**Computer Networks Assignment-4**

**Network simulation and TCP congestion control analysis using ns3**

**Ans 1.**

1. The maximum expected value of **throughput** is **7 Mbps**, which is the node1’s configured **data rate**. This is the bottleneck throughput. This is because:

* The throughput in this simulation is constrained by the node1's data rate (DataRate("7Mbps")) from N1 to N2.
* TCP might dynamically adjust the sending rate based on network conditions. But,
  + The **loss rate (0.000001)** is too low to significantly impact throughput.
  + Assuming no significant queuing delays or drops due to buffer overflow, the queue size is assumed sufficient to handle temporary bursts.

**BDP = Bandwidth × Round Trip Time (RTT)**

Delay from N0 to N1: 100 ms

Delay from N1 to N2: 10 ms

Total one-way delay: 100+10 = 110 ms

**RTT (Round-Trip time)** = 2 × one-way delay = 2 ×110 ms = **220 ms = 0.22 seconds**

The **bandwidth** of the bottleneck link (from N1 to N2) is **7 Mbps (7 ×106 bps).**

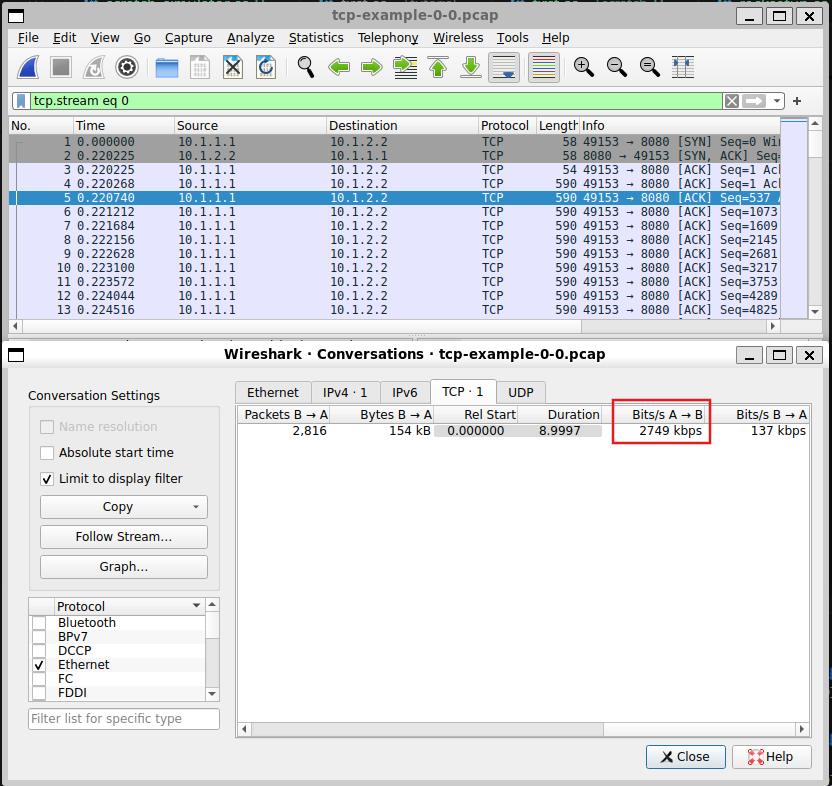
**BDP (in bits) = Bandwidth** **× RTT = 7 ×106 bps × 0.22 seconds = 1540000 = 154 × 104 bits**

