Analysis of a G/G/1 queuing system

Jordi Pique Selles

FIB • UPC

Abstract

In this paper we analyze a G/G/1 queuing system with a long-tailed random distribution for the service time using a simulation. We also compare these empirical results with theoretical values.

1. Introduction

Queuing models are a widely used tool to analyze and predict the behaviour of lots of different systems, from transportation systems to data transfer in a computer network. In this project we want to analyze the behaviour of a G/G/1 queuing system and see the mean occupancy and the mean length of the waiting queue. Section 2 gives a description of the queuing system. In section 3 there's a short explanation about the implementation of this system. Section 4 reports basic statistic of the service time. In section 5 there's a deep analysis of the results of the simulation of the queuing system. Finally, in section 6, we summarize the results and write the conclusions.

2. The queuing problem

As said in the Introduction, in this project we want to analyze the behaviour of a G/G/1 queuing system. G/G/1 means that is a system with a **general random distribution** for the inter-arrival time and for the service time. There's only **one server** and the **queue can grow infinitely**.

Our inter-arrival time follows a Weibull distribution with parameters $a = 2$ and $b = 88$. The service time also follows a Weibull, but with parameter $a = 0.5439$ and b is not defined. The value of the parameter a in the service time creates a probability density function with a long tail. As we will see later, that increases a lot the variance of this random variable.

The goal is to see the value of the mean occupancy of the system and the mean length of the queue in different conditions. We will make different simulations with different loading factors for the system. The desired value for the loading factor will be achieved thanks to the free value of the parameter b of the service time distribution.

3. Implementation of the simulator

The simulator has been implemented in Python.

3.1. The Weibull random generator

In order to have random numbers following a Weibull distribution, we have implemented a function that takes the parameters a and b and return this random number. This function generates a random number x from a uniform distribution $[0, 1]$ using the random package in Python. Then it uses this number x and the inverse of the cumulative distribution function from the Weibull to return the desired number y.

$$f(x, a, b) = \frac{a}{b}\left(\frac{x}{b}\right)^{a-1} e^{-\left(\frac{x}{b}\right)^{a}}$$

$$F(x, a, b) = y = 1 - e^{-\left(\frac{x}{b}\right)^{a}}$$

$$x = b \cdot log(1 - y)^{1/a}$$

3.2. Getting statistical informatiom from the Weibull distribution

In order to have a tool to make the analysis required for Section 4 we have implemented two functions. The first gives the theoretical values for the **mean**, the **variance** and the **coefficient of deviation**. The second generates n random values from the Weibull and calculates the same statistical information but from this empirical data.

3.3. The simulator

The core of the simulator simply repeats the following recursion n times, where n is the number of clients to be simulated.

$t_i = t_{i-1} + weibullArrivalTime$
$ts_i = max(t_i, \omega_{i-1})$
$wq_i = ts_i - t_i$
$w_i = wq_i + weibullServiceTime$
$\omega_i = w_i + t_i$

where

t_i is the time when the user i enters the system
ts_i is the time when the user i enters the server
ω_i is the time when the user i exits the system
wq_i is the time spent in the queue by the user i
w_i is the time spent in the system by the user i

4. Analysis of the service time

In order to show that our distribution for the service time is a long-tailed distribution we have performed first a quick analysis. We have created 10000 numbers following that distribution and we computed the mean, variance and coefficient of deviation. Then, we have compared these results with the theoretical values. The simulated values correspond to a Weibull with $a = 0.5439$ and $b = 31$. This b is the one that gives a $\rho = 0.7$ (mean occupancy of the server).

	Theoretical values	Simulated values
Mean	53.7	52.61
Variance	11467.45	10740.08
Coef Dev	1.99	1.97

As we can see in Figure 1, the distribution has a long tail.

