ECE461 Lab 5 Lab Report

Xiaoyang Guo(999578513)

Mingqi Hou(999767676)

Xiaolong Zhang(997969104)

Exercise 1(B)

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?**

There are 18 packets exchanged in the data transfer. 16 packets are transmitted for UDP datagram. The size of the UDP payload of these packets is 1024.

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

The total number of bytes transmitted is 10*1066+46*6=10936. The amount of application data transmitted is 10*1024=10240 bytes.

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

The source address, destination address, source port, and destination port do not change.

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

Source port is 32770. It's randomly assigned in between a certain range.

Exercise 1(C)

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

There are total 22 packets exchanged. The sizes of TCP segments are 66, 70, 1090 and 1514 bytes.

The following is the selected captured ethereal data.

```
No.  Time     Source           Destination       Protocol Info
1    0.000000 10.0.5.11        10.0.5.22         TCP
46827 > krb524 [SYN] Seq=0 Len=0 MSS=1460 TSV=302456 TSER=0
......
No.  Time     Source           Destination       Protocol Info
2 0.000064    10.0.5.22        10.0.5.11         TCP
krb524 > 46827 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460
TSV=302495 TSER=302456
Frame 2 (70 bytes on wire, 70 bytes captured)
......
No.  Time     Source           Destination       Protocol Info
3    0.000090 10.0.5.11        10.0.5.22         TCP      46827 >
krb524 [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSV=302456 TSER=302495
Frame 3 (66 bytes on wire, 66 bytes captured)
......
```

```
No.  Time     Source           Destination       Protocol Info
4    0.010530 10.0.5.11        10.0.5.22         TCP
46827 > krb524 [PSH, ACK] Seq=1 Ack=1 Win=5840 Len=1024 TSV=302459
TSER=302495
Frame 4 (1090 bytes on wire, 1090 bytes captured)
......
Data (1024 bytes)
No.  Time     Source           Destination       Protocol Info
5    0.010582 10.0.5.11        10.0.5.22         TCP
46827 > krb524 [PSH, ACK] Seq=1025 Ack=1 Win=5840 Len=1024
TSV=302459 TSER=302495
Frame 5 (1090 bytes on wire, 1090 bytes captured)
......
Data (1024 bytes)
No.  Time     Source           Destination       Protocol Info
6    0.010598 10.0.5.11        10.0.5.22         TCP
46827 > krb524 [PSH, ACK] Seq=2049 Ack=1 Win=5840 Len=1024
TSV=302459 TSER=302495
Frame 6 (1090 bytes on wire, 1090 bytes captured)
......
Data (1024 bytes)
No.  Time     Source           Destination       Protocol Info
7    0.010843 10.0.5.22        10.0.5.11         TCP
krb524 > 46827 [ACK] Seq=1 Ack=1025 Win=7168 Len=0 TSV=302498
TSER=302459
Frame 7 (66 bytes on wire, 66 bytes captured)
......
No.  Time     Source           Destination       Protocol Info
8    0.010873 10.0.5.11        10.0.5.22         TCP
46827 > krb524 [ACK] Seq=3073 Ack=1 Win=5840 Len=1448 TSV=302459
TSER=302498
Frame 8 (1514 bytes on wire, 1514 bytes captured)
......
Data (1448 bytes)
No.  Time     Source           Destination       Protocol Info
9    0.010876 10.0.5.11        10.0.5.22         TCP
46827 > krb524 [ACK] Seq=4521 Ack=1 Win=5840 Len=1448 TSV=302459
TSER=302498
