Modeling and Simulation Final Paper

Author: **Abdul Hamid Mangorangca**
Final Revision: **June 8, 2025**

Abstract

I ran a simulation where customers waited in waiting lines (**queue**) to be served by the workers (**server**). Think of it like a person who owns a vehicle waiting in front of the casa or the authorized service center for his car brand so that when it's his turn, his car will be serviced by the mechanic for any repairs, PMS, etc.

I tried three kinds of traffic in my simulation: **(1) slow arrivals (underloaded)**, **(2) equal arrivals and service (balanced)**, and **(3) fast arrivals (overloaded)**. The first one is where customers arrive less than when customers are serviced (meaning by the time the customer is done being serviced, there may be no one waiting or someone just arrived), and you can infer from the rest because it is self explanatory. I also changed how many workers and waiting lines there were.

Some key findings were (some may be obvious but nevertheless), **(a) when arrivals were slow, waiting lines stayed almost empty and nobody was turned away**, **(b) when arrivals matched service speed, waiting lines got a bit long, and adding workers helped more than adding extra waiting lines**, **(c) when arrivals were fast, obviously lines got very long and many customers got turned away, unless we added more workers.**

Introduction

We want to simulate a traffic, whether it be in a road, buying something from the convenience store or in my example, servicing a vehicle. We tried three kinds of traffic as mentioned in the abstract, and we also tested more by giving help to the place (model), by adding more workers or adding more separate waiting lines. Our goal is to figure out the easiest way to keep line short and avoid turning people away.

Methods

The code stuff was written in C++, and also implementing various features took a lot of time. It simulates the model for 10000 minutes, in that time, people arrive at random times using the random function provided by our instructor, Sir Jojo. These random times were given a mean (so on average, every 2 minutes, every 1 minute, or every 0.5 minute, depending on the test). The reason for these numbers is because 1 is a simple number, we either multiply it by twice or half of it which these numbers are easy to get and not too much arbitrary. Each worker (server) takes on average a minute to help (service) one person. When a worker is free, they pick the person who has been waiting the longest across any waiting lines (queue), as opposed to having a dedicated waiting line to each worker.

My experiment parameters

How fast people arrived

- **slow** - one every 2 minutes (underloaded)
- **medium** - one every 1 minute (balanced)
- **fast** - two every 1 minute (overloaded)
  Note that these are generated randomly within those time frame (e.g. randomly within that 2 minutes, or 1 minute).

How many workers (servers): 1, 2 or 3

How many separate waiting lines (queue): 1, 2, or 3

Note that as I mentioned that they are not dedicated to each worker. The waiting lines is like the one in S.M. supermarket malls where they could go to whichever server is free.

I ran every combination three times (3 arrival speeds x 3 worker counts x 3 waiting line counts, repeated 3 runs each). Results shown below are based on the average across all three runs.

Results

Averaged Simulation Results (3 Runs)