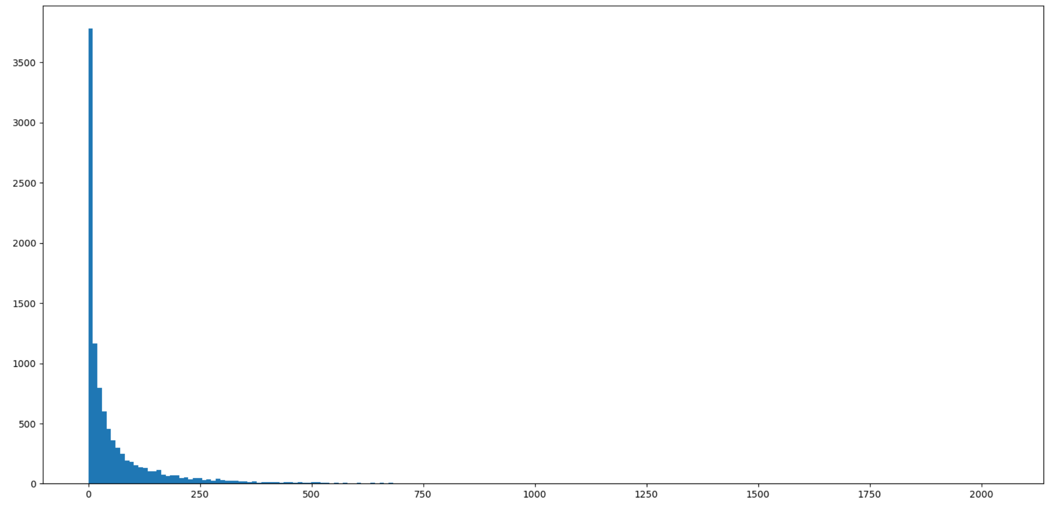

Figure 1: Service time histogram for 10000 samples following a Weibull with a=0.5439 and b=31

5. Analysis of the queuing system

In order to analyze the system, we have executed 10 runs of the simulator for each of the following loading factors: [0.4, 0.7, 0.85, 0.925]. For each execution we have computed the mean occupancy in the system (L), the occupancy in the queue (L_q), the mean waiting time in the system (W) and the mean waiting time in the queue (W_q). In order to be sure that the system reaches the stationary state, we have plot the evolution of L. Then, we have computed the confidence interval for L_q and W_q. Finally, we have compared the empirical results with the **Allen-Cuneen's** approximation formula.

If we look at the plots in the Appendix B we will see the evolution of L. We can consider that the system reaches the stationary state. It's true that we can notice some fluctuation in the plot with $\rho = 0.7$ and in the plot with $\rho = 0.85$ but that's because of the resolution of the vertical axis. If that axis went from 0 to 1 we will see a flat line at the end of each plot. Once we can assume the system reaches the stationary state, we can continue the analysis.

In an optimal system, a system such that the inter-arrival time and the service time are constant, we would see a $L_q = 0$ (mean queue length). However, in our system these times follow random distributions. The service time has a high variance and that gives high values for L_q. Maybe, at a given time few clients arrive to the system and the service time is short and the server is idle. But in another time more clients will arrive and the service time will be longer, because the empirical mean of inter-arrivals and service time must tend to the theoretical mean. That increases the clients that are waiting in the queue and that is reflected to the mean value of L_q.

This same phenomena can be explained using the Allen-Cuneen's approximation formula.

$$L_q \approx \frac{\rho^2}{1-\rho} \cdot \frac{{C_a}^2 + {C_s}^2}{2}$$

If C_a and C_s (coefficients of deviation from inter-arrival time and service time) are 0, then $L_q = 0$. As we increase one of the coefficients of deviations (or both) L_q increases. Another behaviour that can be deduced from the last formula is that L_q increases as ρ increases. That's what we see from our simulation.

The Allen-Cuneen's approximation formula gives a good approximation for the $\rho = 0.7$ and $\rho = 0.85$ simulations. For the $\rho = 0.4$ simulation the approximation is out of the confidence interval for L_q.