Frame 9 (1514 bytes on wire, 1514 bytes captured)
......
Data (1448 bytes)
No.  Time     Source           Destination       Protocol Info
10   0.011023 10.0.5.22        10.0.5.11         TCP
krb524 > 46827 [ACK] Seq=1 Ack=2049 Win=9216 Len=0 TSV=302498
TSER=302459
Frame 10 (66 bytes on wire, 66 bytes captured)
```

```
No.  Time     Source           Destination       Protocol Info
11   0.011046 10.0.5.11        10.0.5.22         TCP
46827 > krb524 [ACK] Seq=5969 Ack=1 Win=5840 Len=1448 TSV=302459
TSER=302498
Frame 11 (1514 bytes on wire, 1514 bytes captured)
......
Data (1448 bytes)
No.  Time     Source           Destination       Protocol Info
12   0.011049 10.0.5.11        10.0.5.22         TCP
46827 > krb524 [ACK] Seq=7417 Ack=1 Win=5840 Len=1448 TSV=302459
TSER=302498
Frame 12 (1514 bytes on wire, 1514 bytes captured)
......
Data (1448 bytes)
No.  Time     Source           Destination       Protocol Info
13   0.011026 10.0.5.22        10.0.5.11         TCP
krb524 > 46827 [ACK] Seq=1 Ack=3073 Win=11264 Len=0 TSV=302498
TSER=302459
Frame 13 (66 bytes on wire, 66 bytes captured)
No.  Time     Source           Destination       Protocol Info
14   0.011055 10.0.5.11        10.0.5.22         TCP
46827 > krb524 [PSH, ACK] Seq=8865 Ack=1 Win=5840 Len=1376
TSV=302459 TSER=302498
Frame 14 (1442 bytes on wire, 1442 bytes captured)
......
Data (1376 bytes)
No.  Time     Source           Destination       Protocol Info
15   0.011439 10.0.5.22        10.0.5.11         TCP
krb524 > 46827 [ACK] Seq=1 Ack=4521 Win=14480 Len=0 TSV=302498
TSER=302459
Frame 15 (66 bytes on wire, 66 bytes captured)
No.  Time     Source           Destination       Protocol Info
16   0.011440 10.0.5.22        10.0.5.11         TCP
krb524 > 46827 [ACK] Seq=1 Ack=5969 Win=17376 Len=0 TSV=302498
TSER=302459
Frame 16 (66 bytes on wire, 66 bytes captured)
No.  Time     Source           Destination       Protocol Info
17   0.011568 10.0.5.22        10.0.5.11         TCP
krb524 > 46827 [ACK] Seq=1 Ack=7417 Win=20272 Len=0 TSV=302498
TSER=302459
Frame 17 (66 bytes on wire, 66 bytes captured)
No.  Time     Source           Destination       Protocol Info
18   0.011613 10.0.5.11        10.0.5.22         TCP
46827 > krb524 [FIN, ACK] Seq=10241 Ack=1 Win=5840 Len=0
TSV=302459 TSER=302498
Frame 18 (66 bytes on wire, 66 bytes captured)
```

```
No.  Time     Source           Destination       Protocol Info
19   0.011723 10.0.5.22        10.0.5.11         TCP
krb524 > 46827 [ACK] Seq=1 Ack=8865 Win=23168 Len=0 TSV=302498
TSER=302459
Frame 19 (66 bytes on wire, 66 bytes captured)
No.  Time     Source           Destination       Protocol Info
20   0.011739 10.0.5.22        10.0.5.11         TCP
krb524 > 46827 [ACK] Seq=1 Ack=10241 Win=26064 Len=0 TSV=302498
TSER=302459
Frame 20 (66 bytes on wire, 66 bytes captured)
No.  Time     Source           Destination       Protocol Info
21   0.011943 10.0.5.22        10.0.5.11         TCP
krb524 > 46827 [FIN, ACK] Seq=1 Ack=10242 Win=26064 Len=0
TSV=302498 TSER=302459
Frame 21 (66 bytes on wire, 66 bytes captured)
No.  Time     Source           Destination       Protocol Info
22 0.011963   10.0.5.11        10.0.5.22         TCP
46827 > krb524 [ACK] Seq=10242 Ack=2 Win=5840 Len=0 TSV=302459
TSER=302498
Frame 22 (66 bytes on wire, 66 bytes captured)
```

