TCP and OSPF Assignment
ELC 3080

Name	ID	Sec / BN
مجدي أحمد عباس عبد الحميد الابرق	9210899	3 / 36

Submitted To: Prof. Dr. Rania Osama

3rd Year – Spring 2024

Contents

1. TCP Lab

Each of the following sections describes an experiment based on the network topology provided by the **Project_OSPF_TCP.imn** file. In your report, divide the answers to correspond to one of these sections in exactly the same sequence and same name, for example use Question 2.1, Question 3.2 and so on.
The Introductory Lab explained and introduced most of the needed tools like iperf3, vtysh, wireshark, etc. If you have not gone through it, do not start working on the assignment below. It is better to start the network emulation once and run Wireshark captures. You can reset Wireshark captures from experiment to another.

1.1. Effect of TCP Window Size

- Set wireshark filter to display TCP packets only.
- Start an iperf3 server on node n11.
- Start an iperf3 client on node n7 connecting to server on node n11 for a duration of 40 seconds and reporting interval of 10. **Note that in iperf, the client is the node sending the traffic and the server simply receives and sends an ACK**.

1.1.1. Vary iperf3 window size from 1Kbytes to 6 Kbytes in increments of 1 Kbyte, then set it to 12, 16, 24, 32 Kbytes. Plot the average throughput and the average number of retransmissions as function of window size. Comment on the results and explain the zigzag behavior noticed for larger window sizes. Note that number of retransmissions is the 5th column in iperf3 default output & throughput is the 4th. You can plot data using MATLAB/python/any online graphing tool (Ex. https://chart-studio.plotly.com/create/#/). The window size is the last option in the iperf command (return to INTRO Lab).

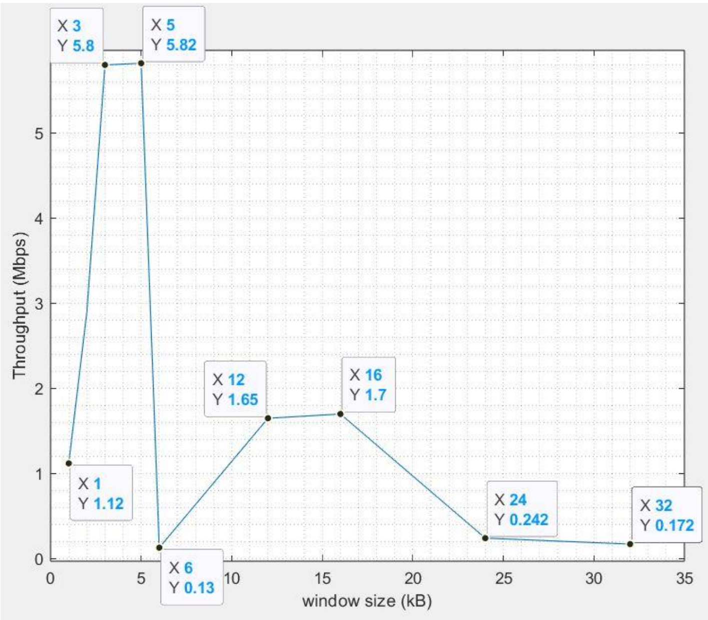


Figure 1: Throughput (Mbps) vs Window Size (kB)

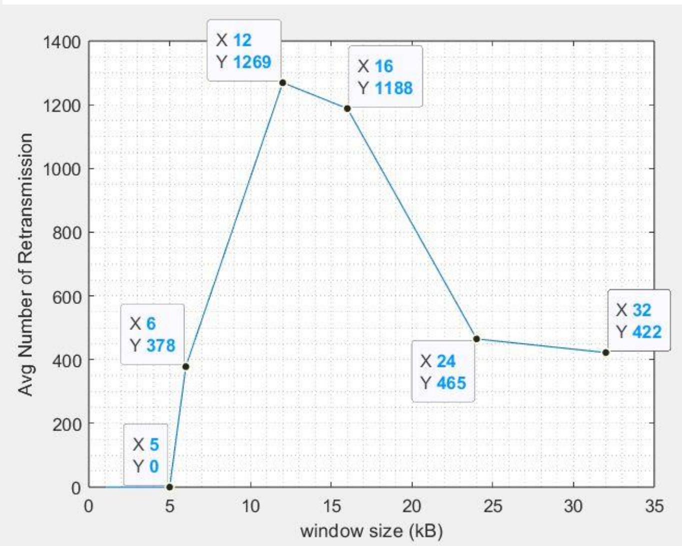


Figure 2: Avg Number of Retransmissions vs Window Size (kB)

Comment on your findings:

We observe from the throughput versus window size graph that there is a zigzag pattern as the window size increases from 1k to 5k. During this range, the number of retransmissions remains zero because the receiver buffer is not yet full. Consequently, as the window size increases, the throughput also rises until the receiver buffer becomes saturated at a window size of 5k. At the window size of 6k, the graph shows a sudden drop in throughput and an increase in retransmissions due to a significant number of packet losses. When this occurs, the transmitter identifies the issue and adjusts by allocating more space in the buffer. As a result, within a window size range between 6k and 12k, the throughput increases again along with a rise in retransmissions. However, if the window size exceeds 16k, the receiver buffer fills up once more, leading to packet losses and a subsequent decrease in throughput and decrease in number of retransmissions.

Table 1: Window Size Table Showing Throughput and Avg Number of Retransmissions Values

Figure 3: 1 kB Window Size

```
root@n7:/tmp/pycore.34469/n7.conf# iperf3 -c 10.0.13.20 -t 40 -i 10 -w 1k
Connecting to host 10.0.13.20, port 5201
[  4] local 10.0.12.20 port 38557 connected to 10.0.13.20 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-10.00  sec  1.34 MBytes  1.12 Mbits/sec    0   5.45 KBytes
[  4]  10.00-20.00  sec  1.33 MBytes  1.12 Mbits/sec    0   5.45 KBytes
[  4]  20.00-30.00  sec  1.33 MBytes  1.12 Mbits/sec    0   5.45 KBytes
[  4]  30.00-40.00  sec  1.33 MBytes  1.12 Mbits/sec    0   5.45 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-40.00  sec  5.33 MBytes  1.12 Mbits/sec    0             sender
[  4]   0.00-40.00  sec  5.33 MBytes  1.12 Mbits/sec                  receiver

iperf Done.
root@n7:/tmp/pycore.34469/n7.conf#
```

Throughput = 1.12 Mbps, Retransmissions = 0

Figure 4: 2 kB Window Size

```
root@n7:/tmp/pycore.34472/n7.conf# iperf3 -c 10.0.13.20 -t 40 -i 10 -w 2k
Connecting to host 10.0.13.20, port 5201
[  4] local 10.0.12.20 port 38560 connected to 10.0.13.20 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-10.00  sec  3.46 MBytes  2.90 Mbits/sec    0   7.07 KBytes
[  4]  10.00-20.00  sec  3.44 MBytes  2.89 Mbits/sec    0   7.07 KBytes
[  4]  20.00-30.00  sec  3.46 MBytes  2.90 Mbits/sec    0   7.07 KBytes
[  4]  30.00-40.00  sec  3.47 MBytes  2.91 Mbits/sec    0   7.07 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-40.00  sec  13.8 MBytes  2.90 Mbits/sec    0             sender
[  4]   0.00-40.00  sec  13.8 MBytes  2.90 Mbits/sec                  receiver

iperf Done.
root@n7:/tmp/pycore.34472/n7.conf#
```

Throughput = 2.9 Mbps, Retransmissions = 0

Figure 5: 3 kB Window Size

```
root@n7:/tmp/pycore.34475/n7.conf# iperf3 -c 10.0.13.20 -t 40 -i 10 -w 3k
Connecting to host 10.0.13.20, port 5201
[  4] local 10.0.12.20 port 38563 connected to 10.0.13.20 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-10.00  sec  6.86 MBytes  5.75 Mbits/sec    0   14.1 KBytes
[  4]  10.00-20.00  sec  6.93 MBytes  5.81 Mbits/sec    0   14.1 KBytes
[  4]  20.00-30.00  sec  6.93 MBytes  5.82 Mbits/sec    0   14.1 KBytes
[  4]  30.00-40.00  sec  6.92 MBytes  5.81 Mbits/sec    0   14.1 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-40.00  sec  27.6 MBytes  5.80 Mbits/sec    0             sender
[  4]   0.00-40.00  sec  27.6 MBytes  5.80 Mbits/sec                  receiver

iperf Done.
root@n7:/tmp/pycore.34475/n7.conf#
```

