

CSCI4211: Introduction to Computer Networks

Homework Assignment II

Due 11:55 PM October 21st, 2015

Help-hot-line: csci4211-help@cs.umn.edu

Name: Mohamed Yunis Student ID: 4597863

Important Notes:

Please submit your solutions as a single archive file (.zip or .tar or .tar.gz) on moodle. You may download the MS Word version of this assignment from moodle and edit it directly. Only online submissions are accepted for this assignment - do not attempt to submit hard copies. All textbook references pertain to the 6th edition.

You may discuss ideas and ask for clarifications freely with others on or off the class forum, and with Professor He or with the TAs. You must not provide or accept other assistance on the assignments. Feel free to post any queries you might have on the [moodle discussion forum](#) for this assignment.

Please do not write anything other than name and student ID on this cover page

For TA only:

Problem	Points	Score
1	10	
2	10	
3	10	
4	10	
5	10	
6	50	
Total	100	

1. Definitions (10 pt. 1 pt. each)

Please read Chapter 3 and define following terminologies briefly.

- Logical Communication
 - ➔ Communication between two hosts that happens as if the two hosts were directly connected even though the two hosts are not physically connected.
- Multiplexing and Demultiplexing
 - ➔ Multiplexing: combining multiple streams of information for transmission over a shared medium
 - ➔ Demultiplexing: Splitting a combined stream from arriving from a shared medium into the original information stream .
- Little Endian
 - > Storing of least significant byte in the smallest address
- Handshaking
 - > The process by which two devices initiate communication
- Flow Control
 - > Mechanism that ensures the rate at which a sender is sending is proportionate to the receiving capabilities of the receiving machine.
- Congestion Control
 - > mechanism to control how much packets (data) a certain node carries during communication between nodes.

- Go-Back-N algorithm
-> Mechanism in which a sender is allowed to transmit multiple packets (when available) without waiting for acknowledgement but is constrained to have no more than some maximum allowable number (N) of unacknowledged packets in the pipeline.
- Selective Repeat algorithm
-> mechanism that avoids unnecessary retransmission by having the sender retransmit only those packets that it suspects were received in error at the receiver.
- Fairness
➔ A term used when the average transmission rate of each connection is approximately R/K . That is when each connection gets an equal share of the link bandwidth.
- Slow Start
-> When the value of the cwnd in a TCP connection starts at 1 MSS and increases by 1 MSS everytime a transmitted segment is first acknowledged.

2. TCP Reliable Data Transfer (10 pts)

Consider two nodes which are connected by an 8Mbps link (assume 1Mbps = 1000Kbps) and RTT is 0.05 sec. Assume the size of each packet is 8K bits.

Answer the following questions for ARQ schemes:

- (a) Assume that the link is error-free: what is the maximum possible rate of transmission for Stop-and-wait, GBN, and SR respectively? Why? **(6 pts)**

$$\text{Stop-and-wait: } d_{\text{trans}} = L/R = 8000 \text{ bits} / (8 * 10^6) = 0.001 \text{ secs}$$

$$\text{Packet transmission time} = 8 \text{ kbps} / (8000 \text{ kbps} * 8 \text{ kbps} * 8 \text{ kbps}) = 0.000015625$$

$$\text{Maximum possible transmission rate} = 8000 \text{ mpbs} / (0.000015625 + 0.05) = 159.95 \text{ kb/s}$$

For both GBN and SR: if window size is large enough for transmitting without waiting for acknowledgement from receiver then maximum possible transmission rate equals the total bandwidth of the link which in this case is 8Mbps.

- (b) For GBN, in order to allow sender to continuously send packets without any waiting, what is the minimum window size in terms of the number of packets?

(1 pt)

$$(8000 / (8 * 8000 * 8000) + 0.05) / (8000 / (8 * 8000 * 8000)) = 3201$$

- (c) Suppose that we transmit 20 packets with sequence number from 1 to 20. The packet with sequence number 16 is lost and all other packets are received correctly. Assuming there is

no ACK lost, for stop-and-wait, GBN, and SR, which packets have been retransmitted? (3 pt)

Stop and wait: packet 16 only

GBN: 16 to 20

SR: 16 only

3. TCP Sequence Numbers (10 pts)

Host A and B are communicating over a TCP connection, and Host B has already received from A all bytes up through byte 168. Suppose that Host A then sends two segments to Host B back-to-back. The first and second segments contain 20 and 40 bytes of data, respectively. In the first segment, the sequence number is 169, source port number is 303, and the destination port number is 80. Host B sends an acknowledgement whenever it receives a segment from Host A.

- a) In the second segment sent from Host A to B, what is the sequence number, source port number, and destination port number? (3pts)

Sequence number = $169 + 20 = 189$

Source port number = 303

Destination port = 80

- b) If the first segment arrives before the second segment, in the acknowledgement of the first arriving segment, what is the acknowledgement number, the source port number, and the destination port number? (3pts)

Acknowledgement number = 189

Source port number = 80

Destination port number = 303

- c) If the second segment arrives before the first segment (out of order arrival), in the acknowledgement of the first arriving segment, what is the acknowledgement number? (1pt)

