ECE466 Computer Networks II
Lab 3 Packet Scheduling
March 24th, 2014

Michael Law
997376343

Part 1. FIFO Scheduling

Exercise 1.1 Traffic generator for compound Poisson traffic

The traffic generator for scaling the Poisson traffic is built in part1/TrafficGeneratorPoissonRescale.java. To scale the Poisson traffic to N Mbps, the delay between each packet was scaled by 1/N.

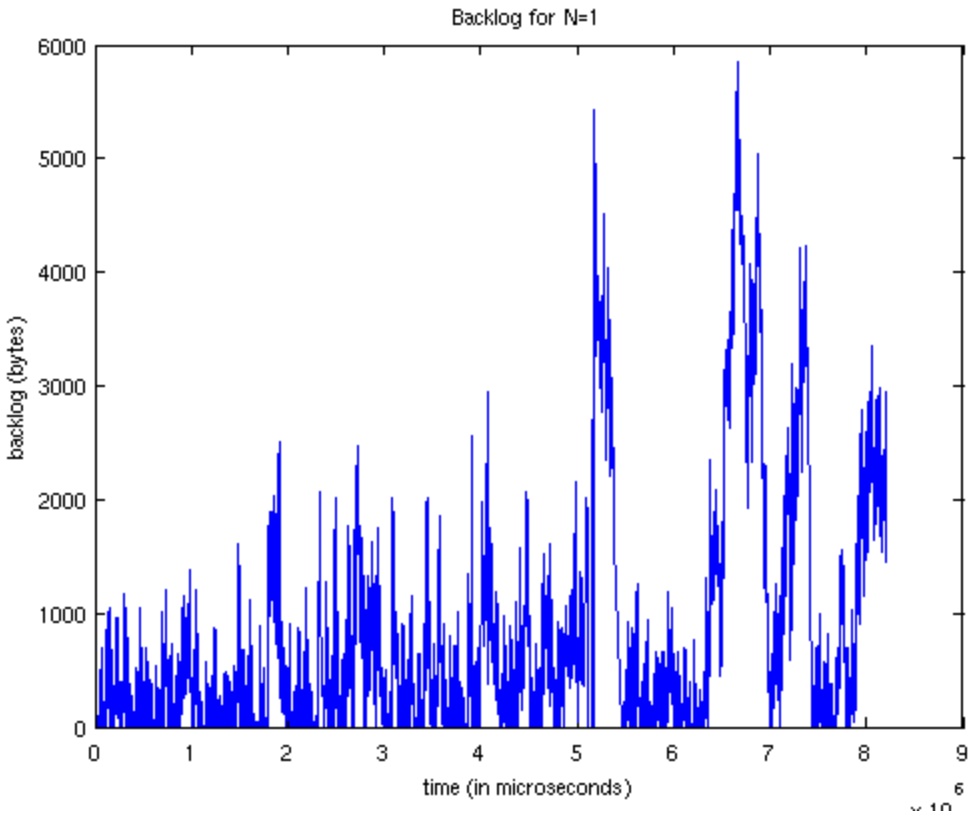
Exercise 1.2 Implement a FIFO Scheduler

The FIFO Scheduler is built with part1/TrafficScheduler.java.

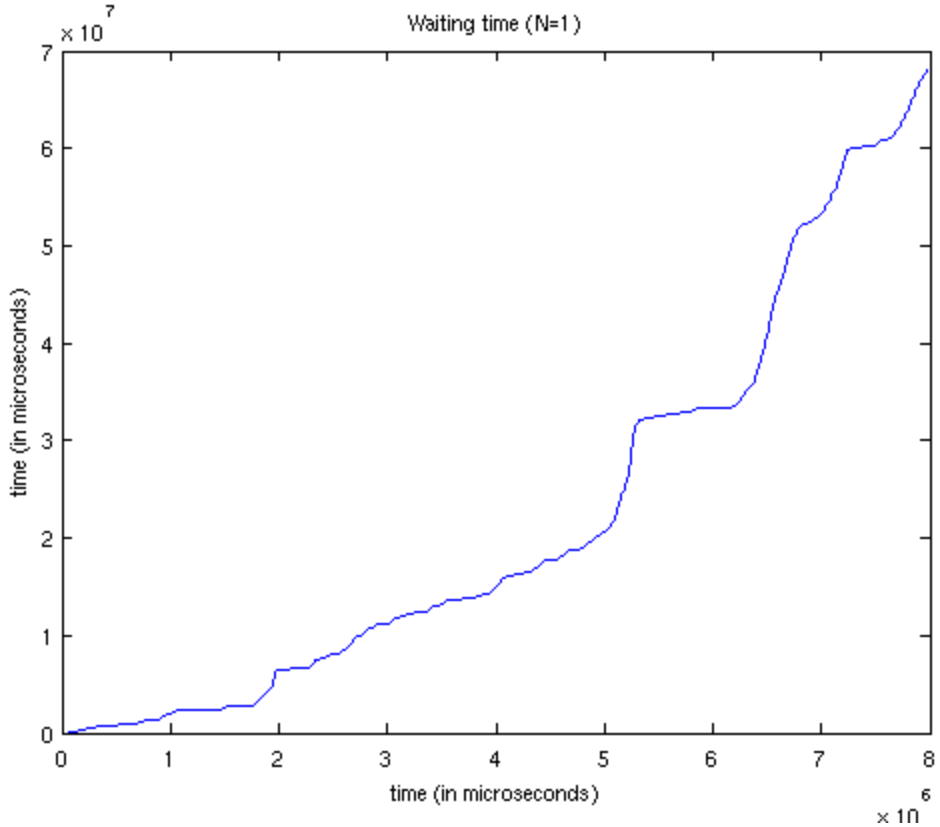
Exercise 1.3 Observing a FIFO Scheduler at different loads

For each value of N, the following plots show the backlog, waiting time, and number of discarded packets as a function of time.

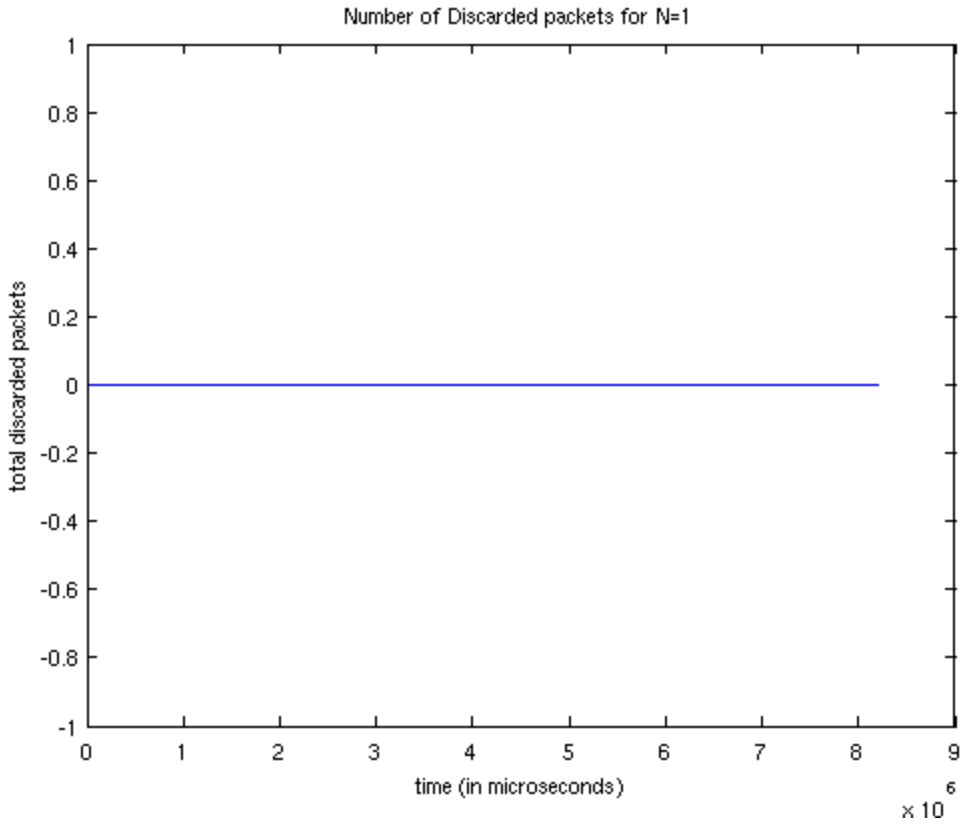
N=1:



The backlog is bounded for the value N=1, which means the FIFO scheduler is able to handle the load.