Each packet size is 1460 Bytes**, Packet size** = 1460 × 8 = **11680 bits**

**BDP (in packets) =**

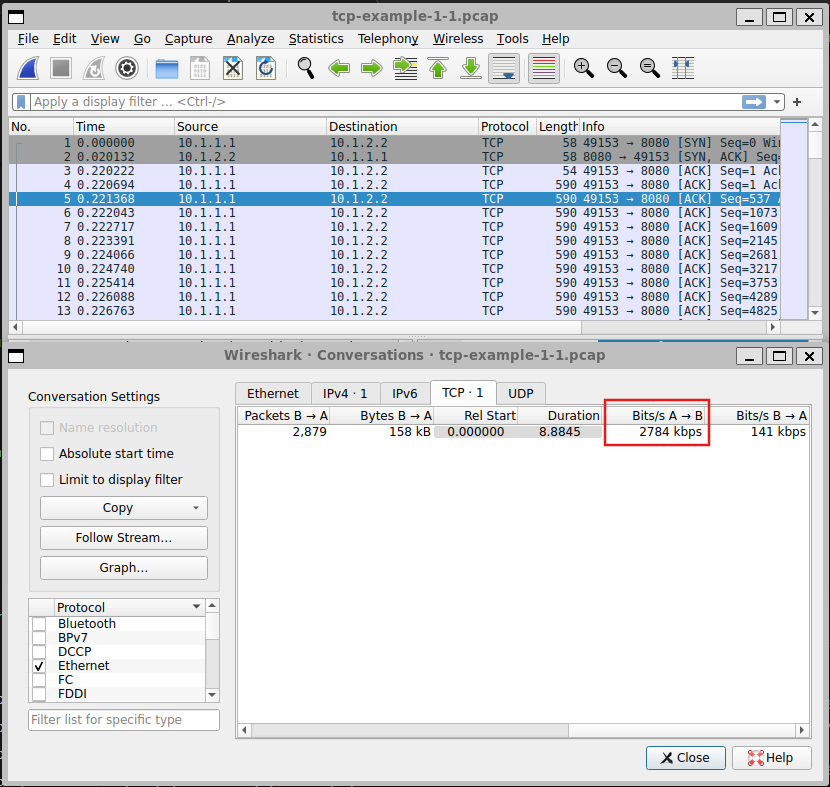
So, the **Bandwidth-Delay Product (BDP)** is **132 packets.**

1. The computed throughput of “**tcp-example-0-0.pcap”: 2.749 Mbps**



This is the throughput from Node0 to Node1.

The computed throughput of **“tcp-example-1-1.pcap”: 2.784 Mbps**



This is the throughput from Node1 to Node2.

The **average computed throughput** of the **TCP transfer**:

1. The achieved throughput is not equal to the maximum expected value. This is because:

* **BDP Utilization:** Maximum throughput assumes that the network is fully utilizing the BDP. However, achieving this requires the sender to transmit at a rate that keeps the pipe full at all times.
* **TCP congestion control:** TCP sender starts with a small congestion window, gradually increasing it (slow start phase). Therefore, the link cannot immediately operate at full capacity because the congestion window grows incrementally.
* **Queueing Delays and Buffer Drops:** At node1, the queue size is limited, and packets exceeding the buffer are dropped. Packet drops lead to retransmissions, further reducing effective throughput.
* **Packet Loss:** A small packet loss rate (0.000001) is present on the link. Even minor packet losses can reduce throughput as TCP interprets packet loss as a sign of congestion, further reducing its congestion window.

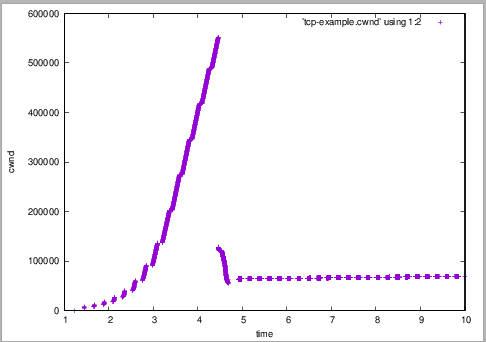
1. To plot the **cwnd v/s time**, create the “**example.gp**” file as given in the pdf

Run: **gnuplot example.gp**

This will create a “**cwnd.eps**” file. To view this file,

Run: **gv cwnd.eps**

This will result in the following plot:



1. To plot the queue delay v/s time, first create a python script that parses the “**tcp-example.tr**” file and outputs the queueing events and their delays as attached in the .zip file named “**extract\_queue\_delay.py**”. This will result in a “**queueing\_delay.dat**” file. Compile the script “**example2.gp**”:

**set term postscript eps color**

**set output ‘queue.eps'**

**set ylabel 'Queueing Delay (s)'**

**set xlabel 'Time (s)'**

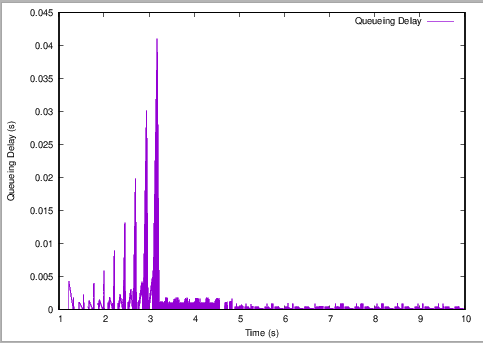
**plot 'queueing\_delay.dat' using 1:2 with lines title 'Queueing Delay'**

Run: **gnuplot example2.gp**

This will create a “**queue.eps**” file. To view this file,

Run: **gv queue.eps**

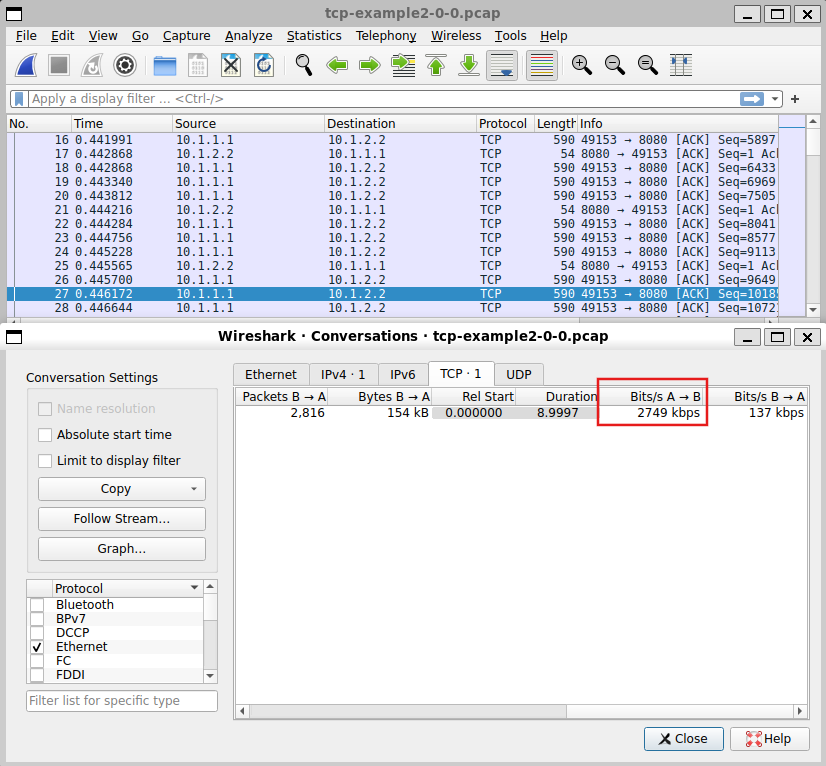
This will result in the following plot:

****

1. The plots in **e)** and **f)** are related as the increase in **cwnd** directly causes higher **queueing delay** due to network congestion, and the reduction in **cwnd** during congestion control helps alleviate **queueing delay**.

