Cover Page
ECE 461 Lab4 Report

Name: Wei Lin (#999595193)
Name: Shenglin Meng (#1000695517)

Exercise 1

1B

1. How many packets are exchanged in the data transfer? How many packets are transmitted for each UDP datagram? What is the size of the UDP payload of these packets?

Ans: There are in total 18 packets are exchanged in the data transfer, in which 16 packets are transmitted for each UDP datagram. The size of the UDP packets are 1032 bytes, which consist of 8 bytes for UDP header and 1024 byte for UDP payload.

2. Compare the total number of bytes transmitted, in both directions, including Ethernet, IP, and UDP headers, to the amount of application data transmitted.

Ans: The total number of bytes transmitted in both directions is 10936 bytes (10x1066+ 46x6) , and the amount of application data transmitted is 10x1032= 10320 bytes in total.

3. Inspect the fields in the UDP headers. Which fields in the headers do not change in different packets?

Ans: The fields that stays unchanged are source address, destination address , source port , destination port.

4. Observe the port numbers in the UDP header. How did the ttcp sender select the source port number?

Ans: The source port is 32270, which is selected randomly by the ttcp sender within a certain range.

1C

1.How many packets are exchanged in the data transfer? What are the sizes of the TCP segments?

Data:

No. Time Source Destination Protocol Info
1 0.000000 10.0.5.11 10.0.5.22 TCP 44354 > 4444 [SYN] Seq=0 Len=0 MSS=1460 TSV=184090 TSER=0
Frame 1 (70 bytes on wire, 70 bytes captured)

No. Time Source Destination Protocol Info
2 0.001777 00:04:5a:7a:c5:b5 ff:ff:ff:ff:ff:ff ARP Who has 10.0.5.11? Tell 10.0.5.22

Frame 2 (60 bytes on wire, 60 bytes captured)

No. Time Source Destination Protocol Info
3 0.001809 00:04:5a:80:76:df 00:04:5a:7a:c5:b5 ARP 10.0.5.11 is at 00:04:5a:80:76:df

Frame 3 (42 bytes on wire, 42 bytes captured)

No. Time Source Destination Protocol Info
4 0.001871 10.0.5.22 10.0.5.11 TCP 4444 > 44354 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 TSV=153064 TSER=184090

Frame 4 (70 bytes on wire, 70 bytes captured)

No. Time Source Destination Protocol Info
5 0.001897 10.0.5.11 10.0.5.22 TCP 44354 > 4444 [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSV=184090 TSER=153064

Frame 5 (66 bytes on wire, 66 bytes captured)

No. Time Source Destination Protocol Info
6 0.010357 10.0.5.11 10.0.5.22 TCP 44354 > 4444 [PSH, ACK] Seq=1 Ack=1 Win=5840 Len=1024 TSV=184093 TSER=153064

Frame 6 (1090 bytes on wire, 1090 bytes captured)

No. Time Source Destination Protocol Info
7 0.010411 10.0.5.11 10.0.5.22 TCP 44354 > 4444 [PSH, ACK] Seq=1025 Ack=1 Win=5840 Len=1024 TSV=184093 TSER=153064

Frame 7 (1090 bytes on wire, 1090 bytes captured)

No. Time Source Destination Protocol Info
8 0.010670 10.0.5.22 10.0.5.11 TCP 4444 > 44354 [ACK] Seq=1 Ack=1025 Win=7168 Len=0 TSV=153067 TSER=184093

Frame 8 (66 bytes on wire, 66 bytes captured)

No. Time Source Destination Protocol Info

9 0.010710 10.0.5.11 10.0.5.22 TCP 44354 > 4444 [ACK] Seq=2049 Ack=1 Win=5840 Len=1448 TSV=184093 TSER=153067

Frame 9 (1514 bytes on wire, 1514 bytes captured)

No. Time Source Destination Protocol Info
10 0.010714 10.0.5.11 10.0.5.22 TCP 44354 > 4444 [ACK] Seq=3497 Ack=1 Win=5840 Len=1448 TSV=184093 TSER=153067

Frame 10 (1514 bytes on wire, 1514 bytes captured)

No. Time Source Destination Protocol Info
11 0.010693 10.0.5.22 10.0.5.11 TCP 4444 > 44354 [ACK] Seq=1 Ack=2049 Win=9216 Len=0 TSV=153067 TSER=184093

Frame 11 (66 bytes on wire, 66 bytes captured)

No. Time Source Destination Protocol Info
12 0.010721 10.0.5.11 10.0.5.22 TCP 44354 > 4444 [ACK] Seq=4945 Ack=1 Win=5840 Len=1448 TSV=184093 TSER=153067

