Lab 2 Documentation

Christopher Jensen (chjjense@ucsc.edu)

How To Use

1. Cd into folder lab02
2. There are 4 folders and 2 files. The two files are Makefile and README.txt. The four folders are doc, bin, build, and src. Doc is where the documentation is located, build is for the .o files, bin is for the binaries, and src is where the source code for this lab is located. In src there is myclient.cpp and myserver.cpp.
3. Run a quick "make" in the top directory where the Makefile is
4. Cd into bin
5. Firstly you need to set up the server. You can do this by running the command:
    a. ./myserver *port_number*
    b. Where port number is a number between 1024 and 65536
6. Once the server is up and running, create a new instance of the terminal and cd back into the bin folder
7. Now you can send requests to the server. You can do this by running the command;
    a. ./myclient *server_ip server_port mtu in_file_path out_file_path*
    b. Where server_ip is the IP address where the server is running, in this case it would be the IP address of the Unix timeshare (192.168.122.1).
    c. Server_port is the port you entered in the previous section when you set up the server.
    d. Mtu is the maximum transmission unit, and for this client protocol, the smallest MTU you can enter is 61. These bytes are what makes up the header, and the rest is made up of payload bytes.
    e. In_file_path is where the file that you want to copy is located
    f. Out_file_path is where you want to put the new copied file. If this directory does not exist it will create it for you.

Internal Design

- The design of this client server echo program is built around the example given in the textbook.
- **Myserver.cpp**: The server did not really change much from the example in the book, besides capturing user input from stdin and error checking all system calls. Another change was the size of the buffer used for recvfrom(), I changed this to the maximum MTU for this assignment, which is 32000. The server waits to receive packets from the client in an infinite loop, and sends the packet back to the client from which it received the packet.
- **Myclient.cpp**: The client had a lot of changes from the example in the book. For simplicity, I used a stop and wait protocol for sending and receiving packets. First, I open the file that the user gives me to copy over. I read from this file in do_client_processing() using a while loop and read() to get the bytes. To split the file into MTU sized packets, I

read() overhead-MTU bytes from the file, and append this information to the end of my headers. The headers include the packet sequence number and the length of the packet. Then once I have the packet ready to send, I pass it to the function send_packet_to_server() which sends the packet using sendto() and gets the echoed response using recvfrom(). Then once the packet is received, it is parsed for the sequence number and compared to a global variable packet_counter which has the total packets sent. If the two don't match, packet loss is detected. Once we're sure the packet is correct, we can parse the echoed packet for the payload and start constructing the original file. We can be sure that the packets are not out of order because we're only sending and parsing one at a time. This process continues until the original file is constructed in the directory given by the user.

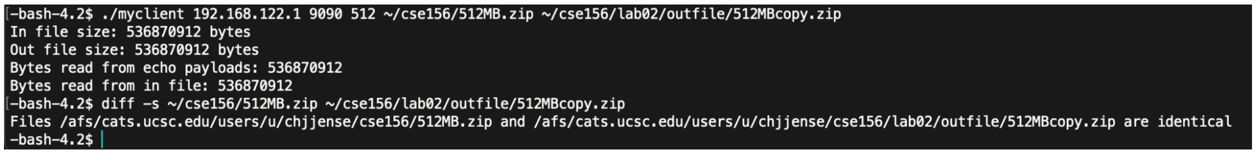
Shortcomings

- I think my client/server is pretty solid, the only thing I wish I had implemented is sending packets using a window. Stop and wait was helpful for this assignment because it makes detecting packet loss a lot easier, but probably hurts the throughput a little.

Testing

- Cout outputs like "In file size: and out file size:" removed in the final src file. These were put here to demonstrate that all bytes from the in file were transferred to the out file.
- Cerr outputs however still appear when program encounters an error
- For the big file tests, the transfer takes a few seconds.

