# CN ASSIGNMENT 3

**Question 1:**

**a.**

Theoretically, the maximum expected value of throughput would be minimum of the 2 connection between the 2 nodes, i.e.,

Throughput = min(bandwidth(N0→N1), bandwidth(N1→N2))
= min(10Mbps, 7 Mbps)
**= 7Mbps**

The reason for the same is that the bandwidth between the N1→N2 nodes is bottleneck bandwidth for the entire communication channel from N0 to N2. So, the data can flow from N1 to N2 at the maximum bandwidth of 7Mbps, which is the rate at which N2 will receive data.

**b.**

<u>Considering Transmission Delay:</u>

For entire connection from N0 to N1:

Propagation Delay = 100+10 = 110ms
Transmission Delay N0-N1 link = (1460x8 bits)/(10x$10^6$ bps) = 1.168ms
Transmission Delay N1-N2 link = (1460x8 bits)/(7x$10^6$ bps) = 1.668ms
End to end delay = 2x110 + 1.168 + 1.668 = 222.836 ms
BDP = 7x$10^6$bps x 222.836x$10^{-3}$s = 1559.852Kb = 1.56Mb
In terms on number of packets:

BDP = 1.56Mb/(1460x8bits)
= 133.54 packets
≈ 134 packets

For link N0-N1:

Propagation Delay = 100ms
Transmission Delay = 1.168ms
BDP = 10Mbps x 201.168ms = 2011.68Kb = 2.012Mb
In terms of number of packets:

BDP = 2.012Mb / (1460 x 8 bits)
= 172.26 packets
≈ 172 packets

For link N1-N2:

Propagation Delay = 10ms
Transmission Delay = 1.668ms
BDP = Mbps x 21.668ms = 151.676Kb = 0.152Mb
In terms of number of packets:

BDP = 0.152Mb / (1460 x 8 bits)

= 12.98 packets
≈ 13 packets

Not Considering Transmission Delay (as its very small):

BDP (for entire connection from N0 to N2):

BDP = Bandwidth x Round Trip Delay
= 7 Mbps x (100x2 ms + 10x2 ms)
= 7 Mbps x 220 ms
= 1540 Kb
= 1.54Mb

In terms of Number of packets:

Each packet is of 1460 bytes = 1460 x 8 bits
BDP = 1540 Kb / (1460 x 8 bits)
= 131.85 packets
**≈ 132 packets**

For individual links:

N0 → N1:

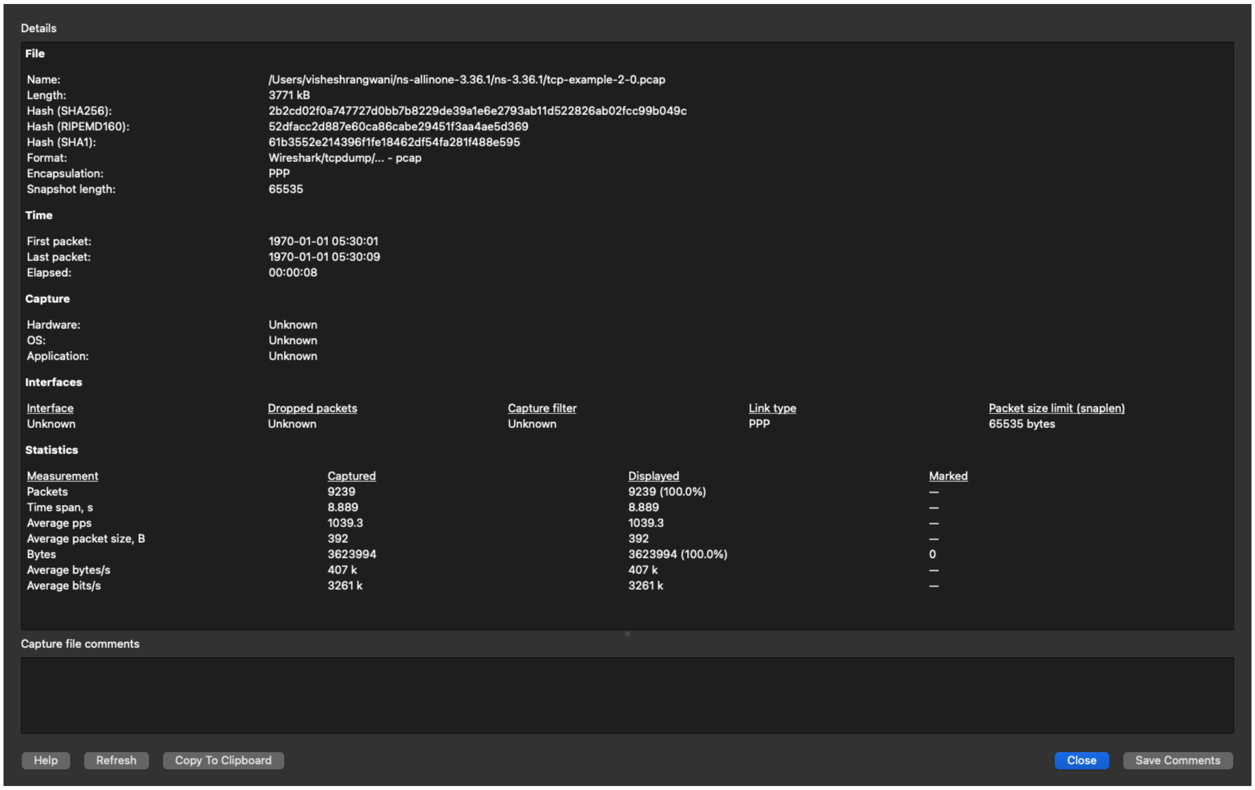
BDP (in no of packets) = 10 Mbps x 200 ms / (1460 x 8 bits)
= 2 Mb / (11680 bits)
= 171.23 packets
**≈ 171 packets**

