CS224M Programming Assignment 2
Understanding TCP

Varun Patil
(160010005)

August 31, 2018

Purpose

To study the working of TCP, including but not limited to the window size for congestion control, queueing delays, how the sizes of the queue are related to various factors, throughput and effects of headers on the same with the help of ns-3 network simulator by simulating a single point-to-point network link. Answers provided to questions given at the course webpage.

Setup

All simulation was perfomed using ns-3, while analysis and plotting of the data was done using Python (Anaconda 4.3.16). All simulations were performed in a unix-like environment.

Configuration

The simulation is carried out by running the simulation code (in C++) inside ns-3. The following configurable options (with default values) are provided in the main function.

`std::string bandwidth = "5Mbps";`
(Adjusts the bandwidth of the link between the two nodes)
`std::string delay = "5ms";`
(Adjusts the propagation delay of the link)
`double error_rate = 0.000001;`
(The fractional error rate of the link)
`int queuesize = 10;`
(Maximum size of the queue to be built at any node)
`int simulation_time = 10;`
(Time for which to run the simulation)
`int InitialSlowStartThreshold = 12000;`
(Initial value of ssthresh in bytes. Is a very large number by default)

Methodology

On running the simulator, packet capture files for both ends, a trace file of all events and a file with data for the size of the congestion window are generated in the parent directory of the simulator. Following this, details of the packets from the capture files are extracted using tshark (see `analyze.sh`) and stored in separate files. Of these, the congestioni window data is used to plot the size of the congestion window against time, the capture file data is used to find out the throughput while the trace file is used for extracting data about queueing at the sending node. It has been assumed (and checked) that no queues are built up at the receiving end and all queueing delays are negligible.

Exercises

1. Run the simulation with the default parameters and answer the following questions.

   - What is the average throughput of the TCP transfer? What is the maximum expected value of throughput? Is the achieved throughput approximately equal to the maximum expected value? If it is not, explain the reason for the difference.
   - How many times did the TCP algorithm reduce the cwnd, and why?

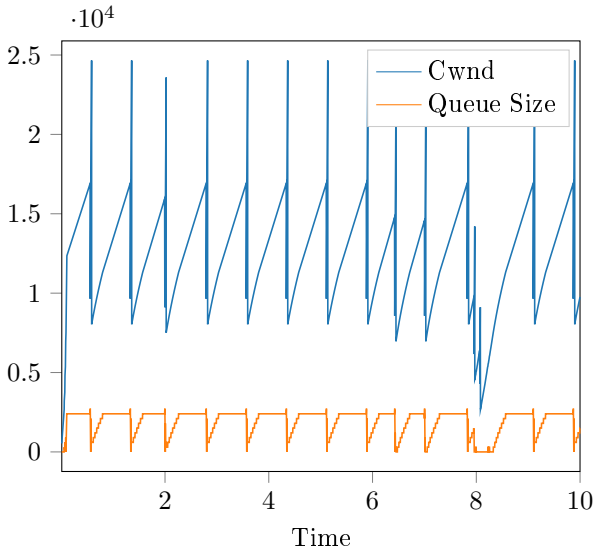
   The parameters used for this question are the same as given in section Configuration.
   Upon summing up the total lengths of the packets that were received and sent and dividing by the total time taken, we obtain the following information.

```
Total Time : 9.999637s

Total TCP Bytes Sent : 5580832
TCP Sending Throughput : 4464 kbps
TCP Receiving Throughput : 4460 kbps
TCP bits lost : 47168
TCP bits lost (as percentage of sent) : 0.11%

Total IP Bytes Sent : 6122364
IP Sending Throughput : 4898 kbps
IP Receiving Throughput : 4892 kbps

Total Bytes Sent (on wire) : 6143192
Sending Throughput : 4914 kbps
Receiving Throughput : 4909 kbps
```