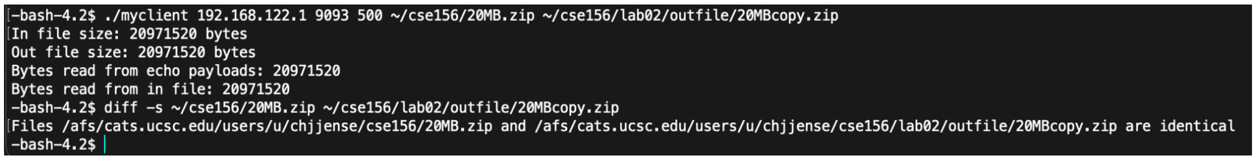
512MB File Test With Diff: (Would try with a 1GB file but timeshare has limited space)



```
-bash-4.2$ ./myclient 192.168.122.1 9090 512 ~/cse156/512MB.zip ~/cse156/lab02/outfile/512MBcopy.zip
In file size: 536870912 bytes
Out file size: 536870912 bytes
Bytes read from echo payloads: 536870912
Bytes read from in file: 536870912
-bash-4.2$ diff -s ~/cse156/512MB.zip ~/cse156/lab02/outfile/512MBcopy.zip
Files /afs/cats.ucsc.edu/users/u/chjjense/cse156/512MB.zip and /afs/cats.ucsc.edu/users/u/chjjense/cse156/lab02/outfile/512MBcopy.zip are identical
-bash-4.2$
```

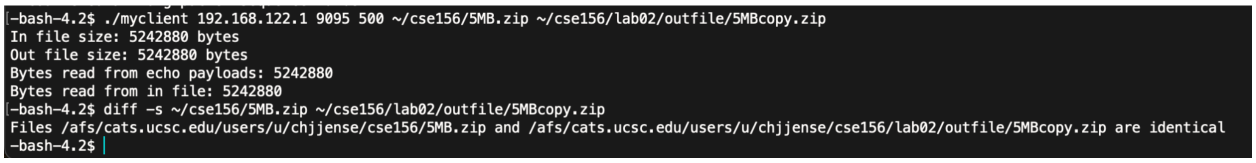
- This test takes a few seconds so be patient when testing

20MB File Test With Diff:



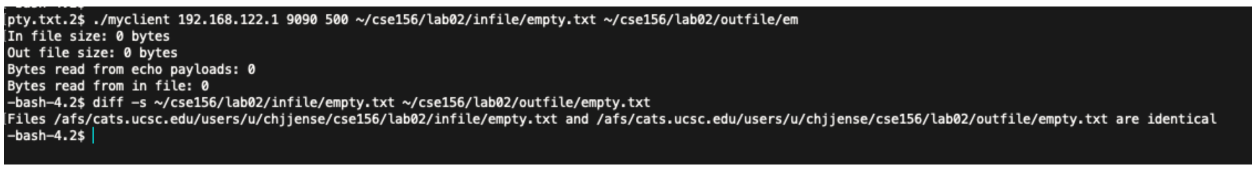
```
-bash-4.2$ ./myclient 192.168.122.1 9093 500 ~/cse156/20MB.zip ~/cse156/lab02/outfile/20MBcopy.zip
In file size: 20971520 bytes
Out file size: 20971520 bytes
Bytes read from echo payloads: 20971520
Bytes read from in file: 20971520
-bash-4.2$ diff -s ~/cse156/20MB.zip ~/cse156/lab02/outfile/20MBcopy.zip
Files /afs/cats.ucsc.edu/users/u/chjjense/cse156/20MB.zip and /afs/cats.ucsc.edu/users/u/chjjense/cse156/lab02/outfile/20MBcopy.zip are identical
-bash-4.2$
```

5MB File Test With Diff:



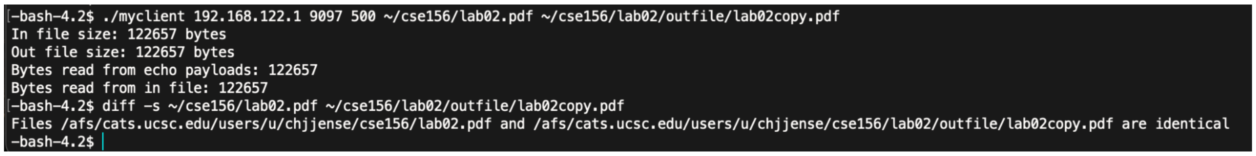
```
-bash-4.2$ ./myclient 192.168.122.1 9095 500 ~/cse156/5MB.zip ~/cse156/lab02/outfile/5MBcopy.zip
In file size: 5242880 bytes
Out file size: 5242880 bytes
Bytes read from echo payloads: 5242880
Bytes read from in file: 5242880
-bash-4.2$ diff -s ~/cse156/5MB.zip ~/cse156/lab02/outfile/5MBcopy.zip
Files /afs/cats.ucsc.edu/users/u/chjjense/cse156/5MB.zip and /afs/cats.ucsc.edu/users/u/chjjense/cse156/lab02/outfile/5MBcopy.zip are identical
-bash-4.2$
```

