COMP 6320 Design and Analysis of Computer Networks: Course Project

James S. L. Browning and Austin David Preston

December 1, 2022

1 Introduction

The purpose of this project was to develop a basic simulation of a queueing system. The system simulates two queues, each with its own server for "processing" simulated packets. The system either assigns new arrivals to a queue randomly, or assigns them to the whichever queue is shortest (randomly if they happen to be the same length when a new packet arrives). In addition to a choice of queue assignment strategy, the simulator takes two integer parameters, λ and μ, representing the average time between arrivals and the average time needed to service a packet, respectively. Our simulator uses these values to generate appropriate interarrival and service times for the simulator, which uses a while loop and several timer variables to simulate the system millisecond-by-millisecond.

2 Usage

The simulator can be compiled using the following command:

```
gcc main.c -lm -o main
```

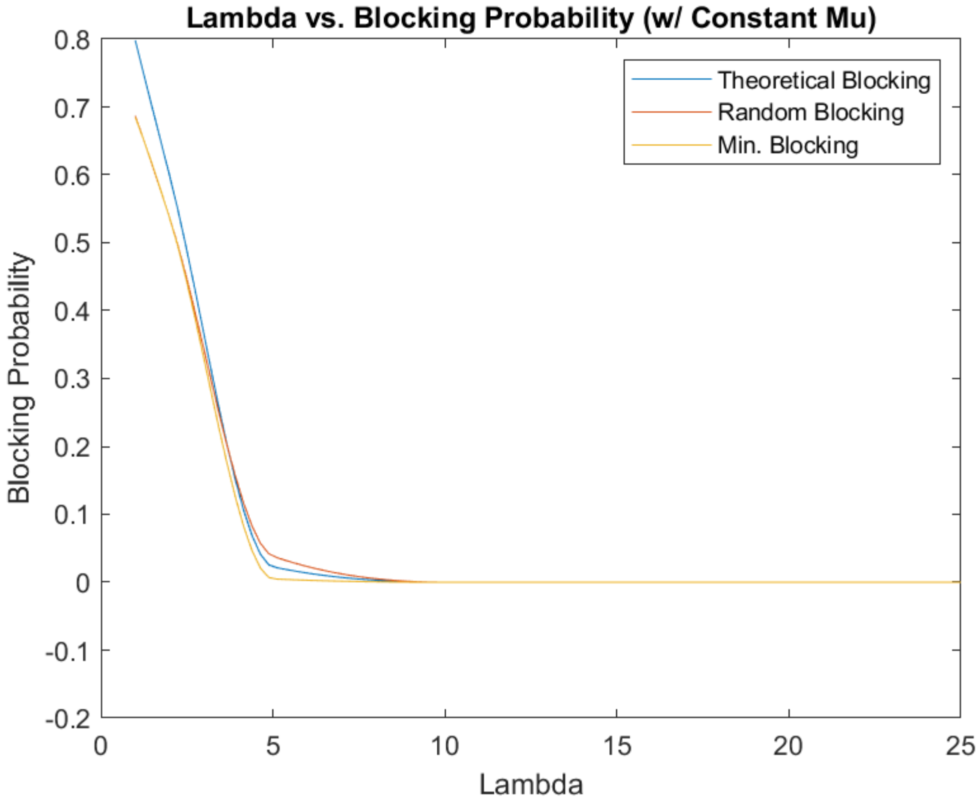
The first parameter passed to it during usage should be either a `1` or a `2`, to select random assignment or minimum queue length assignment, respectively. This should be followed by λ and μ values. For example, this would be a valid usage:

```
./main 1 10 10
```