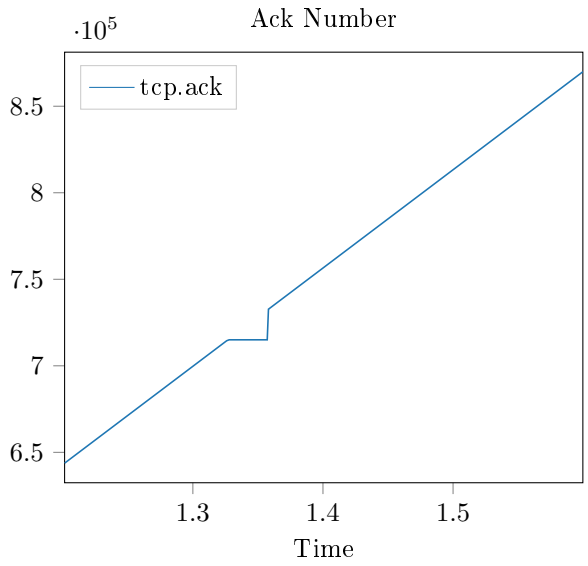
   The following inferences can be drawn from the above data:

   - The raw throughput of the wire (total data transmitted) at 4.9 Mbps is quite close to the maximum expected value of 5 Mbps (as configured).
   - The throughput excluding IP headers is only slightly less than the raw throughput, indicating that IP headers account for only a small amount of data.
   - Throughput excluding TCP headers (i.e. of the actual TCP payload) is significant less at 4.4 Mbps. This indicates that TCP is a much more heavy protocol, with the headers accounting for a significant fraction of the data transmitted.
   - There is a slight difference in the sending and receiving throughput at all levels. This can be accounted for by dropped packets, which are around 0.11% of total.

   For the second part, we plot the size of cwnd against time.



Further, if we plot the sequence number of the ACKs received against time, we observe it to be a straight line. However, if we zoom in to the points where cwnd reduces, we observe the following graph.



The following inferences can be drawn from the two figures above:

- The size of cwnd increases quadratically at the start after Slow Start. After this, the size increases linearly and reduces periodically. During the time of simulation, it reduces 13 times.
- Multiple ACKs with the same sequence number are received before the reduction in cwnd. After cwnd reduces, the sequencce number jumps suddenly to the linearly expected value. This confirms that the size of cwnd reduces after multiple DupACKs are received.
- The queue gets filled quickly before the size of cwnd reduces.

2. Start with the default config. Change the link bandwidth to 50Mbps (from 5 Mbps).

   - What is the average throughput of the TCP transfer? What is the maximum expected value of throughput? Is the achieved throughput approximately equal to the maximum expected value? If it is not, explain the reason for the difference.
   - What other parameters in the simulation (amongst the ones exposed to you above) can you change to make sure that the throughput is close to the maximum expected value, for this link bandwidth? (Try out a few different simulations, and see what gets you close to the maximum.)

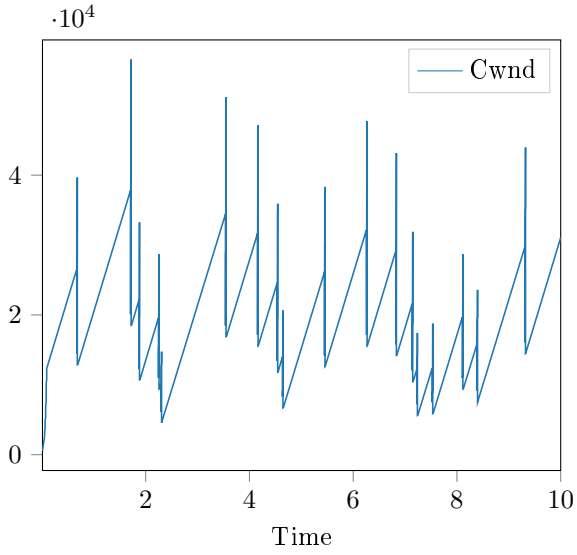
   Setting bandwidth to 50 Mbps, we get the following throughputs:

   ```
   TCP Receiving Throughput : 14668 kbps
   Receiving Throughput : 16145 kbps
   ```

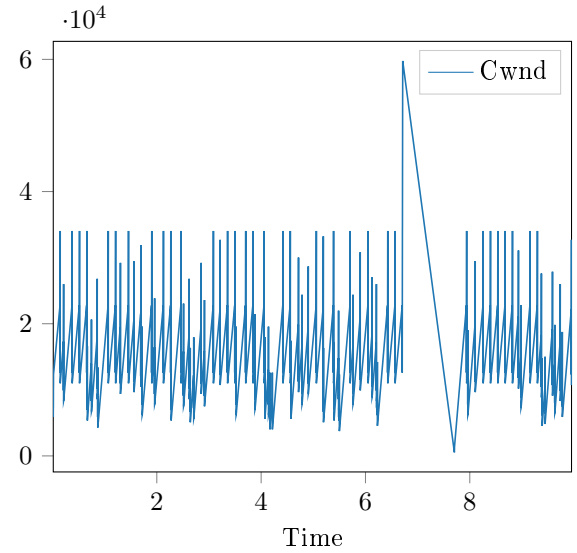
   Thus, while increasing in absolute terms, the throughput gets much worse when compared to what is expected. Further, it can be observed that on reducing the delay to 1ms, we get,

   ```
   TCP Receiving Throughput : 37458 kbps
   Receiving Throughput : 41232 kbps
   ```

   This can be explained by looking at the size of cwnd. At a latency of 5ms, we see the following:



At a latency of 1ms, we get,



This shows that at a delay of 1ms, TCP spends much less time in congestion avoidance than at 5ms, which leads to a much higher throughput, since the average size of cwnd remains higher and the link is utilized much better. Thus, we can say that the throughput is lower at the higher latency because every time after fast recovery, TCP doesn't receive ACKs fast enough to increase cwnd quickly.
(Note: the spike seen at $t \approx 7$ sec was much higher and was reduced just for clarity)

Reducing the error rate a hundred times over while keeping delay 5ms also leads to a rise in the throughput (as expected) to

```
TCP Receiving Throughput : 36278 kbps
Receiving Throughput : 39932 kbps
```

Increasing queue size keeping everything else the same, however, doesn't have any effect on the throughput, since the is almost no queue anyway.

Thus, we can conclue that low latency and a low error rate is required for TCP to be able to fully utilize the link bandwidth.

3. Start with the default config. Change the link delay to 50 ms.

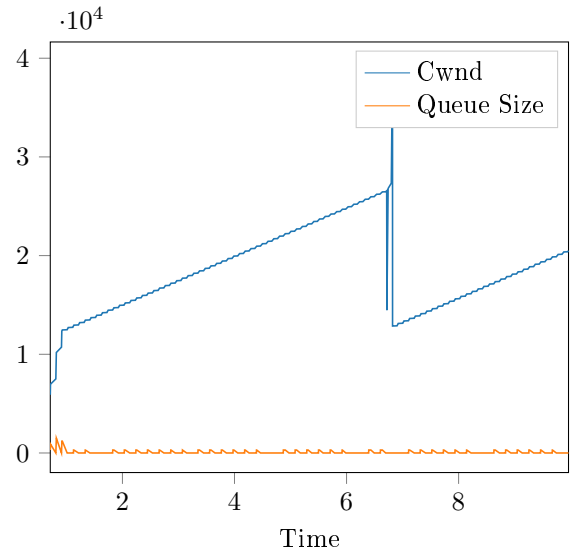
- What is the average throughput of the TCP transfer? What is the maximum expected value of throughput? Is the achieved throughput approximately equal to the maximum expected value? If it is not, explain the reason for the difference.
- What other parameters (amongst the ones exposed to you above) can you change to make sure that the throughput is close to the maximum expected value, for this link delay? (Try out a few different simulations, and see what gets you close to the maximum.)