2. **What is the range of the sequence numbers?**

As shown in the data above, the range is from 0 to 10242.

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

12 packets are transmitted by PC1 (10.0.5.11).10 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?**

There are 14 packets that do not carry a payload.

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

The total number of bytes transmitted is 11712 bytes. The amount of application data transmitted is 10772 bytes

6. **Inspect the fields in the TCP headers. Which packets contain flags in the TCP header? Why?**

All of the packets contain flags, since flags are used to identify the purpose. The following flags are observed: [SYN], [SYN, ACK], [ACK], [FIN], [FIN, ACK], [PSH, ACK]

Lab report

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

TCP transmitted more data than UDP as 11712 (TCP) > 10936 (UDP).

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.**

The biggest UDP datagram is 1060 bytes with 1024 bytes payload. The biggest TCP datagram is 1514 bytes with 1448 bytes payload. TCP sends data in streams of segments, which is usually larger than UDP packets.

Exercise 3(A)

1. **Determine the exact UDP datagram size at which fragment occurs.**

The exact size of UDP Datagram at which fragmentation occurs is 1514 bytes, the data size is 1480 bytes.

The following is the selected data captured by *ethereal*.

```
No.   Time       Source          Destination           Protocol
Info
22    15.955700  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: Nov  7, 2016 16:28:33.235047000
    Packet Length: 1514 bytes
    Capture Length: 1514 bytes
      .....
    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: 0x1009 [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: 23
Data (1480 bytes)
```

2. **Determine the maximum size of a UDP datagram that the system can send and receive, regardless of fragmentation.**

The practical limit for the UDP size which is imposed by the underlying IPv4 protocol is 65,507 bytes (65535 − 8 byte UDP header − 20 byte IP header). When we try to send 65508 bytes, we received the following error message.

```
nttcp-t: socket
nttcp-t: IO: Message too long
errno=90
ttcp-t: buflen=65508, nbuf=10, align=16384/0, port=4444
udp  -> 10.0.2.22
```

Lab Report

Determine the values of the fields in the IP header that are used for fragmentation.
In IP header we have the following fields.

Identification	0	DF	MF	Fragment offset

Before fragmentation occurs the fields are shown as the following.

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

When fragmentation occurs the fields are shown as the following.

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

Exercise 3 (B)

1. **Do you observe fragmentation? Explain your observation.**

No, fragmentations are not observed. TCP segment size is negotiated during three way handshake stage between the source and destination host.

2. **If you observe ICMP error messages, decide how they are used for path MTU discovery.**

No, ICMP error messages were not observed. ICMP messages are used to tell senders to reduce the segment size but the size didn't exceed the MTU.

Exercise 4 (A)

Step 3

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.**

The following is the selected data captured by *ethereal*.

```
No.   Time      Source              Destination         Protocol
Info
3     0.000069  10.0.5.11           10.0.5.22           TCP
48955 > telnet [SYN] Seq=0 Len=0 MSS=1460 TSV=585294 TSER=0
No.   Time      Source              Destination         Protocol
Info
```

```
4     0.000154  10.0.5.22          10.0.5.11          TCP
telnet > 48955 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460
TSV=585339 TSER=585294
No.   Time      Source             Destination        Protocol
Info
5     0.000183  10.0.5.11          10.0.5.22          TCP
48955 > telnet [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSV=585295
TSER=585339
```

1st packet is [SYN]. The client sends request to initialize connection with SYN bit set to 1.
2nd packet is [SYN + ACK]. The server accepts the connection, it will send back a confirm response with both SYN and ACK bits set to 1.
3rd packet is [ACK]. After the client receives the server's response, it sends back an ACK response.

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 are the initial sequence numbers of the TCP client and the TCP server?**

The initial sequence number is 0.

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?**

The first packet that contains application data is Frame 6 captured by ethereal. The sequence number is 1.

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

The client windows size is 5840. The server window size is 5792.

5. **What is the MSS value that is negotiated between the TCP client and the TCP server?**

MSS = 1460 bytes.

6. **How long does it take to open a TCP connection?**

The time taken for the three-way handshake finish, which is the time used for TCP connection, is 0.000183 seconds.

Step 4

1. 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.

1st packet: [FIN]: The client sends request to terminate connection with FIN bit set to 1.
2nd packet: [FIN, ACK]: The server acknowledges the FIN request, and sends back a FIN request to client.
3rd packet: [ACK]: Client sends out ACK.
Then TCP connection terminates.

Step 5

1. **Describe how the closing is different from step 4.**

In this case, the server actively terminates the connection.

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

Telnet waits 60 seconds until it closes the TCP connection.

Exercise 4 (B)

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

6 attempts were observed. The time between consecutive attempts increases.

Attempt 1: time = 0
Attempt 2: time = 2.999487
Attempt 3: time = 8.998497
Attempt 4: time = 20.996503
Attempt 5: time = 44.996525
Attempt 6: time = 92.988573

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

TCP client resets the connection. A packet with [RST] was observed.

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

A static ARP table entry is required as a non-existing IP address is needed.