N1 → N2:

BDP (in no of packets) = 7 Mbps x 20 ms / (1460 x 8 bits)
= 140 Kb / (11680 bits)
= 11.98 packets
**≈ 12 packets**

**c.**

After making required changes to the code and running the program using './ns3 run scratch/tcp-example.cc', we get the 'tcp-example-2-0.pcap' file.
Attached below is a screenshot of the 'Capture File Properties' window in the Statistics tab of Wireshark upon opening the 'tcp-example-2-0.pcap' file in wireshark. In the statistics tab we can see that the throughput is shown to be 3261 Kbps = **3.261 Mbps**.



Details

**File**

| | |
|---|---|
| Name: | /Users/visheshrangwani/ns-allinone-3.36.1/ns-3.36.1/tcp-example-2-0.pcap |
| Length: | 3771 kB |
| Hash (SHA256): | 2b2cd02f0a747727d0bb7b8229de39a1e6e2793ab11d522826ab02fcc99b049c |
| Hash (RIPEMD160): | 52dfacc2d887e60ca86cabe29451f3aa4ae5d369 |
| Hash (SHA1): | 61b3552e214396f1fe18462df54fa281f488e595 |
| Format: | Wireshark/tcpdump/... - pcap |
| Encapsulation: | PPP |
| Snapshot length: | 65535 |

**Time**

| | |
|---|---|
| First packet: | 1970-01-01 05:30:01 |
| Last packet: | 1970-01-01 05:30:09 |
| Elapsed: | 00:00:08 |

**Capture**

| | |
|---|---|
| Hardware: | Unknown |
| OS: | Unknown |
| Application: | Unknown |

**Interfaces**

| Interface | Dropped packets | Capture filter | Link type | Packet size limit (snaplen) |
|---|---|---|---|---|
| Unknown | Unknown | Unknown | PPP | 65535 bytes |

**Statistics**

| Measurement | Captured | Displayed | Marked |
|---|---|---|---|
| Packets | 9239 | 9239 (100.0%) | — |
| Time span, s | 8.889 | 8.889 | — |
| Average pps | 1039.3 | 1039.3 | — |
| Average packet size, B | 392 | 392 | — |
| Bytes | 3623994 | 3623994 (100.0%) | 0 |
| Average bytes/s | 407 k | 407 k | — |
| Average bits/s | 3261 k | 3261 k | — |