Frame 12 (1514 bytes on wire, 1514 bytes captured)

No. Time Source Destination Protocol Info
13 0.010724 10.0.5.11 10.0.5.22 TCP 44354 > 4444 [ACK] Seq=6393 Ack=1 Win=5840 Len=1448 TSV=184093 TSER=153067

Frame 13 (1514 bytes on wire, 1514 bytes captured)

No. Time Source Destination Protocol Info
14 0.011222 10.0.5.22 10.0.5.11 TCP 4444 > 44354 [ACK] Seq=1 Ack=3497 Win=11584 Len=0 TSV=153067 TSER=184093

Frame 14 (66 bytes on wire, 66 bytes captured)

No. Time Source Destination Protocol Info
15 0.011252 10.0.5.11 10.0.5.22 TCP 44354 > 4444 [ACK] Seq=7841 Ack=1 Win=5840 Len=1448 TSV=184093 TSER=153067

Frame 15 (1514 bytes on wire, 1514 bytes captured)

No. Time Source Destination Protocol Info

16 0.011256 10.0.5.11 10.0.5.22 TCP 44354 > 4444 [PSH, ACK] Seq=9289 Ack=1 Win=5840 Len=952 TSV=184093 TSER=153067

Frame 16 (1018 bytes on wire, 1018 bytes captured)

No. Time Source Destination Protocol Info
17 0.011402 10.0.5.11 10.0.5.22 TCP 44354 > 4444 [FIN, ACK] Seq=10241 Ack=1 Win=5840 Len=0 TSV=184093 TSER=153067

Frame 17 (66 bytes on wire, 66 bytes captured)

No. Time Source Destination Protocol Info
18 0.011481 10.0.5.22 10.0.5.11 TCP 4444 > 44354 [ACK] Seq=1 Ack=4945 Win=14480 Len=0 TSV=153067 TSER=184093

Frame 18 (66 bytes on wire, 66 bytes captured)

No. Time Source Destination Protocol Info
19 0.011499 10.0.5.22 10.0.5.11 TCP 4444 > 44354 [ACK] Seq=1 Ack=6393 Win=17376 Len=0 TSV=153067 TSER=184093

Frame 19 (66 bytes on wire, 66 bytes captured)

No. Time Source Destination Protocol Info
20 0.011543 10.0.5.22 10.0.5.11 TCP 4444 > 44354 [ACK] Seq=1 Ack=7841 Win=20272 Len=0 TSV=153067 TSER=184093

Frame 20 (66 bytes on wire, 66 bytes captured)

No. Time Source Destination Protocol Info
21 0.011658 10.0.5.22 10.0.5.11 TCP 4444 > 44354 [ACK] Seq=1 Ack=9289 Win=23168 Len=0 TSV=153067 TSER=184093

Frame 21 (66 bytes on wire, 66 bytes captured)

No. Time Source Destination Protocol Info
22 0.011668 10.0.5.22 10.0.5.11 TCP 4444 > 44354 [ACK] Seq=1 Ack=10241 Win=26064 Len=0 TSV=153067 TSER=184093

Frame 22 (66 bytes on wire, 66 bytes captured)

No. Time Source Destination Protocol Info

23 0.011857 10.0.5.22 10.0.5.11 TCP 4444 > 44354 [FIN, ACK] Seq=1 Ack=10242 Win=26064 Len=0 TSV=153067 TSER=184093

Frame 23 (66 bytes on wire, 66 bytes captured)

No. Time Source Destination Protocol Info
24 0.011883 10.0.5.11 10.0.5.22 TCP 44354 > 4444 [ACK] **Seq=10242** Ack=2 Win=5840 Len=0 TSV=184093 TSER=153067

Frame 24 (66 bytes on wire, 66 bytes captured)

Ans: As the data above shown , there are totally 22 packets are transmitted. The size of the TCP segments are 66 , 70 , 1090.1514 bytes.

2.What is the range of the sequence numbers?

Ans: As the data above shown , the range of sequence number is from 0 to 10242.

3.How many packets are transmitted by PC1 and how many packets are transmitted by PC2?

Ans: There are totally 12 TCP packets transmitted by PC1 (10.0.5.11) , and 10 TCP packets are transmitted by PC2 (10.0.5.22)

4.How many packets do not carry a payload, that is, how many packets are control packets?

Ans: As the data shown above, there are 14 packets not carrying any payload.

5.Compare the total number of bytes transmitted, in both directions, including Ethernet, IP, and UDP headers, to the amount of application data transmitted.