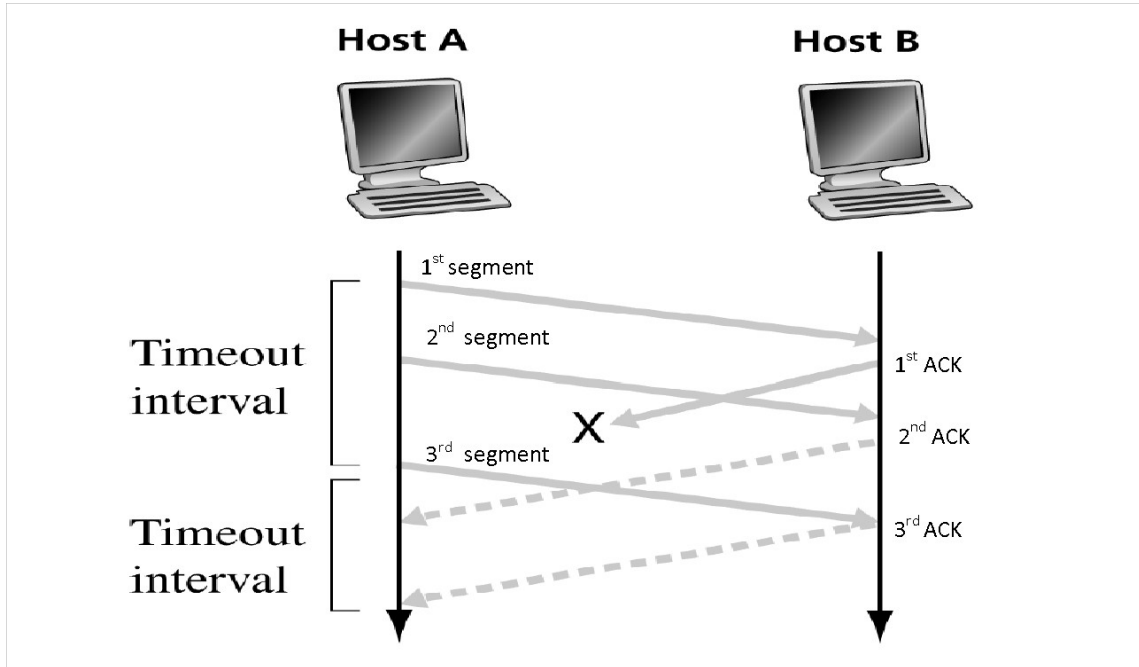
Acknowledgement number is 169

- d) Suppose the two segments sent by A arrive in order at B. The first acknowledgement is lost and the second acknowledgement arrives after the first timeout interval, as shown in the figure below. Please provide the sequence number for the third (retransmitted) data segment (1 pt) and provide the acknowledgement number for the 2nd and 3rd acknowledgement. (2pts)

Sequence number 1st = 169

Acknowledgement 2nd = 189

Acknowledgement 3rd = 189

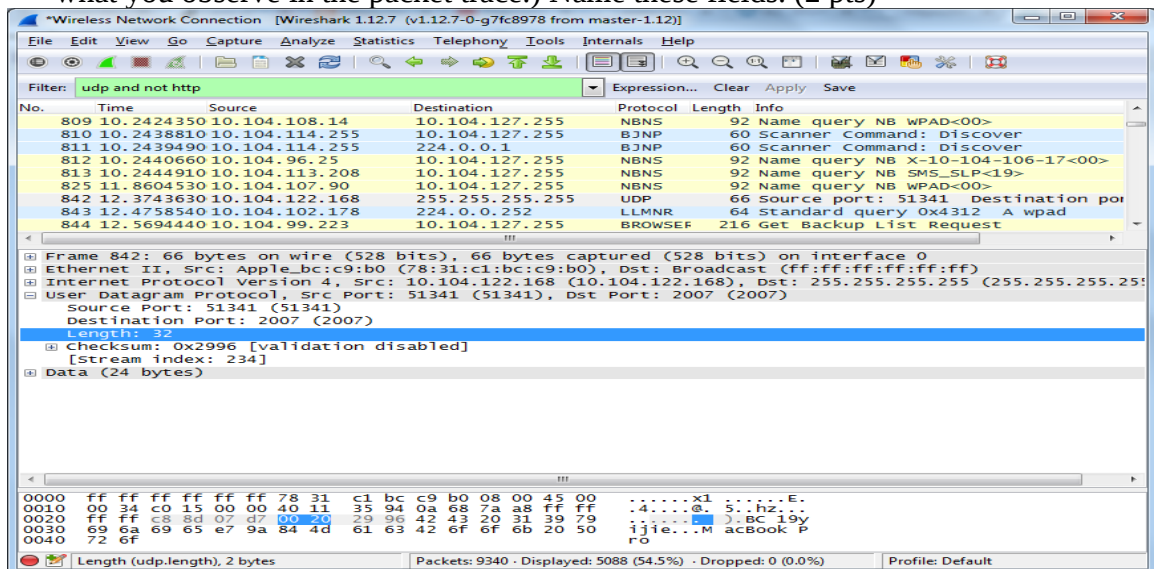


4. Hands-on Practice I: UDP (10 pts)

In this practice, we'll take a quick look at the UDP transport protocol. As we saw in Chapter 3, UDP is a connectionless non-thrills protocol. Start capturing packets in Wireshark and then do something that will cause your host to send and receive several UDP packets. For example, use telnet [domain name].

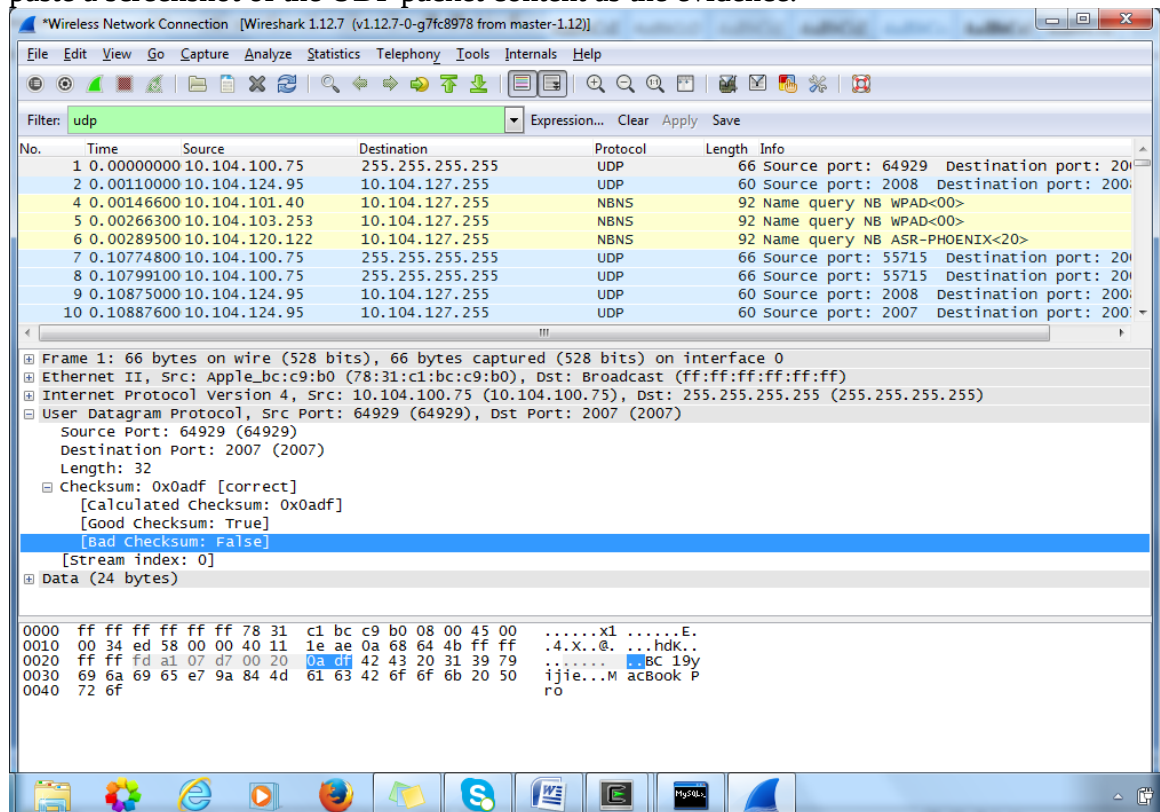
Please answer the following question

- (a) Select one packet. From this packet, determine how many fields there are in the UDP header. (Do not look in the textbook! Answer these questions directly from what you observe in the packet trace.) Name these fields. (2 pts)



There are four fields in UDP header: source port, destination port, length and checksum.