Capture file comments

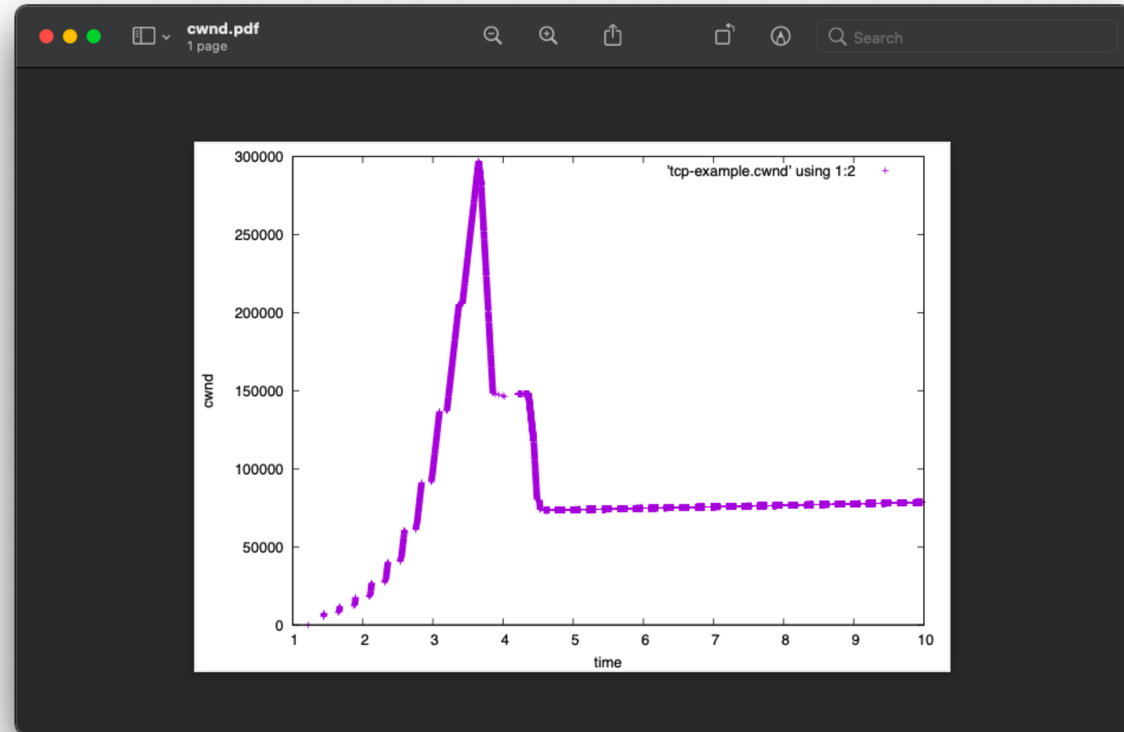
Help Refresh Copy To Clipboard Close Save Comments

**d.**

**No**, the actual average throughput is not equal to the theoretical maximum throughput. The reason for this is that the maximum throughput is the throughput that can be achieved in ideal conditions. But in our scenario, the ideal conditions do not occur as there is a model set up for errors. Not only this, the buffer of the queue is also of limited size, so packets may be dropped and delayed in this buffer. Sometimes, there is redundant retransmission if the timeout value is not enough.

**e.**

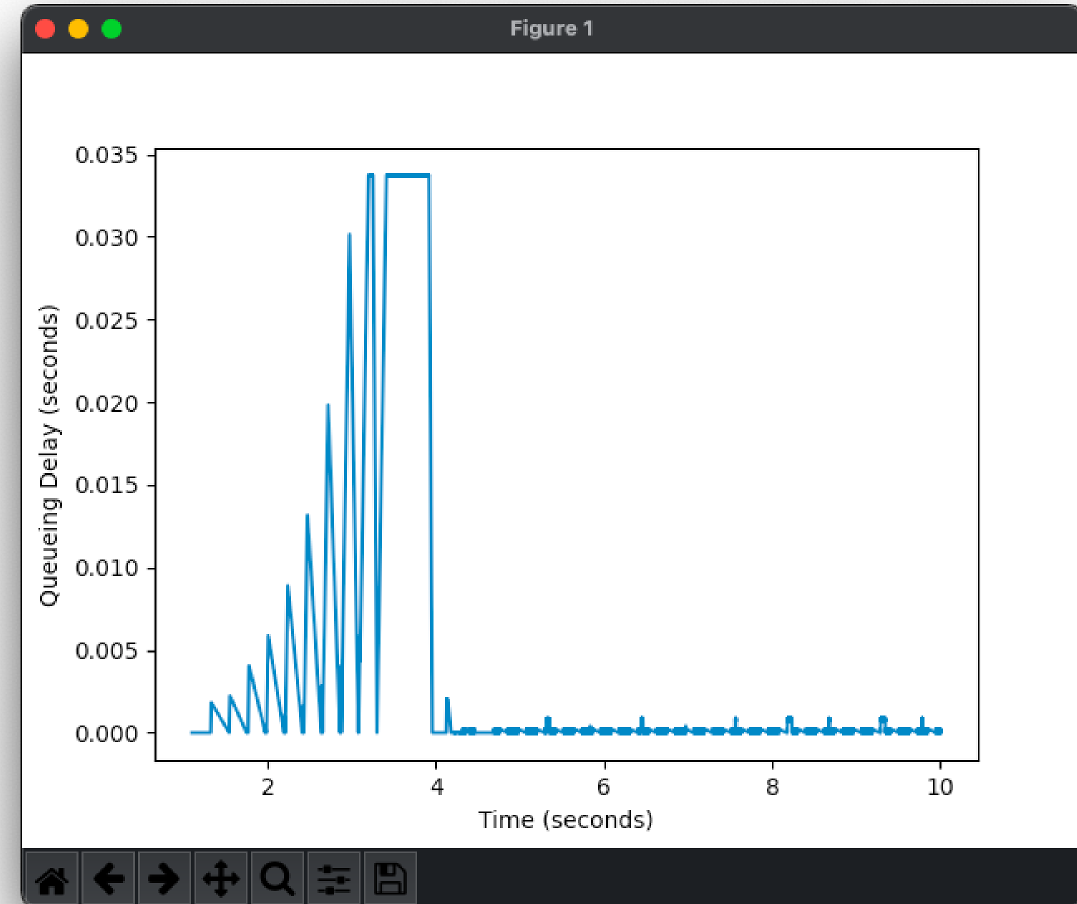
The script as given in assignment's description was used to make the file 'example.gp'. The script uses the file 'tcp-example.cwnd' for data. After this, the command 'gnuplot example.gp' was used to plot the graph between CWND and time ('cwnd.eps') as shown below. The 3 files are attached also.