3 Performance Data

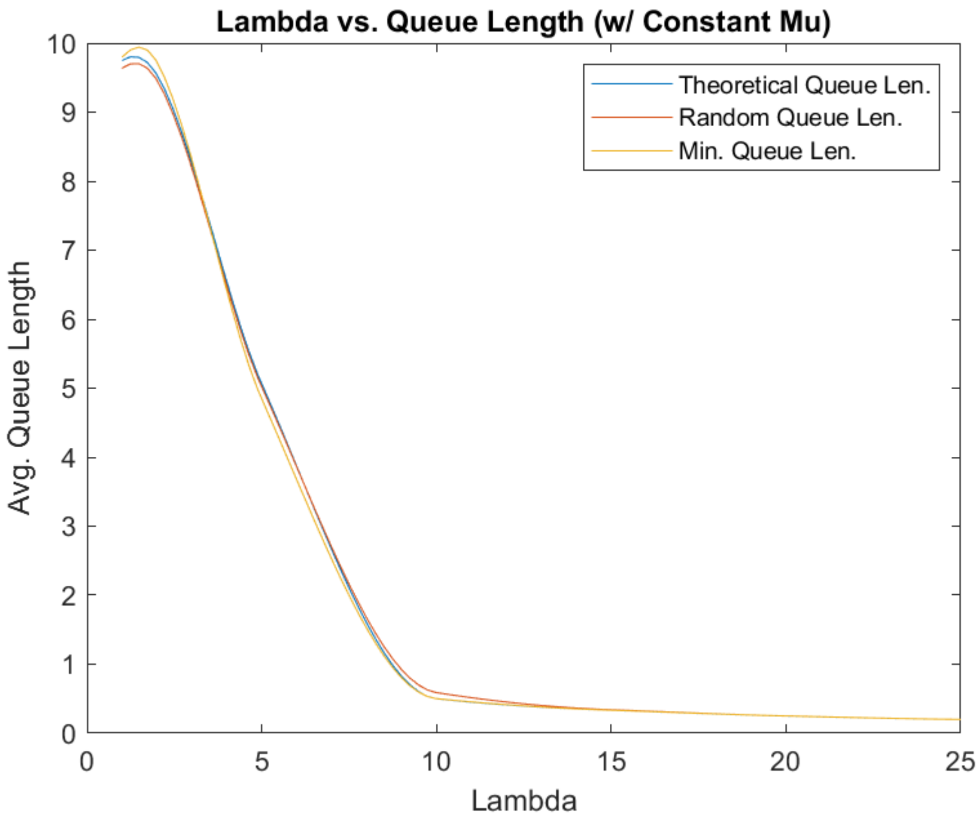
3.1 Lambda

The figure below shows the change in blocking probability relative to λ, assuming a constant average service time (here 10 milliseconds). λ values were shifted from 1 to 25. At around a λ of 5, the service time began to catch up to the growing interarrival time, and beyond a λ of 10, the blocking probability reached zero (meaning all arrivals were able to be processed — this was somewhat expected, as at this point, the interarrival time had exceeded the average service time).