Throughput = 5.8 Mbps, Retransmissions = 0

Figure 6: 4 kB Window Size

```
root@n7:/tmp/pycore.34478/n7.conf# iperf3 -c 10.0.13.20 -t 40 -i 10 -w 4k
Connecting to host 10.0.13.20, port 5201
[  4] local 10.0.12.20 port 38566 connected to 10.0.13.20 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-10.00  sec  6.93 MBytes  5.82 Mbits/sec    0   14.1 KBytes
[  4]  10.00-20.00  sec  6.93 MBytes  5.81 Mbits/sec    0   14.1 KBytes
[  4]  20.00-30.00  sec  6.92 MBytes  5.80 Mbits/sec    0   14.1 KBytes
[  4]  30.00-40.00  sec  6.94 MBytes  5.82 Mbits/sec    0   14.1 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-40.00  sec  27.7 MBytes  5.81 Mbits/sec    0             sender
[  4]   0.00-40.00  sec  27.7 MBytes  5.81 Mbits/sec                  receiver

iperf Done.
root@n7:/tmp/pycore.34478/n7.conf#
```

Throughput = 5.81 Mbps, Retransmissions = 0

Figure 7: 5 kB Window Size

```
root@n7:/tmp/pycore.34481/n7.conf# iperf3 -c 10.0.13.20 -t 40 -i 10 -w 5k
Connecting to host 10.0.13.20, port 5201
[  4] local 10.0.12.20 port 38569 connected to 10.0.13.20 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-10.00  sec  6.94 MBytes  5.82 Mbits/sec    0   14.1 KBytes
[  4]  10.00-20.00  sec  6.94 MBytes  5.82 Mbits/sec    0   14.1 KBytes
[  4]  20.00-30.00  sec  6.94 MBytes  5.82 Mbits/sec    0   14.1 KBytes
[  4]  30.00-40.00  sec  6.93 MBytes  5.82 Mbits/sec    0   14.1 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-40.00  sec  27.7 MBytes  5.82 Mbits/sec    0             sender
[  4]   0.00-40.00  sec  27.7 MBytes  5.82 Mbits/sec                  receiver

iperf Done.
root@n7:/tmp/pycore.34481/n7.conf#
```

Throughput = 5.82 Mbps, Retransmissions = 0

Figure 8: 6 kB Window Size

```
root@n7:/tmp/pycore.34487/n7.conf# iperf3 -c 10.0.13.20 -t 40 -i 10 -w 6k
Connecting to host 10.0.13.20, port 5201
[  4] local 10.0.12.20 port 38575 connected to 10.0.13.20 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-10.00  sec   164 KBytes   134 Kbits/sec   93   4.24 KBytes
[  4]  10.00-20.00  sec   158 KBytes   130 Kbits/sec   95   4.24 KBytes
[  4]  20.00-30.00  sec   158 KBytes   130 Kbits/sec   96   4.24 KBytes
[  4]  30.00-40.00  sec   153 KBytes   125 Kbits/sec   94   4.24 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-40.00  sec   634 KBytes   130 Kbits/sec  378             sender
[  4]   0.00-40.00  sec   624 KBytes   128 Kbits/sec                  receiver

iperf Done.
root@n7:/tmp/pycore.34487/n7.conf#
```

Throughput = 0.13 Mbps, Retransmissions = 378

Figure 9: 12 kB Window Size

```
root@n7:/tmp/pycore.34493/n7.conf# iperf3 -c 10.0.13.20 -t 40 -i 10 -w 12k
Connecting to host 10.0.13.20, port 5201
[  4] local 10.0.12.20 port 38581 connected to 10.0.13.20 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-10.00  sec  1.97 MBytes  1.65 Mbits/sec  321   4.24 KBytes
[  4]  10.00-20.00  sec  1.96 MBytes  1.65 Mbits/sec  316   4.24 KBytes
[  4]  20.00-30.00  sec  1.96 MBytes  1.65 Mbits/sec  316   4.24 KBytes
[  4]  30.00-40.00  sec  1.96 MBytes  1.65 Mbits/sec  316   4.24 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-40.00  sec  7.86 MBytes  1.65 Mbits/sec  1269            sender
[  4]   0.00-40.00  sec  7.84 MBytes  1.64 Mbits/sec                  receiver

iperf Done.
root@n7:/tmp/pycore.34493/n7.conf#
```

Throughput = 1.65 Mbps, Retransmissions = 1269

Figure 10: 16 kB Window Size

```
root@n7:/tmp/pycore.34499/n7.conf# iperf3 -c 10.0.13.20 -t 40 -i 10 -w 16k
Connecting to host 10.0.13.20, port 5201
[  4] local 10.0.12.20 port 38587 connected to 10.0.13.20 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-10.00  sec  1.99 MBytes  1.67 Mbits/sec  299   2.83 KBytes
[  4]  10.00-20.00  sec  2.04 MBytes  1.71 Mbits/sec  297   4.24 KBytes
[  4]  20.00-30.00  sec  2.04 MBytes  1.71 Mbits/sec  296   4.24 KBytes
[  4]  30.00-40.00  sec  2.04 MBytes  1.71 Mbits/sec  296   4.24 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-40.00  sec  8.12 MBytes  1.70 Mbits/sec  1188            sender
[  4]   0.00-40.00  sec  8.10 MBytes  1.70 Mbits/sec                  receiver

iperf Done.
root@n7:/tmp/pycore.34499/n7.conf#
```

Throughput = 1.7 Mbps, Retransmissions = 1188

Figure 11: 24 kB Window Size

```
root@n7:/tmp/pycore.34502/n7.conf# iperf3 -c 10.0.13.20 -t 40 -i 10 -w 24k
Connecting to host 10.0.13.20, port 5201
[  4] local 10.0.12.20 port 38590 connected to 10.0.13.20 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-10.00  sec   329 KBytes   270 Kbits/sec  120   5.66 KBytes
[  4]  10.00-20.00  sec   277 KBytes   227 Kbits/sec  114   8.48 KBytes
[  4]  20.00-30.00  sec   297 KBytes   243 Kbits/sec  118   5.66 KBytes
[  4]  30.00-40.00  sec   277 KBytes   227 Kbits/sec  113   7.07 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-40.00  sec  1.15 MBytes   242 Kbits/sec  465             sender
[  4]   0.00-40.00  sec  1.11 MBytes   233 Kbits/sec                  receiver

iperf Done.
root@n7:/tmp/pycore.34502/n7.conf#
```

Throughput = 0.242 Mbps, Retransmissions = 465

Figure 12: 32 kB Window Size

```
root@n7:/tmp/pycore.34505/n7.conf# iperf3 -c 10.0.13.20 -t 40 -i 10 -w 32k
Connecting to host 10.0.13.20, port 5201
[  4] local 10.0.12.20 port 38593 connected to 10.0.13.20 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-10.00  sec   274 KBytes   225 Kbits/sec  111   7.07 KBytes
[  4]  10.00-20.00  sec   161 KBytes   132 Kbits/sec   96   21.2 KBytes
[  4]  20.00-30.00  sec   242 KBytes   198 Kbits/sec  119   7.07 KBytes
[  4]  30.00-40.00  sec   161 KBytes   132 Kbits/sec   96   21.2 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-40.00  sec   839 KBytes   172 Kbits/sec  422             sender
[  4]   0.00-40.00  sec   795 KBytes   163 Kbits/sec                  receiver

iperf Done.
root@n7:/tmp/pycore.34505/n7.conf#
```