- (b) What is the maximum number of bytes that can be included in a UDP payload?
(1pt)
 $2^{16} - 1$ less the header bytes: $= (2^{16} - 1) - 8 = 65527$ bytes
- (c) What is the protocol number for UDP? Give your answer in both hexadecimal and decimal notation. (To answer this question, you'll need to look into the IP header.)
(2pts)
 IP protocol number for UDP in hex = 0x11 convert to decimal $= (1 * 16^0) + (1 * 16^1) = 17$.
- (d) Search "UDP" in Google and determine the fields over which the UDP checksum is calculated. Capture a VERY SMALL UDP packet. Manually verify the checksum in this packet. Show all work and explain all steps. (5 pts) You need to paste a screenshot of the UDP packet content as the evidence.



Field	Hex value
IP header: Source IP	0a68644b
IP header: Destination	ffff
IP header: protocol number	0011
16 bit UDP length	0x20
UDP header: source port	fda1
UDP header: dest port	07d7
UDP header: length	0020

UDP data	4243
	2031
	3979
	696a
	6965
	e79a
	844d
	6163
	426f
	6f6b
	2050
	726f
Calculated checksum	0x0adf

The UDP checksum is calculated as the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header, and the data. This is padded as needed with zero bytes at the end to make a multiple of two bytes. If the checksum is computed to be 0, it must be set to 0xFFFF

5. Hands-on Practice II: TCP (10 pts)

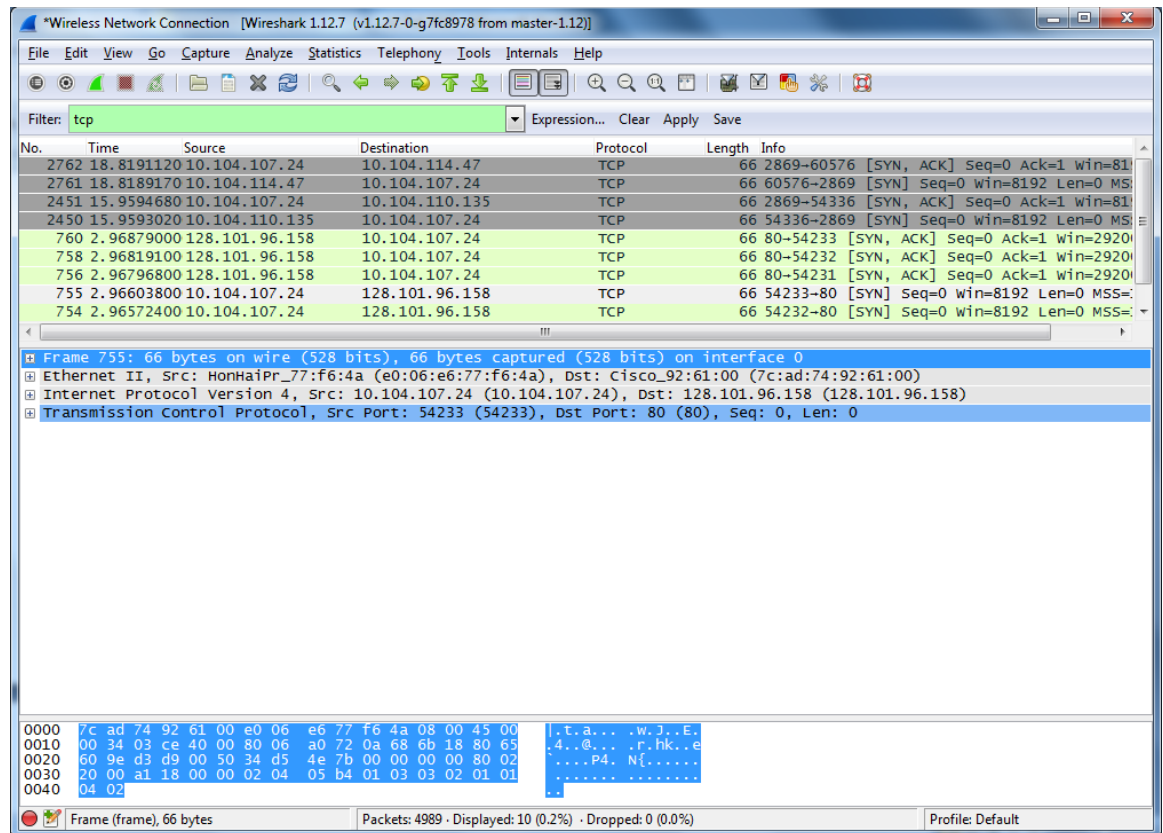
In this practice, we'll investigate the behavior of TCP in detail. Before beginning our exploration of TCP, we'll need to use Wireshark to obtain a packet trace of the TCP. You'll do so by accessing:

<http://www-users.cs.umn.edu/~tianhe/csci4211/HTTP-2.htm>

First, filter the packets displayed in the Wireshark window by entering "tcp" into the display filter window towards the top of the Wireshark window. What you should see is series of TCP and HTTP messages between your computer and

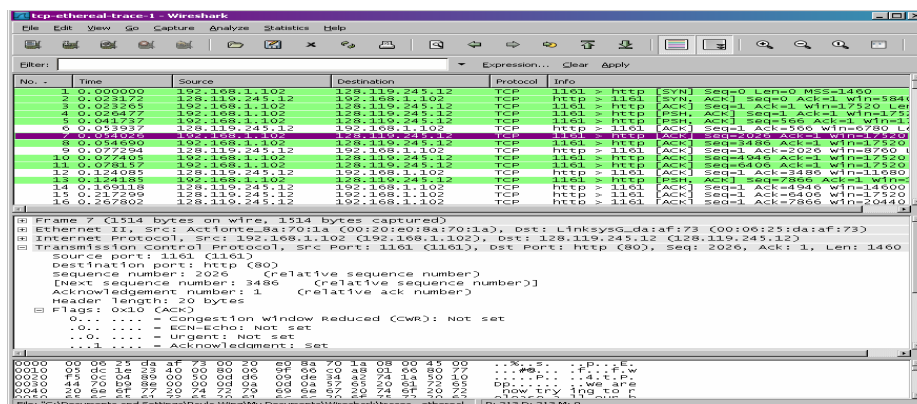
Please answer the following question (along with screenshots as the evidence of your answer)