Ans: The total amount of application data transmitted is 10772 bytes .Whereas the total number of bytes transmitted in both directions is 11712 bytes.

6.Inspect the TCP headers. Which packets contain flags in the TCP header? Which types of flags do you observe?

Ans: Since usually the flag for a packet is an indication of its purpose , every single packet during the transmission has a flag attached to them. The types of flag observed are: SYN , FIN ,ACK, [FIN,ACK], [PSH,ACK], [FIN,ACK]

Lab Report

1.Compare the amount of data transmitted in the TCP and the UDP data transfers.

Ans: The total size of UDP data transmitted (10936 bytes) is slightly less than the TCP data (11712 bytes)

2.Take the biggest UDP datagram and the biggest TCP segment that you observed, and compare the amount of application data that is transmitted in the UDP datagram and the TCP segment.

Ans: The biggest UDP datagram has a size of 1060 bytes compared to the biggest TCP datagram which has a size of 1514 bytes. The payload carried by UDP is 1024 bytes , and the payload carried by TCP is 1448 bytes. The reason why TCP datagram size is larger is because TCP transmits data in segment stream, whereas UDP does not, which eventually cause TCP have bigger packets than UDP.

Exercise 2

2.1 From the timestamps recorded by ethereal, obtain the times it took to transfer the file with FTP and with TFTP? Use your knowledge of FTP, TFTP, TCP, and UDP to explain the outcome.

Ans: FTP takes almost 48 seconds to transfer a file with size of 1MB while TFTP only takes about 5 seconds. FTP is slower than TFTP because FTP is using TCP protocol while TFTP is using UDP protocol. There are 2 main delay that causes TCP to be much slower than UDP. The first one is the buffering delay. TCP will buffer lots of data so that it only needs to send a few big packets instead of lots of packets. This is so-called Nagle's Algorithm in TCP. Another delay is from the data retransmit and data recovery due to packets lost (Note that UDP uses unreliable transfer, so if a packet is lost, UDP will not do anything). Let's say TCP sends segments 5,6,7,8 from the server to the client. Segment 5 is lost in transit. Even though the information in segments 6 to 8 arrive at the client, TCP will not give it to the client until segment 5 arrives. It can take a significant period of time before TCP realises that a segment is lost and resends it.

2.2 Identify the TCP connections that are created in the FTP session, and record the port numbers at the source and at the destination.

Ans: The source port number is 46064, which is randomly generated from client while the destination port number is 21, which is the port number for the FTP application.

Exercise 3

3A

1. Determine the exact UDP datagram size at which fragmentation occurs.

Data:

No. Time Source Destination Protocol Info
22 0.009617 10.0.1.11 10.0.2.22 IP Fragmented IP protocol (proto=UDP 0x11, off=0) [Reassembled in #23]

Frame 22 (1514 bytes on wire, **1514 bytes captured**)
Arrival Time: Jul 19, 2007 09:53:56.311092000
[Time delta from previous packet: 0.000017000 seconds]
[Time since reference or first frame: 0.009617000 seconds]
Frame Number: 22
Packet Length: 1514 bytes
Capture Length: 1514 bytes
[Frame is marked: False]
[Protocols in frame: eth:ip:data]
Ethernet II, Src: 00:04:5a:7a:c8:25 (00:04:5a:7a:c8:25), Dst: 00:04:5a:7a:c6:64 (00:04:5a:7a:c6:64)
Destination: 00:04:5a:7a:c6:64 (00:04:5a:7a:c6:64)
Address: 00:04:5a:7a:c6:64 (00:04:5a:7a:c6:64)
....0 .... = IG bit: Individual address (unicast)
.... ..0. .... = LG bit: Globally unique address (factory default)
Source: 00:04:5a:7a:c8:25 (00:04:5a:7a:c8:25)
Address: 00:04:5a:7a:c8:25 (00:04:5a:7a:c8:25)
....0 .... = IG bit: Individual address (unicast)
.... ..0. .... = LG bit: Globally unique address (factory default)
Data (1480 bytes)

Ans: The size of the UDP datagram when fragment happens is 1514 bytes in total , and the data transmitted is 1480 bytes.

2.Determine the maximum size of the UDP datagram that the system can send and receive, regardless of fragmentation, i.e., fragmentation of data segments occurs until a point beyond which the segment size is too large for to be handled by IP.

Ans: The maximum size of the UDP datagram the system can send or receive is Max size(65535 bytes) - Header size (8 bytes) - IP header(20 bytes) = 65507 bytes. Therefore 65507 is the maximum datagram size that system can transmits.