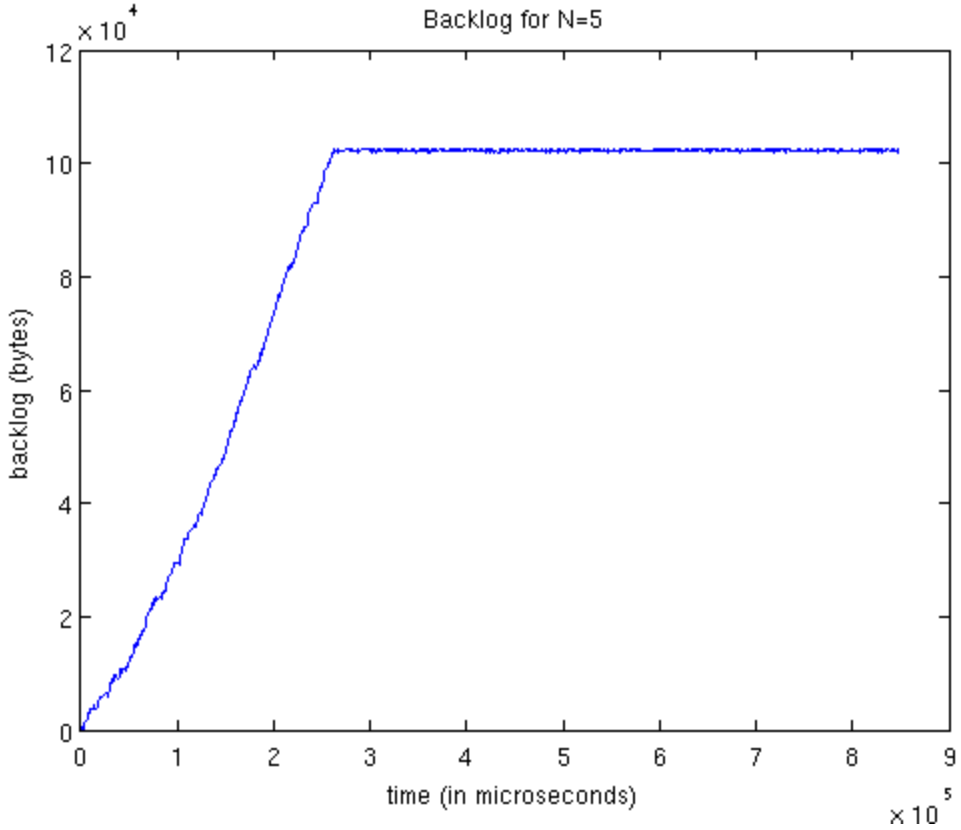


The waiting time shows corresponding spikes to the spikes in the backlog as big packets are generated from the Poisson data.

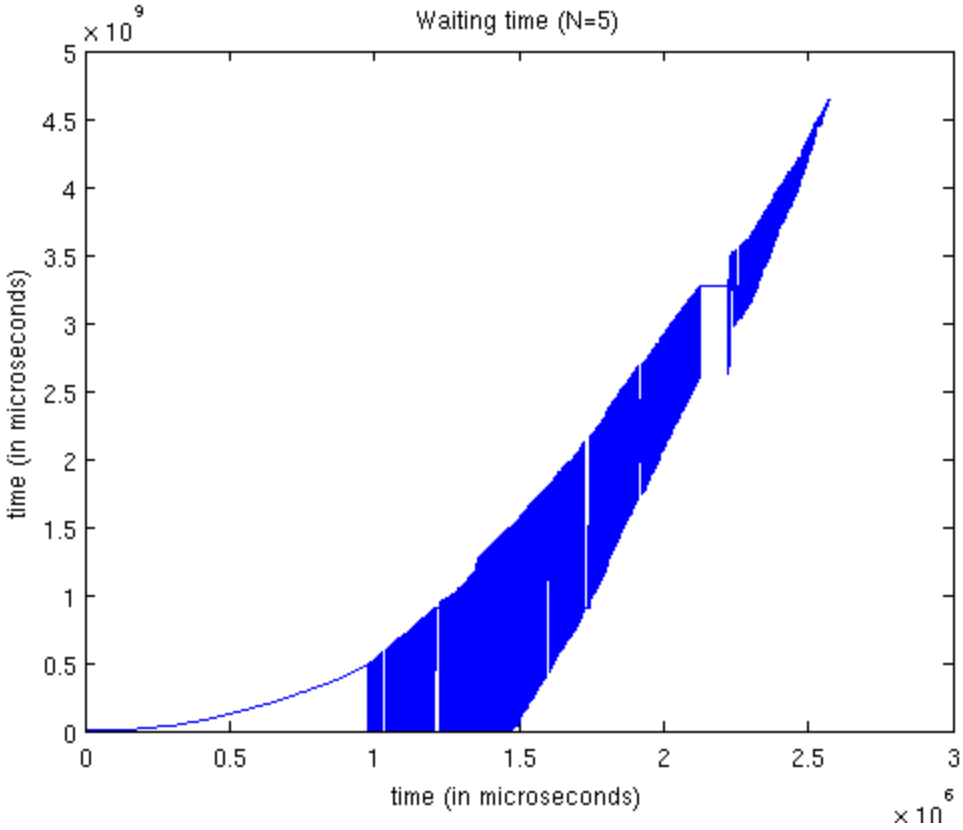


Since we are handling the load and the backlog never overflows, we do not lose any packets.

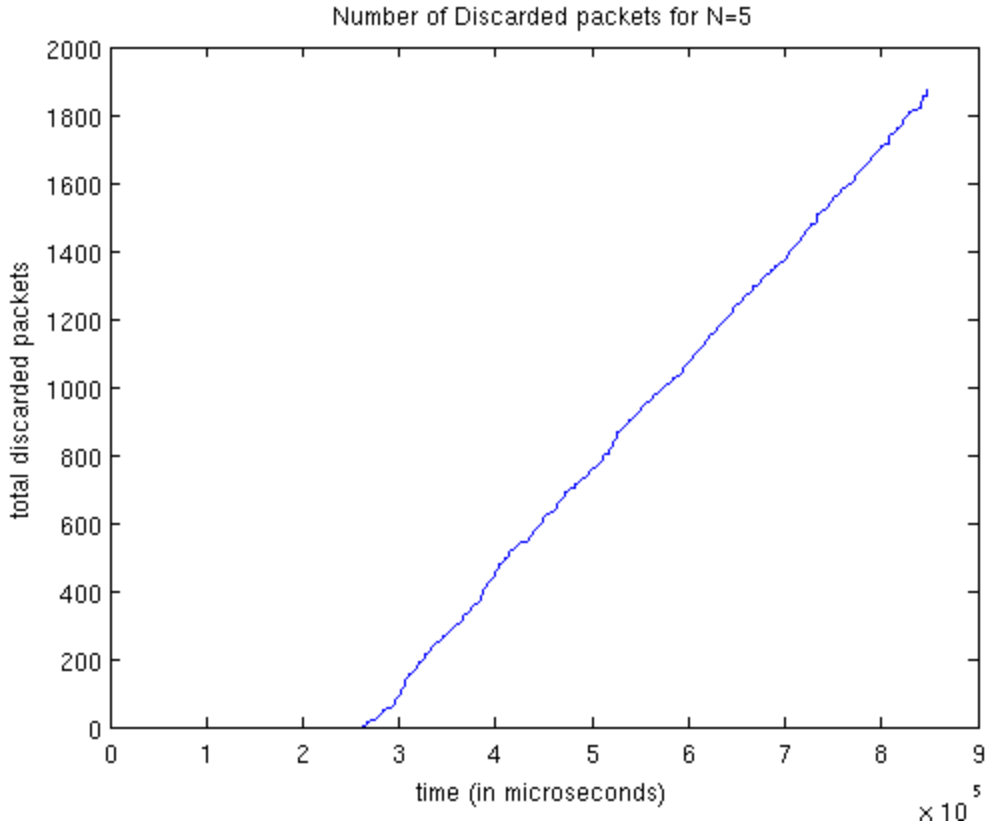
N=5:



The buffer has overflowed with N=5, which transmitted the Poisson traffic at a rate of 5 Mbps.

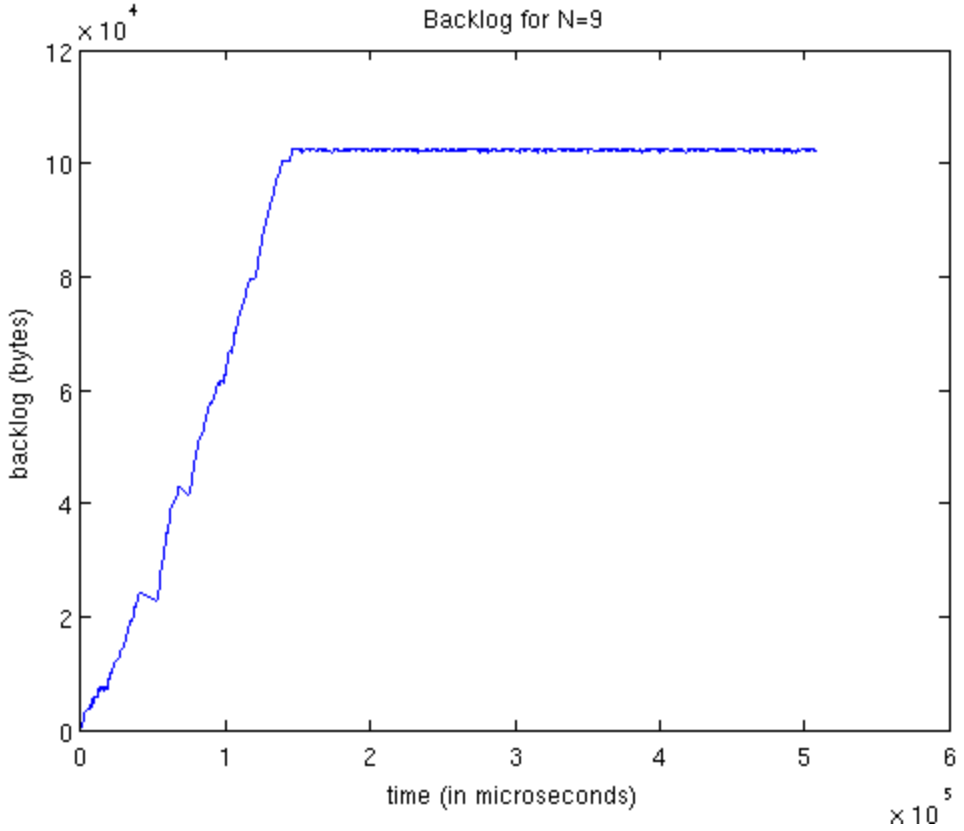


The wait time will significantly increase since our buffer is overflowing.