- (a) What is the IP address and TCP port number used by the client computer (source) that downloads the bill of rights from [www-users.cs.umn.edu](http://www-users.cs.umn.edu/~tianhe/csci4211/HTTP-2.htm)? You need to paste an appropriate screenshot as the evidence. (2pts)



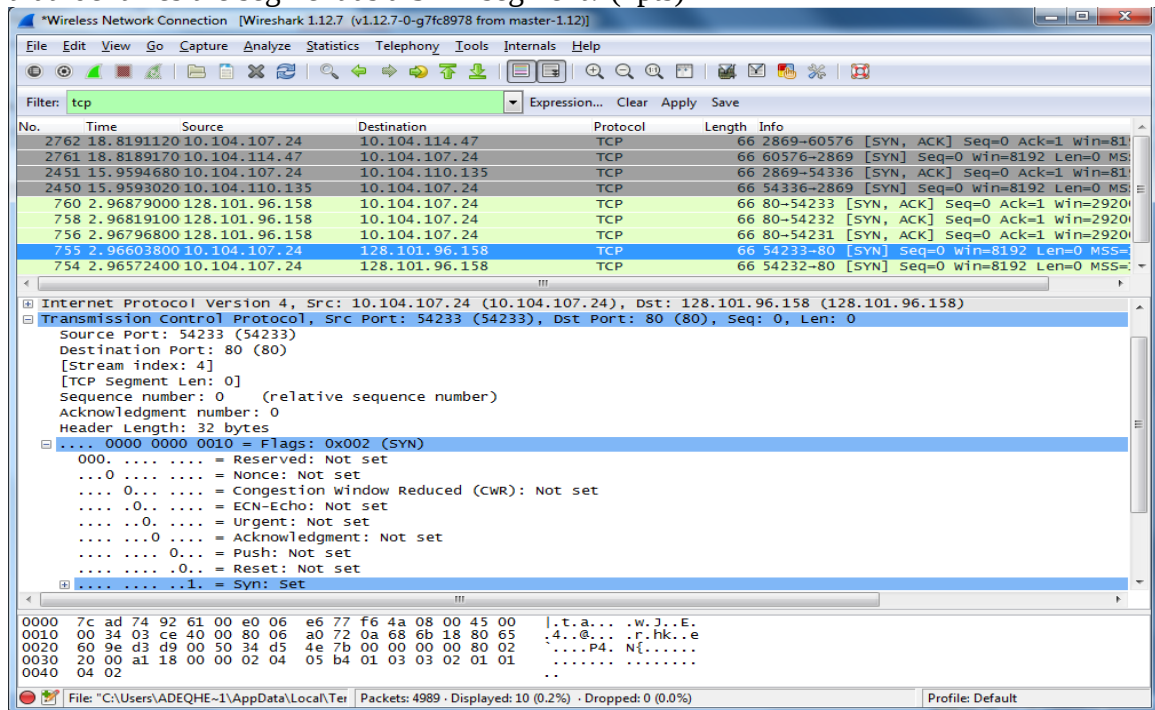
Client computer (source)
 IP: 10.10.107.24
 Source port number: 54233
 Destination port number: 80

Since this lab is about TCP rather than HTTP, let's change Wireshark's "listing of captured packets" window so that it shows information about the TCP segments containing the HTTP messages, rather than about the HTTP messages. To have Wireshark do this, select *Analyze->Enabled Protocols*. Then uncheck the HTTP box and select OK. You should now see a Wireshark window that looks like:



Please answer the following question (along with screenshots as the back-up evidence of your answer, if applicable)

- (b) What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and the webserver? What is it in the segment that identifies the segment as a SYN segment? (2pts)

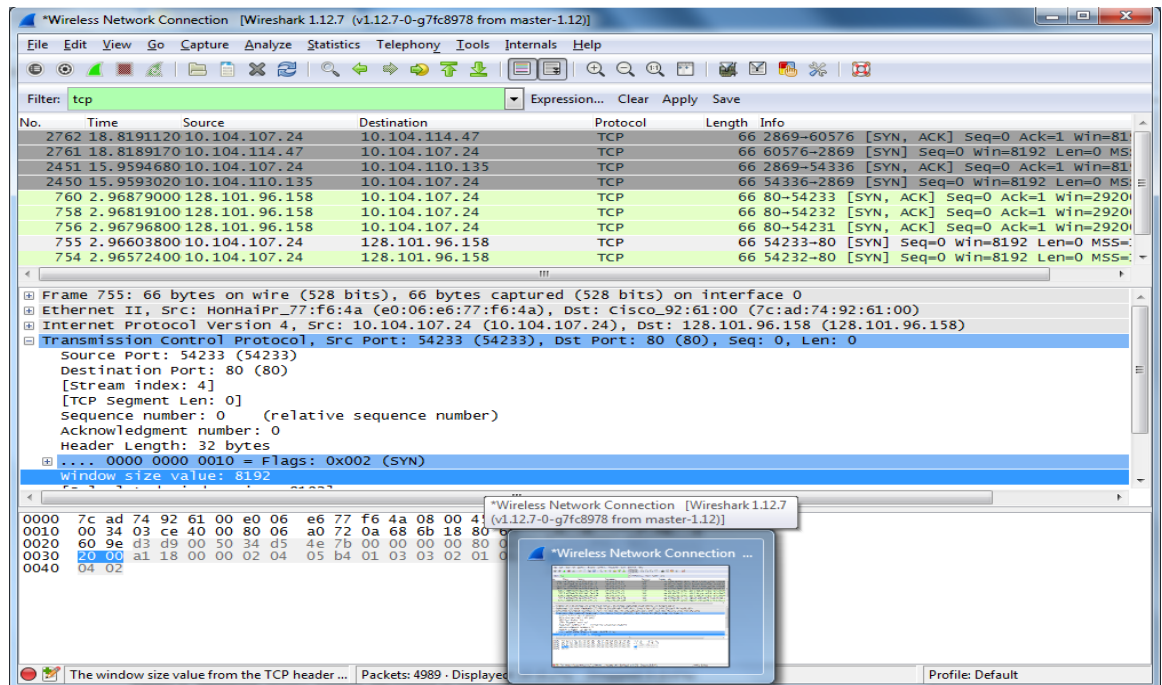


The value of the sequence number of the TCP SYN in this trace is 0
The SYN flag identifies the segment as SYN segment and its set to 1.

- (c) What is the minimum amount of available buffer space advertised at the received for the entire trace? Does the lack of receiver buffer space ever throttle the sender? (2pts)

The minimum available buffers space as shown in the print screen is 8192 bytes.
As far as throttling is concerned, the buffers would never be throttled due to lack of receiver buffer space because it's not close to the maximum available buffer space.