Lab Report

1.From the saved ethereal data, select one IP datagram that is fragmented. Include the complete datagram before fragmentation and include all fragments after fragmentation.

Data:

No. Time Source Destination Protocol Info
6 0.009393 10.0.1.11 10.0.2.22 IP Fragmented IP protocol (proto=UDP 0x11, off=0) [Reassembled in #7]

Frame 6 (1514 bytes on wire, 1514 bytes captured)
Arrival Time: Jul 19, 2007 09:53:56.310868000
[Time delta from previous packet: 0.000018000 seconds]
[Time since reference or first frame: 0.009393000 seconds]
Frame Number: 6
Packet Length: 1514 bytes
Capture Length: 1514 bytes
[Frame is marked: False]
[Protocols in frame: eth:ip:data]
Ethernet II, Src: 00:04:5a:7a:c8:25 (00:04:5a:7a:c8:25), Dst: 00:04:5a:7a:c6:64 (00:04:5a:7a:c6:64)
Destination: 00:04:5a:7a:c6:64 (00:04:5a:7a:c6:64)
Address: 00:04:5a:7a:c6:64 (00:04:5a:7a:c6:64)
....0 .... = IG bit: Individual address (unicast)
.... ..0. .... = LG bit: Globally unique address (factory default)
Source: 00:04:5a:7a:c8:25 (00:04:5a:7a:c8:25)
Address: 00:04:5a:7a:c8:25 (00:04:5a:7a:c8:25)
....0 .... = IG bit: Individual address (unicast)
.... ..0. .... = LG bit: Globally unique address (factory default)
Type: IP (0x0800)
Internet Protocol, Src: 10.0.1.11 (10.0.1.11), Dst: 10.0.2.22 (10.0.2.22)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
0000 00.. = Differentiated Services Codepoint: Default (0x00)
.... ..0. = ECN-Capable Transport (ECT): 0
....0 = ECN-CE: 0
Total Length: 1500
Identification: 0x4888 (18568)
Flags: 0x02 (More Fragments)
0... = Reserved bit: Not set
.0.. = Don't fragment: Not set
..1. = More fragments: Set

```
    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (0x11)
    Header checksum: 0xf568 [correct]
        [Good: True]
        [Bad : False]
    Source: 10.0.1.11 (10.0.1.11)
    Destination: 10.0.2.22 (10.0.2.22)
    Reassembled IP in frame: 7
Data (1480 bytes)

No.     Time        Source                Destination           Protocol Info
      7 0.009396    10.0.1.11             10.0.2.22             UDP      Source port: 32770  Destination port: 4444
[BAD UDP LENGTH 1481 > IP PAYLOAD LENGTH]

Frame 7 (35 bytes on wire, 35 bytes captured)
    Arrival Time: Jul 19, 2007 09:53:56.310871000
    [Time delta from previous packet: 0.000003000 seconds]
    [Time since reference or first frame: 0.009396000 seconds]
    Frame Number: 7
    Packet Length: 35 bytes
    Capture Length: 35 bytes
    [Frame is marked: False]
    [Protocols in frame: eth:ip:udp:data]
    [Coloring Rule Name: UDP]
    [Coloring Rule String: udp]
Ethernet II, Src: 00:04:5a:7a:c8:25 (00:04:5a:7a:c8:25), Dst: 00:04:5a:7a:c6:64 (00:04:5a:7a:c6:64)
    Destination: 00:04:5a:7a:c6:64 (00:04:5a:7a:c6:64)
        Address: 00:04:5a:7a:c6:64 (00:04:5a:7a:c6:64)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
    Source: 00:04:5a:7a:c8:25 (00:04:5a:7a:c8:25)
        Address: 00:04:5a:7a:c8:25 (00:04:5a:7a:c8:25)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
    Type: IP (0x0800)
Internet Protocol, Src: 10.0.1.11 (10.0.1.11), Dst: 10.0.2.22 (10.0.2.22)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
```

Total Length: 21
Identification: 0x4888 (18568)
Flags: 0x00
0... = Reserved bit: Not set
.0.. = Don't fragment: Not set
..0. = More fragments: Not set
Fragment offset: 1480
Time to live: 64
Protocol: UDP (0x11)
Header checksum: 0x1a77 [correct]
[Good: True]
[Bad : False]
Source: 10.0.1.11 (10.0.1.11)
Destination: 10.0.2.22 (10.0.2.22)
[IP Fragments (1481 bytes): #6(1480), #7(1)]
[Frame: 6, payload: 0-1479 (1480 bytes)]
[Frame: 7, payload: 1480-1480 (1 byte)]
User Datagram Protocol, Src Port: 32770 (32770), Dst Port: 4444 (4444)
Source port: 32770 (32770)
Destination port: 4444 (4444)
Length: 1481 (bogus, should be 1)
Checksum: 0x231b [correct]
[Good Checksum: True]
[Bad Checksum: False]
Data (1473 bytes)

For each fragment of this datagram, determine the values of the fields in the IP header that are used for fragmentation (Identification, Fragment Offset, Dont Fragment Bit, More Fragments Bit).