Exercise 4 (C)

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

The remote host closes the connection by resetting the connection with [RST, ACK]. The time difference the [RST, ACK] and the last [SYN] was 0.000146 - 0.000074 = 0.000072 seconds

Exercise 5 (B)

Step3

1. **Observe the number of packets exchanged between the Router 1 and Router 2 for each keystroke. Describe the payload of the packets. Use your knowledge of delayed acknowledgments to explain the sequence of segment transmissions. Explain why you should see 4 packets and explain why you do not see 4 packets per typed character.**

There are 3 packets exchanged for each keystroke. The first packet is the telnet data packet sent from client. The second packet is a telnet data packet with flag [PSH, ACK] sent from server. The third packet is an ACK packet from client to serve. We should see 4 packets as there are 2 data packets. Thus 2 corresponding ACK packets should be sent. We did not see 4 packets as the ACK for the first data packets is piggybacked on the second data packet, which had the flag [PSH, ACK].

2. **When the TCP client receives the echo of a character, it waits a certain time before sending the ACK. Why does the TCP client delay? How long is this delay? How much does the delay vary?**

The TCP client delays to check if there is any data to send so that the ACK can be piggybacked on the next segment. The delay is about 0.00004 seconds and varies by 0.00001 seconds.

3. **What is the time delay associated with the transmission of ACKs from the Telnet server on Router 2?**

The time delay we observed is 0.0007 seconds.

4. **Which flags, if any, are set in the TCP segments that carry typed characters as payload? Explain the meaning of these flags.**

The flag [PSH, ACK] are set. PSH indicates that the packet carries data and ASK indicates acknowledgement.

5. **Why do segments that have an empty payload carry a sequence number? Why does this not result in confusion at the TCP receiver?**

For empty payload segments, ACK packets did not consume any sequence number. The same sequence number was used repeatedly. The reason why there is no confusion is that the sequence number does not change before and after each empty payload segment.

6. **What is the window size that is advertised by the Telnet client and the Telnet server? How does the value of the window size field vary as the connection progresses?**

The client window size is 5840. The server window size is 5792. The value doesn't change as the connection progresses.

Step 4

Do you observe a difference in the transmission of segment payloads and ACKs?

We observed that there is not always a corresponding ACK segment for every payload segment sent. Many of the ACK segments are piggybacked in the next payload segment. In addition, cumulative acknowledgment can be observed.

Exercise 5 (C)

1. **Observe the number of packets exchanged between Router1 and Router2 for each keystroke. Observe how the transmission of packets changes when you type characters more quickly.**

There are always 3 segments for each keystroke. As we typed more quickly, we observed that one segment could carry more than one keystroke.

2. **Do you observe delayed acknowledgments? Why is the outcome expected?**

Yes. The outcome is expected as the ACKs were piggybacked to next segments.

3. **If you type very quickly (i.e., if you hold a key down), you should observe that multiple characters are transmitted in the payload of a segment. Explain this outcome.**

According to the Nagle's algorithm, the messages are buffered and sent at once. TCP sends one character in data packet and waits for the ACK. In the meantime, all the subsequent bytes (as the key is hold down) are buffered. When TCP receives the ACK, it sends a larger packet with all the buffered data. That's why we can see multiple characters sent in the payload of a segment.

Exercise 6 (A)

1. **How frequently does the receiver send ACKs? Is there an ACK sent for each TCP segment or less often? Can you determine the rule used by TCP to send ACKs? Can you explain this rule?**

An ACK is sent for about every two segments from the receiver on average. The rule is cumulative acknowledgment. TCP receiver can acknowledge more than one segments in a single ACK.

2. **How much data (measured in bytes) does the receiver acknowledge in a typical ACK? What is the most data that is acknowledged in a single ACK?**

A typical ACK usually acknowledges 2 to 3 kilobytes of data. The most data we observed in an ACK is 15324.

3. **What is the range of the window sizes advertised by the receiver? How does the window size vary during the lifetime of the TCP connection?**

The first packet sent from the receiver has the window size set to 5792.

```
No.  Time      Source          Destination        Protocol Info
2    0.000064  10.0.5.22       10.0.5.11          TCP
krb524 > 55843 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460
TSV=1886239 TSER=1886176
```

The maximum window size we observe is 65535。

```
No.  Time      Source          Destination        Protocol Info
585  0.076573  10.0.5.22       10.0.5.11          TCP
krb524 > 55843 [FIN, ACK] Seq=1 Ack=500002 Win=65535 Len=0
TSV=1886258 TSER=1886194
```

The window size ranges from 5792 to 65535. The window size increases during the lifetime of the TCP connection.