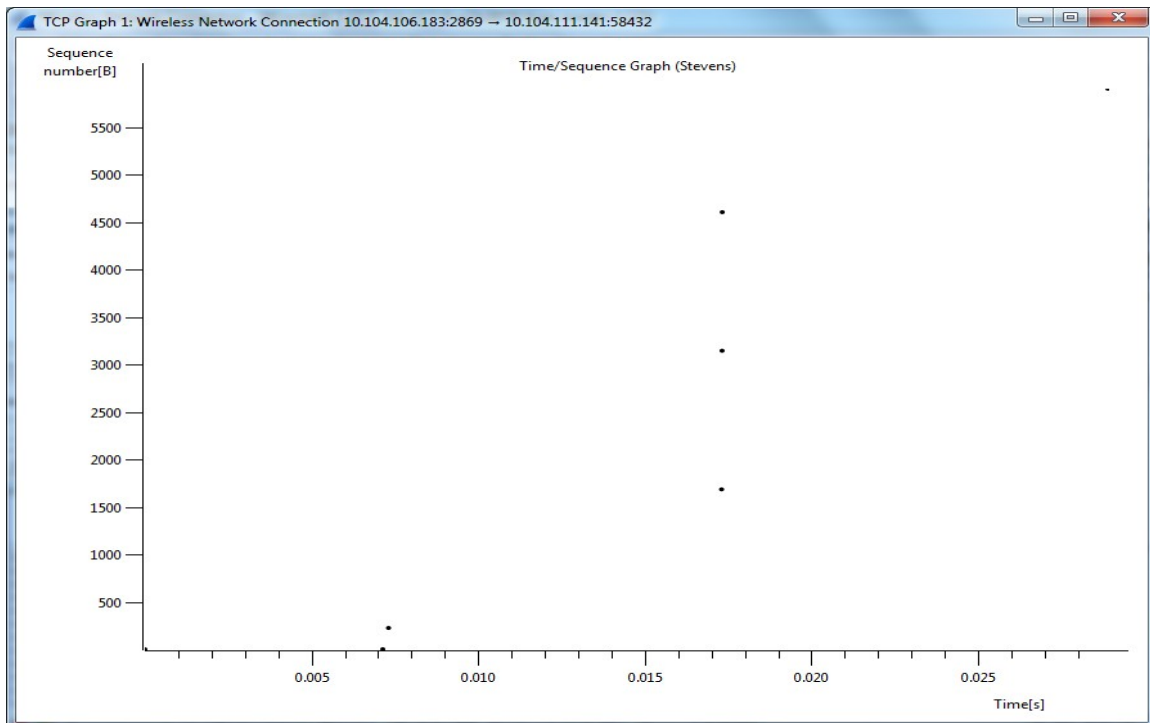
//screenshot below



(d) Are there any retransmitted segments in the trace file? What did you check for in order to answer this question? (2pts) You need to paste a screenshot as the evidence.

There is no retransmission. We can check for this by checking the sequence numbers of the TCP segments

Note: The black dots are tiny but still visible.



- (e) Based on the sequence number, you can calculate how many bytes (TCP payload) have been transferred through this TCP connection. Does the number of total bytes downloaded through this TCP connection equal to the html file size of the bill of rights? Explain why? (2 pts)

The total number of bytes downloaded through the TCP connection is equal to the size of the HTML, we know from the last acknowledgement number that it's not equal to the HTML size. This is because the HTML has been compressed.

6.Programming assignment: Network performance measurement via socket programming (50 pts)

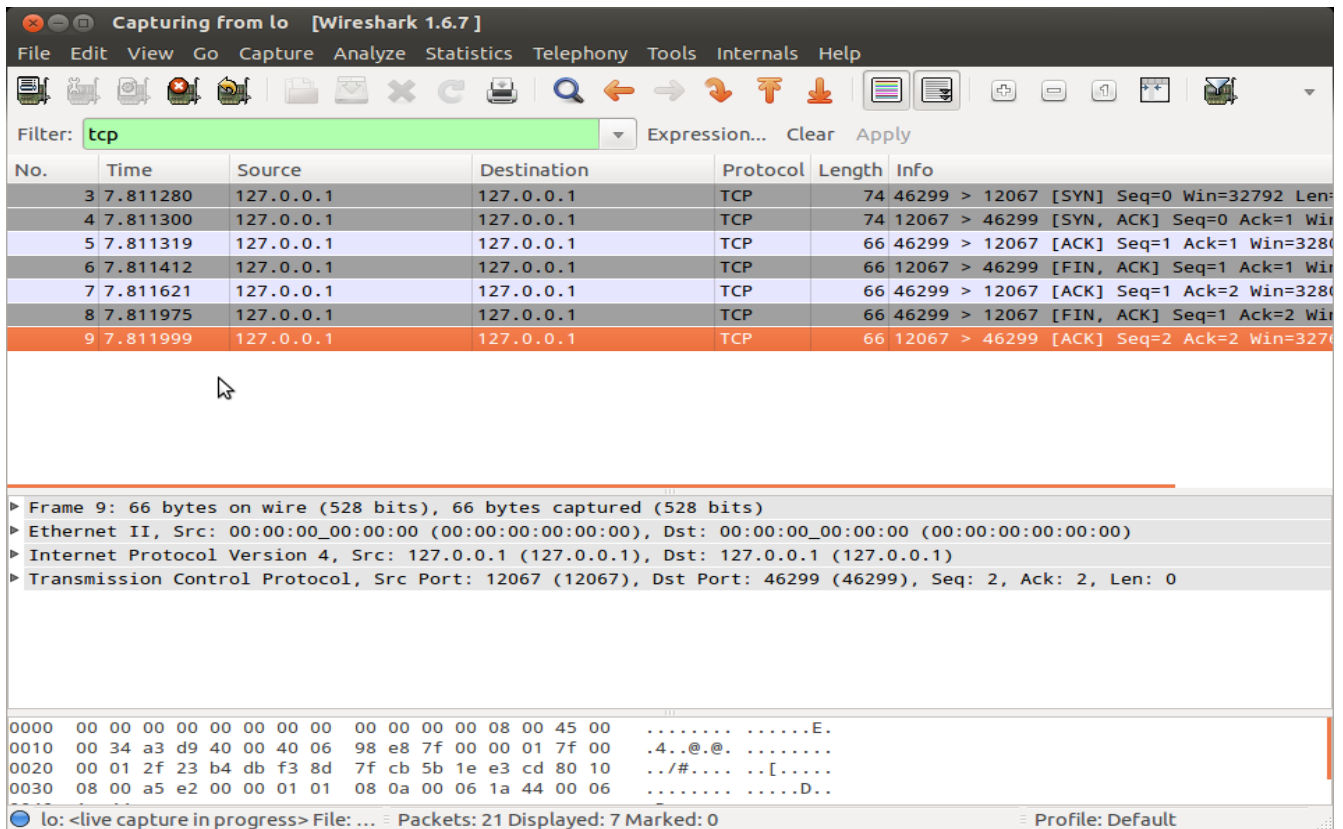
This problem requires building server and client applications in C which exchange packets over UDP and TCP. Using these applications, you will measure the performance of the network path between two hosts.