Ans:

Identification	Reserved bit	DF	MF	Fragment Offset

The IP header has a format of above table and DF stands for don't fragment , MF stands for more fragment, meaning that the current packet is a fragment. When a packet is a not a fragment, as the data shown , the flag is 0x04 , reserved bit : not set (0) ; DF bit : 1 , MF bit : 0 (Not Set) , Fragment Offset : 0 When a packet is a fragment itself. The values are: Flag: 0x02 , Reserved bit: 0 (Not Set), DF bit : 0 , MF bit : 1 , Fragment offset : 0.

Before:
Flags: 0x04 (Don't Fragment)
0... = Reserved bit: Not set
.1.. = Don't fragment: Set
..0. = More fragments: Not set

Fragment offset: 0

After:
Flags: 0x02 (More Fragments)
0... = Reserved bit: Not set
.0.. = Don't fragment: Not set
..1. = More fragments: Set
Fragment offset: 0

2.Include the outcome of the experiment in Step 4. Indicate the UDP datagram size at which fragmentation occurs. Also, determine the maximum size of the UDP datagram that the system can send.

Ans: As the previous questions stated, the UDP datagram size when the fragmentation happens is 1514 bytes in total with a 1480 bytes application data. The maximum size of the UDP datagram the system can send or receive is Max size(65535 bytes) - Header size (8 bytes) - IP header(20 bytes) = 65507 bytes. Therefore 65507 is the maximum datagram size that system can transmits.

Exercise 3B

3.3.1 Do you observe fragmentation? If so, where does it occur? Explain your observation.

Ans: No , no fragmentation is observed. Since the size of TCP packet is determined by the MSS during the three way handshake process between the client (PC1) and server (PC2), which is 500 bytes. The MTU size of internal router (PC3) is 1500, so the TCP packet can safely pass through the PC3 with error message.

3.3.2 Explain why there is no ICMP error message generated in the first part of the experiment (Step 3). Is the DF bit set in the IP datagrams?

Ans: Since the MSS size is smaller than the MTU size of internal router (PC3), so no fragmentation is needed when TCP packet pass through PC3. With DF bit set, there will be no error ICMP message. The DF bit is set. Here is one of the packet:

No. Time Source Destination Protocol Info
3 0.004391 10.0.1.11 10.0.2.22 TCP 52682 > 4444 [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSV=189160 TSER=170386
….
Internet Protocol, Src: 10.0.1.11 (10.0.1.11), Dst: 10.0.2.22 (10.0.2.22)

```
   Version: 4
   Header length: 20 bytes
   Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
      0000 00.. = Differentiated Services Codepoint: Default (0x00)
      .... ..0. = ECN-Capable Transport (ECT): 0
      .... ...0 = ECN-CE: 0
   Total Length: 52
   Identification: 0x5f2e (24366)
   Flags: 0x04 (Don't Fragment)
      0... = Reserved bit: Not set
      .1.. = Don't fragment: Set
      ..0. = More fragments: Not set
   Fragment offset: 0
   Time to live: 64
   Protocol: TCP (0x06)
   Header checksum: 0xc475 [correct]
```

3.5.2 Do you observe fragmentation? If so, where does it occur? Explain your observation.

Ans: Yes, the fragmentation happened in the internal router PC3.

After change the MTU size of eth0 of PC2 to 1500 and eth1 of PC3 to 500, the MSS size between the client PC1 and server PC2 will become 1500. However, the maximum frame size of internal router PC3 will become 500.

In this case, PC1 will send a frame with size = 1500 bytes to PC2 with DF bit set. When the frame reaches PC3, PC3 need to fragment the frame because the MTU of its eth1 interface is 500, but the DF bit is set, which means the TCP packet is not allowed to be fragmented. Therefore, PC3 will send an ICMP error message, Type 3 Code 4 “Fragmentation Needed”, to PC1 to notice PC1 that fragmentation is required.