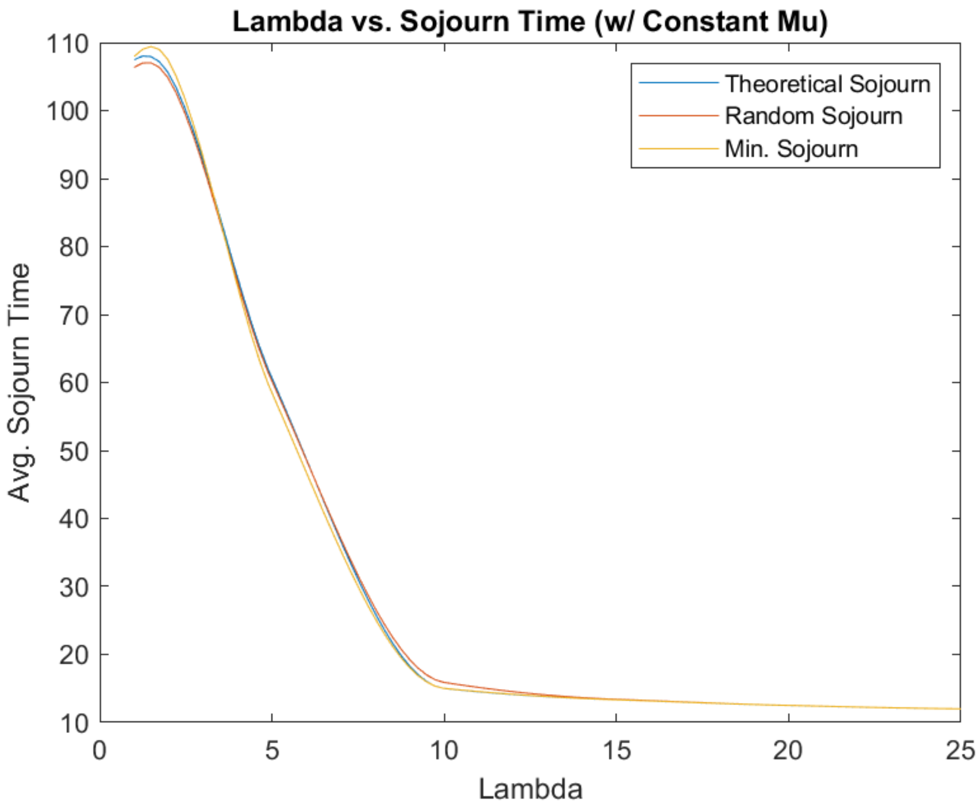
This data demonstrated that λ has a significant impact on system performance, though is only relevant to the extent that it outpaces the service time.

The second figure below shows the change in average queue length (over a run of 10,000 packets) relative to λ.



This value shifts downward more gradually, with rapid arrivals quickly filling up the queues, and slower paces allowing them to stay comparatively short. The split queues could be a contributing factor to changes in λ having less extreme impacts on queue length than blocking probability, as seen in the first figure.

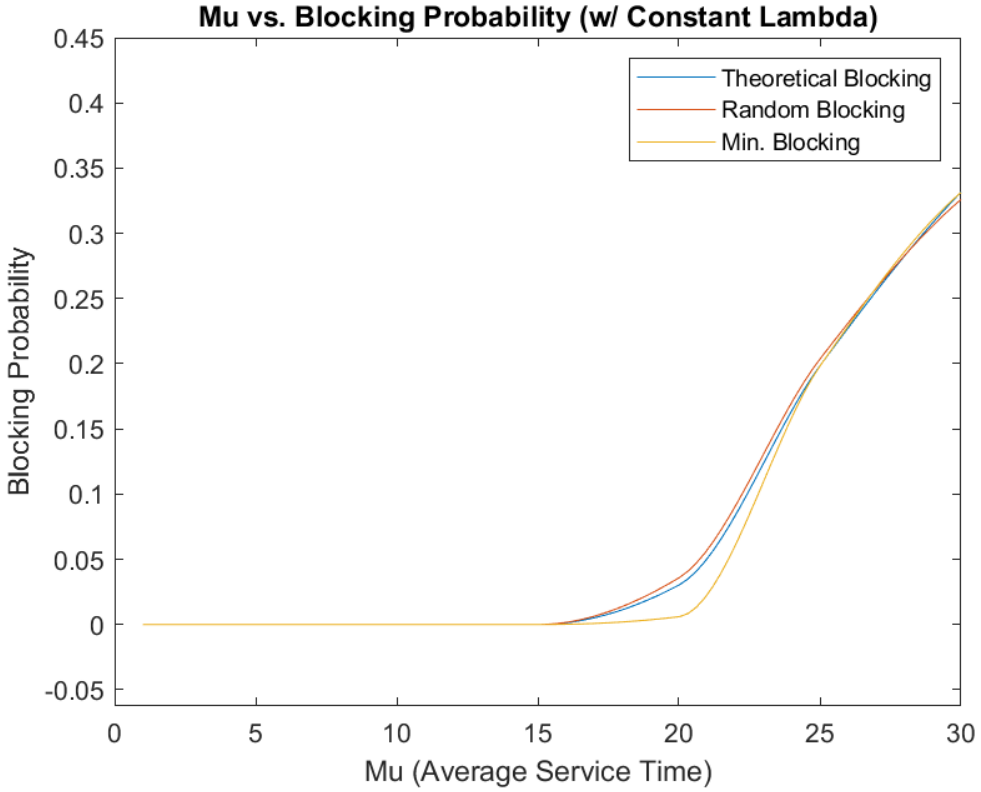
Finally, the third figure below shows λ vs. average sojourn time.