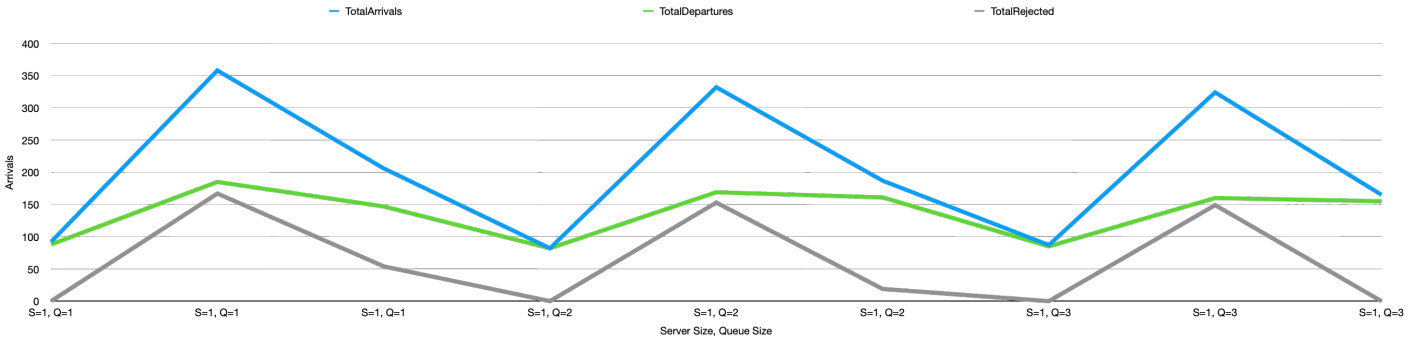
Scenario	Servers	Queues	Mean Interarrival	Mean Service	Label	Total Arrivals	Total Departures	Total Rejected	Avg Queue Length	Server Utilization (%)	Max Queue Observed	Avg Wait Time (min)
1	1	1	2.0	1.0	iA> sD	92.0	88.0	0.0	0.4286	51.97	5.0	48.29
2	1	1	0.5	1.0	iA< sD	358.0	185.0	167.0	4.0441	99.25	5.0	215.65
3	1	1	1.0	1.0	iA==sD	206.0	147.0	54.0	2.8559	89.97	5.0	185.70
4	1	2	2.0	1.0	iA> sD	82.0	82.0	0.0	0.1596	45.09	3.0	19.46
5	1	2	0.5	1.0	iA< sD	332.0	169.0	153.0	8.5265	99.73	10.0	495.10
6	1	2	1.0	1.0	iA==sD	187.0	161.0	19.0	5.1398	90.19	10.0	309.31
7	1	3	2.0	1.0	iA> sD	87.0	85.0	0.0	0.2702	48.14	3.0	31.79
8	1	3	0.5	1.0	iA< sD	324.0	160.0	149.0	13.2221	99.52	15.0	778.15
9	1	3	1.0	1.0	iA==sD	165.0	155.0	0.0	5.8143	87.98	14.0	363.76
10	2	1	2.0	1.0	iA> sD	71.0	71.0	0.0	0.0109	21.55	1.0	1.53
11	2	1	0.5	1.0	iA< sD	350.0	320.0	23.0	1.7859	91.07	5.0	55.02
12	2	1	1.0	1.0	iA==sD	161.0	161.0	0.0	0.1205	47.45	3.0	7.48
13	2	2	2.0	1.0	iA> sD	86.0	86.0	0.0	0.0105	22.37	1.0	1.22
14	2	2	0.5	1.0	iA< sD	328.0	312.0	10.0	2.7243	88.66	10.0	86.66
15	2	2	1.0	1.0	iA==sD	181.0	178.0	1.0	0.5361	58.39	9.0	30.12
16	2	3	2.0	1.0	iA> sD	90.0	90.0	0.0	0.0341	26.01	2.0	3.79
17	2	3	0.5	1.0	iA< sD	340.0	326.0	9.0	6.6390	96.29	15.0	202.89
18	2	3	1.0	1.0	iA==sD	176.0	174.0	0.0	0.1721	46.64	5.0	9.89
19	3	1	2.0	1.0	iA> sD	73.0	73.0	0.0	0.0000	14.65	0.0	0.00
20	3	1	0.5	1.0	iA< sD	360.0	350.0	7.0	0.7692	72.24	5.0	21.98
21	3	1	1.0	1.0	iA==sD	162.0	157.0	0.0	0.0244	27.94	2.0	1.48
22	3	2	2.0	1.0	iA> sD	92.0	91.0	0.0	0.0008	20.02	1.0	0.08
23	3	2	0.5	1.0	iA< sD	356.0	356.0	0.0	0.4712	65.86	7.0	13.24
24	3	2	1.0	1.0	iA==sD	176.0	176.0	0.0	0.0187	32.46	2.0	1.06
25	3	3	2.0	1.0	iA> sD	95.0	95.0	0.0	0.0040	19.18	1.0	0.42
26	3	3	0.5	1.0	iA< sD	347.0	346.0	0.0	0.4142	66.86	5.0	11.97
27	3	3	1.0	1.0	iA==sD	174.0	174.0	0.0	0.0157	31.69	2.0	0.90

Labels,
iA = *mean interarrival*
sD = *mean Service*

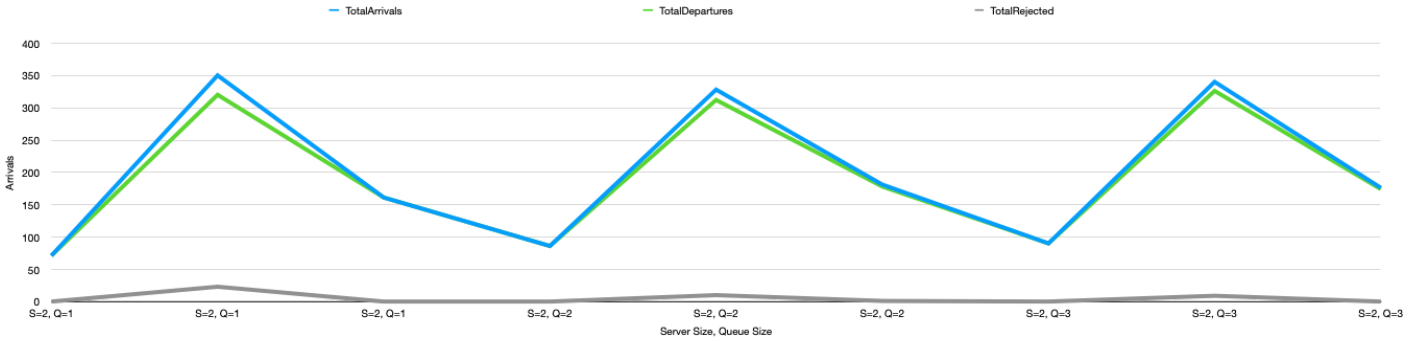
To read the line graph, the blue line (total arrivals) and green line (total departures) should align with each other meaning, what comes in also comes out, but if they are further apart, it means that, there are people still not serviced yet. The closer they are to each other, the better, the further apart they are, the worse. The grey line (total rejected) should be close 0, meaning no one is rejected, if it rises, it means that some are rejected, which is bad, if it's closer to 0, to the ground, the better.