Here is an example of ICMP error message:

```
No.     Time       Source              Destination         Protocol Info
     9 0.018484    10.0.1.33           10.0.1.11           ICMP     Destination unreachable (Fragmentation
needed)
...
   Protocol: ICMP (0x01)
   Header checksum: 0x44a4 [correct]
      [Good: True]
      [Bad : False]
   Source: 10.0.1.33 (10.0.1.33)
   Destination: 10.0.1.11 (10.0.1.11)
Internet Control Message Protocol
```

Type: 3 (Destination unreachable)
Code: 4 (Fragmentation needed)
Checksum: 0xbb3d [correct]
MTU of next hop: 500
Internet Protocol, Src: 10.0.1.11 (10.0.1.11), Dst: 10.0.2.22 (10.0.2.22)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
0000 00.. = Differentiated Services Codepoint: Default (0x00)
.... ..0. = ECN-Capable Transport (ECT): 0
....0 = ECN-CE: 0
Total Length: 1076
Identification: 0x36f9 (14073)
Flags: 0x04 (Don't Fragment)
0... = Reserved bit: Not set
.1.. = Don't fragment: Set
..0. = More fragments: Not set
Fragment offset: 0
Time to live: 63
Protocol: TCP (0x06)
Header checksum: 0xe9aa [correct]

3.5.2. If you observe ICMP error messages, describe how they are used for Path MTU Discovery.

Ans: The ICMP error message can be used for Path MTU Discovery by initially send out a very large ICMP packet. When the large ICMP packet hits a router with smaller MTU size, the router will send back an ICMP error message, “Fragmentation Needed” Type 3 Code 4. Then, after host receives the ICMP error message, the host will try to send a smaller ICMP packet. The process will be repeated until the ICMP message reached the destination.

Exercise 4

4A

4.1 Identify the packets of the three-way handshake. Which flags are set in the TCP headers? Explain how these flags are interpreted by the receiving TCP server or TCP client.

Ans: 1st packet is [SYN]. The client sends request to establish the connection with SYN bit set to 1.
2nd packet is [SYN + ACK]. The server accepts the connection and send back a packet with both SYN and ACK bits set to 1. The SYNC bit means server is used to establish the connection and the ACK bit is used to let client know last segment has been received.
3rd packet is [ACK]. After the client receives the server's response, it sends an ACK packet, which means the connection is established successfully.

Here is the information of TCP header in the three-way handshake, redundant information is omitted and the flag field in highlighted:

```
No.     Time        Source                Destination           Protocol Info
      1 0.000000    10.0.5.11             10.0.5.22             TCP      50093 > telnet [SYN] Seq=0 Len=0
MSS=1460 TSV=1197740 TSER=0
...
Transmission Control Protocol, Src Port: 50093 (50093), Dst Port: telnet (23), Seq: 0, Len: 0
    Source port: 50093 (50093)
    Destination port: telnet (23)
    Sequence number: 0    (relative sequence number)
    Header length: 36 bytes
    Flags: 0x02 (SYN)
        0... .... = Congestion Window Reduced (CWR): Not set
        .0.. .... = ECN-Echo: Not set
        ..0. .... = Urgent: Not set
        ...0 .... = Acknowledgment: Not set
        .... 0... = Push: Not set
        .... .0.. = Reset: Not set
        .... ..1. = Syn: Set
        .... ...0 = Fin: Not set
    Window size: 5840
    Checksum: 0x0b2e [correct]
        [Good Checksum: True]
        [Bad Checksum: False]
    Options: (16 bytes)
        Maximum segment size: 1460 bytes
        SACK permitted
        Timestamps: TSval 1197740, TSecr 0
```

No. Time Source Destination Protocol Info
2 0.000105 10.0.5.22 10.0.5.11 TCP telnet > 50093 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 TSV=1203084 TSER=1197740

…

Transmission Control Protocol, Src Port: telnet (23), Dst Port: 50093 (50093), Seq: 0, Ack: 1, Len: 0
Source port: telnet (23)
Destination port: 50093 (50093)
Sequence number: 0 (relative sequence number)
Acknowledgement number: 1 (relative ack number)
Header length: 36 bytes
Flags: 0x12 (SYN, ACK)
0... .... = Congestion Window Reduced (CWR): Not set
.0.. .... = ECN-Echo: Not set
..0. .... = Urgent: Not set
...1 .... = Acknowledgment: Set
.... 0... = Push: Not set
.... .0.. = Reset: Not set
.... ..1. = Syn: Set
....0 = Fin: Not set
Window size: 5792
Checksum: 0xf93d [correct]
[Good Checksum: True]
[Bad Checksum: False]
Options: (16 bytes)
Maximum segment size: 1460 bytes
SACK permitted
Timestamps: TSval 1203084, TSecr 1197740
[SEQ/ACK analysis]
[This is an ACK to the segment in frame: 1]
[The RTT to ACK the segment was: 0.000105000 seconds]