Empty File Test (0 Bytes):



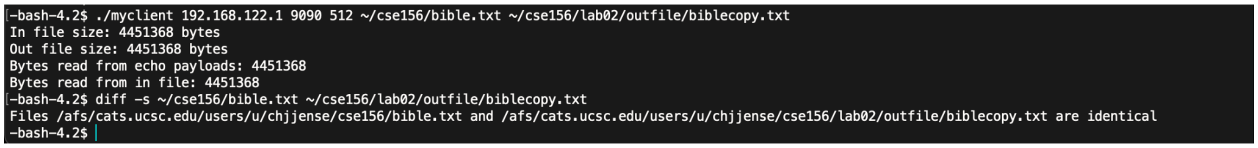
```
pty.txt.2$ ./myclient 192.168.122.1 9090 500 ~/cse156/lab02/infile/empty.txt ~/cse156/lab02/outfile/em
In file size: 0 bytes
Out file size: 0 bytes
Bytes read from echo payloads: 0
Bytes read from in file: 0
-bash-4.2$ diff -s ~/cse156/lab02/infile/empty.txt ~/cse156/lab02/outfile/empty.txt
Files /afs/cats.ucsc.edu/users/u/chjjense/cse156/lab02/infile/empty.txt and /afs/cats.ucsc.edu/users/u/chjjense/cse156/lab02/outfile/empty.txt are identical
-bash-4.2$
```

PDF File Test With Diff:



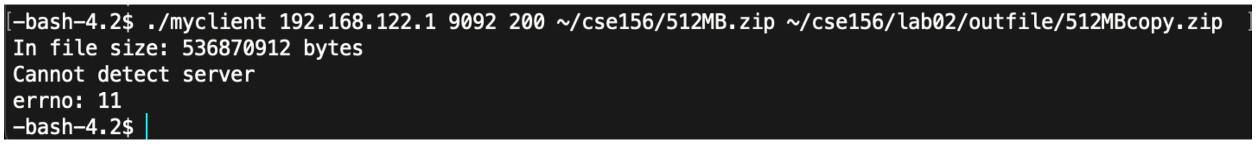
```
-bash-4.2$ ./myclient 192.168.122.1 9097 500 ~/cse156/lab02.pdf ~/cse156/lab02/outfile/lab02copy.pdf
In file size: 122657 bytes
Out file size: 122657 bytes
Bytes read from echo payloads: 122657
Bytes read from in file: 122657
-bash-4.2$ diff -s ~/cse156/lab02.pdf ~/cse156/lab02/outfile/lab02copy.pdf
Files /afs/cats.ucsc.edu/users/u/chjjense/cse156/lab02.pdf and /afs/cats.ucsc.edu/users/u/chjjense/cse156/lab02/outfile/lab02copy.pdf are identical
-bash-4.2$
```

Bible.txt Test (large text file) With Diff:



```
-bash-4.2$ ./myclient 192.168.122.1 9090 512 ~/cse156/bible.txt ~/cse156/lab02/outfile/biblecopy.txt
In file size: 4451368 bytes
Out file size: 4451368 bytes
Bytes read from echo payloads: 4451368
Bytes read from in file: 4451368
-bash-4.2$ diff -s ~/cse156/bible.txt ~/cse156/lab02/outfile/biblecopy.txt
Files /afs/cats.ucsc.edu/users/u/chjjense/cse156/bible.txt and /afs/cats.ucsc.edu/users/u/chjjense/cse156/lab02/outfile/biblecopy.txt are identical
-bash-4.2$
```

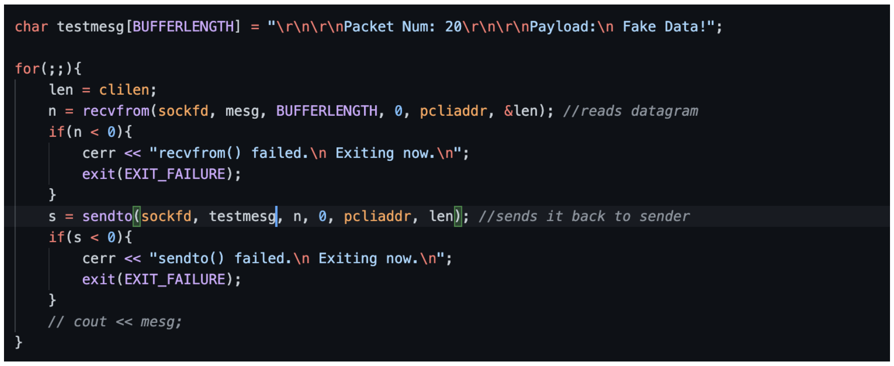
Cannot Detect Server (60s timeout):