**f.**

For this part, a python script is written plotting the queueing delay for the queue of Node 1 for Node 2. Matplotlib is used to get the plot. Screenshot attached below. Both Queueing Delay and Time are in seconds.

In the script, the Queueing Delay for a packet is computed by subtracting the time for enqueueing of the packet from dequeue time and then plotting that against time.




**g.**

Yes, the plots for 1e. and 1f. are related.
From the graphs, we can clearly see that the queueing delays at N1 are increasing with each packet when the CWND is also increasing up to around the 3.5 second mark. The reason for this is that the CWND increases exponentially in the slow start phase. In that case, the packets are transmitted at a high rate and hence the queueing delays also increase. The queueing delays show a saw tooth pattern initially because at N1, the packets according to size of CWND arrive in and as they come, the delay increases and then decreases when a particular window of size CWND is forwarded. In the next window, again more packets arrive hence a higher peak.

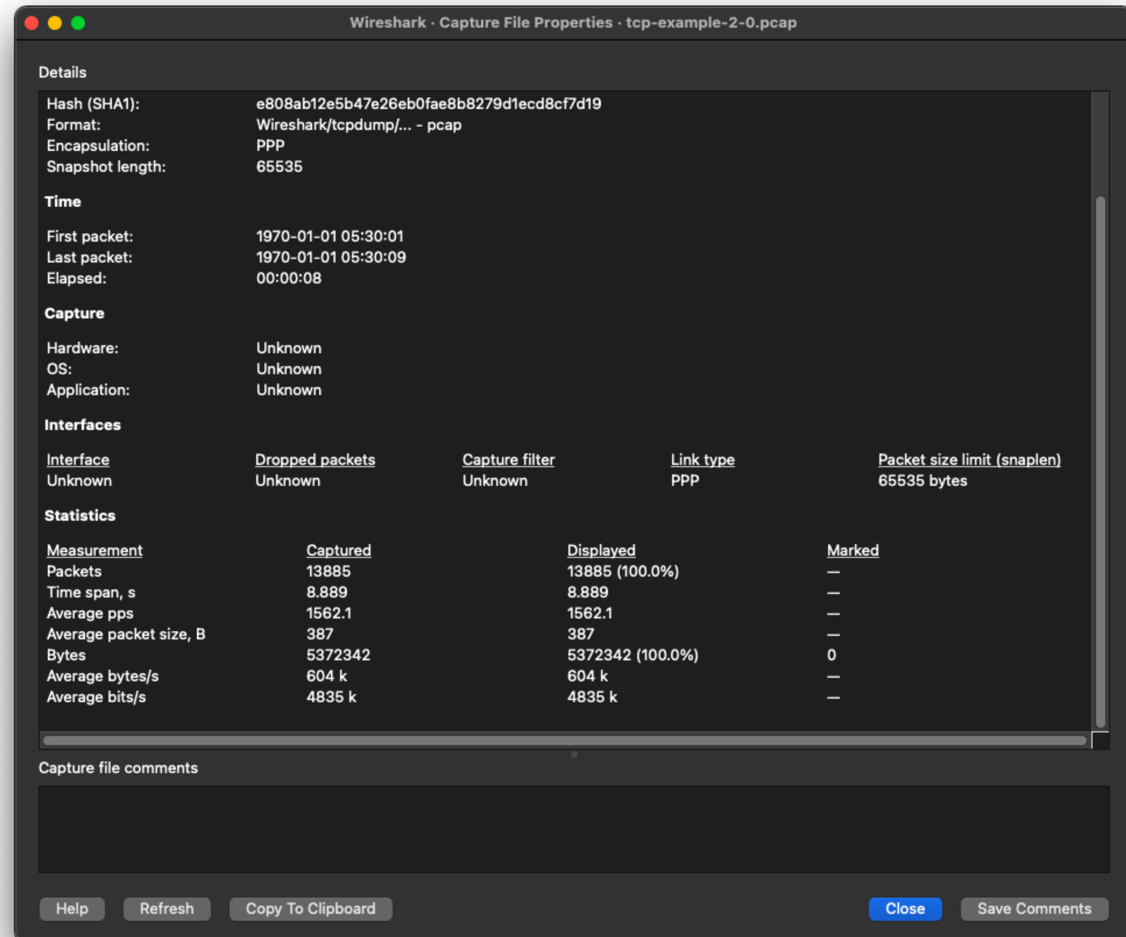
After the slow start phase, ssthresh is set for CWND and after some time, the CWND size becomes more or less stable. Although it varies as per the design of Congestion Avoidance Phase, its value remains in a small window.

