SMDE - Queues project

Balbina Virgili
Laura Cebollero

January 9, 2018

Contents

1 Introduction

The aim of this project is analyze and understand the functionality of a G/G/1 Queueing system that follows a long-tailed distribution of probability. To achieve it, the following points need to be developed:

- Use Allen-Cuneen's approximation formula for analyzing the problem.
- Use a simulator to deeply understand the behaviour of the queueing system.
- Develop our own simulator of the given system.
- Compare and discuss the results obtained.

1.1 Parameters

The parameters used to develop this assignment are the ones from row 48:

- **Arrival**
  - **Distribution**: Weibull
  - **Shape:** $a = 2$
  - **Scale:** $b = 88$
  - **Expectation:** $E[\tau] = 78$
- **Service**
  - **Distribution**: Weibull
  - **Shape:** $a = 0.5425$

All elements needed for fully simulating the arrival distribution are given.

Unfortunately, b value and expectation of service's Weibull distribution are missing.

To be able to compute b, we are going to use the formula stated below:

$$b = \frac{\rho E[\tau]}{\Gamma\frac{a+1}{a})}$$

where, ρ, must take the values $0.4, 0.7, 0.8$ and 0.925, according to the instructions.

As we know, ρ is the loading factor. In other words, it is the proportional difference between the service time and the sojourn time. So, the greater the ρ, the bigger the loading on the system.

$$\rho = \frac{E[x]}{E[\tau]} = \frac{b\Gamma(\frac{a+1}{a})}{E[\tau]}$$

From the previous formula, once ρ has been determined, we can also compute the service expectation E[x].

$$E[x] = \rho E[\tau]$$

So, for each value of ρ, we are going to obtain a different b (scale) and E[x]. The resulting values calculated, which will be needed for the system calculation, are shown in the following table:

	ρ	**Shape, a**	**Scale, b**	**E[τ]**	**E[x]**
1	0.4	0.5425	17.93	78	31.2
2	0.7	0.5425	31.39	78	54.6
3	0.8	0.5425	35.87	78	62.4
4	0.925	0.5425	41.48	78	72.15

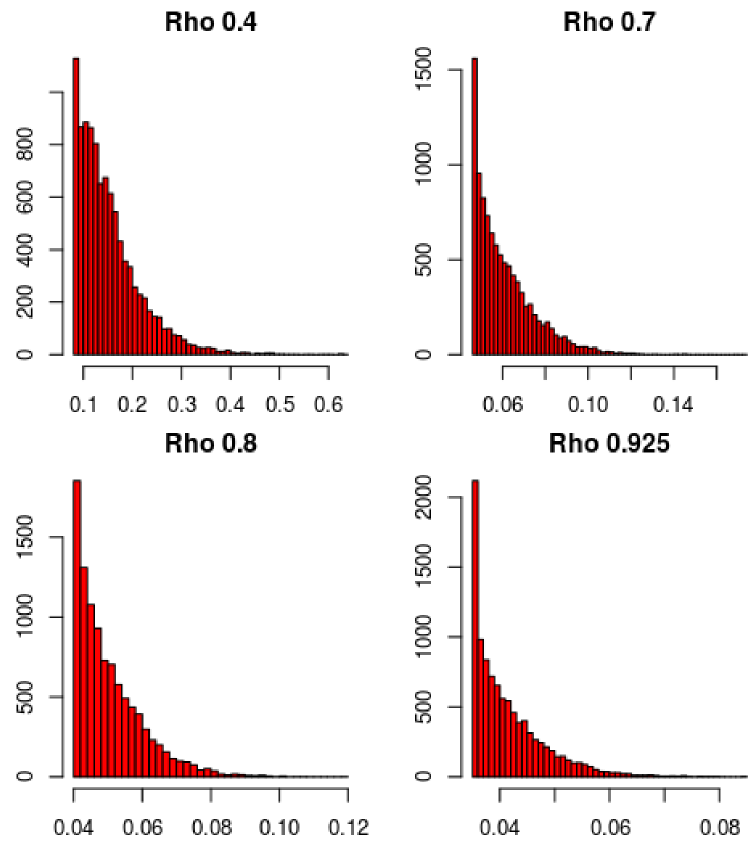
2 Weibull service distribution

First of all, we are going to start analyzing the long-tailed distribution of our system. For it, a small simulation of 10.000 values has been performed and the basic characteristics of the given service distribution are being calculated.

We have implemented a small simulation of the system with 10.000 individuals in R, which can be found attached to this report. It computes the main statistics such as the mean, standard deviation and median, and then prints an histogram for each of the four cases.