Throughput = 0.172 Mbps, Retransmissions = 422

1.1.2. Click on any of the TCP a data segment whose source is node n7, dissect the segment by following different protocol layer headers from TCP->IP->Ethernet identifying how many header bytes are added by each layer, identify the TCP options used.

```
798 65.274124 10.0.12.20 10.0.13.20 TCP 626 [TCP Fast Retransmission] 38599 > targus-...data1 [PSH, ACK] Seq=19568 Ack=1 Win=1116 Len=558 TSval=1137763 TSecr=1137762
▽ Frame 798: 626 bytes on wire (5008 bits), 626 bytes captured (5008 bits)
    Arrival Time: May 17, 2024 16:21:54.058059000 EEST
    Epoch Time: 1715952114.058059000 seconds
    [Time delta from previous captured frame: 0.000481000 seconds]
    [Time delta from previous displayed frame: 0.000481000 seconds]
    [Time since reference or first frame: 65.274124000 seconds]
    Frame Number: 798
    Frame Length: 626 bytes (5008 bits)
    Capture Length: 626 bytes (5008 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: sll:ip:tcp:data]
    [Coloring Rule Name: Bad TCP]
    [Coloring Rule String: tcp.analysis.flags]
▽ Linux cooked capture
    Packet type: Sent by us (4)
    Link-layer address type: 1
    Link-layer address length: 6
    Source: 00:00:00_aa:00:18 (00:00:00:aa:00:18)
    Protocol: IP (0x0800)
▽ Internet Protocol Version 4, Src: 10.0.12.20 (10.0.12.20), Dst: 10.0.13.20 (10.0.13.20)
    Version: 4
    Header length: 20 bytes
  ▽ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
      0000 00.. = Differentiated Services Codepoint: Default (0x00)
      .... ..00 = Explicit Congestion Notification: Not-ECT (Not ECN-Capable Transport) (0x00)
    Total Length: 610
    Identification: 0x89e0 (35296)
  ▽ Flags: 0x02 (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 (6)
  ▽ Header checksum: 0x818e [correct]
      [Good: True]
      [Bad: False]
    Source: 10.0.12.20 (10.0.12.20)
    Destination: 10.0.13.20 (10.0.13.20)
▽ Transmission Control Protocol, Src Port: 38599 (38599), Dst Port: targus-getdata1 (5201), Seq: 19568, Ack: 1, Len: 558
    Source port: 38599 (38599)
    Destination port: targus-getdata1 (5201)
    [Stream index: 3]
    Sequence number: 19568    (relative sequence number)
    [Next sequence number: 20126    (relative sequence number)]
    Acknowledgement number: 1    (relative ack number)
    Header length: 32 bytes
  ▽ Flags: 0x018 (PSH, ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgement: Set
      .... .... 1... = Push: Set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
    Window size value: 1116
    [Calculated window size: 1116]
    [Window size scaling factor: 1]
  ▽ Checksum: 0x2f7c [validation disabled]
      [Good Checksum: False]
      [Bad Checksum: False]
  ▽ Options: (12 bytes)
      No-Operation (NOP)
      No-Operation (NOP)
    ▽ Timestamps: TSval 1137763, TSecr 1137762
        Kind: Timestamp (8)
        Length: 10
        Timestamp value: 1137763
        Timestamp echo reply: 1137762
  ▽ [SEQ/ACK analysis]
      [Bytes in flight: 558]
    ▽ [TCP Analysis Flags]
      ▽ [This frame is a (suspected) fast retransmission]
        ▽ [Expert Info (Warn/Sequence): Fast retransmission (suspected)]
            [Message: Fast retransmission (suspected)]
            [Severity level: Warn]
            [Group: Sequence]
        [This frame is a (suspected) retransmission]
```

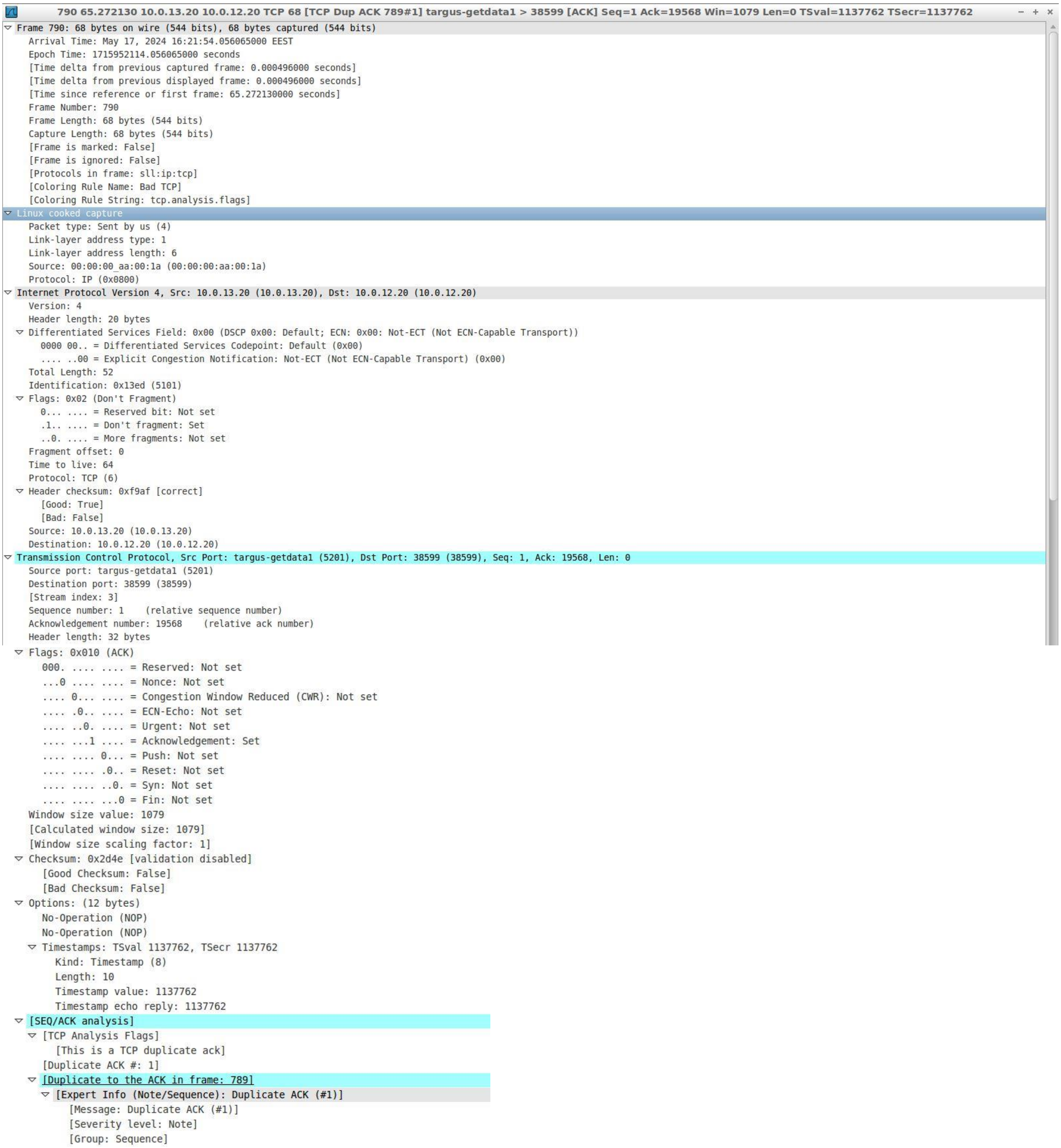
> Note:
> The timestamp option includes a timestamp from the sender, which is then echoed by the receiver in each packet once it's enabled during connection setup. This helps calculate round-trip time samples to estimate packet loss and extends the 32-bit sequence number. On fast connections, sequence numbers can reset quickly, causing potential confusion between old and new data. The PAWS (Protection Against Wrapped Sequence numbers) system resolves this by discarding segments with outdated timestamps.

Figure 13: TCP Packet

Comment on your findings here:

- Frame Length: → 626 bytes.
- IP Header: → 20 bytes
- TCP Header: → 32 bytes (12 Option + 20 Without Option)
- TCP Options: → 12 bytes (2 NOP + 10 Timestamp)
- TCP Options Timestamp: → 10 bytes ($4T_x + 4R_x + 1$ Length + 1 Kind)
- Ethernet Header: = Frame Length − IP Header − TCP Header = $626 - 20 - 32 = 574$ bytes
- As Shown in Fig. 13: Packet Sent from n7 to n11 has large Size (626 B) as it contains Data & Headers.