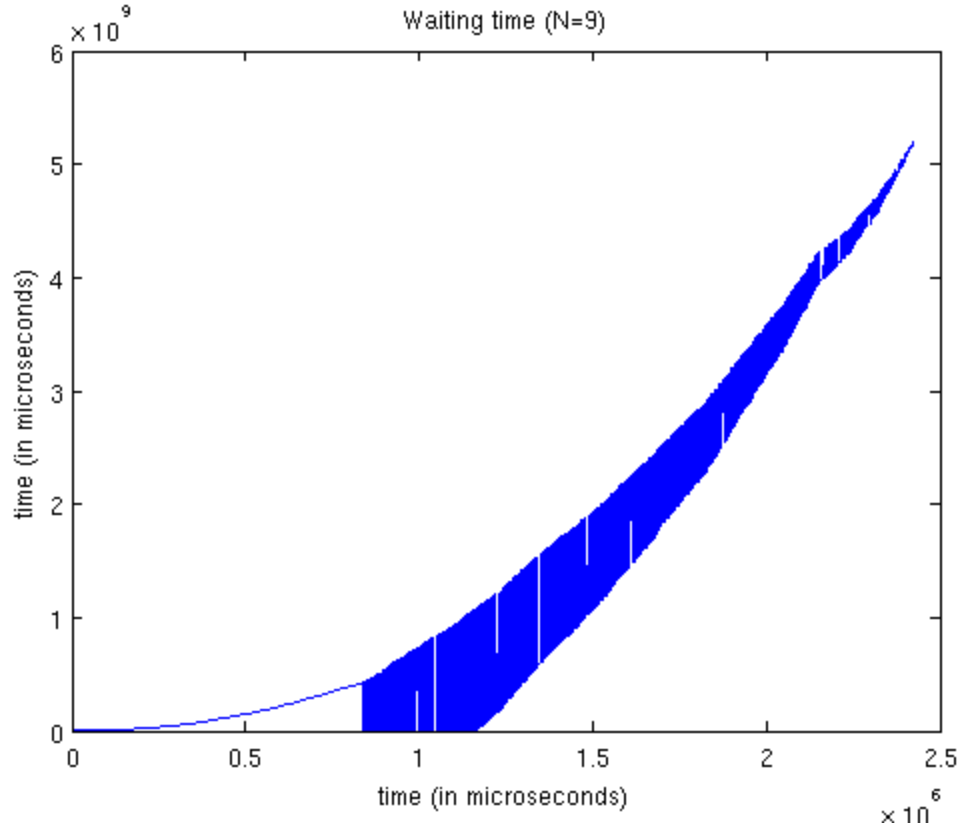


The number of discarded packets linearly increases since buffer has overflowed.

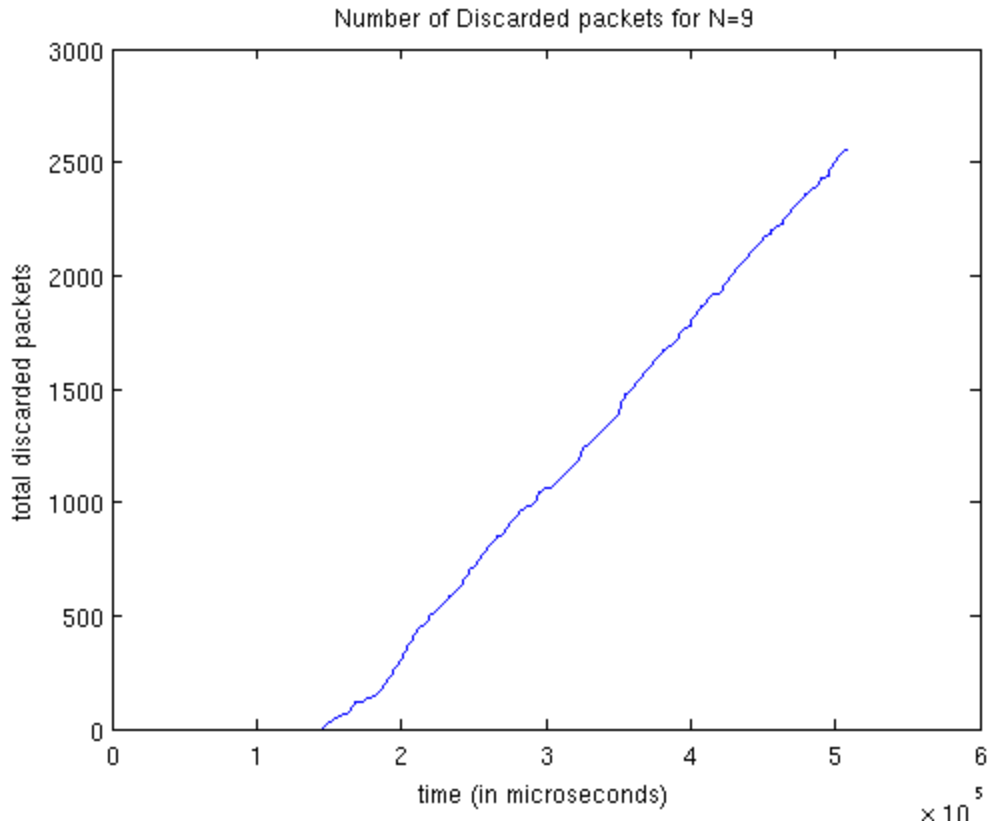
N=9:



With N=9, we can see similar results to N=5 with the backlog increasing even more quickly.



The waiting time also increases linearly as seen before.



The number of discarded packets grows starting at an even earlier time than N=5.

Designate ranges of N, where the FIFO scheduler is in a regime of low load and high load. Justify your choice.

For N<=1, the FIFO scheduler is in a regime of low load as we can see from the experiment with N=1, the backlog is bounded and we do not lose any packets. With N>= 5, the FIFO scheduler is in a regime of high load, specifically we see with N=5 and N=9, the scheduler cannot handle the traffic and discards the packet at an linear rate.

Exercise 1.4 (Optional, 5% extra credit) Unfairness in FIFO

The following experiment uses the traffic sink from part 2 which will allow us to distinguish the output of two different traffic sources by writing the results to different files based on a tag that we are associated with.

The Poisson traffic generator was modified from part 2 to include another parameter to allow changing the classification so we can distinguish between the two Poisson sources being sent at the same time.

Next, the FIFO scheduler was set to 1 Mbps rate with a 100 kB buffer.

Prepare a table that shows the average throughput values and interpret the result.
The average throughput is calculated from the total traffic that was received at the sink divided by the time it took to receive all of that traffic.

experiment	N1	N2	Average Throughput of N1	Average Throughput of N2
1	5	1	1.8625 Mbps	3.9823 Mbps
2	5	5	1.4503 Mbps	1.7506 Mbps
3	5	10	1.9973 Mbps	2.4425 Mbps

Is it possible to write a formula that predicts the throughput as a function of the arrival rate?
Yes, it should be possible since both of our sources produce a linear throughput, with a constant FIFO scheduler, we should be able to determine the linear relationship of the output from these provided inputs. However, in our experiment, the rate capacity of the FIFO scheduler severely limited the scheduler sender from sending the packets quickly and the buffer capacity also caused a lot of dropped packets giving us a throughput that really depends on which packets actually go through to the sink. With an unlimited buffer, we should be able to see a formula that shows a linear relationship such that for N1 to N2, the throughput ratio is N1:N2;

Part 2. Priority Scheduling

Exercise 2.1 Transmission of tagged packets

The traffic generator for Poisson traffic is built in part2/TrafficGeneratorPoissonRescale.java and has been modified to include the packet identification of 0x01 in the first byte of the payload. The traffic generator for Video traffic is built in part2/TrafficGeneratorVideo.java and has been modified to include the packet identification with 0x02 in the first byte of the payload.

Exercise 2.2 Packet classification and priority scheduling

The priority scheduler has been set up with two FIFO queues both set to a maximum buffer size of 100 kB and a transmission rate of 20 Mbps.