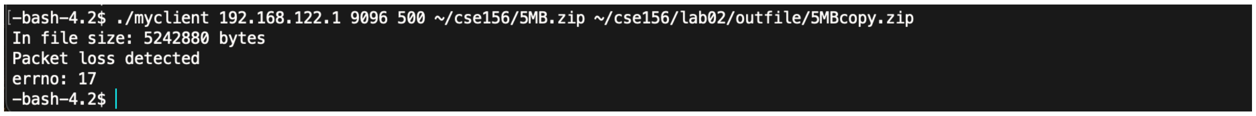
```
-bash-4.2$ ./myclient 192.168.122.1 9092 200 ~/cse156/512MB.zip ~/cse156/lab02/outfile/512MBcopy.zip
In file size: 536870912 bytes
Cannot detect server
errno: 11
-bash-4.2$
```

- To test this one, I started to copy a large file, then quit the server while the client was still running.

Packet Loss:



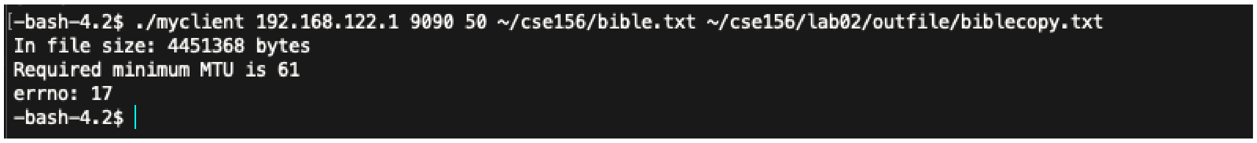
```
char testmesg[BUFFERLENGTH] = "\r\n\r\nPacket Num: 20\r\n\r\nPayload:\n Fake Data!";

for(;;){
    len = clilen;
    n = recvfrom(sockfd, mesg, BUFFERLENGTH, 0, pcliaddr, &len); //reads datagram
    if(n < 0){
        cerr << "recvfrom() failed.\n Exiting now.\n";
        exit(EXIT_FAILURE);
    }
    s = sendto(sockfd, testmesg, n, 0, pcliaddr, len); //sends it back to sender
    if(s < 0){
        cerr << "sendto() failed.\n Exiting now.\n";
        exit(EXIT_FAILURE);
    }
    // cout << mesg;
}
```



```
-bash-4.2$ ./myclient 192.168.122.1 9096 500 ~/cse156/5MB.zip ~/cse156/lab02/outfile/5MBcopy.zip
In file size: 5242880 bytes
Packet loss detected
errno: 17
-bash-4.2$
```

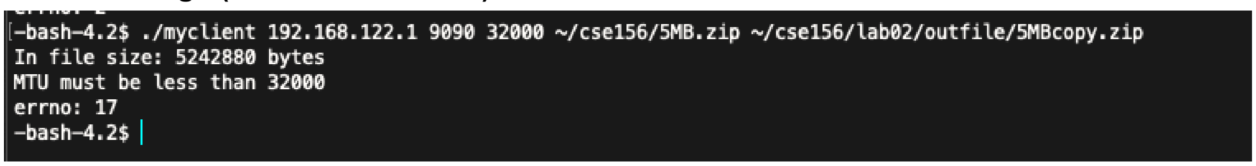
- To test this, I put a dummy packet with the wrong sequence number in the sendto() function on the server side.

MTU Too Small (min MTU is 61):



```
[-bash-4.2$ ./myclient 192.168.122.1 9090 50 ~/cse156/bible.txt ~/cse156/lab02/outfile/biblecopy.txt
 In file size: 4451368 bytes
 Required minimum MTU is 61
 errno: 17
 -bash-4.2$
```

MTU Too Large (max MTU is 31999):



```
[-bash-4.2$ ./myclient 192.168.122.1 9090 32000 ~/cse156/5MB.zip ~/cse156/lab02/outfile/5MBcopy.zip
 In file size: 5242880 bytes
 MTU must be less than 32000
 errno: 17
 -bash-4.2$
```