1.1.3. Repeat for an ACK packet sent from node n11.



```
790 65.272130 10.0.13.20 10.0.12.20 TCP 68 [TCP Dup ACK 789#1] targus-getdata1 > 38599 [ACK] Seq=1 Ack=19568 Win=1079 Len=0 TSval=1137762 TSecr=1137762
Frame 790: 68 bytes on wire (544 bits), 68 bytes captured (544 bits)
    Arrival Time: May 17, 2024 16:21:54.056065000 EEST
    Epoch Time: 1715952114.056065000 seconds
    [Time delta from previous captured frame: 0.000496000 seconds]
    [Time delta from previous displayed frame: 0.000496000 seconds]
    [Time since reference or first frame: 65.272130000 seconds]
    Frame Number: 790
    Frame Length: 68 bytes (544 bits)
    Capture Length: 68 bytes (544 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: sll:ip:tcp]
    [Coloring Rule Name: Bad TCP]
    [Coloring Rule String: tcp.analysis.flags]
Linux cooked capture
    Packet type: Sent by us (4)
    Link-layer address type: 1
    Link-layer address length: 6
    Source: 00:00:00_aa:00:1a (00:00:00:aa:00:1a)
    Protocol: IP (0x0800)
Internet Protocol Version 4, Src: 10.0.13.20 (10.0.13.20), Dst: 10.0.12.20 (10.0.12.20)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..00 = Explicit Congestion Notification: Not-ECT (Not ECN-Capable Transport) (0x00)
    Total Length: 52
    Identification: 0x13ed (5101)
    Flags: 0x02 (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 (6)
    Header checksum: 0xf9af [correct]
        [Good: True]
        [Bad: False]
    Source: 10.0.13.20 (10.0.13.20)
    Destination: 10.0.12.20 (10.0.12.20)
Transmission Control Protocol, Src Port: targus-getdata1 (5201), Dst Port: 38599 (38599), Seq: 1, Ack: 19568, Len: 0
    Source port: targus-getdata1 (5201)
    Destination port: 38599 (38599)
    [Stream index: 3]
    Sequence number: 1    (relative sequence number)
    Acknowledgement number: 19568    (relative ack number)
    Header length: 32 bytes
    Flags: 0x010 (ACK)
        000. .... .... = Reserved: Not set
        ...0 .... .... = Nonce: Not set
        .... 0... .... = Congestion Window Reduced (CWR): Not set
        .... .0.. .... = ECN-Echo: Not set
        .... ..0. .... = Urgent: Not set
        .... ...1 .... = Acknowledgement: Set
        .... .... 0... = Push: Not set
        .... .... .0.. = Reset: Not set
        .... .... ..0. = Syn: Not set
        .... .... ...0 = Fin: Not set
    Window size value: 1079
    [Calculated window size: 1079]
    [Window size scaling factor: 1]
    Checksum: 0x2d4e [validation disabled]
        [Good Checksum: False]
        [Bad Checksum: False]
    Options: (12 bytes)
        No-Operation (NOP)
        No-Operation (NOP)
        Timestamps: TSval 1137762, TSecr 1137762
            Kind: Timestamp (8)
            Length: 10
            Timestamp value: 1137762
            Timestamp echo reply: 1137762
    [SEQ/ACK analysis]
        [TCP Analysis Flags]
            [This is a TCP duplicate ack]
        [Duplicate ACK #: 1]
        [Duplicate to the ACK in frame: 789]
            [Expert Info (Note/Sequence): Duplicate ACK (#1)]
                [Message: Duplicate ACK (#1)]
                [Severity level: Note]
                [Group: Sequence]
```

Figure 14: ACK Packet

Comment on Your findings:

- Frame Length: → 68 bytes.
- IP Header: → 20 bytes
- TCP Header: → 32 bytes (12 Option + 20 Without Option)
- TCP Options: → 12 bytes (2 NOP + 10 Timestamp)
- TCP Options Timestamp: → 10 bytes ($4T_x + 4R_x + 1$ Length $+ 1$ Kind)
- Ethernet Header: = Frame Length − IP Header − TCP Header $= 68 - 20 - 32 = 16$ bytes.
- As Shown in Fig. 14: Packet Sent from n11 to n7 has Small Size (68 B) as it contains Headers only.

1.2. TCP short VS long paths

- Run an iperf3 server on node n8.
- Start an iperf3 client on node n7 connecting to server on node n8 for a duration of 40 seconds and reporting interval of 10 with window size 4K.

1.2.1. Compare the result of throughput with the case when connection was made to node n11. Why throughput drops when connecting to n8 although capacities on the two paths are the same.

```
root@n7:/tmp/pycore.34478/n7.conf# iperf3 -c 10.0.13.20 -t 40 -i 10 -w 4k
Connecting to host 10.0.13.20, port 5201
[  4] local 10.0.12.20 port 38566 connected to 10.0.13.20 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-10.00  sec  6.93 MBytes  5.82 Mbits/sec    0   14.1 KBytes
[  4]  10.00-20.00  sec  6.93 MBytes  5.81 Mbits/sec    0   14.1 KBytes
[  4]  20.00-30.00  sec  6.92 MBytes  5.80 Mbits/sec    0   14.1 KBytes
[  4]  30.00-40.00  sec  6.94 MBytes  5.82 Mbits/sec    0   14.1 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-40.00  sec  27.7 MBytes  5.81 Mbits/sec    0             sender
[  4]   0.00-40.00  sec  27.7 MBytes  5.81 Mbits/sec                  receiver

iperf Done.
root@n7:/tmp/pycore.34478/n7.conf#
```

Figure 15: Short Path (n7 to n11) Throughput = 5.81 Mbps

```
root@n7:/tmp/pycore.34514/n7.conf# iperf3 -c 10.0.11.20 -t 40 -i 10 -w 4k
Connecting to host 10.0.11.20, port 5201
[  4] local 10.0.12.20 port 50864 connected to 10.0.11.20 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-10.00  sec  4.53 MBytes  3.80 Mbits/sec    0   14.1 KBytes
[  4]  10.00-20.00  sec  4.49 MBytes  3.77 Mbits/sec    0   14.1 KBytes
[  4]  20.00-30.00  sec  4.43 MBytes  3.72 Mbits/sec    0   14.1 KBytes
[  4]  30.00-40.00  sec  4.57 MBytes  3.83 Mbits/sec    0   14.1 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-40.00  sec  18.0 MBytes  3.78 Mbits/sec    0             sender
[  4]   0.00-40.00  sec  18.0 MBytes  3.78 Mbits/sec                  receiver

iperf Done.
root@n7:/tmp/pycore.34514/n7.conf#
```

Figure 16: Long Path (n7 to n8) Throughput = 3.78 Mbps

Comment in your findings here:

- ➢ Throughput from n7 to n11 (Short Path): = 5.81 Mbps
- ➢ Throughput from n7 to n8 (Long Path): = 3.78 Mbps
- ➢ We observe that when n8 starts serving instead of n11, the throughput decreases because the path from n7 to n8 is longer than the path from n7 to n11, resulting in delay increase
(RTT and Time out is Increased).
- ➢ By increasing Time out with the same window size, the network utilization is decreased.
So, throughput decreases as shown in the Fig. 15 & Fig. 16.

1.3. Higher link Capacity with Drops VS Reliable Lower Capacity

This questions for this part are based on the path between n7 and n11.

For each case of the following, run iperf3 client from n7 to n11 with window size 4K.

A Select link between n4 and n5, configure it to have capacity of 10 Mbps with zero loss in both directions.

B Select link between n4 and n5, configure it to have capacity of 3 Mbps with zero loss in both directions.

C Select link between n4 and n5, configure it to have capacity of 10 Mbps with 5% loss in both directions.

D Select link between n4 and n5, configure it to have capacity of 100 Mbps with 10% loss in both directions.

E Select link between n4 and n5, configure it to have capacity of 10 Mbps with 1% loss in direction from n4 to n5 and 0% loss in the other direction.

F Select link between n4 and n5, configure it to have capacity of 10 Mbps with 0% loss in direction from n4 to n5 and 1% loss in the other direction.