The modifications to SchedulerReceiver.java will allow it to classify tagged packets with 0x02 to be placed in buffers[1] and tagged packets with 0x01 to be placed in buffers[0]. This classifies our buffers[0] as low priority traffic and buffers[1] as high priority traffic.

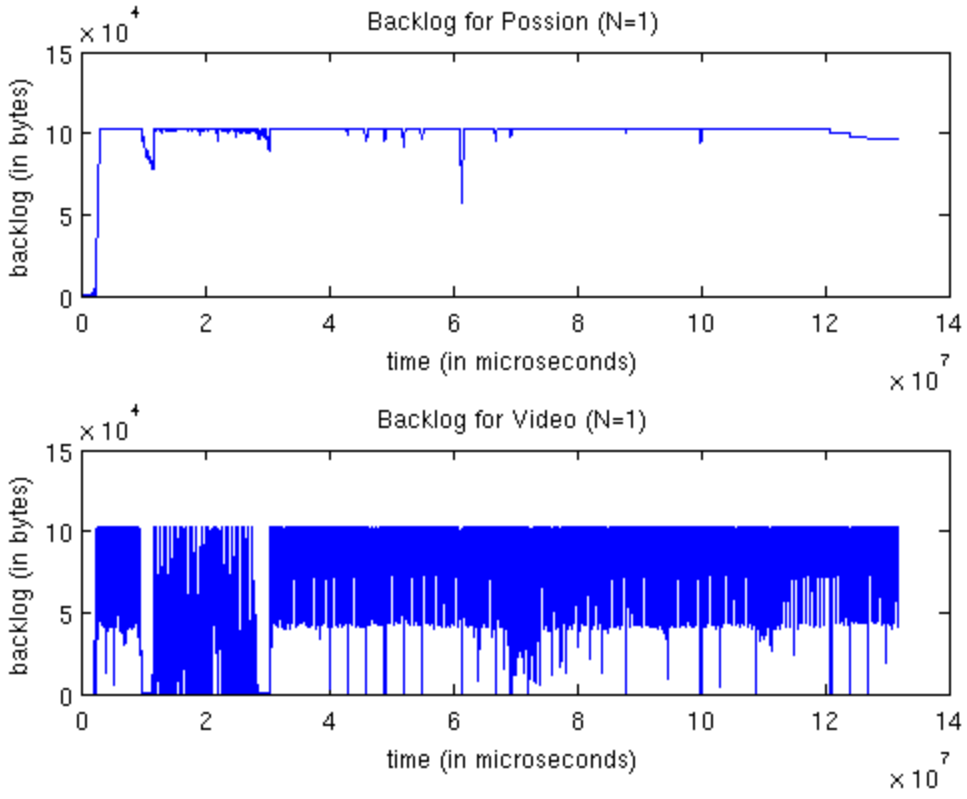
In SchedulerSender.java, we do the corresponding modification that will take data from buffers[1] and if empty, will take data from buffers[0], establishing the high/low priority scheduling.

The implementation was tested with TrafficGeneratorPoissonRescale providing us with the Poisson traffic for N=1 and TrafficGeneratorVideo providing us the video traffic, and the TrafficSink recording the output to two separate sinks based on the priority of the traffic so that we can analysis it for Exercise 2.3.

Exercise 2.3 Evaluation of the priority scheduler

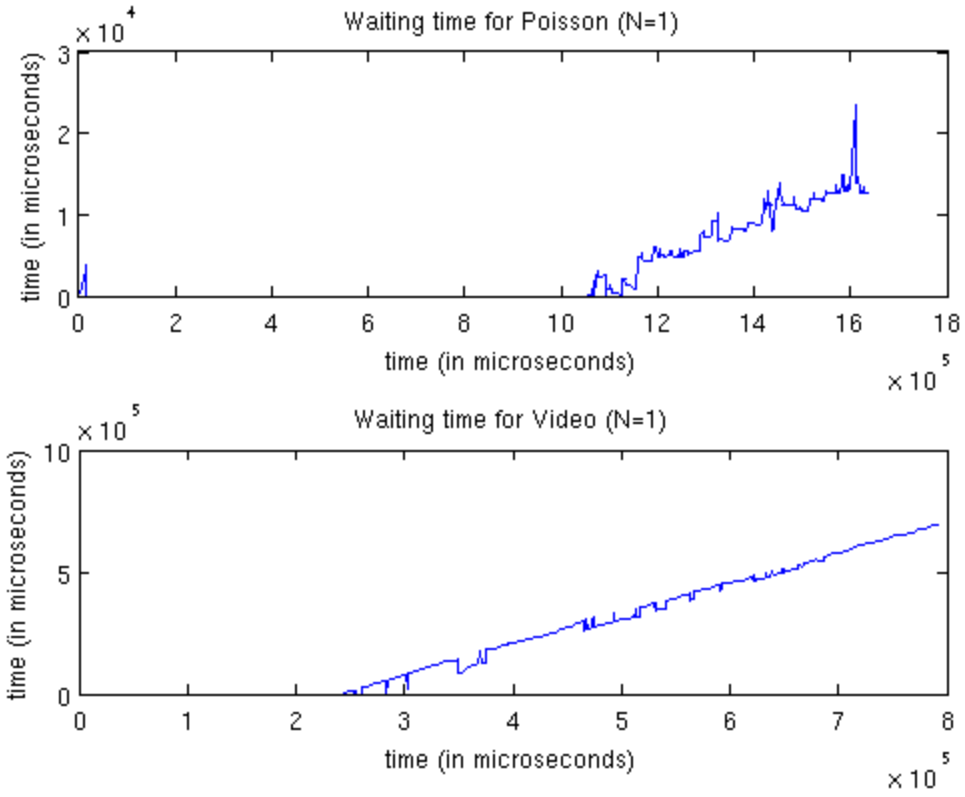
The priority scheduler was evaluated with the two traffic sources, with the Poisson source transmitting with an average rate of N=1, 5, and 9. The following plots have been generated to show the backlog, waiting time, and number of discarded packets as a function of time for both the high priority video traffic and low priority Poisson traffic. The results are then compared with the outcome to Exercise 1.3

N=1:

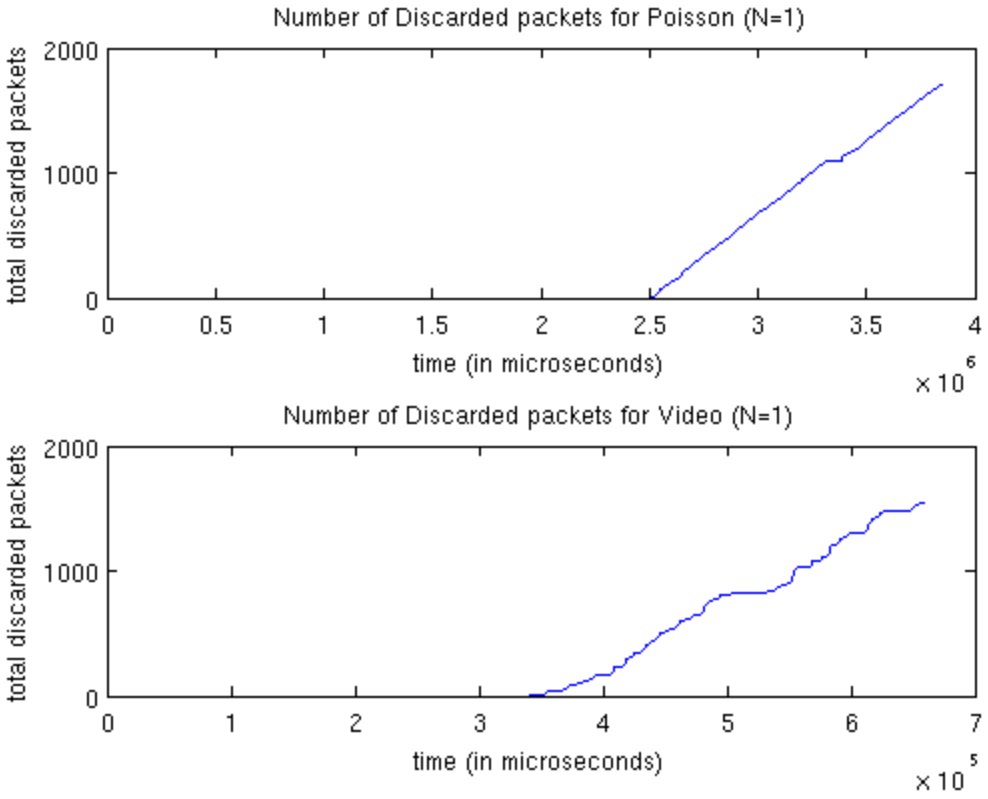


The Poisson traffic is served at the beginning while the amount of video traffic is very little, then the traffic scheduler prioritizes the video traffic once the size of the video packets become very large and the Poisson's backlog overflows. The small dips in the backlog for the video

traffic also allows the backlog for the Poisson traffic to dip as well since the scheduler will get a chance to prioritize the lower priority traffic.

