1. Run the simulation with the default parameters and answer the following questions.
What is the average throughput of the TCP transfer?
We see line total 4994985 Bytes of information is transferred.
For each 536 bytes information, the packet size is 590 bytes (open pcap file in wireshark).
Thus, [(4994985*590/536)*8]Bits of total information along with header is recorded.
This gives an average throughput of 4.887Mbps.
This can also be seen directly in conversations through statistics.

What is the maximum expected value of throughput?
Maximum value of throughput is 5Mbps.

Is the achieved throughput approximately equal to the maximum expected value? If it is not, explain the reason for the difference.
Despite the fact, the values are quite close, they aren't equal. This is primarily because that packets are dropped when travelling across a computer network. The dropped packets need to be resent accounting to lowering throughput. Also, the packets we captured, do not include the header size of network card such as ethernet header. This utilises additional bytes which when unaccounted leads to lower than ideal values.

How many times did the TCP algorithm reduce the cwnd, and why?
The cwnd reduced 43 time in the total duration as seen from running the cwnd_drop.py script.

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?
We see that the achieved average throughput is 16.04Mbps which is quite far from the ideal value(Maximum expected value) of 50Mbps.

If it is not, explain the reason for the difference.
The non-ideal is because the delay time gives a more significant role than it had to play in bandwidth of 5Mbps. Though TCP performs quite efficiently in the congestion control but it lags in its slow start algorithm. During the slow start, till the acknowlegment reaches to client, the cwnd cannot increase its size and thus cannot send a packet furthur which results in decrease of average throughput than expected.

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.)
As expected, decreasing the delay time greatly increases the average throughput. Also when we increase the simulation time, we see the average throughput increase, this is because the slow start phase has gone and TCP has estimated the threshold level. Now, the congestion avoidance phase is efficiently managed, so increasing the average throughput.

3. Start with the default config. Change the link delay to 50 ms.
What is the average throughput of the TCP transfer?
Average thoughput = 0.777Mbps.

What is the maximum expected value of throughput?
Maximum value of throughput is 5Mbps.

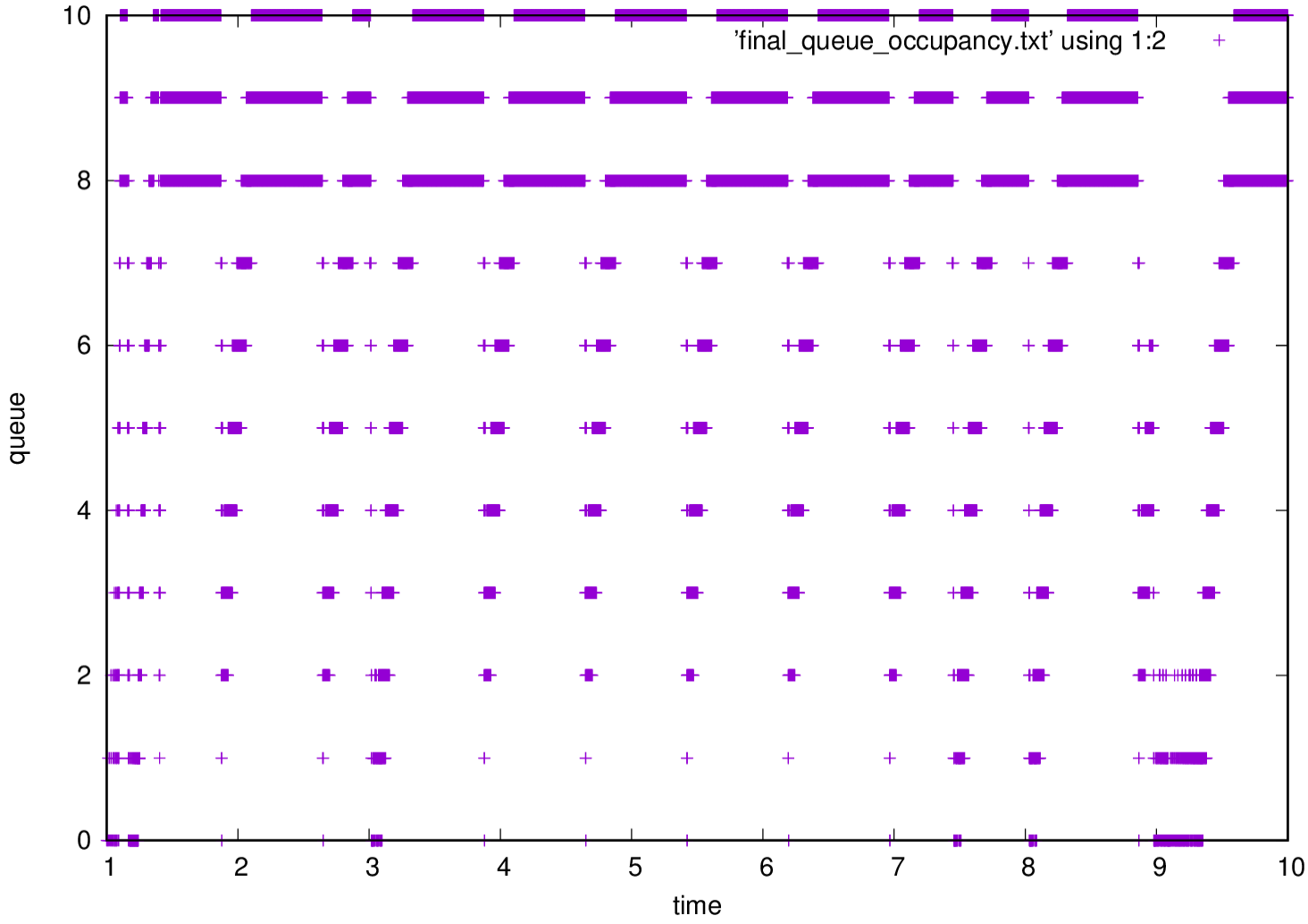
Is the achieved throughput approximately equal to the maximum expected value? If it is not, explain the reason for the difference.
Even this case has a huge diviation from the ideal value This is basically because of the mechanism by which Reno TCP works. As the time delay is high, the time taken for recieveing acknowlegement is high and thus, the cwnd retains its size and thus no packet is sent. This results in a huge delay during the slow start. Thus the overall average falls.