Case	Throughput	Figures
A	5.81 Mbps	root@n7:/tmp/pycore.34478/n7.conf# iperf3 -c 10.0.13.20 -t 40 -i 10 -w 4k Connecting to host 10.0.13.20, port 5201 [4] local 10.0.12.20 port 38566 connected to 10.0.13.20 port 5201 [ID] Interval Transfer Bandwidth Retr Cwnd [4] 0.00-10.00 sec 6.93 MBytes 5.82 Mbits/sec 0 14.1 KBytes [4] 10.00-20.00 sec 6.93 MBytes 5.81 Mbits/sec 0 14.1 KBytes [4] 20.00-30.00 sec 6.92 MBytes 5.80 Mbits/sec 0 14.1 KBytes [4] 30.00-40.00 sec 6.94 MBytes 5.82 Mbits/sec 0 14.1 KBytes - [ID] Interval Transfer Bandwidth Retr [4] 0.00-40.00 sec 27.7 MBytes 5.81 Mbits/sec 0 sender [4] 0.00-40.00 sec 27.7 MBytes 5.81 Mbits/sec receiver iperf Done. root@n7:/tmp/pycore.34478/n7.conf#
B	2.85 Mbps	root@n7:/tmp/pycore.34523/n7.conf# iperf3 -c 10.0.13.20 -t 40 -i 10 -w 4k Connecting to host 10.0.13.20, port 5201 [4] local 10.0.12.20 port 38611 connected to 10.0.13.20 port 5201 [ID] Interval Transfer Bandwidth Retr Cwnd [4] 0.00-10.00 sec 3.39 MBytes 2.85 Mbits/sec 0 14.1 KBytes [4] 10.00-20.00 sec 3.41 MBytes 2.86 Mbits/sec 0 14.1 KBytes [4] 20.00-30.00 sec 3.39 MBytes 2.84 Mbits/sec 0 14.1 KBytes [4] 30.00-40.00 sec 3.41 MBytes 2.86 Mbits/sec 0 14.1 KBytes - [ID] Interval Transfer Bandwidth Retr [4] 0.00-40.00 sec 13.6 MBytes 2.85 Mbits/sec 0 sender [4] 0.00-40.00 sec 13.6 MBytes 2.85 Mbits/sec receiver iperf Done. root@n7:/tmp/pycore.34523/n7.conf#
C	493 kbps	root@n7:/tmp/pycore.34526/n7.conf# iperf3 -c 10.0.13.20 -t 40 -i 10 -w 4k Connecting to host 10.0.13.20, port 5201 [4] local 10.0.12.20 port 38614 connected to 10.0.13.20 port 5201 [ID] Interval Transfer Bandwidth Retr Cwnd [4] 0.00-10.00 sec 737 KBytes 603 Kbits/sec 39 4.24 KBytes [4] 10.00-20.00 sec 669 KBytes 548 Kbits/sec 38 4.24 KBytes [4] 20.00-30.00 sec 696 KBytes 570 Kbits/sec 38 4.24 KBytes [4] 30.00-40.00 sec 305 KBytes 250 Kbits/sec 22 2.83 KBytes - [ID] Interval Transfer Bandwidth Retr [4] 0.00-40.00 sec 2.35 MBytes 493 Kbits/sec 137 sender [4] 0.00-40.00 sec 2.34 MBytes 492 Kbits/sec receiver iperf Done. root@n7:/tmp/pycore.34526/n7.conf#
D	132 kbps	root@n7:/tmp/pycore.34530/n7.conf# iperf3 -c 10.0.13.20 -t 40 -i 10 -w 4k Connecting to host 10.0.13.20, port 5201 [4] local 10.0.12.20 port 38618 connected to 10.0.13.20 port 5201 [ID] Interval Transfer Bandwidth Retr Cwnd [4] 0.00-10.00 sec 233 KBytes 191 Kbits/sec 25 4.24 KBytes [4] 10.00-20.00 sec 33.9 KBytes 27.8 Kbits/sec 9 4.24 KBytes [4] 20.00-30.00 sec 120 KBytes 98.5 Kbits/sec 10 4.24 KBytes [4] 30.00-40.00 sec 263 KBytes 215 Kbits/sec 25 4.24 KBytes - [ID] Interval Transfer Bandwidth Retr [4] 0.00-40.00 sec 650 KBytes 133 Kbits/sec 69 sender [4] 0.00-40.00 sec 645 KBytes 132 Kbits/sec receiver iperf Done. root@n7:/tmp/pycore.34530/n7.conf#
E	3.56 Mbps	root@n7:/tmp/pycore.34533/n7.conf# iperf3 -c 10.0.13.20 -t 40 -i 10 -w 4k Connecting to host 10.0.13.20, port 5201 [4] local 10.0.12.20 port 38621 connected to 10.0.13.20 port 5201 [ID] Interval Transfer Bandwidth Retr Cwnd [4] 0.00-10.00 sec 4.13 MBytes 3.47 Mbits/sec 18 8.48 KBytes [4] 10.00-20.00 sec 4.72 MBytes 3.96 Mbits/sec 15 8.48 KBytes [4] 20.00-30.00 sec 3.70 MBytes 3.10 Mbits/sec 20 8.48 KBytes [4] 30.00-40.00 sec 4.42 MBytes 3.71 Mbits/sec 17 8.48 KBytes - [ID] Interval Transfer Bandwidth Retr [4] 0.00-40.00 sec 17.0 MBytes 3.56 Mbits/sec 70 sender [4] 0.00-40.00 sec 17.0 MBytes 3.56 Mbits/sec receiver iperf Done. root@n7:/tmp/pycore.34533/n7.conf#
F	2.8 Mbps	root@n7:/tmp/pycore.34537/n7.conf# iperf3 -c 10.0.13.20 -t 40 -i 10 -w 4k Connecting to host 10.0.13.20, port 5201 [4] local 10.0.12.20 port 38625 connected to 10.0.13.20 port 5201 [ID] Interval Transfer Bandwidth Retr Cwnd [4] 0.00-10.00 sec 3.82 MBytes 3.21 Mbits/sec 22 5.66 KBytes [4] 10.00-20.00 sec 3.34 MBytes 2.80 Mbits/sec 25 8.48 KBytes [4] 20.00-30.00 sec 2.78 MBytes 2.33 Mbits/sec 29 4.24 KBytes [4] 30.00-40.00 sec 3.41 MBytes 2.86 Mbits/sec 24 4.24 KBytes - [ID] Interval Transfer Bandwidth Retr [4] 0.00-40.00 sec 13.3 MBytes 2.80 Mbits/sec 100 sender [4] 0.00-40.00 sec 13.3 MBytes 2.80 Mbits/sec receiver iperf Done. root@n7:/tmp/pycore.34537/n7.conf#

1.3.1. Compare throughputs in cases a, b, c. Why b is better than c?
Comment in your findings here:

We observe that the throughput in case (A) is higher than in case (B), and the throughput in case (B) is higher than in case (C). This indicates that case (B) performs better than case (C), Although the link capacity in case (C) '10 Mbps' is greater than in case (B) '3Mbps' but in case (C) there are higher losses (5%) that necessitate retransmissions, unlike in case (B) that has loss rate is 0%. In addition to that, the throughput in case (A) is greater than that of case (B) due to the greater link capacity in case (A) compared to case (B) as both case (A) and case (B) have equal losses.

1.3.2. Compare throughputs in cases b, c and d. Which is better? Why?
Comment in your findings here:

Case (B) is better than cases (C) and (D) [$\text{Throughput}_B > \text{Throughput}_C > \text{Throughput}_D$] because the loss in case (B) is zero in both directions, eliminating the need for the client or server to retransmit any packets. Although the capacity in case (B) is lower than in cases (C) and (D), packet retransmissions are necessary in cases (C) and (D) due to non-zero loss (for case (C) = 5% loss & for case (D) = 10% loss).

1.3.3. Compare throughputs in cases e and f? Which is better? Why?
Comment in your findings here:

Throughput for Case (E) is better than case (F) because large packets are often sent from n5 to n4, while the packets sent from n4 to n5 are usually smaller. Consequently, losses will affect the (n5 to n4) link more than (n4 to n5) link.

2. OSPF Lab

Each of the following sections describes an experiment based on the network topology provided by the **Project_OSPF_TCP.imn** file.

The Introductory Lab explained and introduced most of the needed tools like iperf3, vtysh, wireshark, etc. If you have not gone through it, do not start working on the assignment below.

It is better to start the network emulation once and run Wireshark captures. You can reset Wireshark captures from experiment to another.

2.1. OSPF Link Cost Change

1. Stop any running iperf3 clients.

2. Set all links to have zero loss in the two directions with 100 Mbps speed.

3. Run iperf3 between n7 and n11 for a duration of 500 seconds or longer. Identify the path between n7 and n11.

4. Open vtysh on node n5 by opening a bash terminal and typing vtysh. Now we can configure the router and link costs.

5. Type the following in vtysh: show Ip ospf interface eth1 (To know the ethernet number of each interface you can choose toolbar->view->show->interface names) This displays information about interface eth1. Note its ospf cost.
6. Type the following configure terminal.
   - interface eth1 (the interface for the link between n5 and n4)
   - ospf cost 40 (set cost to 40)

2.1.1. Check what happens to the path between n7 and n11 (as seen after steps 3 and 6)? Explain what happens. To help visualize the path choose toolbar->widgets-> Throughput.

```
root@n5:/tmp/pycore.41071/n5.conf# vtysh

Hello, this is Quagga (version 0.99.20.1).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

n5# show ip ospf interface eth1
eth1 is up
  ifindex 116, MTU 1500 bytes, BW 0 Kbit <UP,BROADCAST,RUNNING,MULTICAST>
  Internet Address 10.0.4.2/24, Area 0.0.0.0
  MTU mismatch detection:enabled
  Router ID 10.0.5.1, Network Type BROADCAST, Cost: 10
  Transmit Delay is 1 sec, State DR, Priority 1
  Designated Router (ID) 10.0.5.1, Interface Address 10.0.4.2
  Backup Designated Router (ID) 10.0.3.1, Interface Address 10.0.4.1
  Multicast group memberships: OSPFAllRouters OSPFDesignatedRouters
  Timer intervals configured, Hello 10s, Dead 40s, Wait 40s, Retransmit 5
    Hello due in 7.095s
  Neighbor Count is 1, Adjacent neighbor count is 1
n5#
```