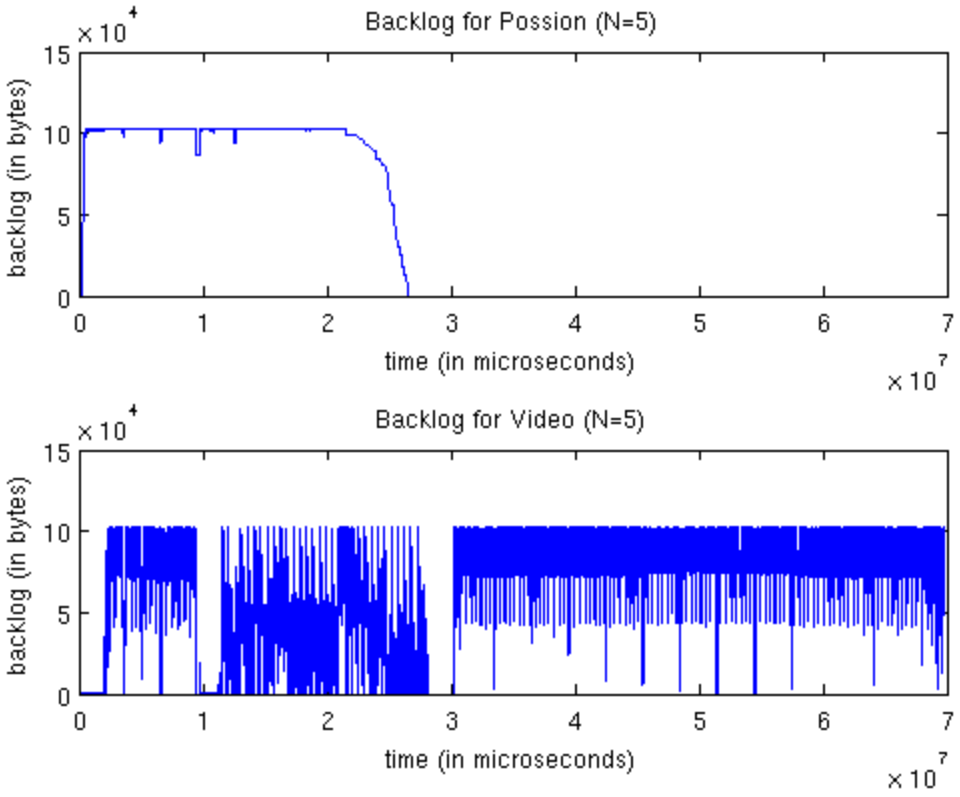


The waiting time grows linearly since the backlog overflows.

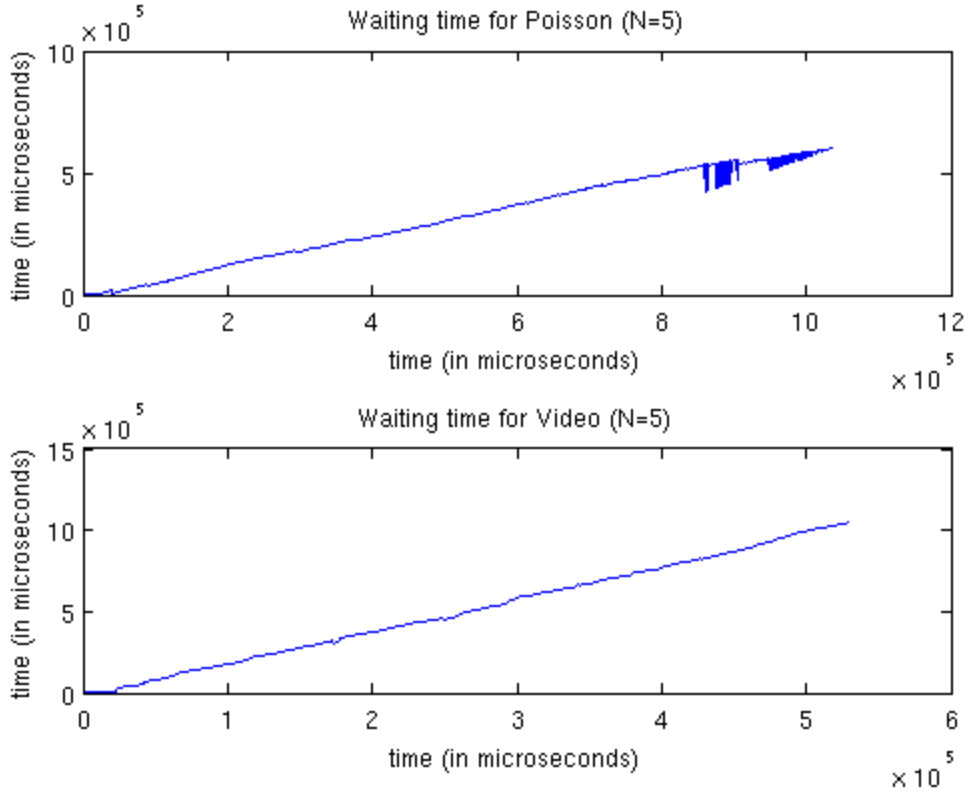


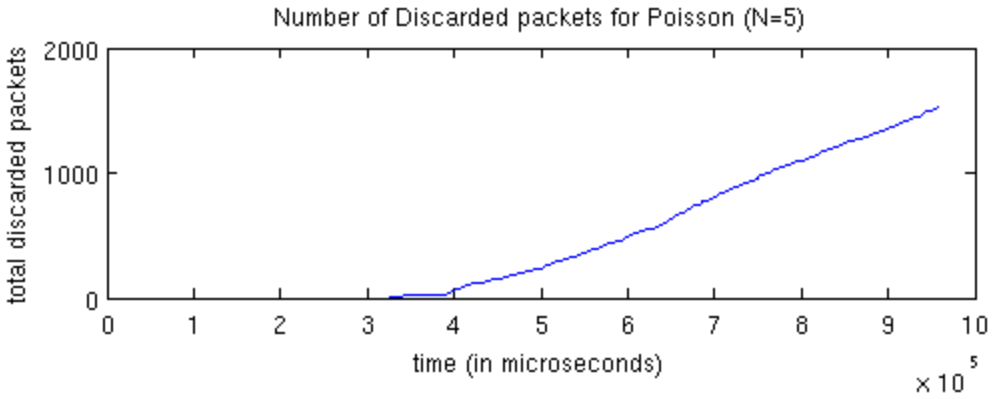
The number of packets are also discarded and grows linearly. A plateau can be seen in the plot for the discarded packets for Video showing less intensive video traffic to be passed through the traffic scheduler.

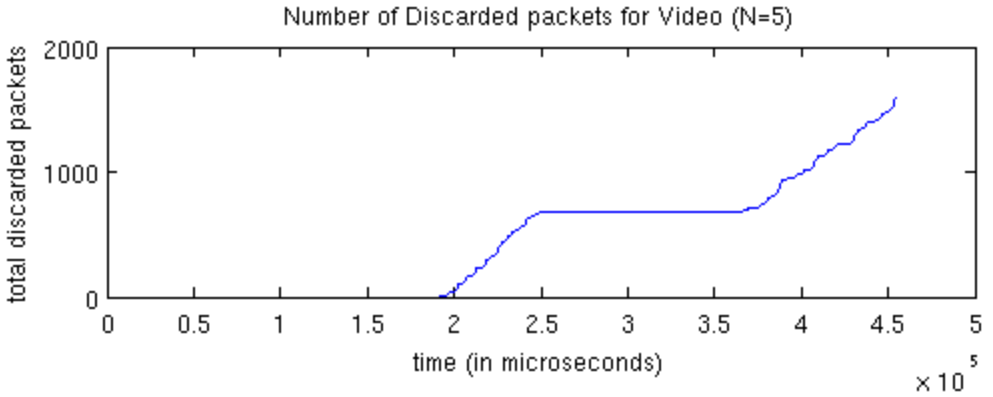
N=5:



With the increased in rate for the Poisson traffic We can see that it finishes much quicker and allows the scheduler to focus on sending the video traffic.