TCP performance measurement:

Build server and client applications that communicate over TCP sockets. All measurement must be performed on the client. Read and understand the questions prior to implementation.

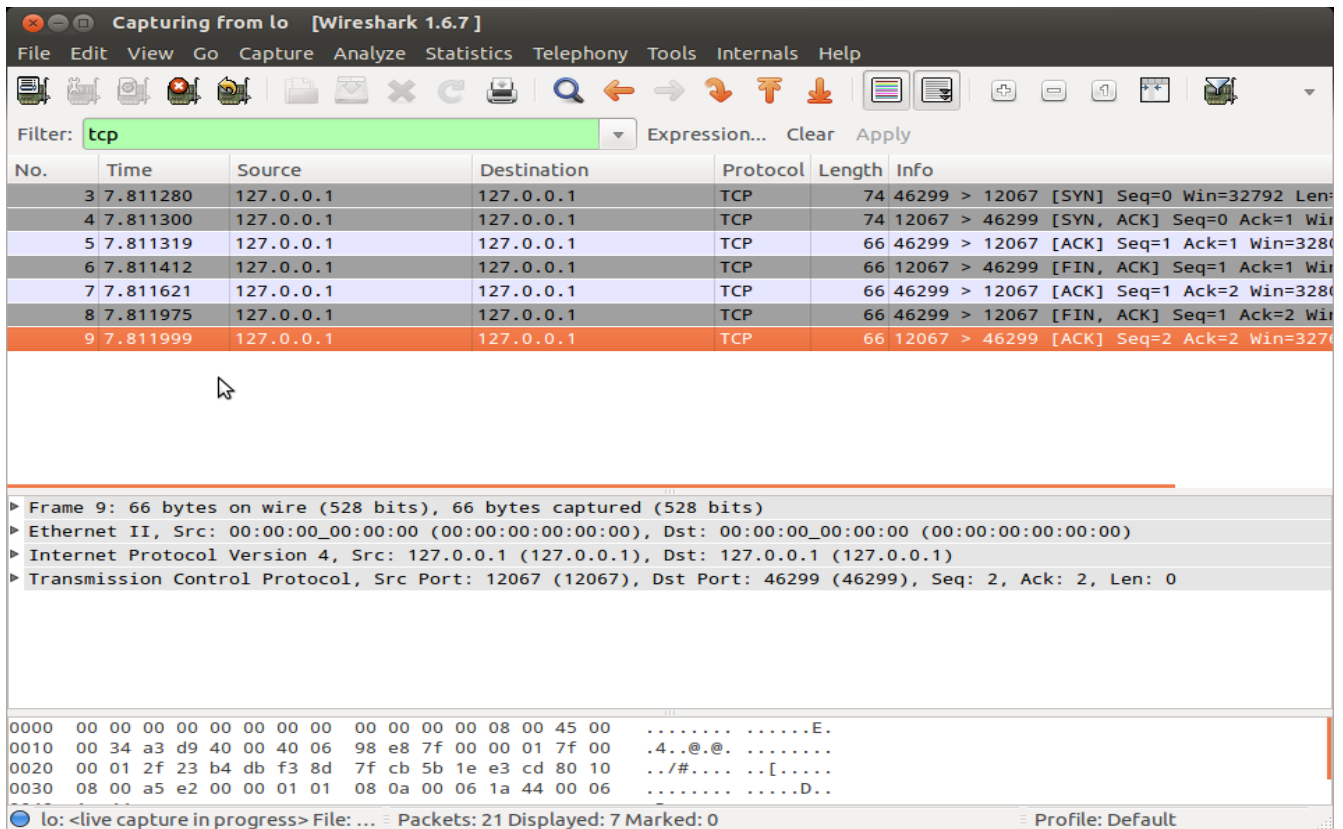
Run both the client and the server on your home LAN for parts (a) & (b).

- a. Using Wireshark, observe the TCP connection establishment process (namely, three way handshake). Repeat 5 times and provide the average time taken for connection establishment. Ignore the time for the last ACK in three way handshaking to reach the server, as this cannot be measured from the client side. Simply put, the establishment time is defined as the duration from when "synchronize" (SYN) segment (first segment) was sent out from the client until when ACK (third segment) was sent out from the client (in response to SYN & ACK segment (second segment) from the server. Also please provide a screenshot of Wireshark showing TCP connection establishment. (5pts)
- Average connection establishment time = 0.000124



- Note: this is in the time unit used in wireshark, which I believe is nanoseconds.
- b. Using Wireshark, observe the TCP connection termination process, initiated by the client. Repeat 5 times and provide the average time taken for connection termination. Ignore the time for the last ACK to reach the server, as this cannot be measured from the client side. Simply put, the establishment time is defined as the duration from when FIN segment (First segment) was sent out from the client until when ACK (fourth segment) was sent out from the client in response to FIN segment (third segment) from the server. Also please provide a screenshot of Wireshark showing TCP connection termination. (5pts)

Average connection termination time = 0.0002546



Now, run the client and the server on the [CSELabs UNIX machines](#): The server can be on one of the machines in [Keller Hall 4-250](#) and the client can be on one of the machines in [Lind Hall 40](#).

- c. Show the network path between the server and the client. Note that they need to be at least 2 hops apart. Please provide a screenshot of the traceroute command on Linux as well as a table including all the IP addresses on the path (including the client and the server) (5 pts)

IP on the Path:

Name	IP
Server	134.84.62.102
Telecomb-en connection	134.84.62.254
Kh4250.-07	128.101.37.7

```

lind40-02.cselabs.umn.edu - PuTTY
noorx004@csel-lind40-02 (/home/noorx004/CSCI4211/pa2/client) % traceroute kh4250-07
traceroute to kh4250-07 (128.101.37.7), 30 hops max, 60 byte packets
 1 telecomb-cn-01-v879.ggnnet.umn.edu (134.84.62.254)  4.667 ms  4.966 ms  5.276 ms
 2 csel-kh4250-07.cselabs.umn.edu (128.101.37.7)  0.339 ms  0.345 ms  0.335 ms
noorx004@csel-lind40-02 (/home/noorx004/CSCI4211/pa2/client) %

```

Ippaddresses on the path
134.84.62.254
128.101.37.7

Implement the message communication protocol (which is described in the last portion of the handout) on the TCP based server and client. The client should download a file from the server using the message protocol. **You must submit the code for this part of the assignment.**

- d. The client should measure and report the time taken to download the file from the server.
Repeat this step for different values of BUF_SZ (256B, 512B, 1024B & 1536B) and the three input files.
NOTE: The time measurement should only include the time taken for recv() and should not include any other system/function calls (like time for I/O). You may use gettimeofday() to measure the time.