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.)
As, the bandwidth isn't used completely, so, we can lower the bandwidth so that we get average thorughput equal to maximum expected value. Though reducing the number of nodes would practically help, in this case we have only 2 nodes which are absolutely necessary(client and server)

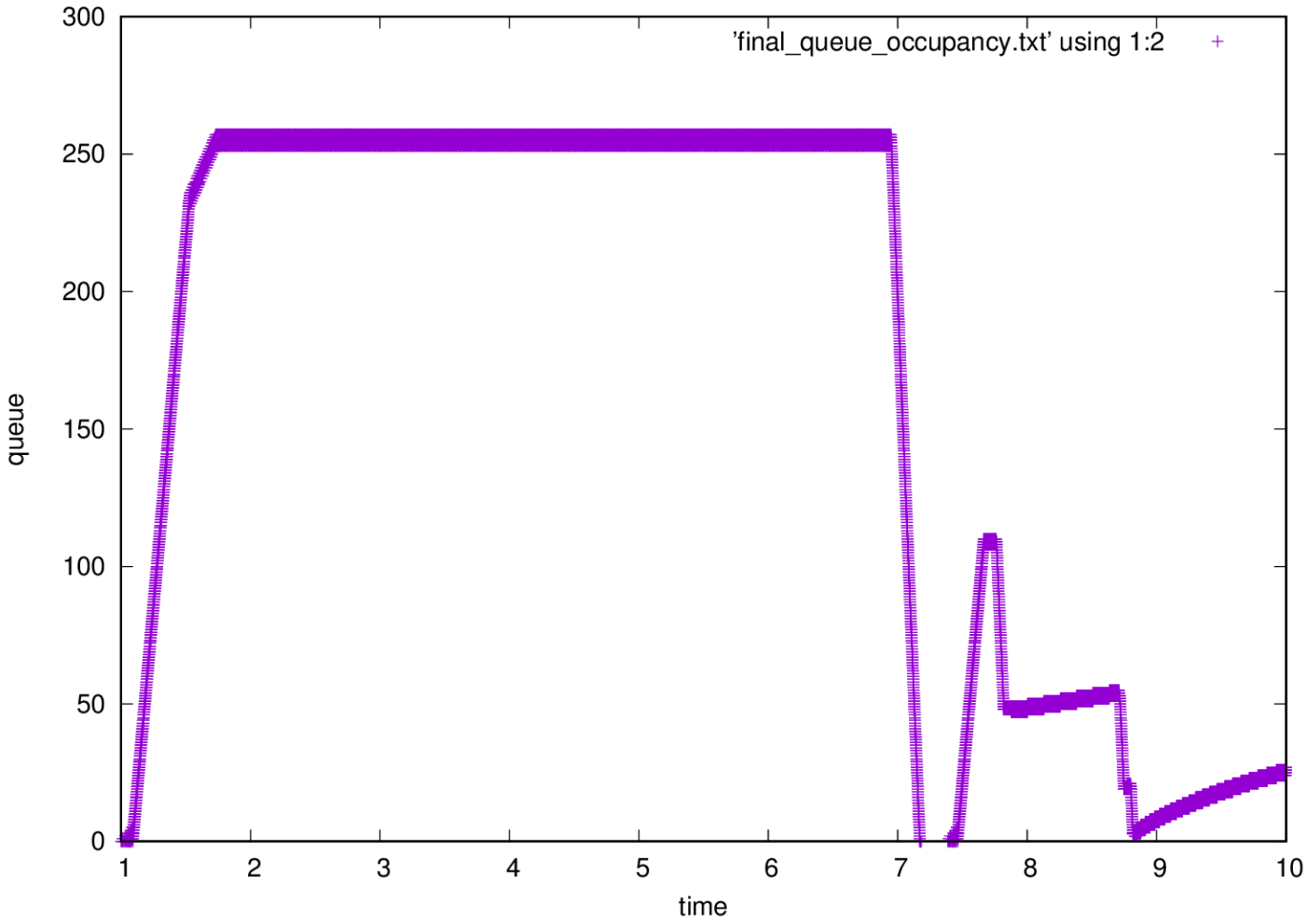
4. Start with the default config. Change the queue size to 1000 packets.
Compare the TCP throughput in both cases and explain what you see.
When changing the queue size to 1000 we get average throughput = 4.79Mbps, as compared to 4.887Mbps when queue size is 10.

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