Figure 17: n5 interface 1

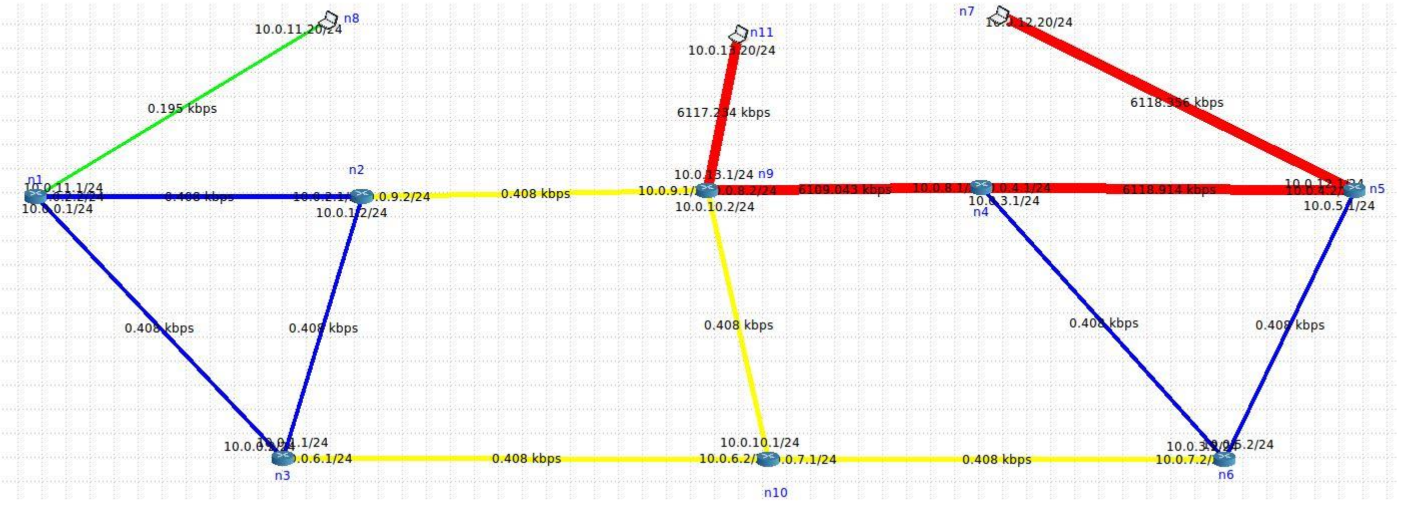


Figure 18: Path From n7 to n11 before changing n5 cost.

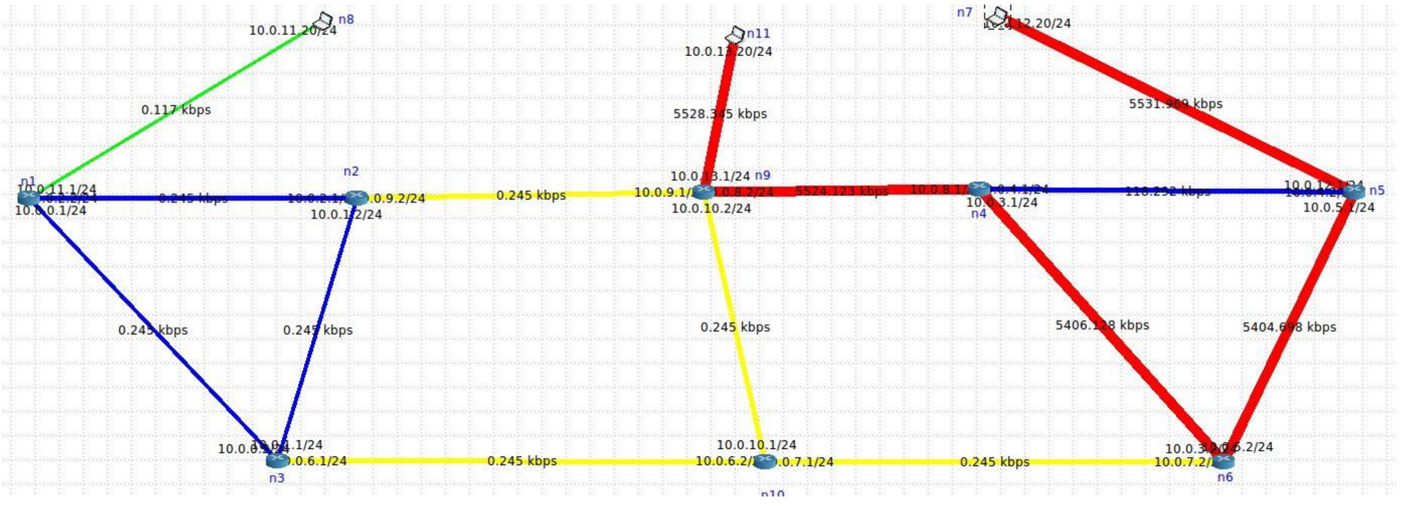


Figure 19: Path From n7 to n11 after changing n5 cost to 40

Comment on your findings:

As illustrated in Fig. 18 & 19, increasing the cost of the interface between n4 and n5 causes the data packets to take an alternative route to achieve a lower overall path cost, with the help of Dijkstra's Algorithm.
The routers begin to recalculate the best route to determine the shortest path between n7 and n11 as shown in Fig. 19

2.1.2. Set the cost of eth1 at node n5 back to 10. Establish two iperf3 connections: one from n7 to n11 and the second from n11 to n7 both for duration of 500 seconds. Now go to node n4 and set interface cost for interface connecting n4 with n5 to 40. What happens in the paths of the two connections? Explain what happens. What do you conclude?

```
root@n4:/tmp/pycore.59083/n4.conf# vtysh

Hello, this is Quagga (version 0.99.20.1).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

n4# show ip ospf interface eth1
eth1 is up
  ifindex 361, MTU 1500 bytes, BW 0 Kbit <UP,BROADCAST,RUNNING,MULTICAST>
  Internet Address 10.0.4.1/24, Area 0.0.0.0
  MTU mismatch detection:enabled
  Router ID 10.0.3.1, Network Type BROADCAST, Cost: 10
  Transmit Delay is 1 sec, State Backup, Priority 1
  Designated Router (ID) 10.0.5.1, Interface Address 10.0.4.2
  Backup Designated Router (ID) 10.0.3.1, Interface Address 10.0.4.1
  Multicast group memberships: OSPFAllRouters OSPFDesignatedRouters
  Timer intervals configured, Hello 10s, Dead 40s, Wait 40s, Retransmit 5
    Hello due in 3.903s
  Neighbor Count is 1, Adjacent neighbor count is 1
n4#
```

Figure 20: n4 interface 1

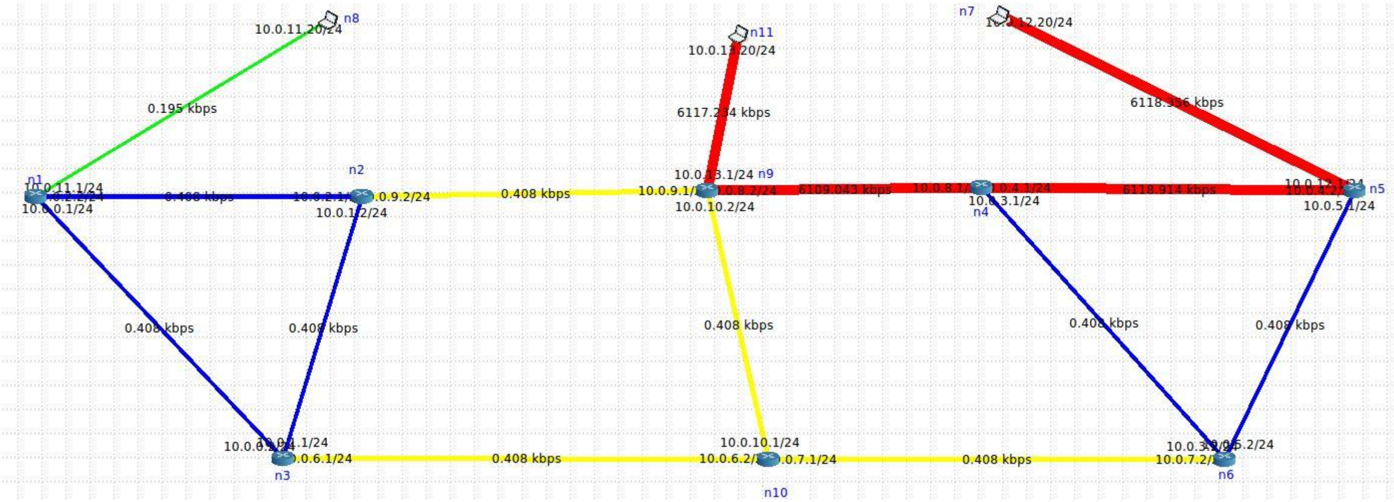


Figure 21: Path from n7 to n11 and vice versa before changing n4 cost.