Rho	**Mean**	**Median**	**Std Dev**
0.4	0.1516	0.1348	0.0642
0.7	0.0610	0.0571	0.0139
0.8	0.0506	0.0478	0.0099
0.925	0.0416	0.0397	0.0067

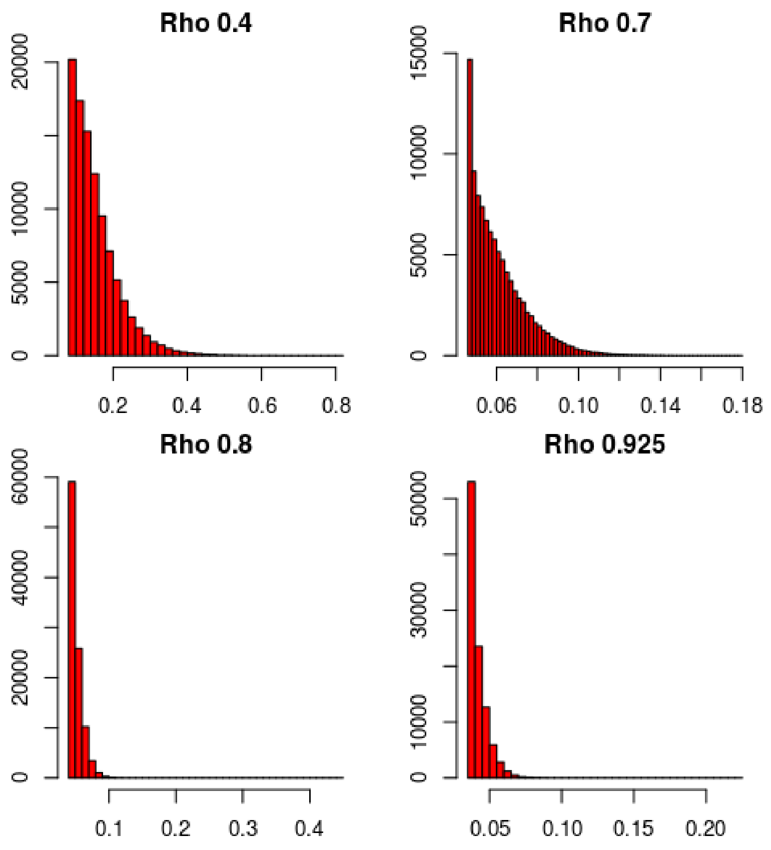
Resulting in these 4 histograms, where the density function has been applied:



As we know, theoretically, Weibull distribution slightly changes its behaviour according to the b value, which represents the scale of the function. So, we can see with the results retrieved that our implemented distribution is working well, as each rho is showing the expected results with different scale. And we can also conclude that as larger the value of b is, as faster the Weibull distribution arrives to the maximum values and biggest numbers are reached.

Now, we are going to retrieve the same results but we are going to increase the individuals from 10.000 to 100.000.

Rho	**Mean**	**Median**	**Std. dev**
0.4	0.152	0.136	0.064
0.7	0.061	0.057	0.014
0.8	0.051	0.048	0.010



With the results shown, we can see that the values of mean, median and std. dev have increased slightly, but not much. This way, we can confirm the behavior observed with 10000 individuals of the the long-tailed distributions, specially seen in the histograms.

After analyzing the basic characteristics and behaviour of Weibull service distribution, now, we can start the comparison between the approximation of Allen Cuneen, the values obtained from the Java simulator and our own implementation of the system.

3 Queueing system development

3.1 Allen - Cuneen approximation

The Allen Cuneen approximation formula has been computed manually via R and each variable has been individually computed. The source code created can be found attached to this report.

The formula is the following one:

$$\mathrm{E}[w_q] = W_q \approx \frac{C(s,\theta)(\lambda^2\sigma_\tau^2 + \mu^2\sigma_x^2)}{2s\mu(1-\rho)}$$

where:

- $\sigma_\tau^2 = Var[\tau]$;
- $\sigma_x^2 = Var[x]$;
- $\Theta = \frac{\lambda}{\mu}$

$$C(s,\Theta) = P_{M/M/1} = \frac{\frac{\Theta}{1-\rho}}{\frac{\Theta^l}{l!} + \frac{\Theta}{1-\rho}}$$

And to compute L_q: $L_q = \lambda * W_q$.