4. **Select an arbitrary ACK packet in ethereal sent by PC2 to PC1. Locate the acknowledgment number in the TCP header. Now relate this ACK to a segment sent by PC1. Identify this segment in the ethereal output. How long did it take from the transmission of the segment until the ACK arrives at PC1?**

Time taken from the transmission of the segment until the ACK arrives at PC1 is 0.001928 seconds. The following is the selected data captured by *ethereal*.

```
No.  Time      Source      Destination      Protocol Info
45   0.023891  10.0.5.11   10.0.5.22        TCP
45840> krb524  [ACK]  Seq=45810 Ack=1 Win=5840 Len=1000 TSV=2008215
TSER=2032564
......
No.  Time      Source      Destination      Protocol Info
48   0.021963  10.0.5.22   10.0.5.11        TCP
krb524 > 45840  [ACK] Seq=1  Ack=45810 Win=7000 Len=0 TSV=2032565
TSER=2008215
```

5. **Determine whether or not the TCP sender generally transmits the maximum amount of data allowed by the advertised Window. Explain.**

No. The reason is that the amount of data transmitted is limited by the MSS to avoid congestion.

6. **When the ttcp sender has transmitted all this data, it closes the connection, but acknowledgements from PC2 still trickle in. What does PC2 do when it has sent all ACKs?**

After all the ACKs sent, PC2 sent a [FIN, ACK] to PC1 for termination. PC1 sends back an ACK to that packet. The following is the selected data captured by *ethereal*.

```
No.  Time      Source      Destination      Protocol Info
585  0.076573  10.0.5.22   10.0.5.11        TCP
krb524 > 55843 [FIN, ACK] Seq=1 Ack=500002 Win=65535 Len=0
TSV=1886258 TSER=1886194
......
No.  Time      Source      Destination      Protocol Info
586  0.076597  10.0.5.11   10.0.5.22        TCP
55843 > krb524 [ACK] Seq=500002 Ack=2 Win=5840 Len=0 TSV=1886196
TSER=1886258
```

Exercise 6(B)

1. **How does the pattern of data segments and ACK change, as compared to the fast Ethernet link?**

ACK is sent for every one or two TCP data segments. In the fast Ethernet link,a number of data TCP segments are sent, followed by a number of ACKs. The following is the selected data captured by *ethereal*.

```
No.  Time        Source         Destination      Protocol Info
33   68.256046   10.0.2.22      10.0.1.11        TCP
krb524 > 49303 [ACK] Seq=1 Ack=4897 Win=14480 Len=0 TSV=2100402
TSER=2105799
......
No.  Time        Source         Destination      Protocol Info
36   68.353148   10.0.2.22      10.0.1.11        TCP
```

```
krb524 > 49303 [ACK] Seq=1 Ack=6345 Win=17376 Len=0 TSV=2100427
TSER=2105816
......
No.  Time        Source         Destination      Protocol Info
39   68.450385   10.0.2.22      10.0.1.11        TCP
krb524 > 49303 [ACK] Seq=1 Ack=7793 Win=20272 Len=0 TSV=2100451
TSER=2105816
```

2. **Does the frequency of ACKs change?**

ACKs are sent more frequently.

3. **Is the range of window sizes advertised by the receiver different from those in Exercise 6(A)?**