Fill in the values in the table for each input file. You may consider only messages of type MSG_TYPE_RESP_GET for computation. **(10 pts)**

Filename	Input_small.txt			
buffer size /	256B	512B	1024B	1536B

parameter				
Download time	2.733	1.61	3.4	1.43
No. of messages	1	1	1	1
Total bytes transferred	254	254	254	254

Filename	Input_medium.txt			
buffer size / parameter	256B	512B	1024B	1536B
Download time	87.08	59.57	54.13	51.18
No. of messages	1	1	1	1
Total bytes transferred	1000000	1000000	1000000	1000000

Filename	Input_large.txt			
buffer size / parameter	256B	512B	1024B	1536B
Download time	874.967	541.08	763.97	307.77
No. of messages	1	1	1	1
Total bytes transferred	10000000	10000000	10000000	10000000

Note: I used clock_t which gives CPU time used by a task.

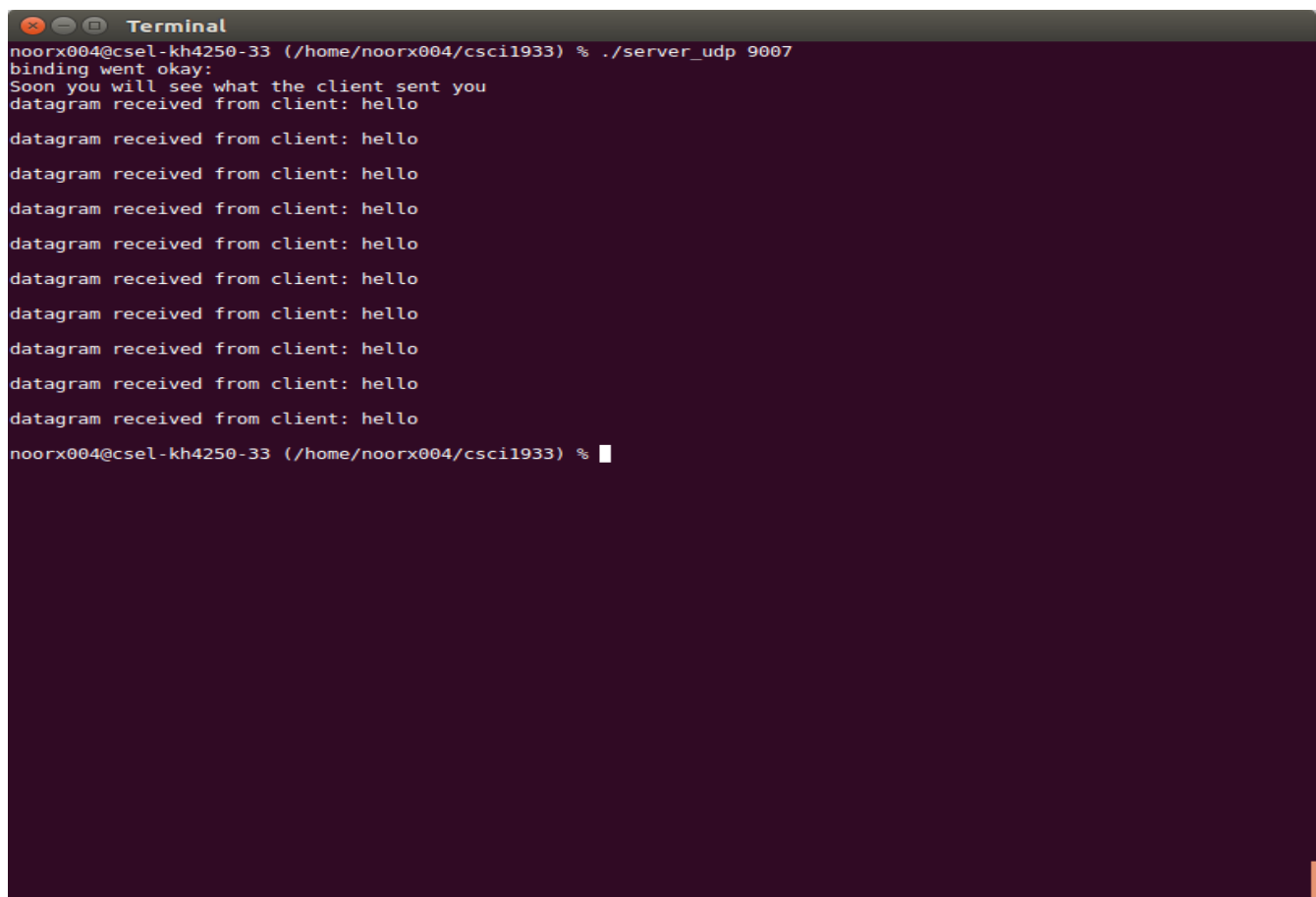
UDP performance measurement:

Build server and client applications that communicate over UDP sockets. All measurement must be performed on the client. Read and understand the questions prior to implementation.

Run the client and the server on the [CSELabs UNIX machines](#): The server can be on one of the machines in [Keller Hall 4-250](#) and the client can be on one of the machines in [Lind Hall 40](#).

- e. Use your own program to measure RTT (Round Trip Time) of UDP. Send 10 UDP datagrams from the client to server. The server should reply with a UDP datagram whenever it receives a datagram from the client. RTT can be obtained by computing the difference in time from when a datagram was sent out from client until the time when the corresponding reply was received by the client. Please provide the average, minimum and standard deviation of RTT and a screenshot of 10 UDP exchanges. (5pts)

Minimum RTT = 5.6 Standard deviation RTT = 0.6364, Average RTT= 6.5mb/s

A terminal window titled "Terminal" with a dark background. The prompt is "noorx004@cse-kh4250-33 (/home/noorx004/csci1933) %". The user has entered the command "./server_udp 9007". The output shows "binding went okay:" followed by "Soon you will see what the client sent you". Then, ten lines of "datagram received from client: hello" are displayed, one for each of the ten UDP datagrams sent. The prompt is visible again at the bottom.

```
noorx004@cse-kh4250-33 (/home/noorx004/csci1933) % ./server_udp 9007
binding went okay:
Soon you will see what the client sent you
datagram received from client: hello
datagram received from client: hello
datagram received from client: hello
datagram received from client: hello
datagram received from client: hello
datagram received from client: hello
datagram received from client: hello
datagram received from client: hello
datagram received from client: hello
datagram received from client: hello
noorx004@cse-kh4250-33 (/home/noorx004/csci1933) %
```

Implement the message communication protocol (which is described in the last portion of the handout) on the UDP based server and client. The client should download a file from the server using the message protocol. **You must submit the code for this part of the assignment.**

- f. The client should measure and report the time taken to download the file from the server. Repeat this step for different values of BUF_SZ (256B, 512B, 1024B & 1536B) and the three input files.