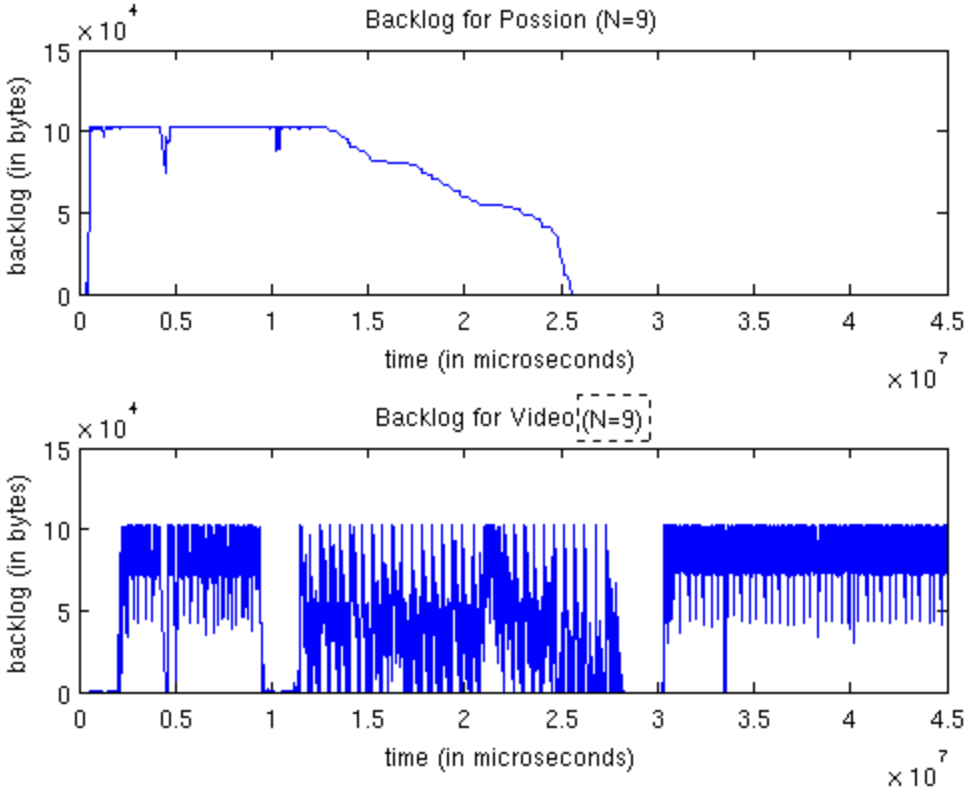




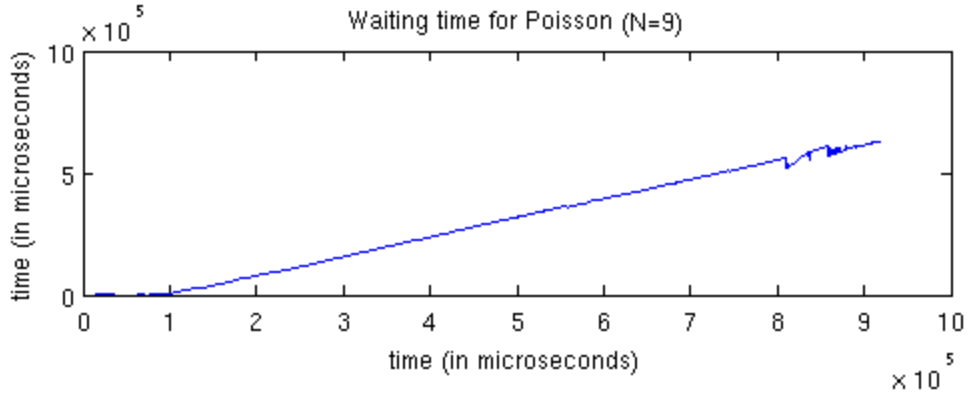


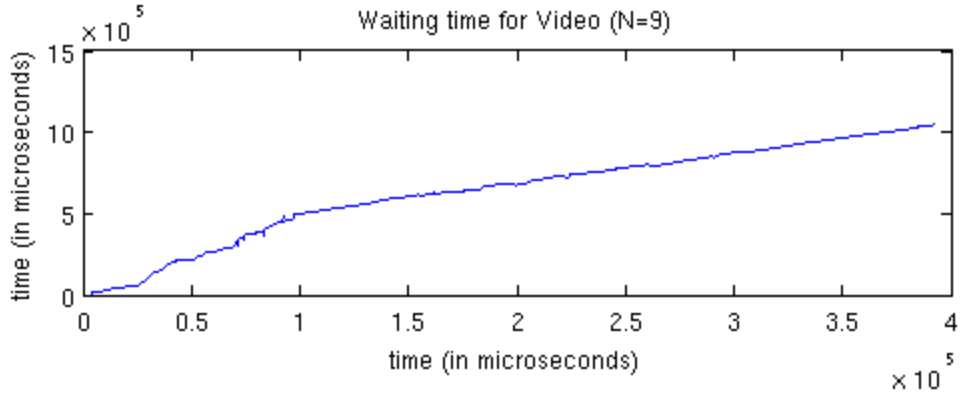
We can see a significant plateau for the number of discarded packets for video.

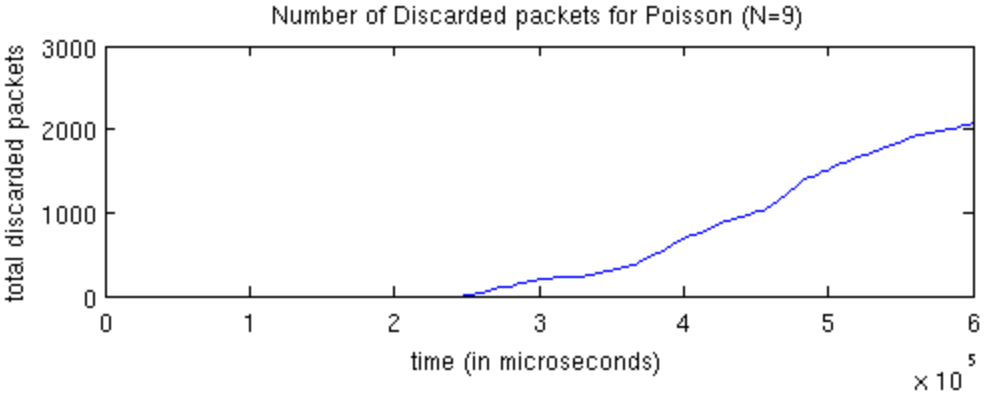
N=9:

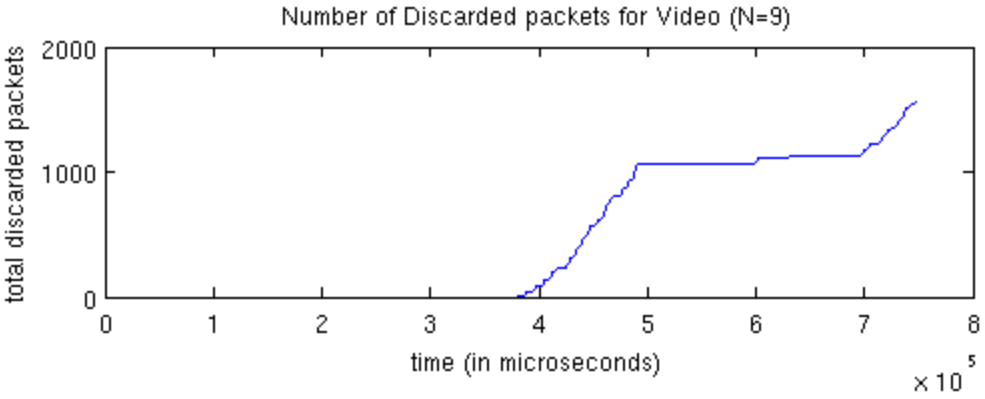


Similar to N=5, the backlog for Poisson quickly overflows and falls back down to zero. Again we see gaps of low video traffic allowing the Poisson backlog to retreat a bit before overflowing again.









Compare the outcome to Exercise 1.3

Unlike Exercise 1.3, the Poisson traffic had to share the traffic scheduler with the Video traffic, causing it to overflow even with N=1. We can see the backlog spiking up once large amounts of

video traffic begins to flow in and the scheduler prioritizes the high priority video traffic over the Poisson traffic.

We can also notice the large dips of the video backlog retreating which corresponds to smaller amounts of video traffic. This is enough to allow the scheduler to prioritize the Poisson traffic that is buffered and allow the buffered Poisson traffic to be served a bit before overflowing again.

The waiting time and number of discarded packets are similar to Exercise 1.3's experiment with N >=5 where the FIFO scheduler is in a regime of high load.

Exercise 2.4 (Optional, 5% extra credit) Starvation in priority schedulers

The Poisson traffic generator was that was modified from part 2 to do Exercise 1.4 is used here to include another parameter to allow changing the classification so we can distinguish between the two Poisson sources being sent at the same time. The difference here is that we are using a priority scheduler we built in Exercise 2.3 to exhibit the starvation of low priority traffic.

Prepare a table that shows the average throughput values and interpret the result.

The average throughput is calculated from the total traffic that was received at the sink divided by the time it took to receive all of that traffic.

experiment	N1	N2	Average Throughput of N1	Average Throughput of N2
1	5	1	5.1342 Mbps	9.5627 Mbps
2	5	5	3.4224 Mbps	9.2344 Mbps
3	5	9	1.8944 Mbps	9.2211 Mbps

Compare the outcome to Exercise 1.4

Compared to Exercise 1.4, in this experiment, we were able to transfer much more traffic since our scheduler capacity rate was set to 10 Mbps.

In each experiment, we were able to see that N2's high priority traffic overcame N1's low priority traffic. N2's buffer gets sent to the sink quickly while N1 backlogs and once N2 is done, N1's backlog will brought back down. In experiment 1, we saw that a combination of high priority traffic that is at a low load can compete with the low priority traffic that is of high load.In Experiment 2, the high load high priority traffic starved the lower priority traffic. In experiment 3, the same case can be seen.