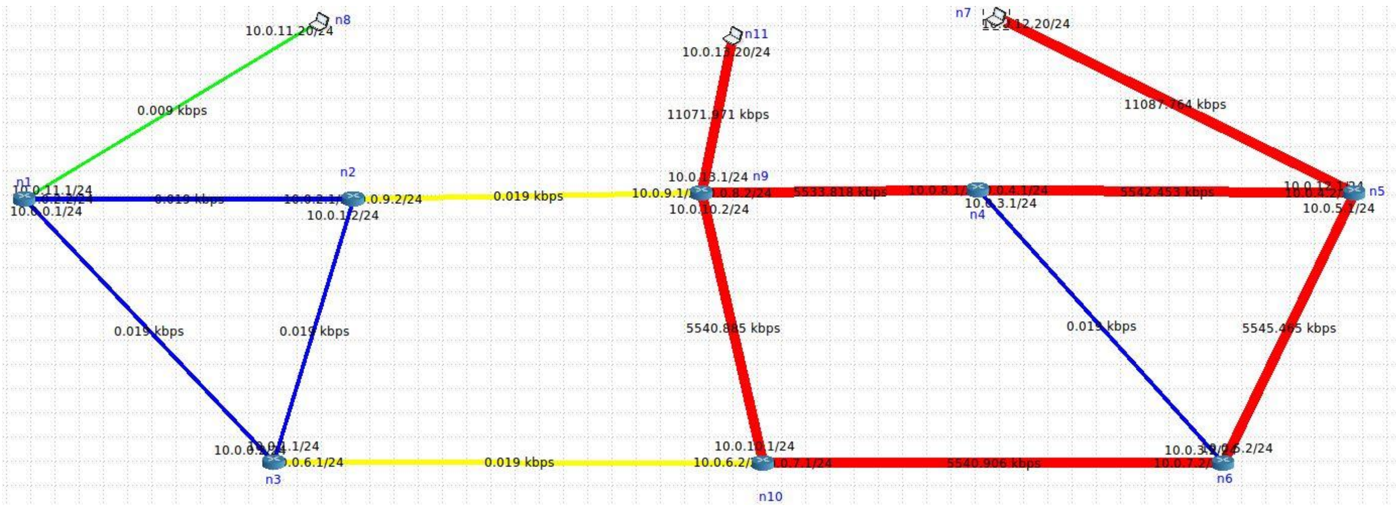


Figure 22: Path from n7 to n11 and vice versa after changing n4 cost to 40

Comment on your findings:

For the same reason mentioned above, the path becomes much longer. The increasing cost of a link forces the router to choose an alternative path, even if it is longer, to reduce congestion.
As OSPF could handle such situations.by increasing the cost of n4 → n5 and leaving n5 → n4 the same. The shortest path from n7 to n11 is not the same as the shortest path from n11 to n7.
Paths are (n7 → n5 → n4 → n9 → n11) & (n11 → n9 → n10 → n6 → n5 → n7)

2.2. OSPF Database Updates

1. Start wireshark and have its filter to capture only OSPF related packets.

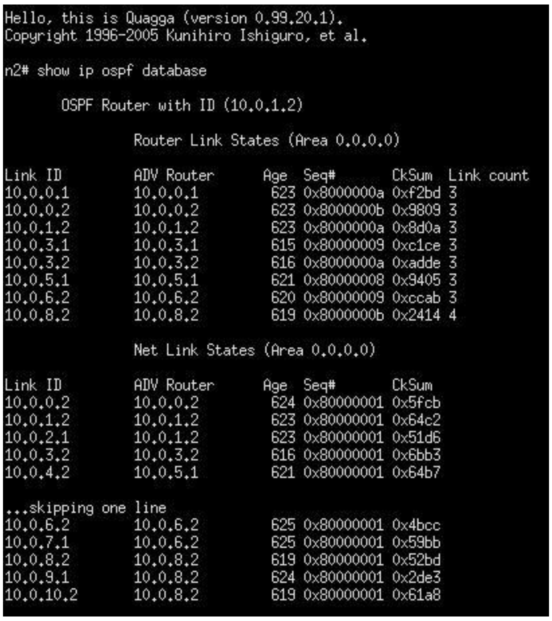
2. Go to another router say router n2, open vtysh and issue the commands:
    - config terminal
    - show in ospf database (check this link to understand more)
    - show in ospf route.

3. On router n4, set link cost of eth1 to 20. Capture the link state packets advertised in Wireshark.

4. Go to router n4, go out of vtysh, or open new bash terminal and issue the following commands to disconnect router n4 from the network:

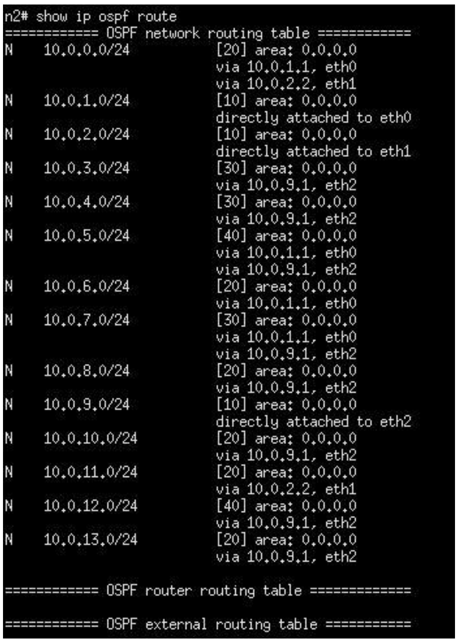
    - ifconfig eth0 down
    - ifconfig eth1 down
    - ifconfig eth2 down

2.2.1. Capture and explain the outputs due to execution of step 2. Why some destinations have more than route in the routing tables?



```
Hello, this is Quagga (version 0.99.20.1).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

n2# show ip ospf database

       OSPF Router with ID (10.0.1.2)

                Router Link States (Area 0.0.0.0)

Link ID         ADV Router      Age  Seq#       CkSum  Link count
10.0.0.1        10.0.0.1         623 0x8000000a 0xf2bd 3
10.0.0.2        10.0.0.2         623 0x8000000b 0x9809 3
10.0.1.2        10.0.1.2         623 0x8000000a 0x8d0a 3
10.0.3.1        10.0.3.1         615 0x80000009 0xc1ce 3
10.0.3.2        10.0.3.2         616 0x8000000a 0xadde 3
10.0.5.1        10.0.5.1         621 0x80000008 0x9405 3
10.0.6.2        10.0.6.2         620 0x80000009 0xccab 3
10.0.8.2        10.0.8.2         619 0x8000000b 0x2414 4

                Net Link States (Area 0.0.0.0)

Link ID         ADV Router      Age  Seq#       CkSum
10.0.0.2        10.0.0.2         624 0x80000001 0x5fcb
10.0.1.2        10.0.1.2         623 0x80000001 0x64c2
10.0.2.1        10.0.1.2         623 0x80000001 0x51d6
10.0.3.2        10.0.3.2         616 0x80000001 0x6bb3
10.0.4.2        10.0.5.1         621 0x80000001 0x64b7

...skipping one line
10.0.6.2        10.0.6.2         625 0x80000001 0x4bcc
10.0.7.1        10.0.6.2         625 0x80000001 0x59bb
10.0.8.2        10.0.8.2         619 0x80000001 0x52bd
10.0.9.1        10.0.8.2         624 0x80000001 0x2de3
10.0.10.2       10.0.8.2         619 0x80000001 0x61a8
```

Figure 23: n2 Database



```
n2# show ip ospf route
============ OSPF network routing table ============
N    10.0.0.0/24           [20] area: 0.0.0.0
                           via 10.0.1.1, eth0
                           via 10.0.2.2, eth1
N    10.0.1.0/24           [10] area: 0.0.0.0
                           directly attached to eth0
N    10.0.2.0/24           [10] area: 0.0.0.0
                           directly attached to eth1
N    10.0.3.0/24           [30] area: 0.0.0.0
                           via 10.0.9.1, eth2
N    10.0.4.0/24           [30] area: 0.0.0.0
                           via 10.0.9.1, eth2
N    10.0.5.0/24           [40] area: 0.0.0.0
                           via 10.0.1.1, eth0
                           via 10.0.9.1, eth2
N    10.0.6.0/24           [20] area: 0.0.0.0
                           via 10.0.1.1, eth0
N    10.0.7.0/24           [30] area: 0.0.0.0
                           via 10.0.1.1, eth0
                           via 10.0.9.1, eth2
N    10.0.8.0/24           [20] area: 0.0.0.0
                           via 10.0.9.1, eth2
N    10.0.9.0/24           [10] area: 0.0.0.0
                           directly attached to eth2
N    10.0.10.0/24          [20] area: 0.0.0.0
                           via 10.0.9.1, eth2
N    10.0.11.0/24          [20] area: 0.0.0.0
                           via 10.0.2.2, eth1
N    10.0.12.0/24          [40] area: 0.0.0.0
                           via 10.0.9.1, eth2
N    10.0.13.0/24          [20] area: 0.0.0.0
                           via 10.0.9.1, eth2

============ OSPF router routing table =============

============ OSPF external routing table ===========
```

Figure 24: n2 Routing table.