No. Time Source Destination Protocol Info
3 0.000053 10.0.5.11 10.0.5.22 TCP 50093 > telnet [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSV=1197740 TSER=1203084

…

Transmission Control Protocol, Src Port: 50093 (50093), Dst Port: telnet (23), Seq: 1, Ack: 1, Len: 0
Source port: 50093 (50093)
Destination port: telnet (23)
Sequence number: 1 (relative sequence number)
Acknowledgement number: 1 (relative ack number)
Header length: 32 bytes
Flags: 0x10 (ACK)
0... .... = Congestion Window Reduced (CWR): Not set

.0.. .... = ECN-Echo: Not set
..0. .... = Urgent: Not set
...1 .... = Acknowledgment: Set
.... 0... = Push: Not set
.... .0.. = Reset: Not set
.... ..0. = Syn: Not set
....0 = Fin: Not set
Window size: 5840
Checksum: 0x13cc [correct]
[Good Checksum: True]
[Bad Checksum: False]
Options: (12 bytes)
NOP
NOP
Timestamps: TSval 1197740, TSecr 1203084
[SEQ/ACK analysis]
[This is an ACK to the segment in frame: 2]
[The RTT to ACK the segment was: -0.000052000 seconds]

4.2 During the connection setup, the TCP client and TCP server tell each other the first sequence number they will use for data transmission. What is the initial sequence number of the TCP client and the TCP server?

Ans: The initial sequence numbers are both 0 for client and server. The sequence numbers are also highlighted in the header information in section 4.1.

4.3 Identify the first packet that contains application data? What is the sequence number used in the first byte of application data sent from the TCP client to the TCP server?

Ans: The first packet that contains application data is Frame 4. The sequence number is 1. Here is the telnet data captured by ethereal, the sequence number are highlighted.

No. Time Source Destination Protocol Info
4 0.010490 10.0.5.11 10.0.5.22 TELNET Telnet Data …
...
Transmission Control Protocol, Src Port: 50093 (50093), Dst Port: telnet (23), Seq: 1, Ack: 1, Len: 33
Source port: 50093 (50093)
Destination port: telnet (23)
Sequence number: 1 (relative sequence number)
[Next sequence number: 34 (relative sequence number)]
Acknowledgement number: 1 (relative ack number)
Header length: 32 bytes

Flags: 0x18 (PSH, ACK)
0... .... = Congestion Window Reduced (CWR): Not set
.0.. .... = ECN-Echo: Not set
..0. .... = Urgent: Not set
...1 .... = Acknowledgment: Set
.... 1... = Push: Set
.... .0.. = Reset: Not set
.... ..0. = Syn: Not set
....0 = Fin: Not set
Window size: 5840
Checksum: 0x722e [correct]
[Good Checksum: True]
[Bad Checksum: False]
Options: (12 bytes)
NOP
NOP
Timestamps: TSval 1197743, TSecr 1203084
Telnet
Command: Do Encryption Option
Command: Will Encryption Option
Command: Do Suppress Go Ahead
Command: Will Terminal Type
Command: Will Negotiate About Window Size
Command: Will Terminal Speed
Command: Will Remote Flow Control
Command: Will Linemode
Command: Will New Environment Option
Command: Do Status
Command: Will X Display Location

4.4 The TCP client and TCP server exchange window sizes to get the maximum amount of data that the other side can sent at any time. Determine the values of the window sizes for the TCP client and the TCP server.

Ans: The client windows size is 5840. The server window size is 5792. The client windows size is highlighted in the TCP header attached in section 4.3. Here is the telnet data from the server, which shows the window size of server in the TCP header.

No. Time Source Destination Protocol Info
5 0.010523 10.0.5.22 10.0.5.11 TCP telnet > 50093 [ACK] Seq=1 Ack=34 **Win=5792** Len=0 TSV=1203087 TSER=1197743
...

