR* TO SEND
char *wrap_msg(int opcode, int block_number, char *file_buffer, int length){
    char *out;
    if(opcode == DATA){
        out=(char *)malloc((length + 4)*sizeof(char));
        convert(out,opcode);
        convert(out + 2,block_number);
        memcpy(out + 4,file_buffer,length);
    }
    if(opcode == ERROR){
        out=(char *)malloc((length+5)*sizeof(char));
        convert(out,opcode);
        convert(out + 2,block_number);
        memcpy(out + 4,file_buffer,length);
        memset(out + 4 + length,'\0',1);
    }
    return out;
}

int readable_timeout(int fd, int sec){
    fd_set rset;
      struct timeval tv;

    FD_ZERO(&rset);
    FD_SET(fd, &rset);

    tv.tv_sec = sec;
    tv.tv_usec = 0;

    return (select(fd + 1, &rset, NULL, NULL, &tv));
```

```
}

int main(int argc, char **argv){
    if (argc != 3){
        errno = EPERM;
        perror("Illegal Input! Please only enter your IP addr and server port");
        exit(EXIT_FAILURE);
    }
    std::string IP_addr = argv[1];
    std::string port_str = argv[2];

    // fd_list saves all file descriptors
    fd_set fd_list;
    // curr_fd_list reads all current file descriptors
    fd_set curr_fd_list;
    FD_ZERO(&fd_list);
    FD_ZERO(&curr_fd_list);
    int max_fd;
    int socket_fd;  // server's socket
    int new_socketfd;

    char buffer[512];
    char s[INET_ADDRSTRLEN];
    int opt = 1;

    socklen_t client_addr_size;
    struct addrinfo addressinfo;
    struct addrinfo *server_addrinfo;
    struct addrinfo *curr_addrinfo;
    struct sockaddr_storage client_addr;
    struct sockaddr addr;
    struct sockaddr_in *address;
    memset(&addressinfo, 0, sizeof(addressinfo));
    pid_t child;
    int bytes;

    addressinfo.ai_family = AF_INET;
    addressinfo.ai_socktype = SOCK_DGRAM;   // SET THE PROTOCOL TO UDP
    if ((getaddrinfo(argv[1], argv[2], &addressinfo, &server_addrinfo)) != 0) {
        errno = EPERM;
        perror("Fail to set address...");
        exit(EXIT_FAILURE);
    }
    for (curr_addrinfo = server_addrinfo; curr_addrinfo != NULL; curr_addrinfo =
curr_addrinfo->ai_next){
        // CREATE A SOCKET WITH A DESCRIPTER socket_fd WHICH SUPPORT IPv4
        if ((socket_fd = socket(curr_addrinfo->ai_family, curr_addrinfo->ai_socktype,
curr_addrinfo->ai_protocol)) < 0){
            errno = ETIMEDOUT;
            continue;
        }
        print_status("Socket has been created...");
        setsockopt(socket_fd, SOL_SOCKET, SO_REUSEADDR, &opt, sizeof(opt));

        if (::bind(socket_fd, curr_addrinfo->ai_addr, curr_addrinfo->ai_addrlen) < 0){
            errno = EADDRINUSE;
            continue;
        }
        break;
    }

    if (curr_addrinfo == NULL){
        errno = EPERM;
        perror("Fail to create a socket and bind...");
        exit(EXIT_FAILURE);
    }

    print_status("Socket has been binded successfully!");
    print_status("Wait for connection");

    // int bytes;
    addr = *(curr_addrinfo->ai_addr);
    address = (struct sockaddr_in *) &addr;
    address->sin_port = htons(0);

    struct sigaction sa;
    sa.sa_handler = handle_sigchld; // wipe out all dead processes
    sigemptyset(&sa.sa_mask);
    sa.sa_flags = SA_RESTART;
    if (sigaction(SIGCHLD, &sa, NULL) == -1) {
        perror("sigaction");
        exit(EXIT_FAILURE);
    }

    char nextchar = -1;

    while (true){
        client_addr_size = sizeof(client_addr);

        if (recvfrom(socket_fd, buffer, 511, 0, (struct sockaddr *)&client_addr,
&client_addr_size) < 0){
            exit(EXIT_FAILURE);
        }
        struct sockaddr_in* new_client_addr = (struct sockaddr_in*) &client_addr;

        cout << "Received packet from IP: " << inet_ntop(client_addr.ss_family,&(((struct
sockaddr_in*)address)->sin_addr),s, sizeof s) << endl;

        struct TFTP_MSG *to_send;
        to_send = readFile(buffer);
        if (to_send->opcode != RRQ){
            if (to_send->opcode == WRQ)
              print_status("Currently WRQ is not valid");
            else
            print_status("Invalid request");
            continue;
        }

        if (fork() == false){
            close(socket_fd);
            if ((socket_fd = socket(curr_addrinfo->ai_family, curr_addrinfo->ai_socktype,
curr_addrinfo->ai_protocol)) < 0){
              exit(EXIT_FAILURE);
            }
            setsockopt(socket_fd, SOL_SOCKET, SO_REUSEADDR, &opt, sizeof(opt));
            if (::bind(socket_fd, &addr, sizeof(addr)) < 0){
              errno = EADDRINUSE;
              perror("Fail to bind...");
            }
            socklen_t new_len = sizeof(addr);
            if(getsockname(socket_fd, &addr, &new_len) == -1){
              perror("Error");
            }