Comment on your findings:

From the above routing table there are multiple paths available that have the same cost. This means that the network switch can choose any one of these paths to send data. Additionally, if one of these paths becomes unavailable (goes down), there are still other paths that can be used, ensuring continued network connectivity.
The database for router n2 is illustrated in the figures, and it consists of two segments:
Router (n2) Link State: This describes the network's routers, links, and topology, constructed at each router after exchanging hello packets with other routers.
Designated Router Link State: Similar to the router link state but generated by the designated router, which is elected to manage communication on a multi-access network segment, reducing overhead and ensuring efficient data exchange.
Figures 23 and 24 show the n2 database and routing table, respectively. Both tables have fields including:
Link ID: Identifies the router or network for which the LSA is originated.
ADV Router: Specifies the originating router of the LSA.
Age: Indicates the time since the LSA was last updated.
SEQ#: The sequence number of the LSA, used to compare and accept new information.

Checksum: Validates the LSA packet.
Link Count: Number of links advertised within the LSA.
The routing table in figure 24 includes destination addresses and their corresponding routes. OSPF supports Equal Cost Multipath (ECMP), meaning multiple paths can be assigned to the same destination if they have equal cost, balancing load and increasing network fault tolerance. Some destinations have multiple equal-cost paths, while others have a single path.

2.2.2. After executing step 3, determine how long it took the network to exchange link state packets and adjust routing tables. (Hint: you can calculate the required time by observing the time of first OSPF update message and the last ACK from Wireshark).

No.	Time	Source	Destination	Protocol	Length	Info
10322	1380.229640	10.0.6.2	224.0.0.5	OSPF	84	Hello Packet
10323	1380.229645	10.0.7.1	224.0.0.5	OSPF	84	Hello Packet
10324	1380.229647	10.0.10.1	224.0.0.5	OSPF	84	Hello Packet
10325	1388.370918	10.0.4.1	224.0.0.5	OSPF	208	LS Update
10326	1388.370924	10.0.4.1	224.0.0.5	OSPF	208	LS Update
10327	1388.370933	10.0.8.1	224.0.0.5	OSPF	208	LS Update
10328	1388.370936	10.0.8.1	224.0.0.5	OSPF	208	LS Update
10329	1388.371087	10.0.8.1	224.0.0.5	OSPF	208	LS Update
10330	1388.371090	10.0.4.1	224.0.0.5	OSPF	208	LS Update
10331	1388.371166	10.0.9.1	224.0.0.5	OSPF	112	LS Update
10332	1388.371170	10.0.9.1	224.0.0.5	OSPF	112	LS Update
10333	1388.371179	10.0.10.2	224.0.0.5	OSPF	112	LS Update
10334	1388.371182	10.0.10.2	224.0.0.5	OSPF	112	LS Update
10335	1388.371224	10.0.5.1	224.0.0.5	OSPF	112	LS Update
10336	1388.371226	10.0.5.1	224.0.0.5	OSPF	112	LS Update
10337	1388.371335	10.0.9.1	224.0.0.5	OSPF	112	LS Update

Figure 25: n2 First Update Packet

No.	Time	Source	Destination	Protocol	Length	Info
10389	1388.730011	10.0.2.2	224.0.0.5	OSPF	80	LS Acknowledge
10390	1388.730014	10.0.2.2	224.0.0.5	OSPF	80	LS Acknowledge
10391	1388.730439	10.0.0.1	224.0.0.5	OSPF	80	LS Acknowledge
10392	1388.730443	10.0.2.2	224.0.0.5	OSPF	80	LS Acknowledge
10393	1388.884656	10.0.10.1	224.0.0.5	OSPF	80	LS Acknowledge
10394	1388.884671	10.0.10.1	224.0.0.5	OSPF	80	LS Acknowledge
10395	1388.885069	10.0.10.1	224.0.0.5	OSPF	80	LS Acknowledge
10396	1390.108222	10.0.3.2	224.0.0.5	OSPF	84	Hello Packet
10397	1390.108222	10.0.1.2	224.0.0.5	OSPF	84	Hello Packet
10398	1390.108231	10.0.3.2	224.0.0.5	OSPF	84	Hello Packet
10399	1390.108237	10.0.1.2	224.0.0.5	OSPF	84	Hello Packet
10400	1390.108246	10.0.5.2	224.0.0.5	OSPF	84	Hello Packet
10401	1390.108252	10.0.2.1	224.0.0.5	OSPF	84	Hello Packet
10402	1390.108253	10.0.5.2	224.0.0.5	OSPF	84	Hello Packet

Figure 26: n2 Last ACK Packet

Comment on your findings:

From the above graphs we conclude that the time taken by the Network to exchange link state packets and adjust routing tables → Time taken $= 1388.885069 - 1388.370918 = 0.514151$ sec

2.2.3. After execution of step 4, identify the new routing table and router database at router n2. Explain the updates in the new routing table and the new database.

```
Hello, this is Quagga (version 0.99.20.1).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

n2# show ip ospf database

       OSPF Router with ID (10.0.1.2)

                Router Link States (Area 0.0.0.0)

Link ID         ADV Router      Age  Seq#       CkSum  Link count
10.0.0.1        10.0.0.1        1307 0x8000000b 0xf0be 3
10.0.0.2        10.0.0.2        1357 0x8000000c 0x960a 3
10.0.1.2        10.0.1.2        1356 0x8000000b 0x8b0b 3
10.0.3.1        10.0.3.1        1311 0x8000000e 0xba3f 1
10.0.3.2        10.0.3.2        1669 0x8000000b 0xf4a6 3
10.0.5.1        10.0.5.1        1280 0x8000000a 0xffa8 3
10.0.6.2        10.0.6.2        1288 0x8000000a 0xcaac 3
10.0.8.2        10.0.8.2        1267 0x8000000d 0x2823 4

                Net Link States (Area 0.0.0.0)

Link ID         ADV Router      Age  Seq#       CkSum
10.0.0.2        10.0.0.2         737 0x80000002 0x5dcc
10.0.1.2        10.0.1.2         736 0x80000002 0x62c3
10.0.2.1        10.0.1.2        1656 0x80000002 0x4fd7
10.0.5.1        10.0.5.1         720 0x80000002 0x67b2
10.0.6.2        10.0.6.2          78 0x80000003 0x47ce

...skipping one line
10.0.9.1        10.0.8.2         707 0x80000002 0x2be4
10.0.10.2       10.0.8.2         257 0x80000003 0x5daa
```

Figure 27: n2 Database after Shutting down n4

```
n2# show ip ospf route
============ OSPF network routing table ============
N    10.0.0.0/24           [20] area: 0.0.0.0
                           via 10.0.1.1, eth0
                           via 10.0.2.2, eth1
N    10.0.1.0/24           [10] area: 0.0.0.0
                           directly attached to eth0
N    10.0.2.0/24           [10] area: 0.0.0.0
                           directly attached to eth1
N    10.0.3.0/24           [40] area: 0.0.0.0
                           via 10.0.1.1, eth0
                           via 10.0.9.1, eth2
N    10.0.4.0/24           [50] area: 0.0.0.0
                           via 10.0.1.1, eth0
                           via 10.0.9.1, eth2
N    10.0.5.0/24           [40] area: 0.0.0.0
                           via 10.0.1.1, eth0
                           via 10.0.9.1, eth2
N    10.0.6.0/24           [20] area: 0.0.0.0
                           via 10.0.1.1, eth0
N    10.0.7.0/24           [30] area: 0.0.0.0
                           via 10.0.1.1, eth0
                           via 10.0.9.1, eth2
N    10.0.8.0/24           [20] area: 0.0.0.0
                           via 10.0.9.1, eth2
N    10.0.9.0/24           [10] area: 0.0.0.0
                           directly attached to eth2
N    10.0.10.0/24          [20] area: 0.0.0.0
                           via 10.0.9.1, eth2
N    10.0.11.0/24          [20] area: 0.0.0.0
                           via 10.0.2.2, eth1
N    10.0.12.0/24          [50] area: 0.0.0.0
                           via 10.0.1.1, eth0
                           via 10.0.9.1, eth2
N    10.0.13.0/24          [20] area: 0.0.0.0
                           via 10.0.9.1, eth2

============ OSPF router routing table =============

============ OSPF external routing table ===========
```

Figure 28: n2 Routing Table after Shutting down n4

Comment on your findings:

If the node (n4) is disconnected, another node will take its place which is (n2). Although n2 has a higher cost than n4, the network will update its database to reflect this change. This ensures that data can still be routed effectively, even though the overall path cost will be higher.