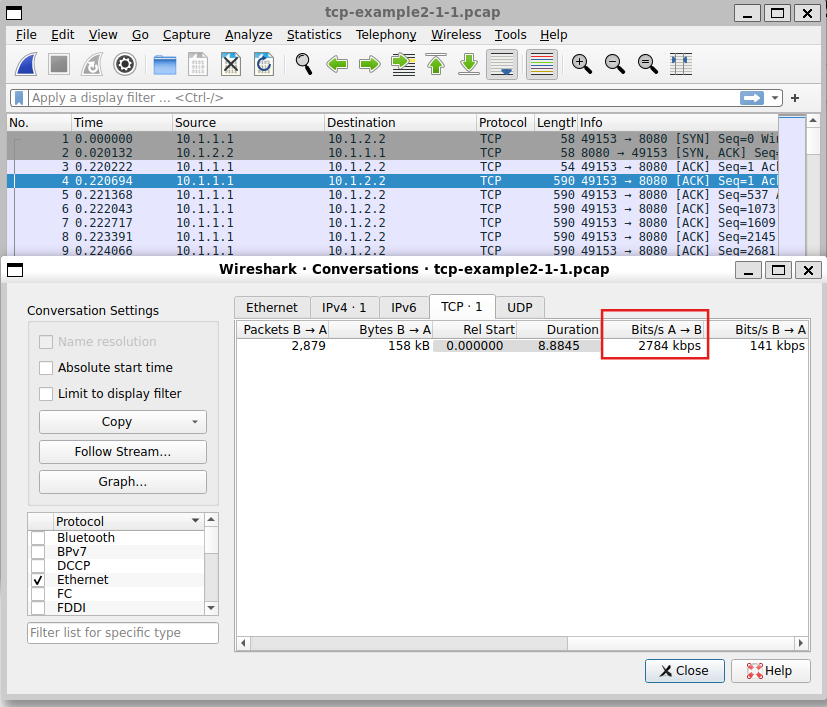
**Ans 2.**

1. The computed throughput of “**tcp-example2-0-0.pcap”: 2.749 Mbps**

****

This is the throughput from Node0 to Node1.

The computed throughput of **“tcp-example2-1-1.pcap”: 2.784 Mbps**

****

This is the throughput from Node1 to Node2.

The **average computed throughput** of the **TCP transfer**:

1. To plot the **cwnd v/s time**, create the “**cwnd.gp**” file:

**set term postscript eps color**

**set output 'cwnd2.eps'**

**set ylabel 'cwnd'**

**set xlabel 'time'**

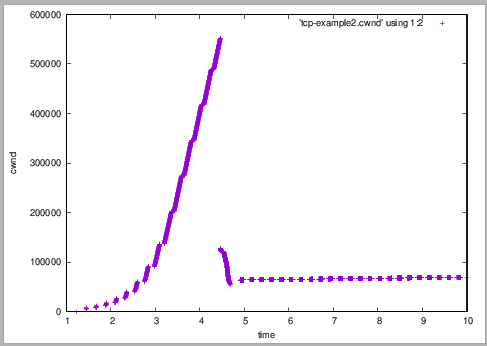
**plot 'tcp-example2.cwnd' using 1:2**

Run: **gnuplot cwnd.gp**

This will create a “**cwnd2.eps**” file. To view this file,

Run: **gv cwnd2.eps**

This will result in the following plot:

****

1. To plot the queue delay v/s time, first create a python script that parses the “**tcp-example2.tr**” file and outputs the queueing events and their delays as attached in the .zip file named “**extract\_queue\_delay.py**”. This will result in a “**queueing\_delay2.dat**” file. Compile the script “**queue.gp**”:

**set term postscript eps color**

**set output ‘queue2.eps'**

**set ylabel 'Queueing Delay (s)'**

**set xlabel 'Time (s)'**

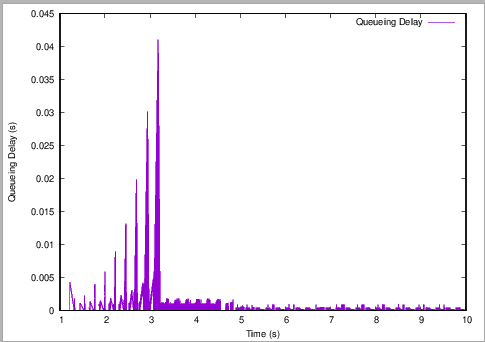
**plot 'queueing\_delay2.dat' using 1:2 with lines title 'Queueing Delay'**

Run: **gnuplot queue.gp**

This will create a “**queue2.eps**” file. To view this file,

Run: **gv queue2.eps**

This will result in the following plot:



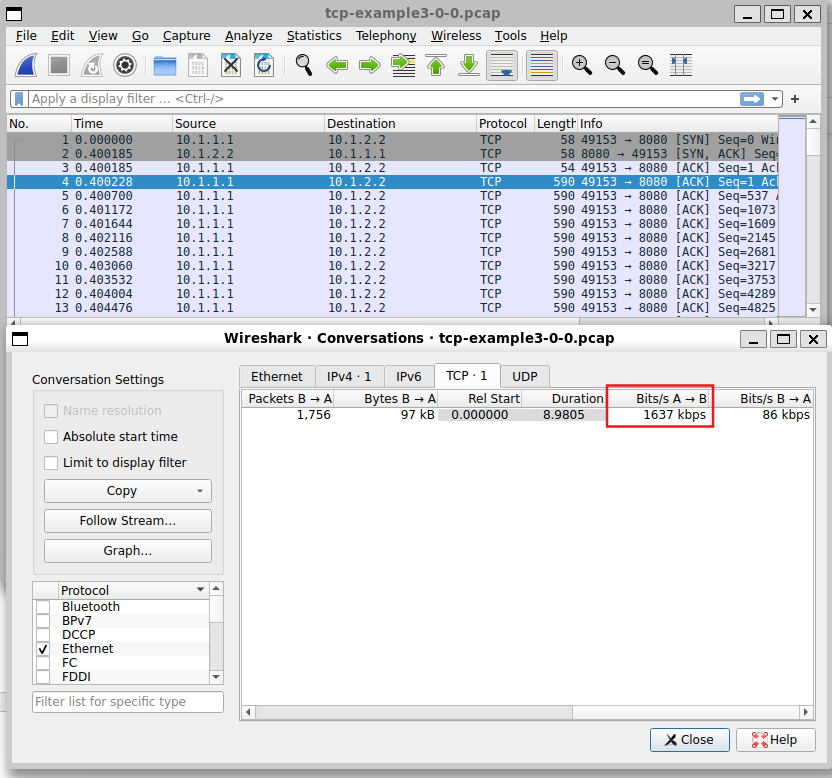
1. In both plots, the congestion window grows exponentially during the slow start phase until a packet loss or congestion occurs.

For a queue size of 5 packets, the congestion window reaches a lower peak before experiencing packet loss, indicating an earlier onset of congestion. With a queue size of 100 packets, the congestion window grows much larger before packet loss occurs, suggesting that the queue can absorb more packets before dropping any.

For the queue size of 5 packets, the steady-state cwnd is much smaller compared to the 100-packet queue.