This figure is quite similar to the second figure, showing λ vs. queue length, which makes sense — queue length directly correlates with sojourn time, as the number of packets waiting in front of a given packet determines how long that packet will have to wait to be serviced.

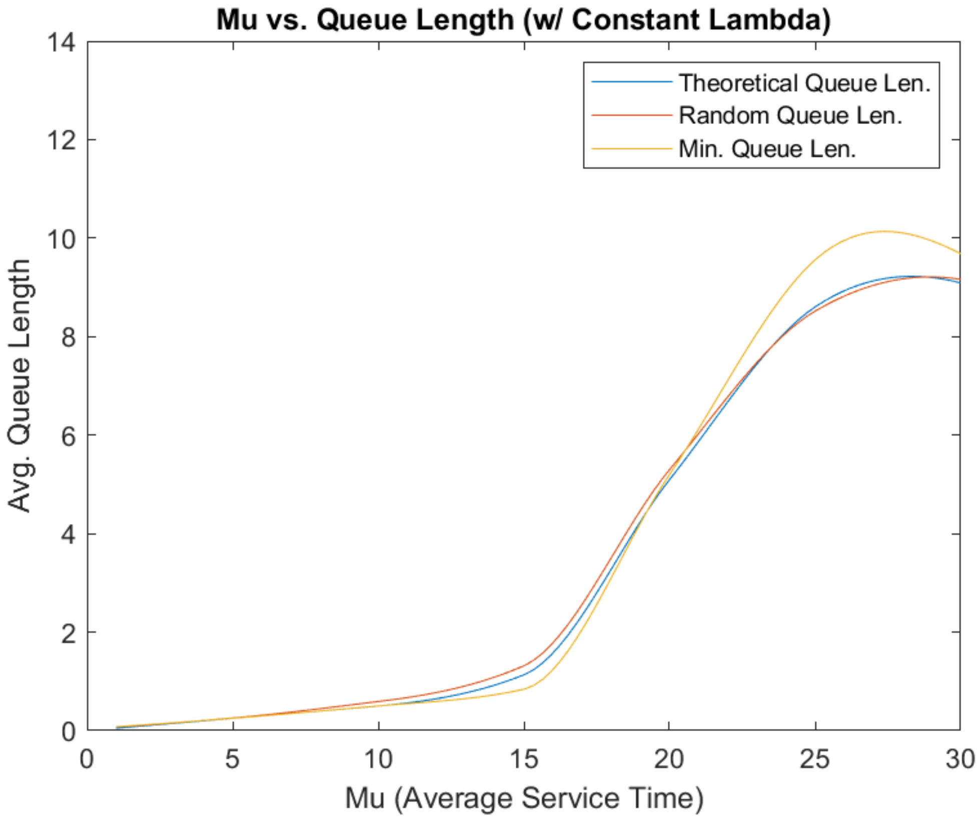
3.2 Mu

The next figure shows average service time vs. blocking probability.