In this case, the throughput and lost packets obtained initially are

```
TCP Receiving Throughput : 1299 kbps
Receiving Throughput : 1430 kbps
TCP bits lost (as percentage of sent) : 1.18%
```

Thus, the throughput is much lower than the expected 5 Mbps. We can also see that amount of dropped packets has increased to 1.18% from 0.11% for a latency of 5ms.

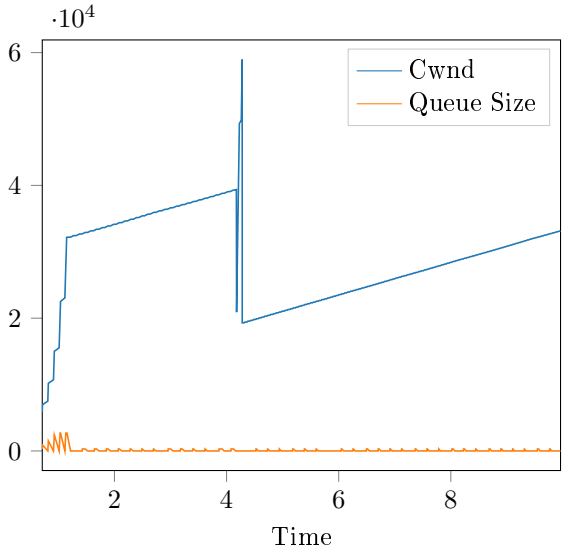
As with Q2, this can be explained on the basis of the size of cwnd. Here, we observe that the rate of increase in the size of cwnd is very low due to the high delay, leading it to spending most of its time in congestion avoidance.



Since this is a short simulation, increasing the initial ssthresh has a positive effect on throughput increasing it to

```
TCP Receiving Throughput : 2068 kbps
Receiving Throughput : 2276 kbps
```

The reason for this can be seen easily in the following figure, since increasing ssthresh reduces the time span of the congestion avoidance phase till the first recovery.



This can be overcome by running a longer simulation of 100 seconds, where the effect of the initial time becomes negligible, giving,

```
TCP Receiving Throughput : 1466 kbps
Receiving Throughput : 1614 kbps
```

Reducing or increasing the link speed has little to no effect on the throughput for a link with such a high delay. As earlier, a decrease in the error rate has a huge positive effect and an increase in the maximum queue size has no effect, since the queue is unoccupied (as can be seen from the figure).