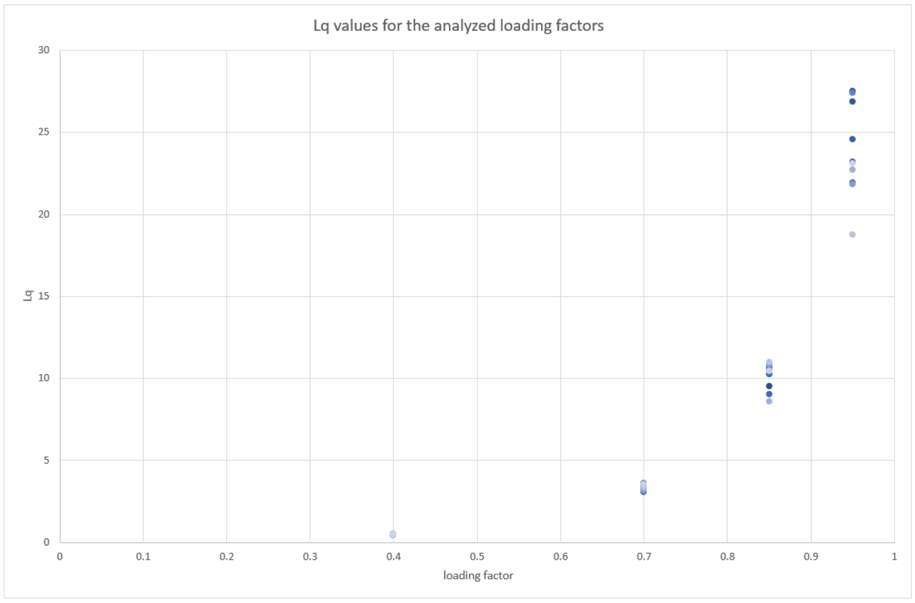

Figure 2: L_q values for the different loading factors

ρ	Approximation	Confidence interval	
0.4	0.57	0.46	0.49
0.7	3.47	3.25	3.51
0.85	10.23	9.54	10.69
0.925	24.24	21.78	25.81

Table 1: L_q approximation and confidence interval for different loading factors

6. Conclusions

In this paper we have seen the effect of a long-tailed random distribution in a G/G/1 queuing system. In our simulation, we have seen that effect in the mean value of L, L_q, W and W_q. This values are considerably large and they increase as ρ increases. This same effect can be deduced from the Allen's Cuneen approximation.

Appendix A. Instructions for the execution of the simulator

The simulator is simply a Python file. You need to call it from the terminal.

```
python3 runSimulation.py numberUsers numberRepetitions \
                                [listLoadingFactor] [-v] [-p] [-w]
```

- **numberUsers:** The number of users that will enter the system.
- **numberRepetitions:** The number of times each simulations will be executed.
- **listLoadingFactor:** (Optional) The a list of loading factors. For each loading factor, *numberRepetitions* will be executed. It must have this format: [p1, p2, p3, ...]. If it is not specified, then the program will be executed with [0.4, 0.7,0.85, 0.925].
- **-v:** (Optional) Enable verbose mode.
- **-p:** (Optional) Show the plots of the evolution of L.
- **-w:** (Optional) Write the results of each execution in a csv file.

Appendix B. Results

$\rho = 0.4$			
L	Lq	W	Wq
0.85	0.45	66.56	35.48
0.86	0.46	67.13	35.86
0.84	0.45	65.66	34.68
0.88	0.48	68.4	37.39
0.89	0.48	68.83	37.5
0.87	0.47	68.04	36.83
0.89	0.49	69.73	38.15
0.84	0.44	65.85	34.7
0.91	0.51	70.79	39.37
0.88	0.48	68.76	37.58

$\rho = 0.7$			
L	Lq	W	Wq
3.98	3.28	310.97	256.24
4.27	3.57	333.03	278.03
4.09	3.39	318.68	264.12
3.74	3.04	291.27	236.73
4.31	3.61	335.78	281.33
4.18	3.48	325.98	271.27
3.88	3.18	302.8	248.27
3.99	3.29	311.5	256.62
4.06	3.37	317.1	262.66
4.27	3.56	332.51	277.44

$\rho = 0.85$			
L	Lq	W	Wq
10.38	9.54	809.39	743.52
11.11	10.26	867.71	801.51
11.54	10.68	897.42	830.82
9.86	9.02	772.04	705.95
11.11	10.26	866.36	800.0
11.38	10.53	888.52	821.65
11.68	10.82	908.99	842.35
9.44	8.6	736.74	671.35
11.84	10.99	926.54	859.59
11.3	10.46	879.99	814.26

$\rho = 0.925$			
L	Lq	W	Wq
28.41	27.49	2219.13	2147.17
27.78	26.85	2169.71	2097.59
25.5	24.58	1989.09	1916.9
24.13	23.21	1880.27	1808.43
22.87	21.95	1781.17	1709.1
28.35	27.42	2212.36	2139.71
22.76	21.84	1770.04	1698.16
23.66	22.74	1842.2	1770.45
19.68	18.76	1538.89	1467.19
24.06	23.13	1877.99	1805.84

