John Luscombe
Grace Rose

READEME

RFT Project – COS 331

1. Description:

This project is a demonstration of the Transport Layer's UDP protocol. Using C, both a client and server can send and receive packets over a uniquely built UDP protocol that provides reliable transfer and correct packet ordering. The client can request a file from the server and stores it in a local file.

2. Build:

Using CLion:

1. Open the directory and select the file
2. Go to: File > Reload CMake Project

- if not an option -

1. Open the directory and select CMakeLists.txt
2. Go to: File > Reload CMake Project
3. Select a new file
4. Go to: File > Reload CMake Project (again)

Using Linux:

1. Run commands:
   gcc -o [program-name] [program.c] … {any other dependent files}

3. Specifications:

- The goal of this program is for the client to request a file from the server and then for the server to send the correct file back to the client. The client begins a handshake with the server by sending a packet with a sequence number of 0, end of file (eof) flag set to 0, and a body containing the remote path of the file is requesting. Once the server receives the handshake, it immediately begins to send data back to the client, increasing the sequence number by 1. Once the client receives a packet, it checks if the sequence number is correct: the packet sequence number must be one more than the client's record of the previous sequence number. Then the client updates its sequence number to what was most recently received from the server and sends an ACK. The client also writes the data received into a local file. Once the server receives the ACK from the client, it checks that the sequence number is the same as the packet it previously sent. If that is correct, the server increments its sequence number and sends the next packet of data. Once the server reaches the end of the file, it sets the eof flag to 1. If the client receives this last packet, it reads the eof flag, sends one final ACK, and disconnects from the server.

- If there is a dropped packet (ACK) on the way to the server, the server will not send a new packet unless it receives an ACK from the client. The client will be waiting for the next packet in the sequence but receives none. There is a timeout for each ACK that is sent, and if the timeout is reached eight times, the client assumes it cannot connect to the server and disconnects.

- If there is a dropped packet (data) on the way to the client, the client will resend the same ACK eight times before it decides it cannot reach the server and disconnects. The server will not send any more data until it receives the correct ACK.

- If data is received out of order, the client will not accept the packet and send the ACK of the packet it wants in the correct order. Only when the data is received at the client in the correct order will its sequence number increase.

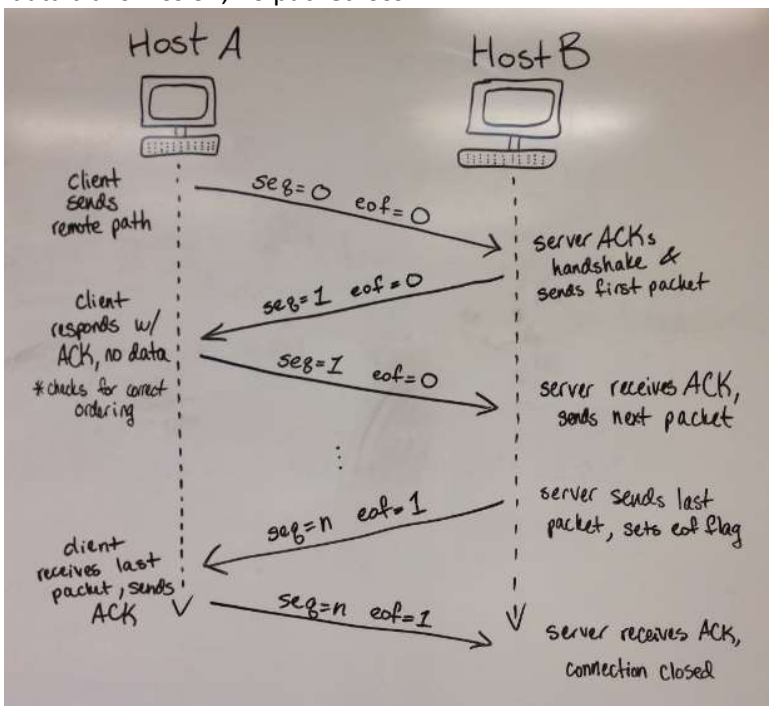
- To use this program, run the server's executable first. The 1 parameter it requires is the listening port number. Then run the client's executable. The 4 parameters it requires is the IP address of the server, the listening port number, the remote path (the file it is requesting), and the local path (file the data will be stored into).