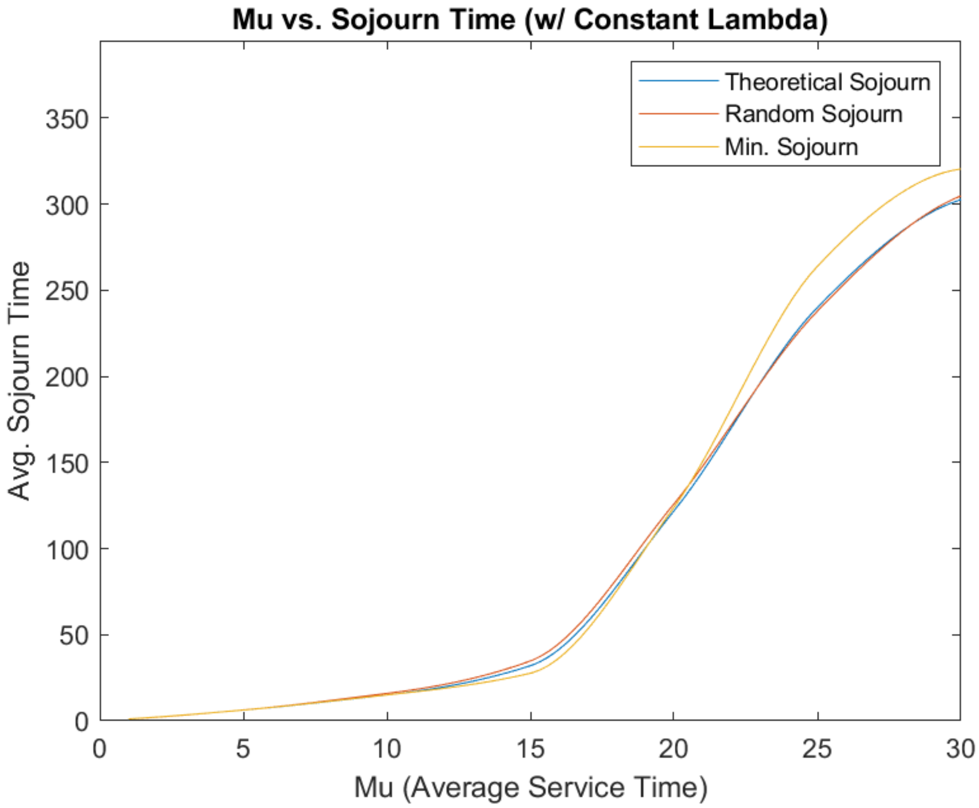
This figure is essentially an inverse of λ vs. blocking probability, as increasing the amount of time needed for a packet to be serviced would necessarily increase the odds that some packets would be left behind entirely. The blocking probability does appear to climb more gradually here than it shrank in the first figure, however, perhaps indicating that λ carries more weight for this performance metric.

The fifth figure below shows average service time vs. average queue length.



As one might expect, slowing down service time leads to an increase in average queue length. It is interesting to note that the minimum queue length assignment strategy displays a noticeable bump in average queue length at slow service rate values — this is likely because the minimum queue length strategy ensures both queues are being filled equally (meaning they both end up being fully taken advantage of), slightly increasing the average queue length.

The sixth figure below shows average service time vs. average sojourn time.