            FILE *f;
            f = fopen(to_send->filename, "rb");

            if (f == NULL){
              print_status("Fail to open target file");
              char error_msg[512];
              strcpy(error_msg, "Fail to open target file");
              int error_length = strlen(error_msg);
              char *error_buffer = wrap_msg(ERROR, 1, error_msg, error_length);
              if ((sendto(socket_fd, error_buffer, error_length + 5, 0, (struct
sockaddr*)&client_addr, client_addr_size)) < 0){
          perror("Fail to send error message...");
          continue;
              }
              exit(EXIT_FAILURE);
            }

            //  SEARCH FOR THE FILE SIZE AND RESET THE POINTER FILE *f TO THE
BEGINNING OF FILE
            fseek(f, 0, SEEK_END);
            int file_size = ftell(f);
            fseek(f, 0, SEEK_SET);
            int num_packet = file_size / 512 + 1;

            cout << "The file size is: " << file_size << endl;
            //cout << "Number of packets needed: " << num_packet << endl;

            char file_buffer[512];
            int block_num = 0;
            int block_cnt = 0;
            int last_ACK = 0;
            int file_length = 0;
            int special_letter = 0;
            char *ptr = file_buffer;
            int totalcnt = 0;
            short int c;
            int overhead = 0;
                     int timeout, sel_timeout;
                     int opc = -1;

            FD_SET(socket_fd, &fd_list);
            max_fd = socket_fd;

            while (last_ACK < num_packet){
              //  RUN THE ROLL OVER TO AVOID DEFAULT CLIENT TERMINATE
              if (block_num == 65536){
          block_num = 0;
              }

              if (last_ACK == block_num){
          for (int cnt = 0; cnt < 512; cnt++){
            c = getc(f);
            //  KEEP READING EVEN COME ACROSS EOF, THE FOLLOWING
LOGIC WILL RESOVE THE LAST PACKET ISSUE
            if (c == EOF){
              if (ferror(f))
                perror("Error");
              break;
            }
            *ptr++ = c;
            file_length += 1;
          }
          cout << "The file length is " << file_length << endl;
          block_num += 1;
          block_cnt += 1;
          totalcnt += 1;
          ptr = file_buffer;
              }
              //  THIS IF CONDITION PLAYS THE ROLE TO DETECT WHETHER THIS IS
THE LAST PACKET
              if (block_cnt == num_packet)
          file_length = file_size % 512;

              char* data_packet = wrap_msg(DATA, block_num, file_buffer, file_length);
                       char* recv_packet = (char*) malloc(4 * sizeof(char));
              //cout << "Block number is " << block_num << endl;
                       for (timeout = 0; timeout < 10; timeout++) { // if timeout over 10
times, the trasmission will be stopped and socket  will be closed
                  if ((sendto(socket_fd, data_packet, file_length + 4, 0,
(struct sockaddr *)&client_addr, client_addr_size)) < 0){
                      perror("Fail to send...");
                      exit(EXIT_FAILURE);
                  }
                  sel_timeout = readable_timeout(socket_fd, 1); // set time
out to 1 second
                  if (sel_timeout < 0) {
                      perror("Select error: Receiving ACK...");
                      exit(EXIT_FAILURE);
                  }
                  else if (sel_timeout == 0) {
                      cout << "Timeout of receiving ACK: " <<
last_ACK << endl;
                  }
                  else {
                      if (recvfrom(socket_fd, recv_packet,
sizeof(recv_packet), 0, (struct sockaddr *)&client_addr, &client_addr_size) < 0){
                          perror("Fail to receive ACK...");
                          exit(EXIT_FAILURE);
                      }
                      opc = convert_network_byte(recv_packet);
                      if (opc == 4) { // Check if the received message is
ACK
                          cout << "ACK received: " << last_ACK <<
endl;
                          break;
                      }
                  }
              }
              if (timeout == 10) { // timeout equal to 10 times, close socket and
child process
                      cout << "Timeout happended over 10 times: close
the socket." << endl;
                      fclose(f);
                      close(socket_fd);
                      _exit(2); // close child process
              }
              file_length = 0;
              //free(data_packet);
              //memset(file_buffer, '\0', 512);
              /*ptr = file_buffer;
              curr_fd_list = fd_list;
              if (select(max_fd + 1, &curr_fd_list, NULL, NULL, NULL) < 0){
                  perror("Fail to get update");
                  exit(EXIT_FAILURE);
              }
              if(FD_ISSET(socket_fd, &curr_fd_list)){
                  struct sockaddr_storage curr_addr;
                  socklen_t curr_length = sizeof(curr_addr);

                  if((bytes = recvfrom(socket_fd, buffer, 4, 0, (struct
sockaddr *)&curr_addr, &curr_length)) == -1){
                      perror("Fail to receive from socket");
                      exit(EXIT_FAILURE);
                  }
              }
              */
     struct sockaddr_in* curr_client_addr = (struct sockaddr_in*)& client_addr;
     if(curr_client_addr->sin_addr.s_addr == new_client_addr->sin_addr.s_addr){
       to_send = readFile(recv_packet);
       last_ACK = to_send->block_num;
     }
              //cout << "block_cnt = " << block_cnt << ", num_packet = " <<
num_packet <<endl;
     if (block_cnt > num_packet)
       break;
              free(data_packet);
              free(recv_packet);
    }
          cout << "RRQ service finished, closing the socket" << endl;
          close(socket_fd);
    fclose(f);
          _exit(2); // close child process
```

```
        }
    }
    close(socket_fd);
}
```