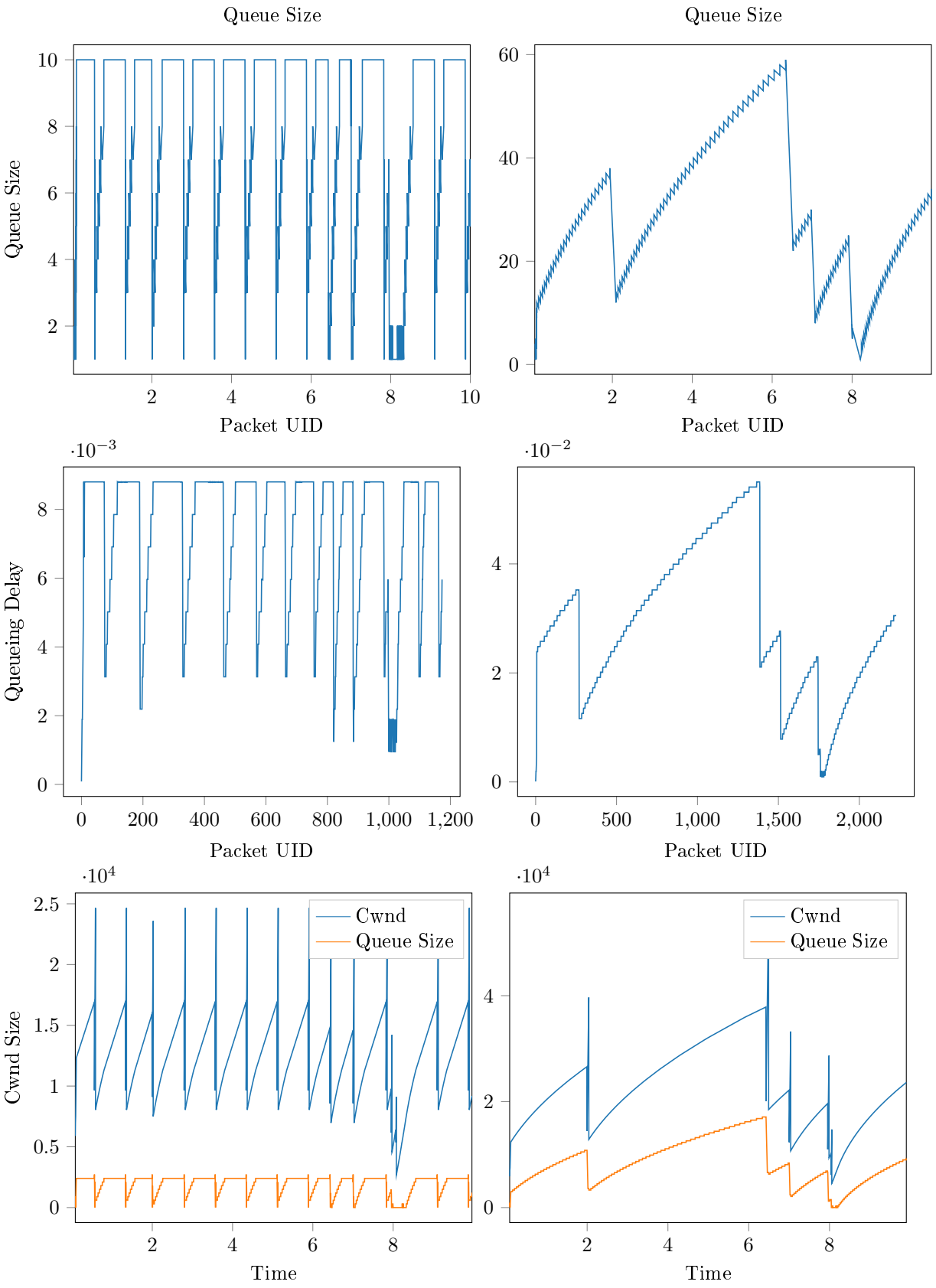
4. Start with the default config. Change the queue size to 1000 packets.

   - Compare the TCP throughput in both cases and explain what you see. Further, explain what happens to the queue occupancy, queueing delay, and cwnd in both cases. (If you cannot write scripts to get the queueing delay and queue occupancy, make an educated guess.)
   - What is the optimum queue size that must be used in this simulation? Justify your answer.

   Comparing various parameters before and after change in maximum queue size,

   ```
   //queuesize = 10
   TCP Receiving Throughput : 4460 kbps
   Receiving Throughput : 4909 kbps

   //queuesize = 100
   TCP Receiving Throughput : 4497 kbps
   Receiving Throughput : 4950 kbps
   ```

   The following plots show the difference between the various observables as time proceeds (Note: Figures on right represent a 1000 packet queue size. Some details have been removed due to technical issues. In the plots produced by python, the queueing delay and the queue size can be seen oscillating as they approach maximum. Please also note that the figures have **different scales**)



It can be seen that the throughput is slightly higher with a larger queue. This is because while a larger queue helps keep more packets with the device, hence increasing cwnd, the queueing delay also increases due to a bigger queue. This balances out and the throughput increases only by a small amount. An ideal maximum queue size would be around the number

of packets that get filled up at the end of the congestion avoidance phase. This way, there will not be a big queue, but the queue would not be filled completely either.

Created using LaTeX with Overleaf and TexStudio with MiKTeX