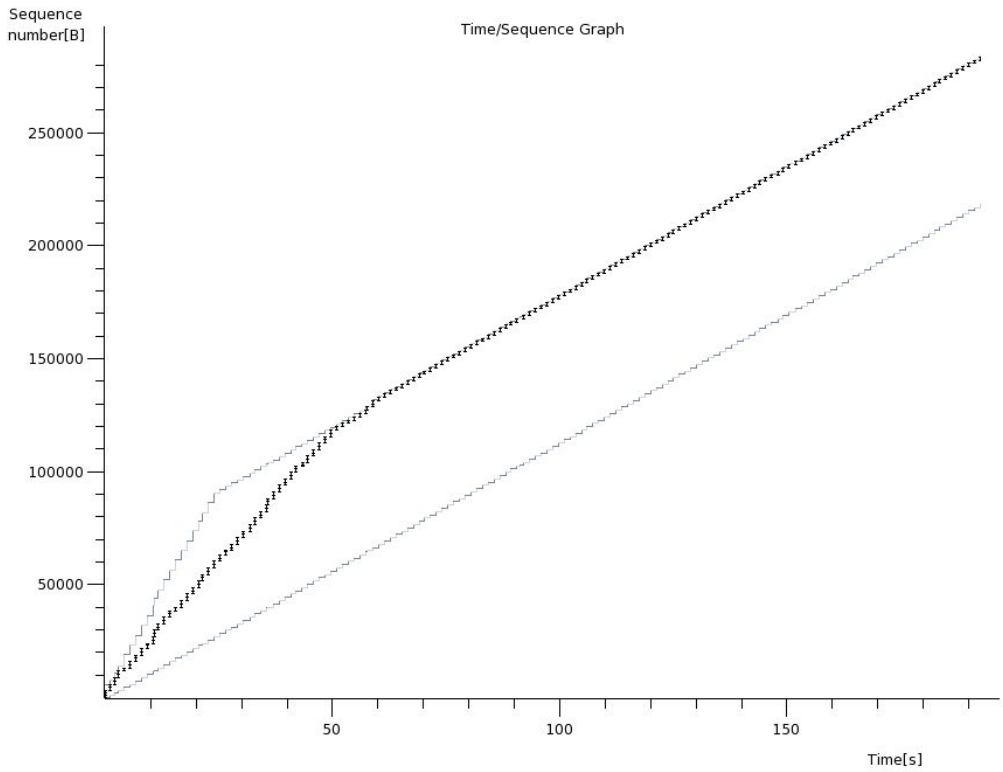
No. The maximum window size is 65535, the same as that of Ex 6(A). The following is the selected data captured by *ethereal*.

```
No.  Time        Source         Destination      Protocol Info
211  74.720378   10.0.2.22      10.0.1.11        TCP
krb524 > 49303 [ACK] Seq=1 Ack=101001 Win=65535 Len=0 TSV=2102019
TSER=2106620
```

4. **Does the TCP sender generally transmit the maximum amount of data allowed by the advertised window? Explain your answer.**

No. According to our observation, TCP does not generally transmit the maximum amount of data allowed. The reason is that the amount of data transmitted is often limited by the MSS to avoid congestion.

Exercise 6(C)



Exercise 7(A)

1. **When the connection is created, do the TCP sender and TCP receiver negotiate to permit SACKs? Describe the process of the negotiation.**

Yes. They negotiate on the permission of SACKs during the three-way handshake in the SACK option.

2. **Observe the time instants when retransmissions take place. How many packets are transmitted at one time?**

One packet was transmitted at a time.

3. **Try derive the algorithm that sets the time when a packet is retransmitted. Is there a maximum time interval between retransmissions?**

According to Karn's algorithm, $RTO_{n+1} = \min (2\ RTO_n , 64)$. The maximum time interval for retransmission is 64 seconds.

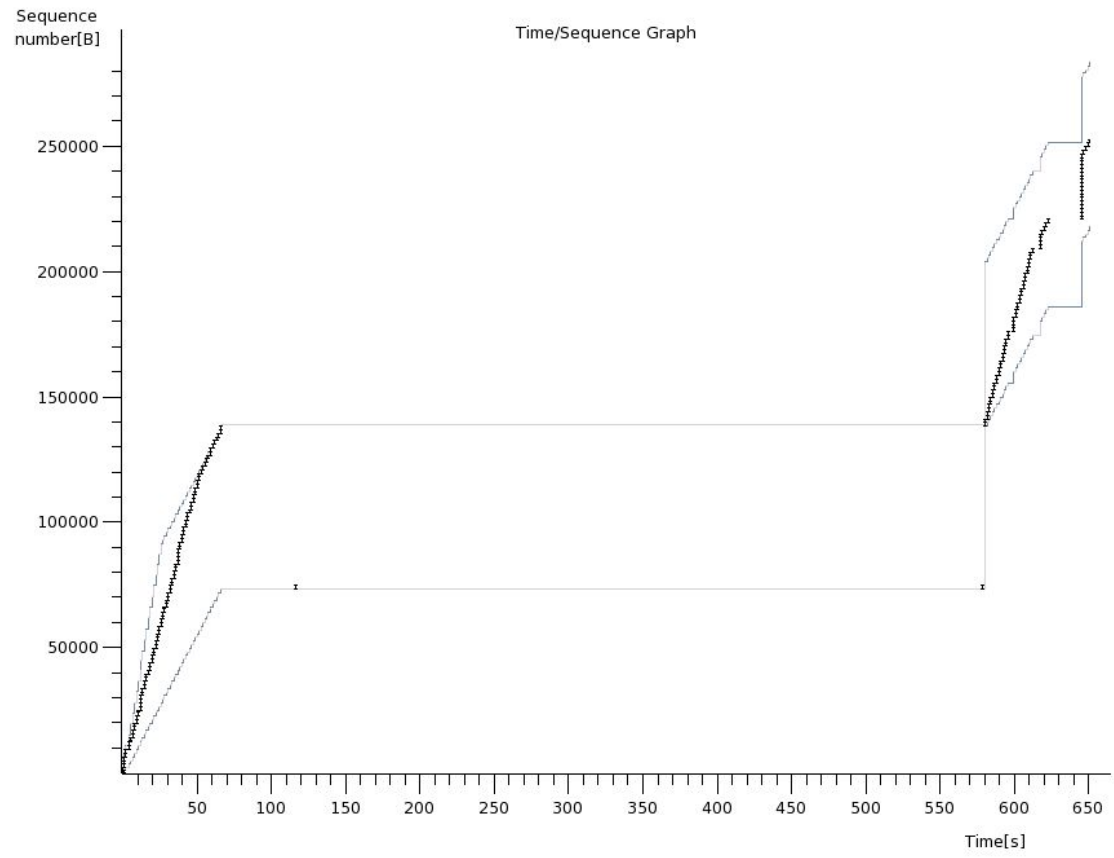
4. **After how many retransmissions, if at all, does the TCP sender stop to retransmit the segment? Describe your observations.**