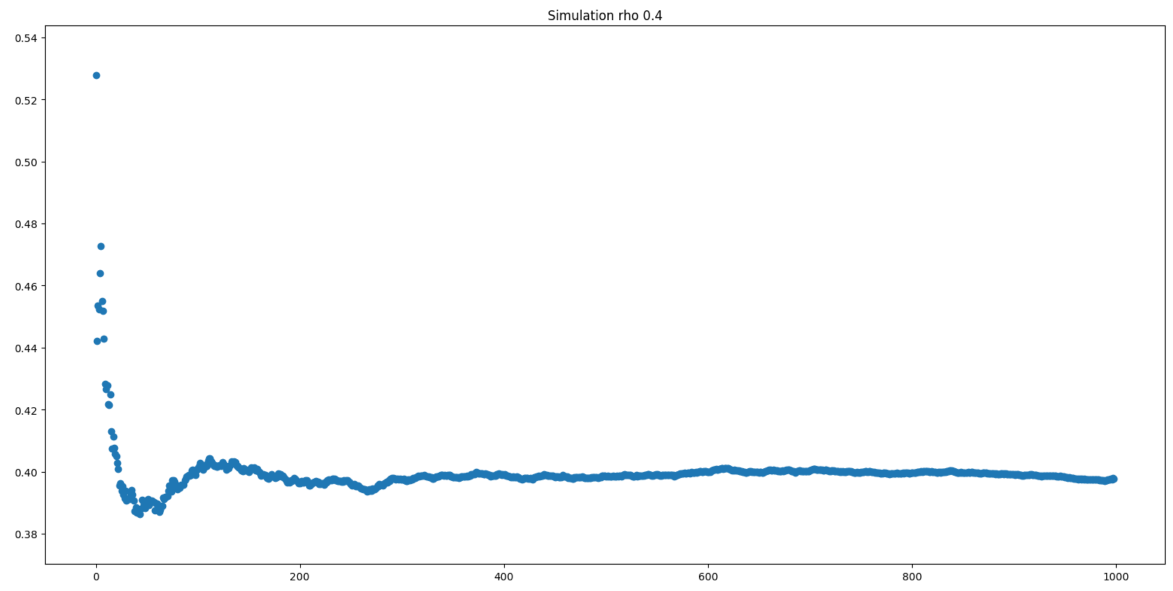

Figure B.3: Evolution of the occupancy for $\rho = 0.4$

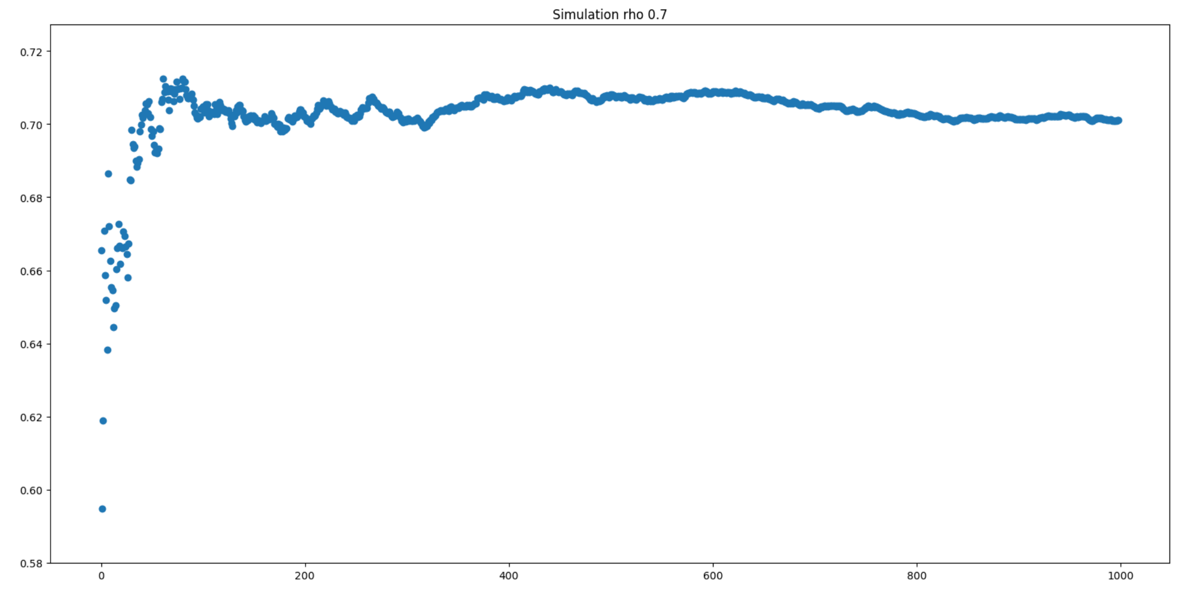

Figure B.4: Evolution of the occupancy for $\rho = 0.7$

