Lab 3 Documentation

Christopher Jensen (chjjense@ucsc.edu)

How To Use

1. Cd into folder lab03
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 droppc*
    b. Where port number is a number between 1024 and 65536
    c. And droppc is a percentage value between 0-100. This is the percentage of packets/ACKSs that will be dropped on the server side.
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 winsz 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. Winsz is the size of the window used for the “go back n” protocol implemented in this client.
    f. In_file_path is where the file that you want to send is located
    g. Out_file_path is where you want to put the new copied file on the server's system. If this directory does not exist it will create it for you.

Internal Design

- **Myserver.cpp**: This server can support multiple clients by storing them each in a vector of struct client_info, which includes information that can identify each client every time a packet is sent. The server listens for 3 types of packets. Firstly a header packet, which contains information about the client and the file it is preparing to send. It also listens for an ender packet, which tells the server that we’re done sending it information and it’s time to close the file and remove the client from the client vector. Finally, it listens for data packets, which is where the bulk of the work is done. If the server receives a data packet, it first determines which client sent the packet, then processes the payload and

writes the data into the correct file. In accordance with the go back n protocol, the server keeps track of the sequence numbers and drops any out of order packets as they come. The server is also given an input droppc, which drops the given percentage of packets sent to the server, used for debugging. I used a random number generator to do this, every time through the loop generating a new random number between 0 and 1. Then, this number is compared to the percentage value droppc/100, and if the random number is less than or equal to that value, the packet is dropped.

- **Myclient.cpp**: This client is designed using a go back n protocol. I kept track of the resend window using a list data structure. Firstly, winsz packets are sent to the client, the pointer nextsn is moved forward and the sent packets are pushed to the resend window. After sending the packets, the client calls recvfrom() with a timeout, waiting for either an ack from the server or a timeout. If we receive an ack, the pointer basesn is moved forward and the acked packets are popped off of the list. If we receive a timeout, the client resends every packet in the window. If the client times out on the same packet too many times, it exits with “reached max re-transmission limit”. This process is repeated until the file descriptor is empty and the client has received the final ack. The client sends an ender packet to the server, telling it to close the file. The client then closes with success.

Shortcomings

- At first, my client was sending packets very slowly to the server due to the timeout value being too large. After reducing the timeout value to a reasonable amount, the client has sped up significantly. The only shortcoming I’d say is that sometimes if the server sets the dropped packet percentage too high, like 20% and above, it's possible that the client hits the max retransmission limit. It almost never hits the limit from 0-10%.

Testing

512MB File Test With 3% Packet Loss (window size 1000, mtu 31000):

- ./myclient 192.168.122.1 9037 31000 1000 ~/cse156/512MB.zip ~/cse156/lab03/outfile/512MBcopy.zip
- ./myserver 9037 3
- This test takes a few minutes to complete, so be patient

```
Bytes read from in file: 536870912
In file size: 536870912
-bash-4.2$ diff -s ~/cse156/512MB.zip ~/cse156/lab03/outfile/512MBcopy.zip
Files /afs/cats.ucsc.edu/users/u/chjjense/cse156/512MB.zip and /afs/cats.ucsc.edu/users/u/chjjense/cse156/lab03/outfile/512MBcopy.zip are identical
-bash-4.2$
```

20MB File Test With 10% Packet Loss (window size 100, mtu 6000):

- ./myclient 192.168.122.1 9039 6000 100 ~/cse156/20MB.zip ~/cse156/lab03/outfile/20MBcopy.zip
- ./myserver 9039 10

```
Bytes read from in file: 20971520
In file size: 20971520
-bash-4.2$ diff -s ~/cse156/20MB.zip ~/cse156/lab03/outfile/20MBcopy.zip
Files /afs/cats.ucsc.edu/users/u/chjjense/cse156/20MB.zip and /afs/cats.ucsc.edu/users/u/chjjense/cse156/lab03/outfile/20MBcopy.zip are identical
-bash-4.2$
```

5MB File Test With 10% Packet Loss (window size 50, mtu 3000):

- ./myclient 192.168.122.1 9038 3000 50 ~/cse156/5MB.zip ~/cse156/lab03/outfile/5MBcopy.zip
- ./myserver 9038 10

```
Bytes read from in file: 5242880
In file size: 5242880
-bash-4.2$ diff -s ~/cse156/5MB.zip ~/cse156/lab03/outfile/5MBcopy.zip
Files /afs/cats.ucsc.edu/users/u/chjjense/cse156/5MB.zip and /afs/cats.ucsc.edu/users/u/chjjense/cse156/lab03/outfile/5MBcopy.zip are identical
-bash-4.2$
```

Large Text File Test With 5% Packet Loss (window size 100, mtu 2000):

- ./myclient 192.168.122.1 9041 2000 100 ~/cse156/bible.txt ~/cse156/lab03/outfile/biblecopy.txt
- ./myserver 9041 5

```
Bytes read from in file: 4451368
In file size: 4451368
-bash-4.2$ diff -s ~/cse156/bible.txt ~/cse156/lab03/outfile/biblecopy.txt
Files /afs/cats.ucsc.edu/users/u/chjjense/cse156/bible.txt and /afs/cats.ucsc.edu/users/u/chjjense/cse156/lab03/outfile/biblecopy.txt are identical
-bash-4.2$
```

Too Many Losses Test:

- Set server to drop 90% of packets: ./myserver 9042 90
- ./myclient 192.168.122.1 9042 2000 100 ~/cse156/bible.txt ~/cse156/lab03/outfile/biblecopy.txt

```
2023-02-18T00:28:06Z, DATA, 111, 13, 113
2023-02-18T00:28:06Z, DATA, 112, 13, 113
Packet loss detected
Reached max re-transmission limit
errno: 11
-bash-4.2$
```

Server Not Available:

- Send client request to a non-running server port and wait for client to time out with error

```
-bash-4.2$
-bash-4.2$ ./myclient 192.168.122.1 9050 2000 100 ~/cse156/bible.txt ~/cse156/lab03/outfile/biblecopy.txt
sending header packet...

Cannot detect server
errno: 11
-bash-4.2$
```

Server Drops Packets According to Droppc:

- Server set to drop 10% of packets: ./myserver 9043 10

Server output:

```
In file size: 4451368 bytes
Out file size: 4451368 bytes
Client terminated, erasing client from vector
Percent packets dropped: 9.53311%
```