Used script1.py then script2_extracting_final_queue_occupancy.py to analyse data and lastly plot_queue.gp to plot data

Default queuing occupance:



When queue size is changed to 1000:



TCP Reno is a cliff-based strategy: packets must be lost before the sender reduces the window size.

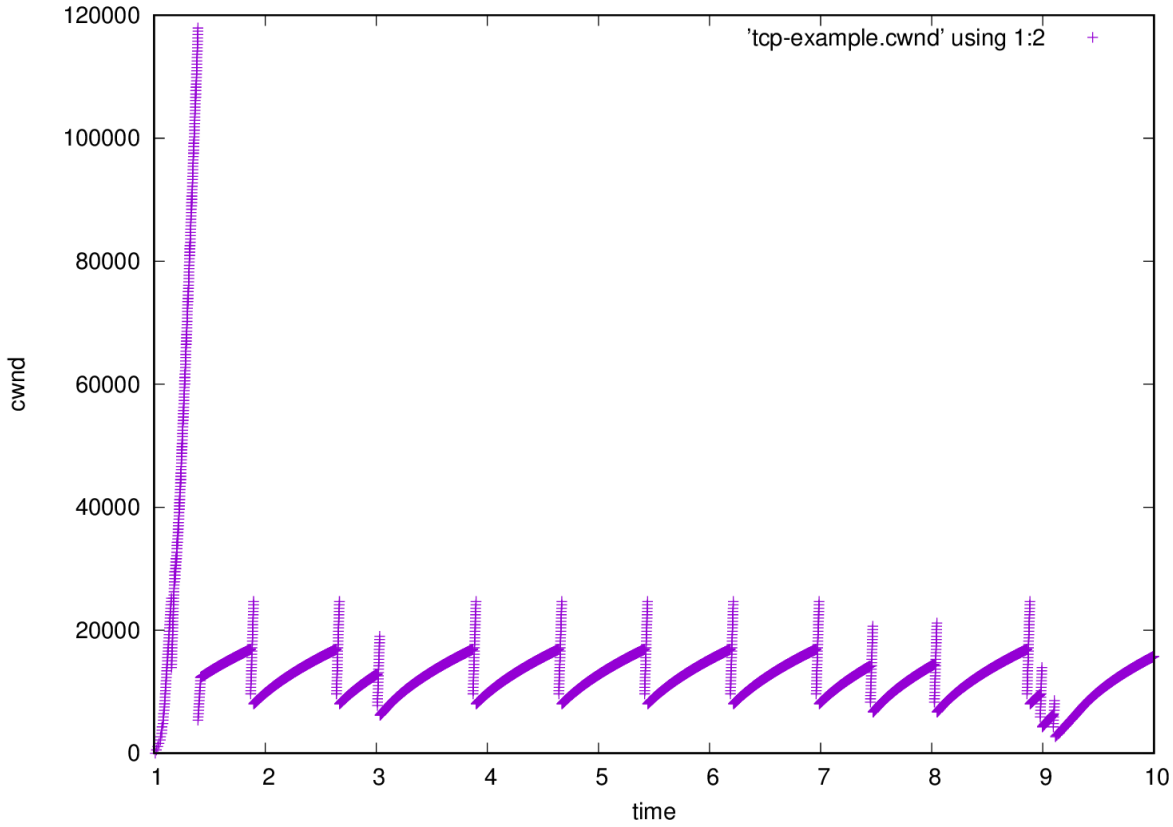
We see that when queue size is increased, TCP is trying to utilize the complete bandwidth which was not previously possible when queue size was just 10. queue is being filled up because of the delay in transmissions. When queue size was just 10, the packets would fill up queue, TCP would realize this quite early and reduce it's transmission so to clear the queue thus increasing the average throughput than than the case when we have enough buffer available.