As we marked in the graph, it can be clearly seen that 2 attempts were made.

5. **Are the retransmissions different from those in Step 5? Specifically, do you observe fast retransmits and SACKs?**

Yes. For this case, the retransmission was successful and no SACKs were allowed.

Step 8



Exercise 7(B)

1. **Describe the losses that occur in the graph when the ping command is started. Do losses occur in regular intervals or irregularly?**

Four packet losses were captured after the ping command. From the intervals shown in the graph, the losses occur regularly.

2. **From the graph, describe the size of the advertised window changes when the flooding ping is started.**

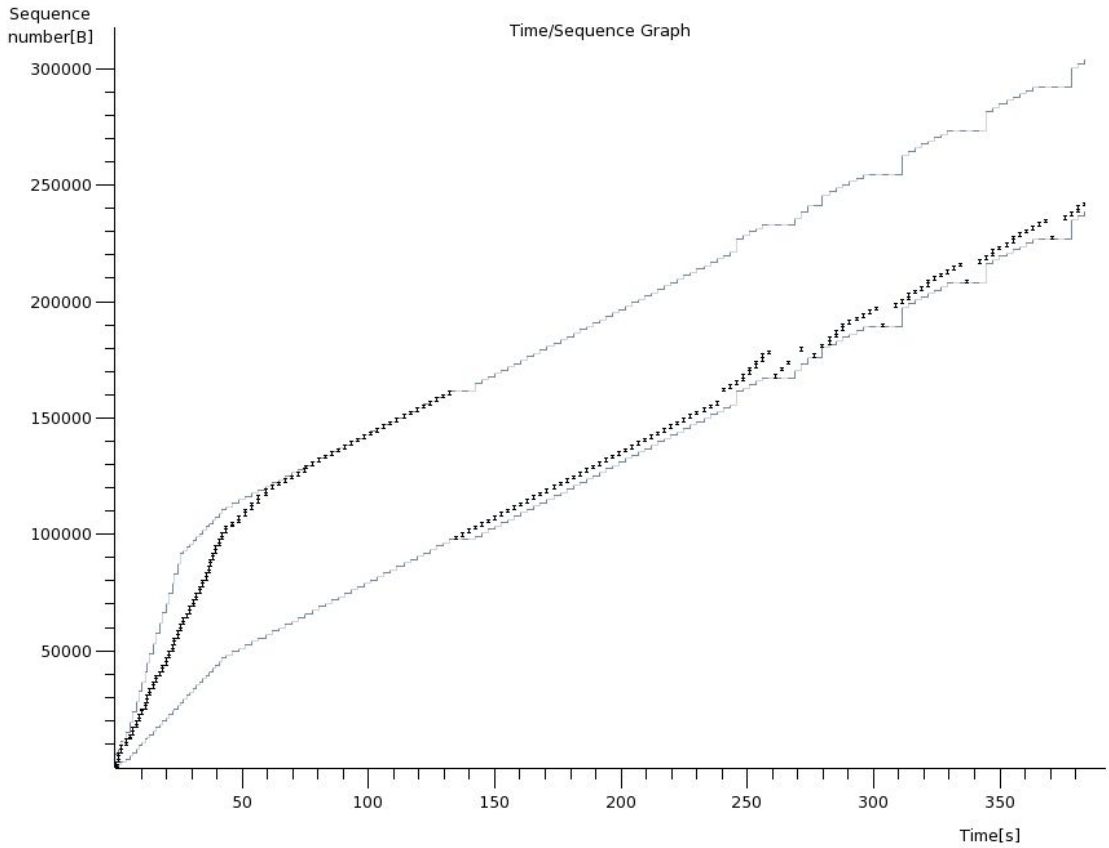
The window size was increasing until a loss occurred.