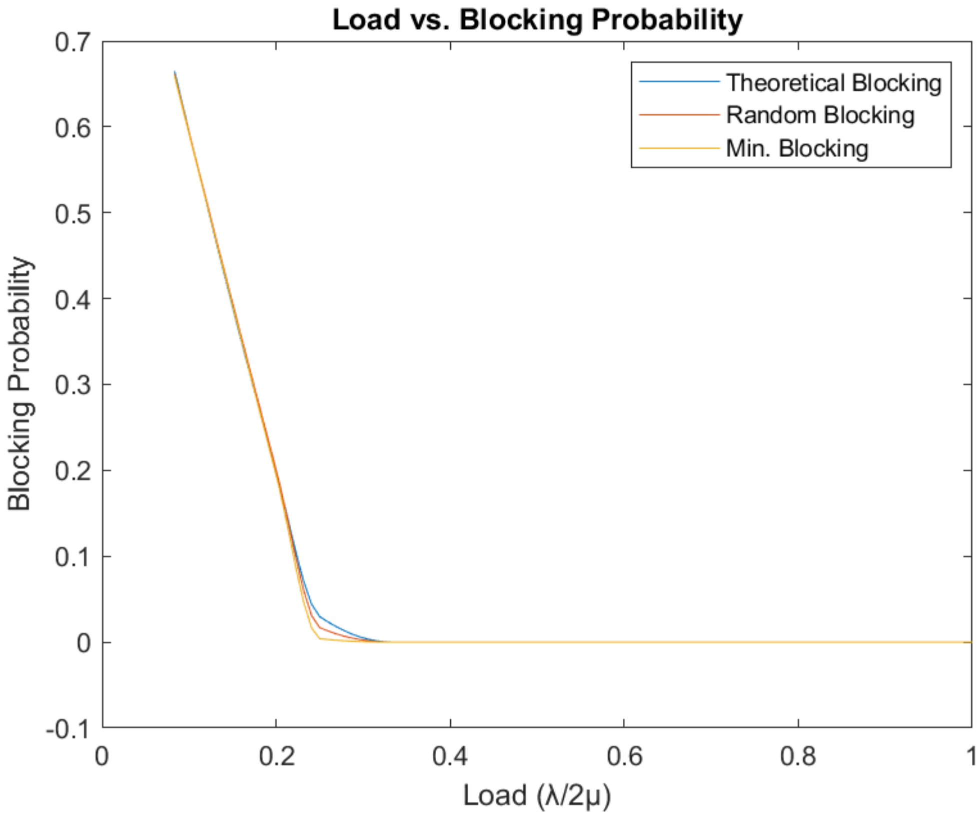


Once again, the outcome here is what one would expect — slowing down service increases sojourn time. As with the figures in the subsection above, this one displays a similarity to service time vs. blocking probability, as the longer a packet is likely to wait for service, the more likely a subsequent packet is to be to dropped entirely.

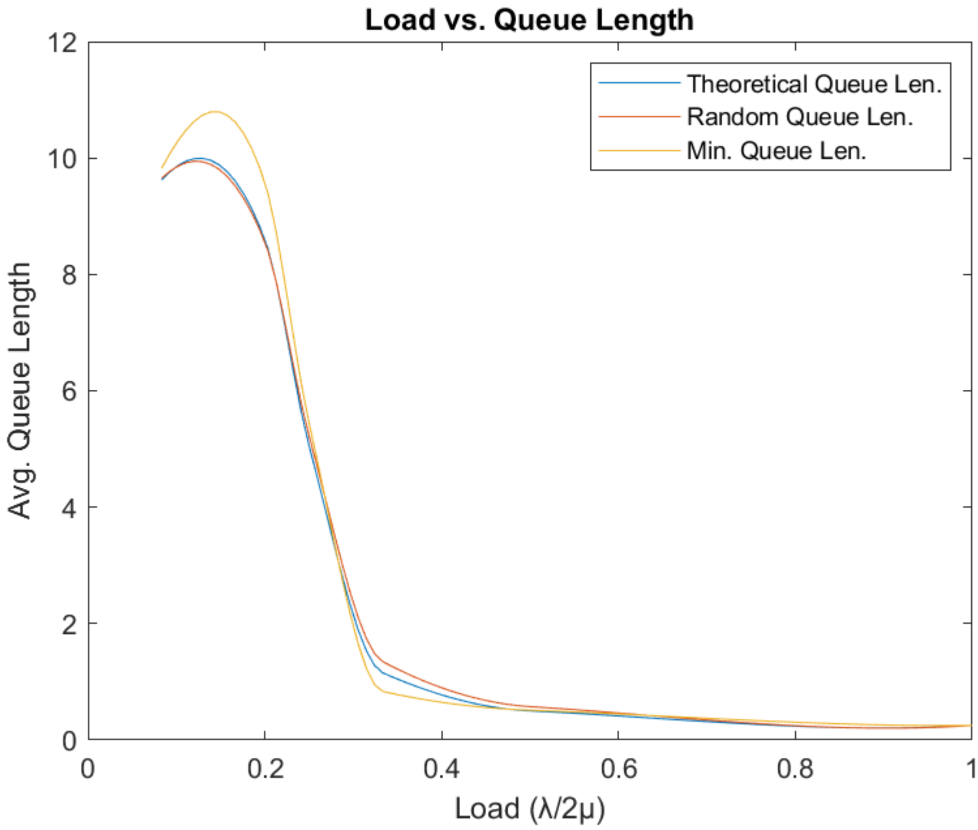
3.3 Load

These next three figures show the above performance metrics relative to the overall load on the system, equal to $\frac{\lambda}{2\mu}$.