NOTE: The time measurement should only include the time taken for recv() and should not include any other system/function calls (like time for I/O). You may use gettimeofday() to measure the time.

Present your results in a tabular form, like you did for part (d) above. Compare the timing results with the results you obtained for the TCP case in part (d). Explain the difference in delay. **(10 pts)**

Message communication protocol:

- Servers and clients using this protocol can communicate with each other by sending messages. Each message is a C structure of type struct msg_t. The servers and clients can only exchange messages in this format. (This means that the buffer passed in the send() and recv() system calls is always a struct msg_t).

```
#define BUF_SZ 1024
struct msg_t {
    enum msg_type_t msg_type;    /* message type */
    int cur_seq;                 /* current seq number */
    int max_seq;                 /* max seq number */
    int payload_len;             /* length of payload */
    unsigned char payload[BUF_SZ]; /* buffer for data */
};
```

- Some details about struct msg_t:
 - a. msg_type: This indicates the type of the message. Five message types have been predefined.
 - b. cur_seq, max_seq: If a set of related messages have to be transmitted, then cur_seq indicates the sequence number of the current message and max_seq indicates the total number of messages in this sequence
 - c. payload: This is a buffer of size BUF_SZ (1024B) which you can use to fill in any kind of data.
 - d. payload_len: This will indicate the size/length of valid data in the buffer.
- Message types explained:
 - a. MSG_TYPE_GET: Use this message to request a file for download
 - b. MSG_TYPE_GET_ERR: Use this message to indicate errors in obtaining the file (if any)
 - c. MSG_TYPE_GET_RESP: Use this message to send the file across the network
 - d. MSG_TYPE_GET_ACK: Use this message to acknowledge a MSG_TYPE_GET_RESP message.
 - e. MSG_TYPE_FINISH: Use this message to indicate end of session.

Requirements:

1. The TCP server executable must be named `server_tcp` and the UDP server executable must be named `server_udp`.
2. The TCP client executable must be named `client_tcp` and the UDP client executable must be named `client_udp`.
3. Provide two Makefiles, one in the server directory and one in the client directory, which can build these executables.
4. Please note that we will try to download files of different sizes while grading - it is your job to ensure that files of any size can be downloaded. You may test your programs against the provided input files (`input_small.txt`, `input_medium.txt`, `input_large.txt`)
5. The submission must include a README file which clearly contains:
 - a. Name of the student, student ID and x500
 - b. A brief description of how files are downloaded.

Server requirements:

The server program will be executed as follows:

```
$ ./server_tcp <port>
```

```
$ ./server_udp <port>
```

1. The server must listen for clients on a socket bound to the specified port.
2. Print the following message on the screen whenever a message is received:
`server: RX <msg_type> <cur_seq> <max_seq> <payload_len>`
3. The server must send a file to the client when the client requests it.
4. If the requested file is not found in the current working directory, then the server must respond to the client with an appropriate error message
(hint: message type - `MSG_TYPE_GET_ERR`)

Client requirements:

The client program will be executed as follows:

```
$ ./client_tcp <server-ip> <port> <filename>
```

```
$ ./client_udp <server-ip> <port> <filename>
```

1. The client must first connect to the specified server on the specified port.
2. The client must then attempt to download the specified file from the server and save it in the current working directory.
3. File integrity must be preserved when downloading files. This means that downloaded file must exactly match the file on the server. (Hint: Use the `diff` utility to compare the two files). You will lose significant points if the downloaded file differs from the file on the server.
4. Print the following message on the screen whenever a message is received:
`client: RX <msg_type> <cur_seq> <max_seq> <payload_len>`

Sample server:

A sample TCP server application has been provided. The server will allow a client to connect and download a file. You may test your client with this server initially. However,

you are required to develop and submit your own servers and clients for this assignment. The sample server works as follows:

1. Start the TCP server
2. Start the TCP client & connect to the server.
3. Send a `MSG_TYPE_GET` from the client to the server, with the name of the file to be download in the payload.
4. If the server cannot find the file, it will respond with a `MSG_TYPE_GET_ERR` message. If the file is found, the server will break the file into chunks and transmit each chunk in a `MSG_TYPE_GET_RESP` message. Depending on the file size, there will be multiple such messages. The actual file contents will be stored in the payload in each message.
5. The client must send a `MSG_TYPE_GET_ACK` message for each `MSG_TYPE_GET_RESP` it receives - or else the server will not respond with the next message.

You can use the same mechanism or a different mechanism to download the files. The README file should clearly explain the mechanism you use to download the file.

Execution environment:

1. Please ensure that your code compiles and executes on the [CSELabs UNIX machines](#). You will lose significant points if your code cannot be compiled/executed on these machines.
2. While developing/implementing your solution, the server and client can run on the same machine - You can use the IP address as localhost or 127.0.0.1.
3. When it comes to actual measurement, the server and client should run on different machines, ideally multiple hops away. Our suggestion is to use two CSE Lab machines as described below:
 - Run the server on one of the machines in [Keller Hall 4-250](#).
 - Run the client on one of the machines in [Lind Hall 40](#).
 - Please use only port numbers between **9000** and **10000**. (These ports are currently allowed by the system staff).
 - You can use `'ip addr show'` or `'ipconfig'` to get the IP address of the machine on which the server is running.

If your code does not execute in this scenario, you will lose significant points on your submission.

Deliverables:

1. You must upload a single archive file (.zip or .tar or .tar.gz) on moodle.
When extracted, the archive file must be a single folder. The name of the folder should be your **student ID**. The folder should contain the following files:
 - Readme
 - server source files & Makefile
 - client source files & Makefile
 - message.h

- MS Word/PDF document
2. DO NOT include the test files (input_small.txt, input_medium.txt & input_large.txt)
 3. DO NOT include any executable files - we will build the executables using your Makefiles.

For example, here is a sample submission:

```
1234567/  
  Readme  
  PA2.doc  
  message.h  
  server/  
    server_tcp.c  
    server_udp.c  
    Makefile  
  client/  
    client_tcp.c  
    client_udp.c  
    Makefile
```

You can create an archive file from the contents of the above directory as follows:

```
$ tar cvf 1234567.tar.gz 1234567/
```

Grading:

README, Makefiles, comments, readability: **4 points**

Packaging the submission as specified: **3 points**

Logging messages on the screen in the correct format: **1 points**

File download: **2 points**