  EXAMPLE:

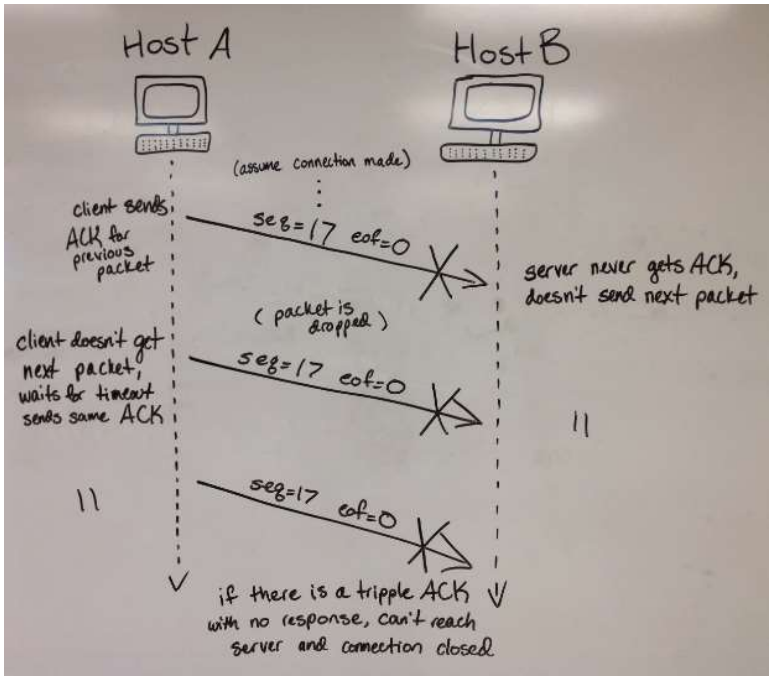
Server Terminal	Client Terminal
> ./rft_server 8080	>
…	…
>	> ./rft_client 127.0.0.1 8080 ../test.txt ../output.txt

4. Packet sequence diagrams

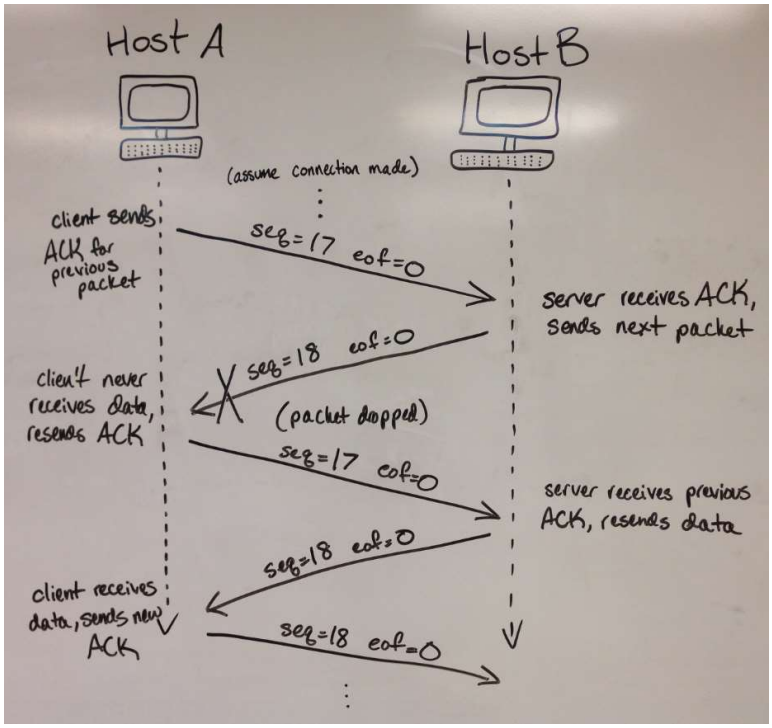
1. Optimal data transmission, no packet loss



2. Packet loss from Client to Server



3. Packet loss from Server to Host



5. Test:

BASIC FILE TRANSFER ON LOCALHOST:

1. Create a file test.txt with sample data, located in same directory as C files
   > touch test.txt
   > wget

2. Create a file output.txt where data will be stored, located in same directory as C files.
   > touch output.txt

3. Run the server executable with listening port number 8080
   > ./rft_server 8080

4. Run the client executable with IP address 127.0.0.1, listening port number 8080, remote path ../test.txt, and local path ../output.txt
   > ./rft_client 127.0.0.1 8080 .../test.txt ../output.txt

5. Test that the two files are the same
   > diff output.txt test.txt

   If there is nothing sent to standard output, then the files are the same!

NETWORK WITH 40% DROPPED PACKETS ON VM:

1. Setup two individual virtual machines and a network that connects them

2. Using the **tc qdisc** command, set the packet loss to 40% and reordering on the Host1
   > tc qdisc add dev 192.168.1.102 ens33 root netem loss 40%
   > tc qdisc add dev 192.168.1.102 ens33 root netem delay 100ms reorder 25% 50%

3. On Host1, run the server (same as above)

4. On Host2, run the client (same as above)

5. The connection works for a time, but there can be up to 8 ACKS or timeouts, or else client disconnects from server. But the timeout dynamically responds to congestion and increases if need be. Therefore, reliable transfer is still guaranteed at 40% packet loss.