When this happens the queuing delays also fluctuate minutely from 0 as the network begins transmission at a somewhat stable rate.

**Question 2:**

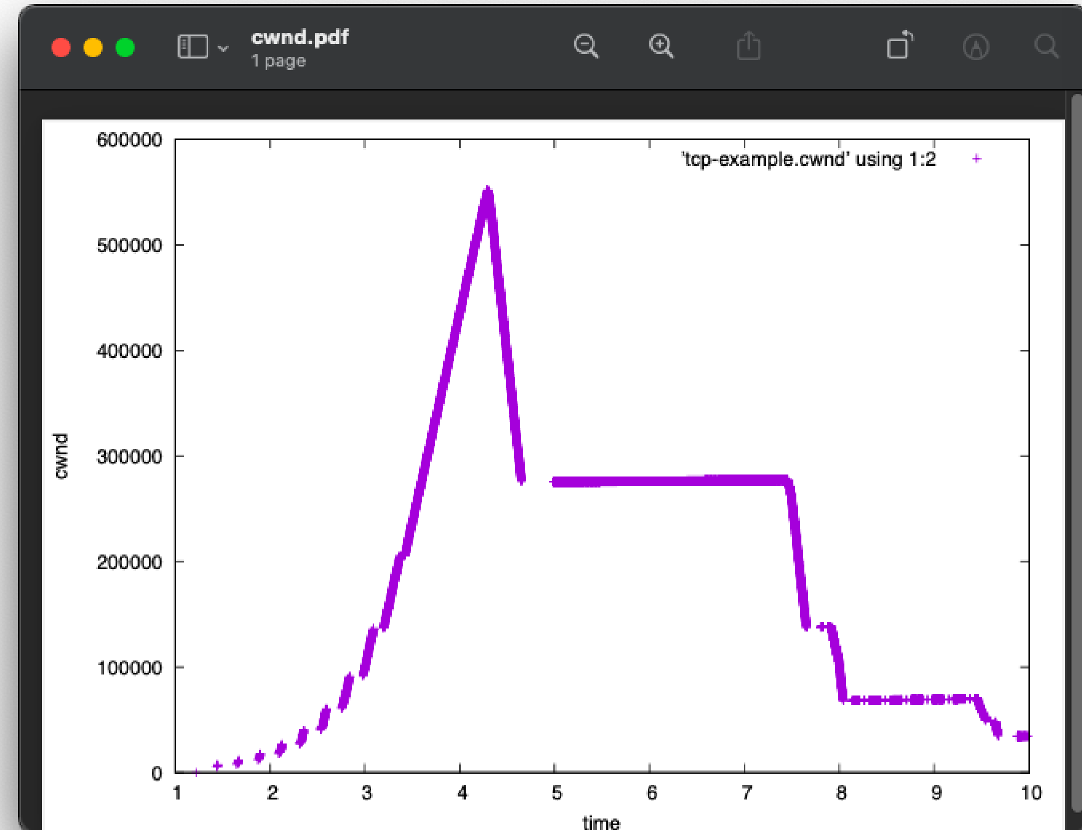
**a.**

Average computed throughput as shown in Wireshark (Statistics > Capture File Properties): 4835 Kbps = 4.835 Mbps