Transmission Control Protocol, Src Port: telnet (23), Dst Port: 50093 (50093), Seq: 1, Ack: 34, Len: 0
Source port: telnet (23)
Destination port: 50093 (50093)
Sequence number: 1 (relative sequence number)
Acknowledgement number: 34 (relative ack number)
Header length: 32 bytes
Flags: 0x10 (ACK)
0... .... = Congestion Window Reduced (CWR): Not set
.0.. .... = ECN-Echo: Not set
..0. .... = Urgent: Not set
...1 .... = Acknowledgment: Set
.... 0... = Push: Not set
.... .0.. = Reset: Not set
.... ..0. = Syn: Not set
....0 = Fin: Not set
Window size: 5792
Checksum: 0x13d5 [correct]
[Good Checksum: True]
[Bad Checksum: False]
Options: (12 bytes)
NOP
NOP
Timestamps: TSval 1203087, TSecr 1197743
[SEQ/ACK analysis]
[This is an ACK to the segment in frame: 4]
[The RTT to ACK the segment was: 0.000033000 seconds]

4.5 What is the MSS value that is negotiated between the TCP client and the TCP server?

Ans: From the TCP header in the previous section, the MSS is 1460 bytes, which is what we expected. Since the MTU is set to 1500 bytes currently, the MSS is equal to 1500 - 40 (TCP + IP header) = 1460.

4.6 How long does it take to open a TCP connection?

Ans: The time to open a TCP connection is 0.000053s. I got the information by observing the first 3 frames for link establishment from ethereal. The 3 frames information are also attached in the previous section 4.1.

4.7 Identify the packets that are involved in closing the TCP connection. Which flags are set in these packets? Explain how these flags are interpreted by the receiving TCP server or TCP client.

Ans:

1st packet [FIN]: The client sends a FIN request to terminate connection with FIN bit set to 1.
2nd packet [FIN, ACK]: The server received the FIN request, acknowledge it and sends back a FIN request to the client as well.
3rd packet: [ACK]: Client received the FIN request from server and sends back an ACK. Then, the TCP connection terminates.

4.8 Describe how the closing is different from step 4.

Ans: Because the client is not sending data for a long period, the server decided to actively terminates the connection. In the step 4, the termination is initiated by the client but not server.

4.9. How long does the telnet wait until it closes the TCP connection?

Ans: Telnet waits 1 minutes.

4B

4.10. How often does the TCP client try to establish a connection? How much time elapses between repeated attempts to open a connection?

Ans: 6 attempts were observed.
Attempt 1: time = 0
Attempt 2: time = 3.000459
Attempt 3: time = 8.999580
Attempt 4: time = 20.997821
Attempt 5: time = 44.994305
Attempt 6: time = 92.988564

Output:

```
No.  Time      Source        Destination   Protocol Info
   1 0.000000  10.0.5.11     10.0.5.100    TCP      40419 > telnet [SYN] Seq=0 Len=0
MSS=1460 TSV=1269402 TSER=0
No.  Time      Source        Destination   Protocol Info
   2 3.000459  10.0.5.11     10.0.5.100    TCP      40419 > telnet [SYN] Seq=0 Len=0
MSS=1460 TSV=1270152 TSER=0
No.  Time      Source        Destination   Protocol Info
   3 8.999580  10.0.5.11     10.0.5.100    TCP      40419 > telnet [SYN] Seq=0 Len=0
MSS=1460 TSV=1271652 TSER=0
No.  Time      Source        Destination   Protocol Info
   4 20.997821 10.0.5.11     10.0.5.100    TCP      40419 > telnet [SYN] Seq=0 Len=0
MSS=1460 TSV=1274652 TSER=0
No.  Time      Source        Destination   Protocol Info
```

5 44.994305 10.0.5.11 10.0.5.100 TCP 40419 > telnet [SYN] Seq=0 Len=0 MSS=1460 TSV=1280652 TSER=0

No. Time Source Destination Protocol Info

5 92.988564 10.0.5.11 10.0.5.100 TCP 40419 > telnet [SYN] Seq=0 Len=0 MSS=1460 TSV=1286652 TSER=0

4.11. Does the TCP client terminate or reset the connection when it gives up trying to establish a connection?

Ans: Yes, TCP client resets the connection and a [RST] packet was captured.

4.12. Why does this experiment require setting a static ARP table entry?

Ans: A static ARP table entry is required because an non-existing IP address is required.

4C

4.13 How does TCP at the remote host closes this connection? How long does the process of ending the connection take?

Ans: The remote host closed and reset the connection by sending [RST, ACK] packet.
The time for the ending process can be observed by the [RST, ACK] and the last [SYN]. The process will take 0.000134 - 0.000061 = 0.000073 seconds. Here is the [RST, ACK] and last [SYN] captured by ethereal:

No. Time Source Destination Protocol Info

3 0.000061 10.0.5.11 10.0.5.22 TCP 39114 > http [SYN] Seq=0 Len=0 MSS=1460 TSV=1315337 TSER=0

No. Time Source Destination Protocol Info

4 0.000134 10.0.5.22 10.0.5.11 TCP http > 39114 [RST, ACK] Seq=0 Ack=1 Win=0 Len=0