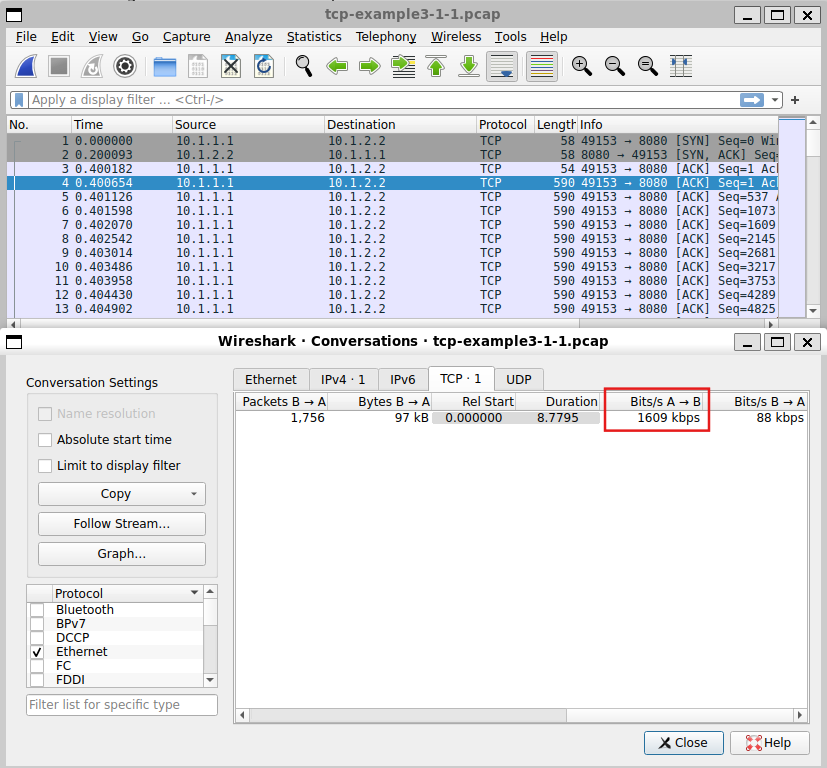
**Ans 3.**

1. The computed throughput of “**tcp-example3-0-0.pcap”: 1.637 Mbps**

****

This is the throughput from Node0 to Node1.

The computed throughput of **“tcp-example3-1-1.pcap”: 1.609 Mbps**

****

This is the throughput from Node1 to Node2.

The **average computed throughput** of the **TCP transfer**:

1. To plot the **cwnd v/s time**, create the “**cwnd3.gp**” file:

**set term postscript eps color**

**set output 'cwnd3.eps'**

**set ylabel 'cwnd'**

**set xlabel 'time'**

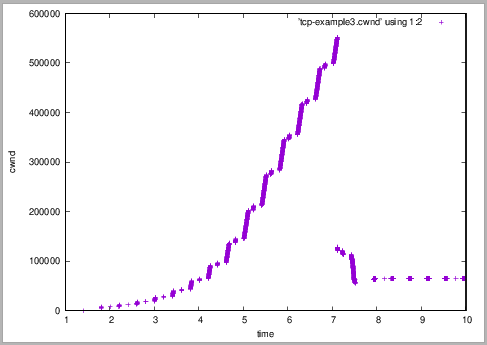
**plot 'tcp-example3.cwnd' using 1:2**

Run: **gnuplot cwnd3.gp**

This will create a “**cwnd3.eps**” file. To view this file,

Run: **gv cwnd3.eps**

This will result in the following plot:



1. To plot the queue delay v/s time, first create a python script that parses the “**tcp-example3.tr**” file and outputs the queueing events and their delays as attached in the .zip file named “**extract\_queue\_delay.py**”. This will result in a “**queueing\_delay3.dat**” file. Compile the script “**queue3.gp**”:

**set term postscript eps color**

**set output ‘queue3.eps'**

**set ylabel 'Queueing Delay (s)'**

**set xlabel 'Time (s)'**

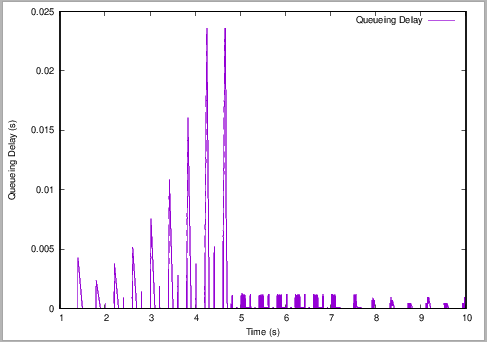
**plot 'queueing\_delay3.dat' using 1:2 with lines title 'Queueing Delay'**

Run: **gnuplot queue3.gp**

This will create a “**queue3.eps**” file. To view this file,

Run: **gv queue3.eps**

This will result in the following plot:

****

1. In the first plot (7 Mbps, 10 ms), the queuing delay peaks higher before reducing.

In the second plot (10 Mbps, 100 ms), the queuing delay peaks at a lower value.

Higher base delay (100 ms) in the second plot results in longer feedback loops.

The first plot demonstrates that with lower bandwidth (7 Mbps), the link becomes saturated sooner, leading to higher queuing delays.

A higher base delay (100 ms) increases the end-to-end round-trip time (RTT), which could affect how quickly TCP reacts to congestion. However, in this case, the higher bandwidth compensates by reducing the queuing delay.