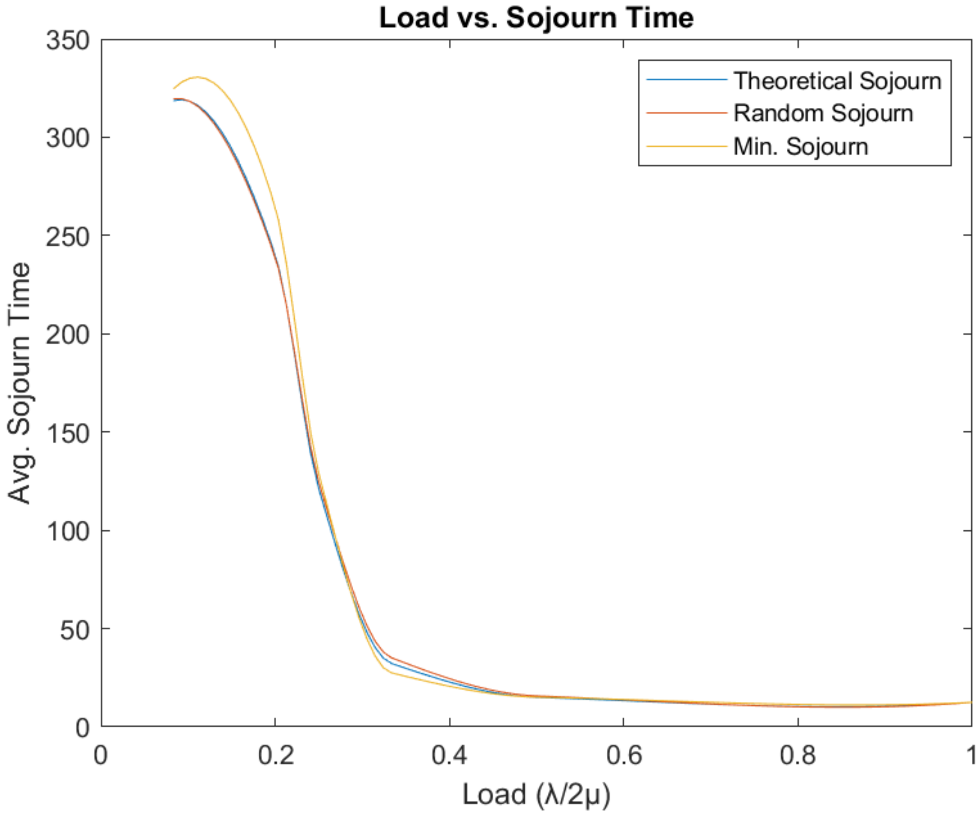
The first figure shows load vs. blocking probability.



The data above shows that the blocking probability of the system decreases as the average service time speeds up and the interarrival time for packets slows down. The system performs at its best when arrivals are few and far between, and when the servers are able to service those packets quickly.



The figure above paints a similar picture — as arrivals become more common and servicing takes longer, the average queue length grows, as more and more packets get backed up in the system.



The third and final figure above is quite similar to the previous figure. As queue length decreases, so too does sojourn time, as packets have fewer packets waiting in front of them.

4 Final Thoughts

This experiment showed that interarrival time and service time both have an impact on the performance of a system. Interarrival time has a stronger correlation with average queue length, as can be seen in the second figure in section 3.1. More common arrivals mean more packets get backed up, but as long as the average service time is decent, packets will still be processed fast enough for most of them to make it through the system. It is only at extraordinarily high arrival rates that the blocking probability makes a sharp increase, indicating that overall, the service rate has a more consistent correlation with blocking probability (as can be seen in the first figure of section 3.2).