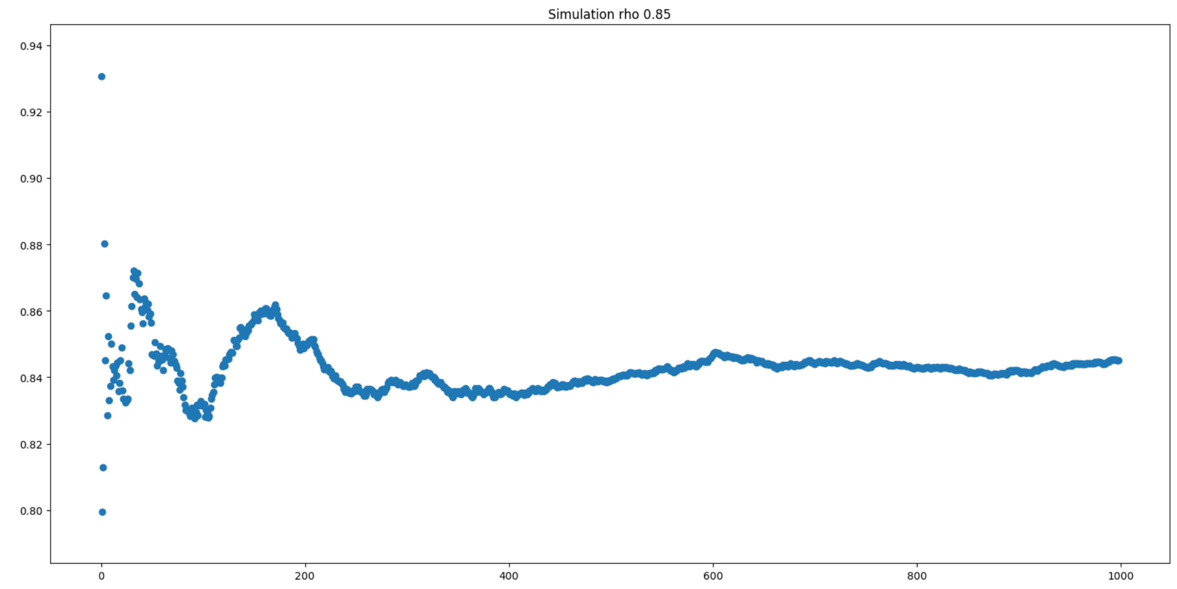

Figure B.5: Evolution of the occupancy for $\rho = 0.85$

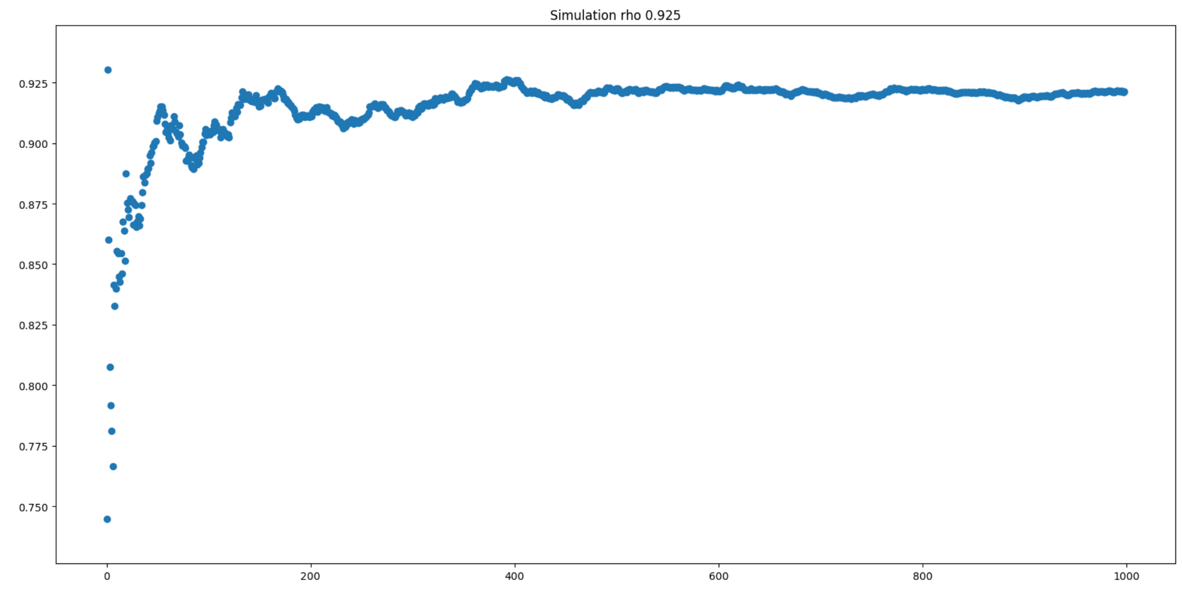

Figure B.6: Evolution of the occupancy for $\rho = 0.925$