To read the column graph, the higher the column bar (rises to the top), the worse it is.

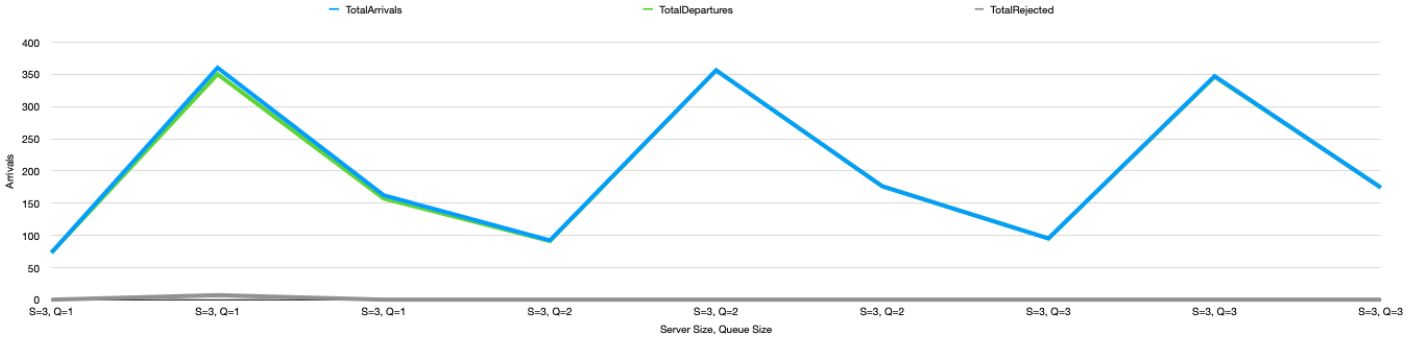
1 Server, 1-3 Queue(s)



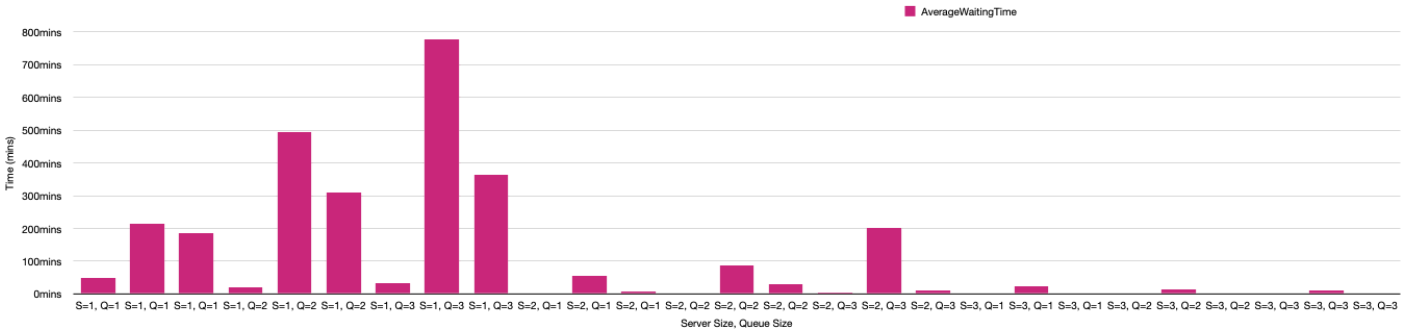
2 Servers, 1-3 Queue(s)



3 Servers, 1-3 Queue(s)



1-3 Server(s), 1-3 Queue(s)



Discussion

Slow Arrivals:

- Lines barely formed, so workers often idle.
- Obviously, no one was turned away.
- Adding more workers or waiting lines doesn't make sense and help when traffic is light.

Balanced Arrivals

- Lines form more often.
- Adding a worker (server) cuts waiting time far more than adding extra waiting lines (queue).
- Once enough workers are added, waiting lines stay short and no one is turned away.

Fast Arrivals

- Obviously, waiting lines get very long and many people are turned away if workers are too few.
- Adding workers (servers) quickly fixes long waits and prevents rejections.
- Adding waiting lines (queues) without extra workers makes things **worse.**
- With 3 workers (servers) and 2 or 3 waiting lines (queues), the system handled fast arrivals with no rejections and low waiting times.

Limitations

- The waiting line's maximum people (max queue length or the buffer size) is 5.
- I used only one queue selection method (choose the shortest line for the interarrival)
- Service times follow a simple random distribution. It could be further improved, where a certain line services a people with less load (like in the context of supermarket, a person have a basket which carries less items than a cart).
- Same results were seen consistently across 3 runs.

Conclusion

Under medium or high load, **add workers (servers) first**. Adding more separate waiting lines (queue) helps only after enough workers (servers) are in place. So for businesses that has space enough for three workers (servers), like a Supermarket or Casa (Car Authorized Service Center), I recommend to choose two or three waiting lines (queues), where **three** if you prioritize *customer service* (customers are more happier this way), or if you prioritize *costs and profits*, then choose **two**. Either way, both are optimal and proven in the data I have gathered, it comes down to which you prioritize.