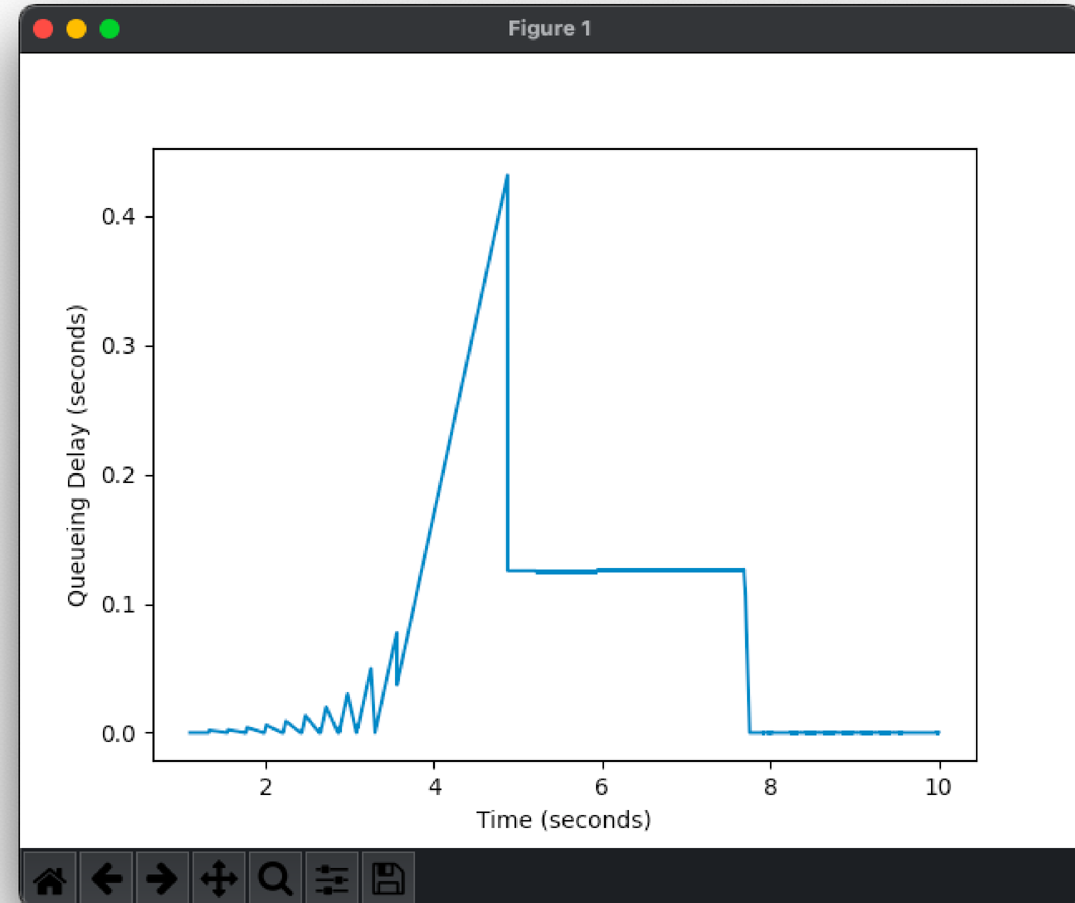

**b.**

As in part (e) of the previous question, we can plot the graph for CWND vs time using the given script.
Screenshot below:




**c.**

The script given in ‘queueScript.py’ (explained in the first question) is used to plot the Queueing Delay vs Time. Both Queueing Delay and Time are in seconds. Screenshot attached below.




**d.**

On comparing the 2 plots, we can see that the peak for CWND in the latter case is higher than the first case. Also, the Slow Start phase exists for more time in the second case. The reason for the 2 observations is that since the queue size in the latter case is higher, the packet loss and timeout in the network occurs later as compared to the first case.