Queue occupancy is more when queue size is more because we get average thorughput less and thus on an average each packet will need to wait a little longer than it had to for lower queue size.

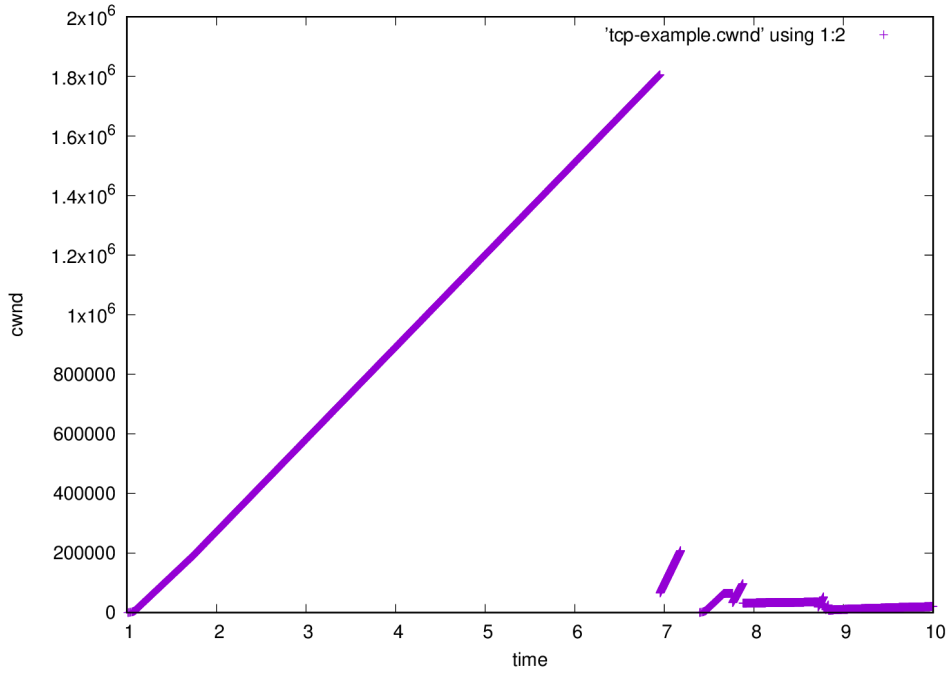
On varying the queue length in simulation, we see that queue length around 14 gives the fastest average throughput of 4.909Mbps (found by running waf scripts multiple times changing queuesize).

Cwnd:

Queue=10



Queue=1000



We see that the cwnd increases for certain time but then decreases abruptly. This abrupt decrease comes when packets start to drop and TCP comes to know network is being conjested.