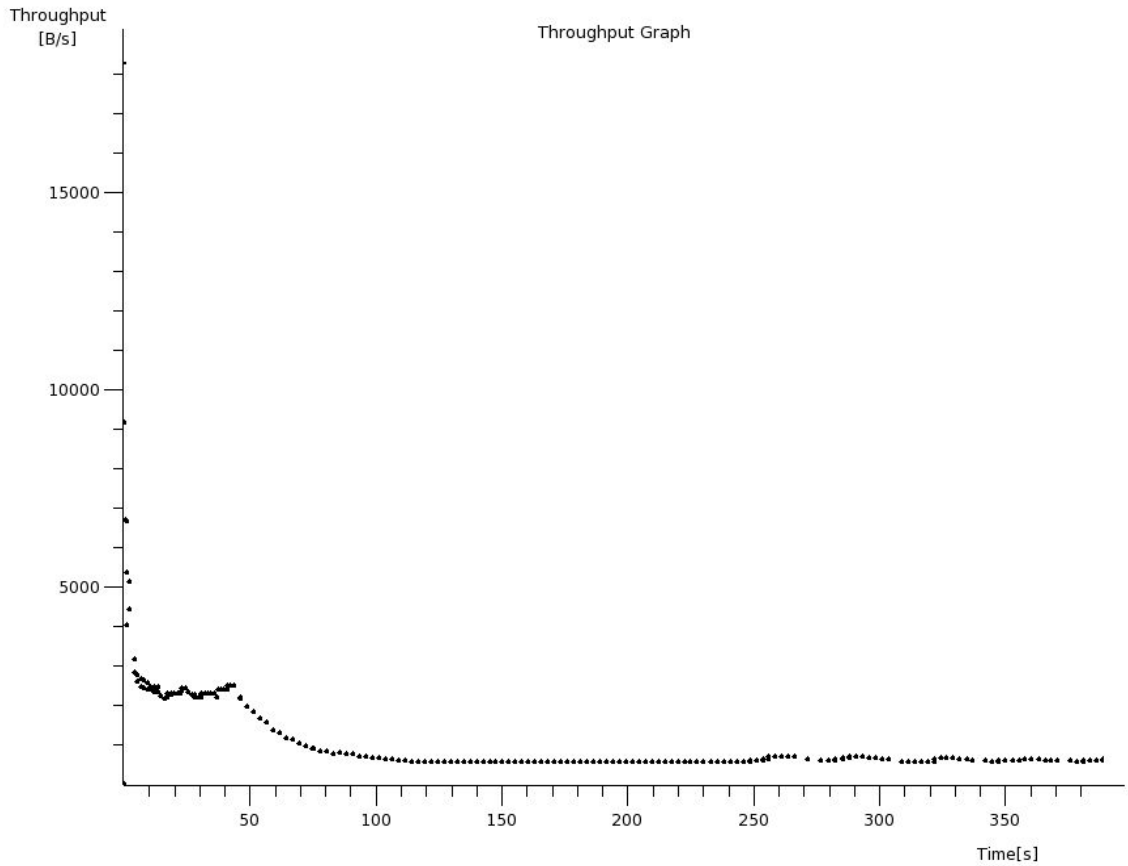
3. **Try to determine if retransmissions occur due to fast retransmit or due to timeouts of the retransmission timers. How can you determine which type of retransmissions you observe?**

The retransmissions are fast retransmits since the time intervals between transmissions are quite short.

4. **How does the throughput changes after the flood of pings is started?**

Throughput decreased after the flood of pings was started




Throughput
[B/s]
Throughput Graph
15000
10000
5000
50
100
150
200
250
300
350
Time[s]