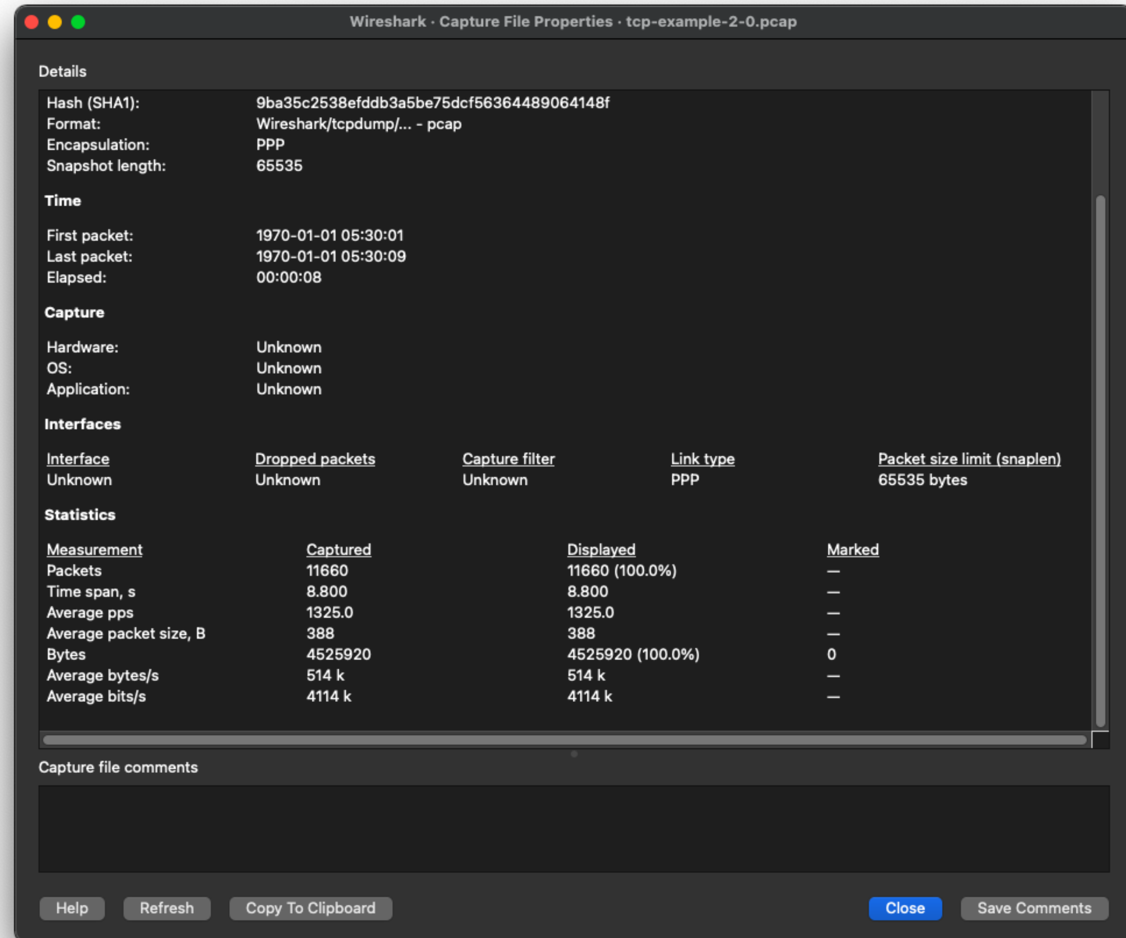
Following the same reasoning, the CWND during the congestion avoidance phase in the 2nd case, the ssthresh is higher.

Towards the end CWND sizes are close in both cases.