Part 3. Weighted Round Robin (WRR) Scheduler

Exercise 3.1 Build a WRR scheduler

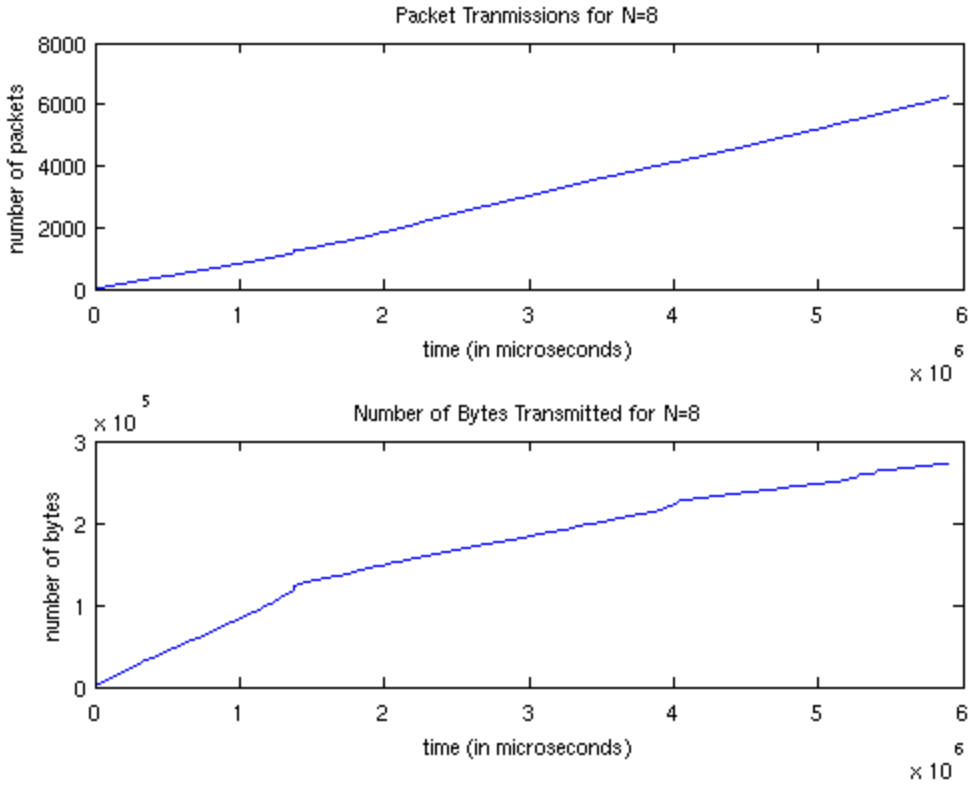
The TrafficScheduler has been modified to 3 queues, one FIFO buffer for each flow, with a transmission rate of 10 Mbps, and maximum buffer size of 100 kB. The WRR scheduler has been implemented in part3/SchedulerSender.java to include the calculation weights, shares, and logic to iterate through the buffers.

The Poisson traffic generator is extended from part 2 which tagged the packets with “1” and “2” to include the labelling of “3”. The sink has been extended to capture the packets with label “3” in a third output file.

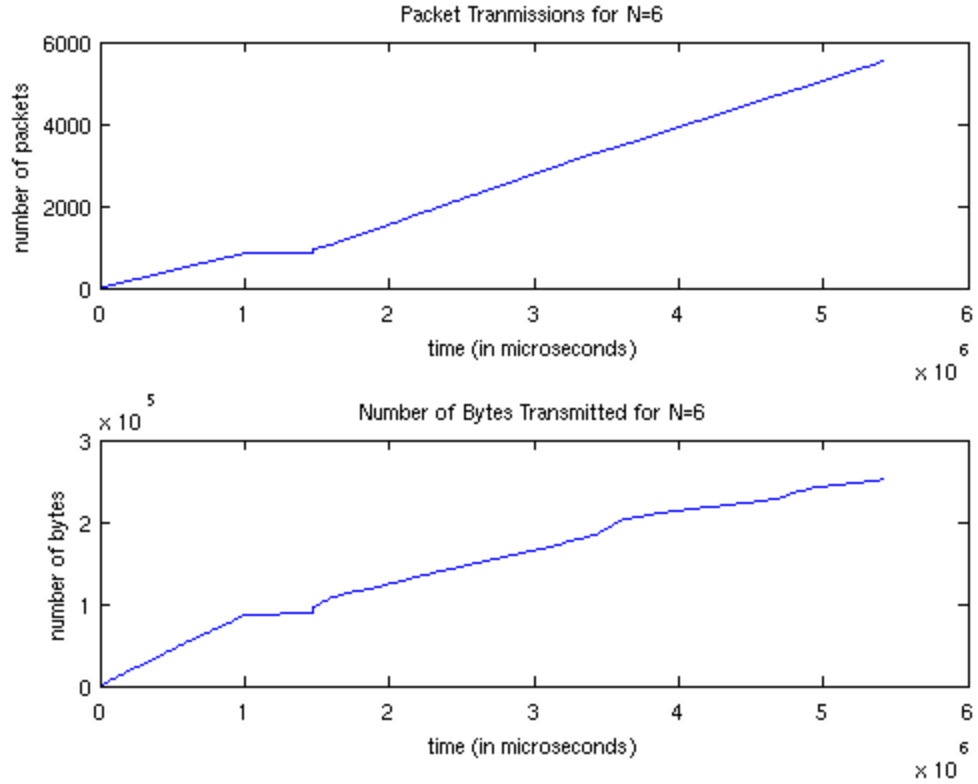
Exercise 3.2 Evaluation of a WRR scheduler: Equal weights

With the configuration of the three Poisson generators from Exercise 3.1 and weights of the queue set to $w_1 = w_2 = w_3 = 1$, the evaluation of the WRR scheduler was done and the results were plotted to show the number of packets transmissions and the number of transmitted bytes from a particular source on the y-axis as a function of time on the x-axis.

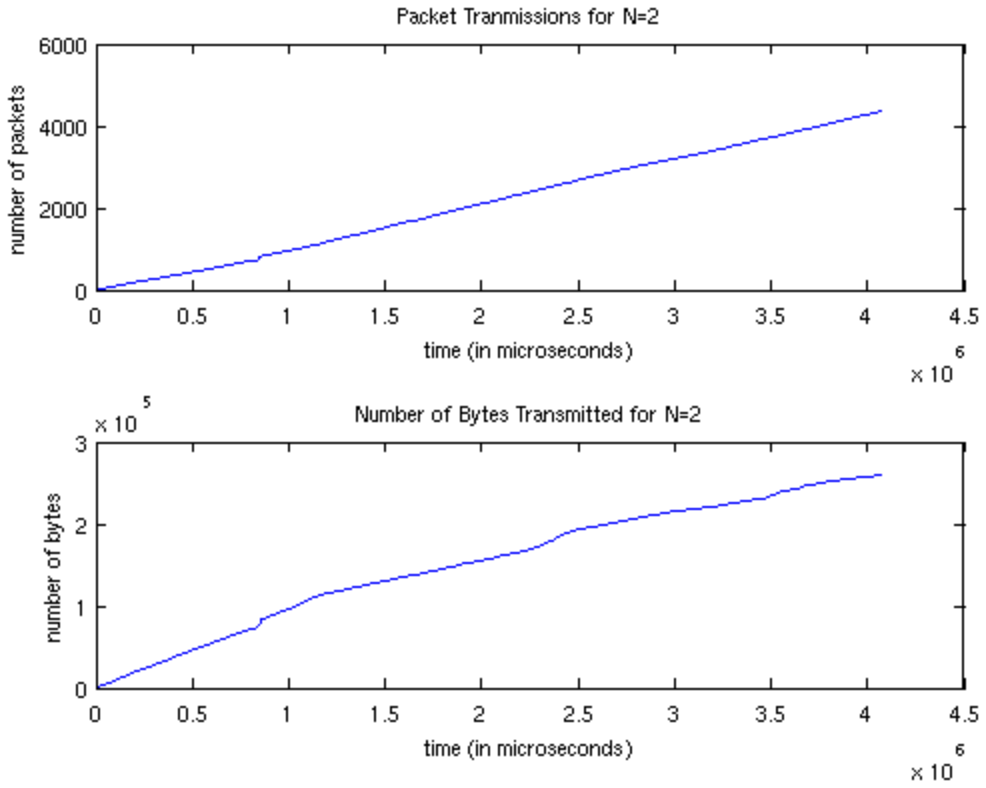
Poisson traffic flow 1, N = 8:



Poisson traffic flow 2, N = 6:



Poisson traffic flow 3, N = 2:

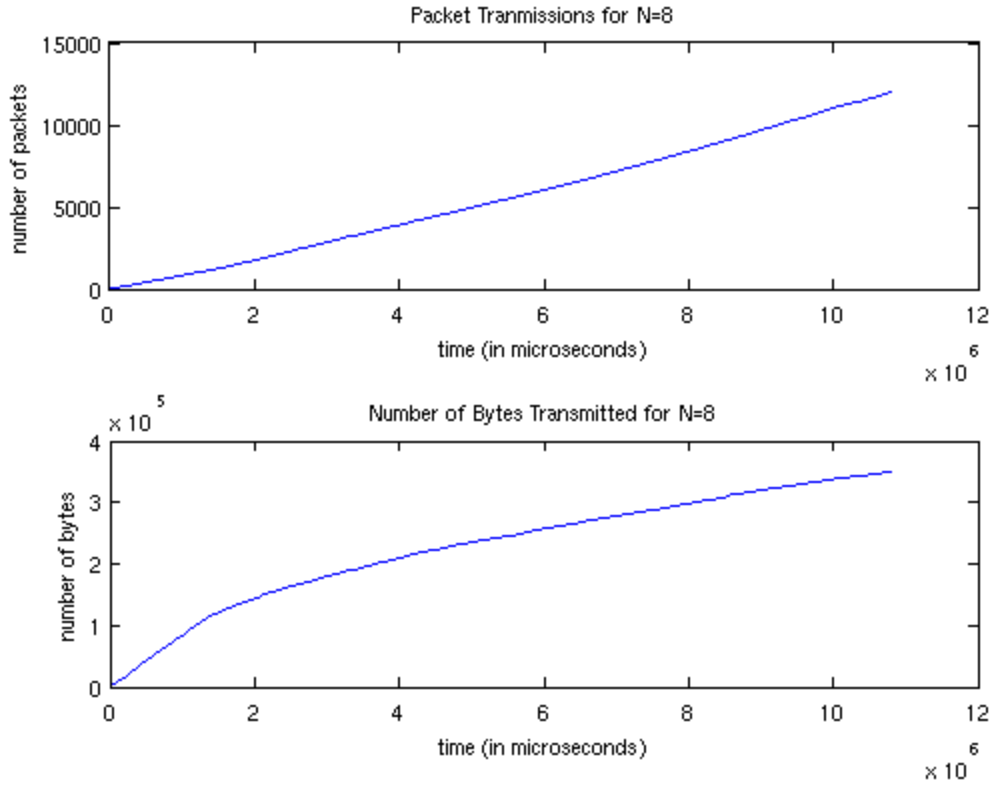


Compare the plots with the theoretically expected values of a PS scheduler.
The ratios of the flows are approximately 6000:6000:4000 which is 3:3:2, which is close to the PS scheduler's 4:4:2. The approximation is due to the traffic scheduler dropping packets when the buffer becomes overflowed, and delays to send the traffic once its capacity limit is reached.

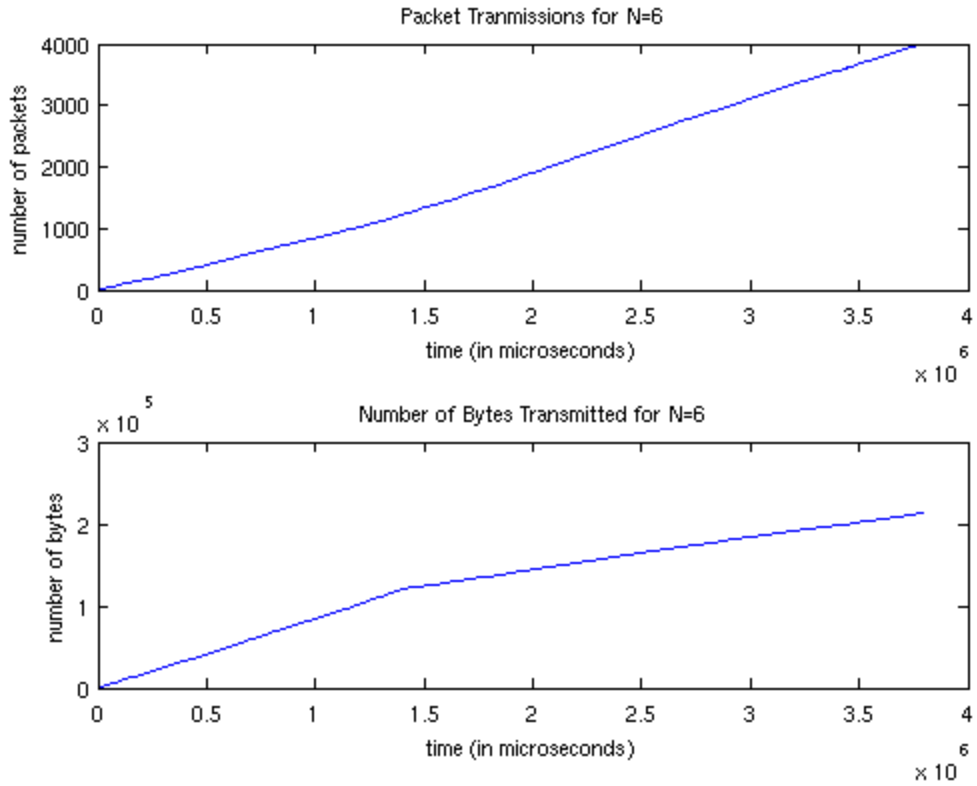
Exercise 3.3 Evaluation of a WRR scheduler: Different weights

The experiment in Exercise 3.2 is repeated but with weights, $w_1 = 3$, and $w_2 = w_3 = 1$. The following plots for the number of packet transmissions on a time scale of 10 ms for each data point and number of transmitted bytes on a time scale of 10 ms per data point is below.

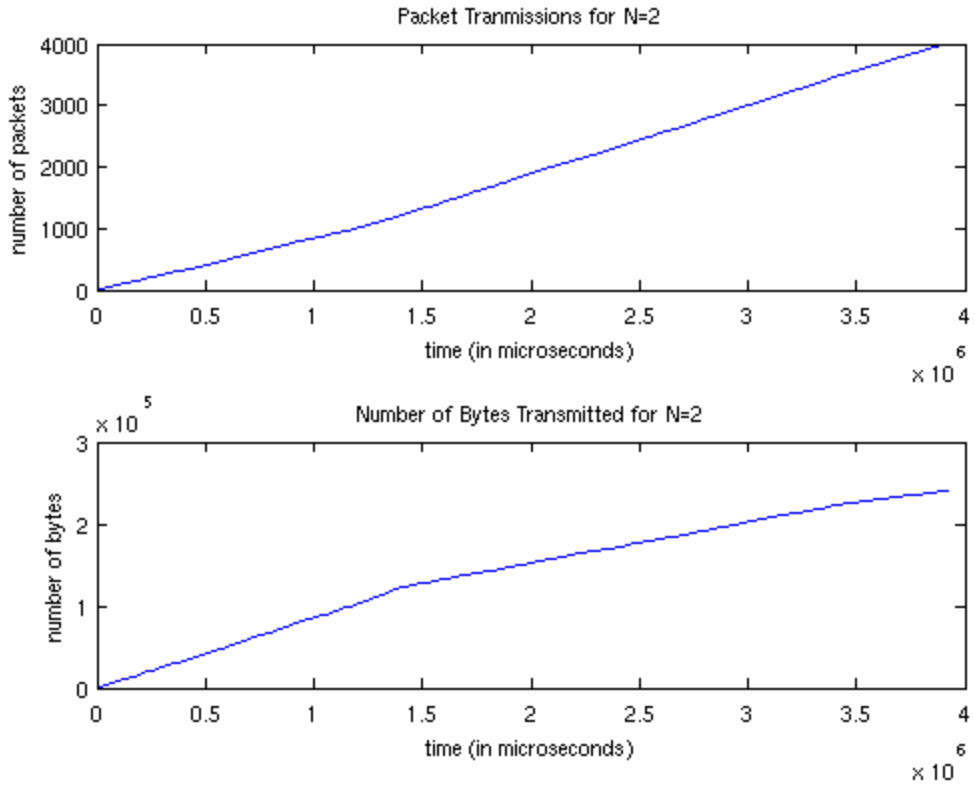
Poisson traffic flow 1, N = 8:



Poisson traffic flow 1, N = 6:



Poisson traffic flow 1, N = 2:



Compare the plots with the theoretically expected values of a GPS scheduler

We see about a ratio of 12000:4000:4000, which is about 12:4:4, or 6:2:2. Again, the approximation difference can be accounted for due to the lost in packets and delay in transmission.

Exercise 3.4 (Optional, 5% extra credit) No Unfairness and no Starvation in WRR

The following experiment tries to show that WRR is fair and free from starvation. With the WRR scheduler still set to the same configuration, we will send Poisson traffic of various rates, and record the throughput in the following table.

experiment	N1	N2	Average Throughput of N1	Average Throughput of N2
1	5	5	7.7593 Mbps	7.3451 Mbps
2	5	9	7.9099 Mbps	7.3311 Mbps
3	5	15	7.1643 Mbps	7.9931 Mbps

As we can see, the flows are transmitted at about the same rate. N1 gets exactly half the transmission capacity, resulting in an average throughput calculated to be almost equal to each other.

Compare the outcome to Exercises 1.4 and 2.4

In Exercise 1.4, the amount of bandwidth should have been distributed among N1 and N2. In Exercise 2.4, the higher priority always starved the lower priority and took most of the bandwidth.