**Question 3:**

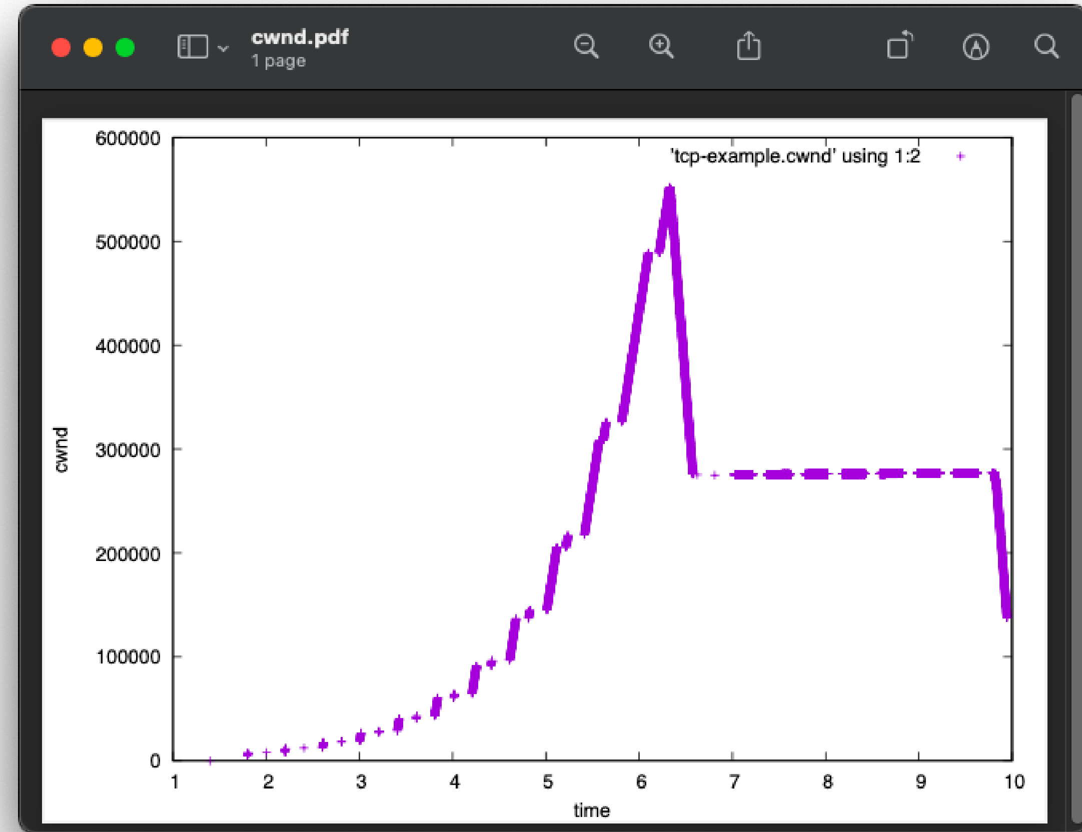
**a.**

Average computed throughput as shown in Wireshark (Statistics > Capture File Properties): 4114 Kbps = 4.114 Mbps



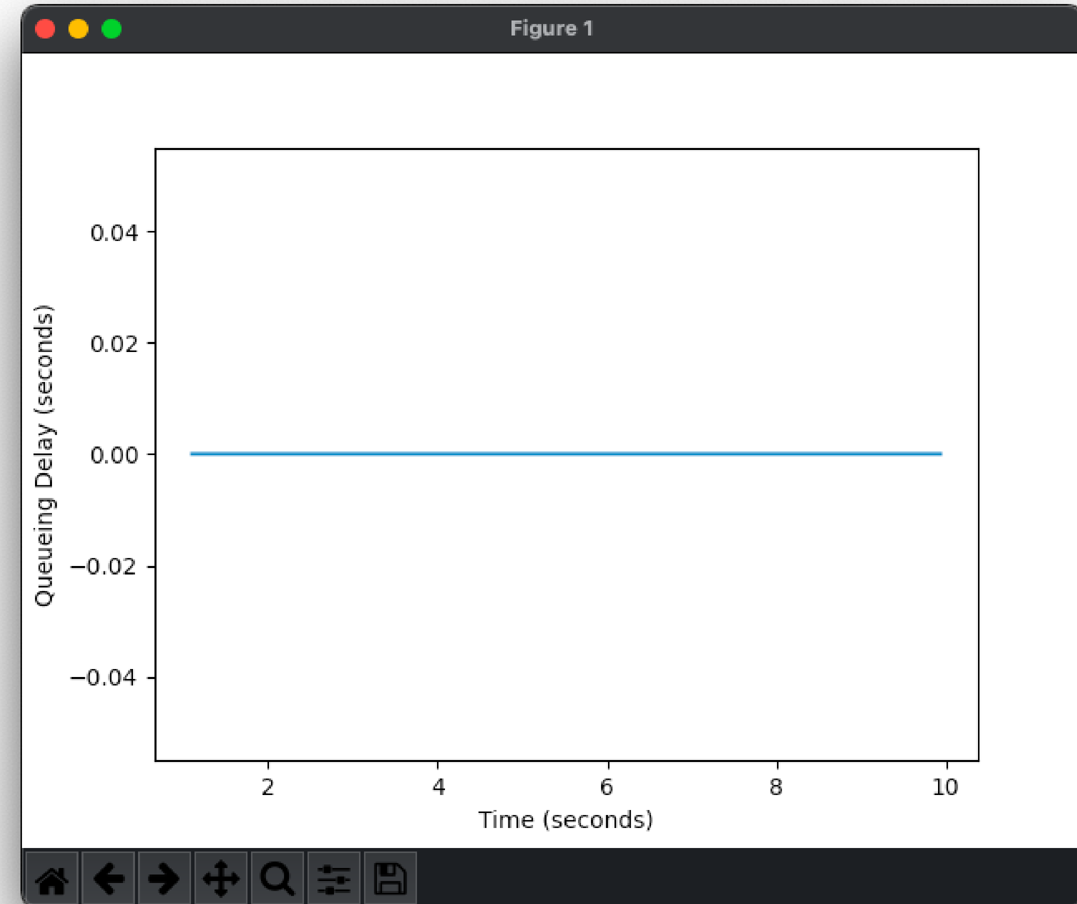

**b.**

The plot is made using the given script. Screenshot is attached.




**c.**

The graph has been plotted using the script ‘queueScript.py’, which is explained in the 1st Question. Screenshot attached:




**d.**

The Queueing delay plot for Question 3 is constant 0 throughout. The reason for the same is that the bandwidth as well as the delay for both N0-N1 link and N1-N2 link is the same. So, there would be no queueing happening at the router (N1) as the data incoming and outgoing rate would exactly be the same.

On the other hand, in case of question 1, the bandwidth as well as the delays are different for the 2 links, hence the Queueing delay is significant and varies depending on the CWND.