Our results obtained using the formulas just explained are summarized in the following table:

	ρ	**a**	**b**	**E**$[\tau]$	**E**$[x]$	$\sigma^2(\tau)$	$\sigma^2(x)$
1	0,4	0,5425	17,938	78,000	31,200	1661,877	3897,424
2	0,7	0,5425	31,391	78,000	54,600	1661,877	11935,862
3	0,8	0,5425	35,875	78,000	62,400	1661,877	15589,697
4	0,925	0,5425	41,481	78,000	72,150	1661,877	20842,085

	$\sigma(\tau)$	$\sigma(x)$	λ	μ	W_q	L_q
1	40,766	62,429	0,013	0,032	44,480	0,570
2	40,766	109,251	0,013	0,018	272,440	3,492
3	40,766	124,859	0,013	0,016	533,760	6,843
4	40,766	144,368	0,013	0,014	1902,908	24,396

As it can be seen, according to Allen-Cuneen formula, the loading factor ρ is crucial on the waiting queue, both on the avg. waiting time as on the length of it.

While upping the ρ value, we can see that the Wq increases greatly. The same happens with Lq.

We have made two plots to see it easily.

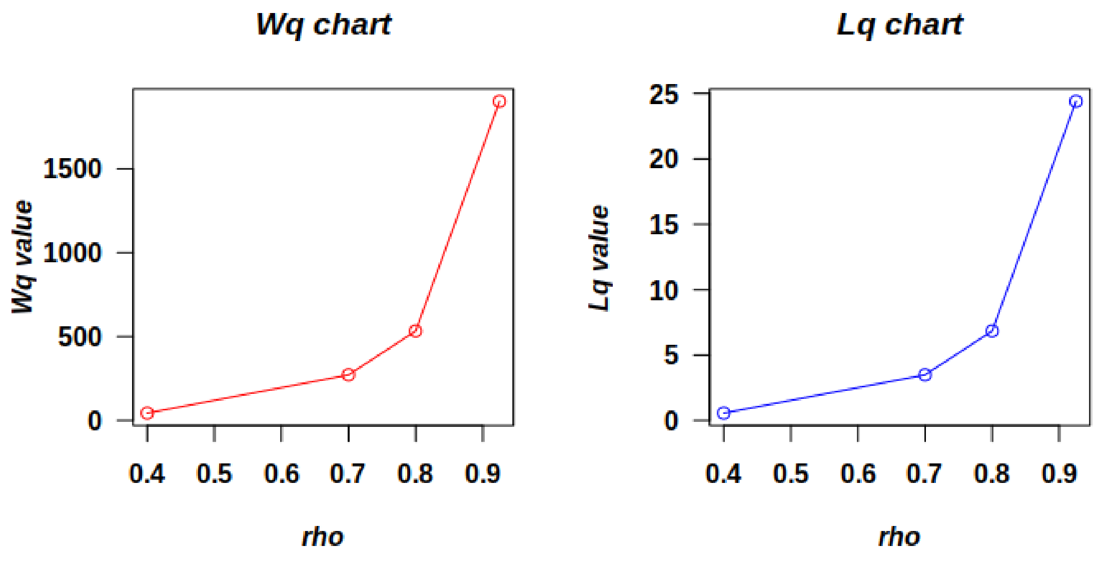


Figure 1: Allen Cuneen Wq and Lq approximation

3.2 Java simulator

Now, we are going to simulate the system using the Java simulator provided in this course.

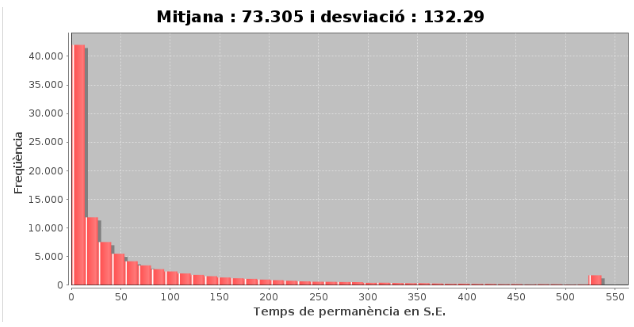
We have performed 10 executions with different seeds for each ρ and computed the mean of all of them. However, we want to exemplify each result with the graphics of one execution for each of them.

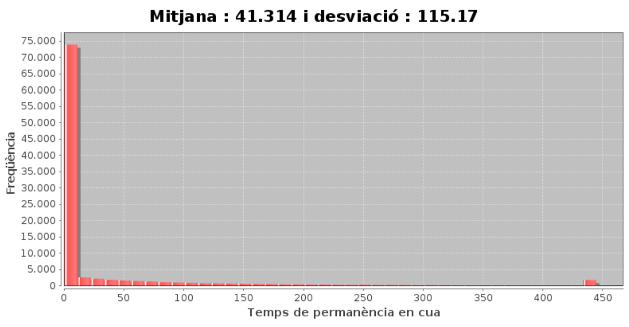
The parameters used have been the ones calculated at the start of this report, which are:

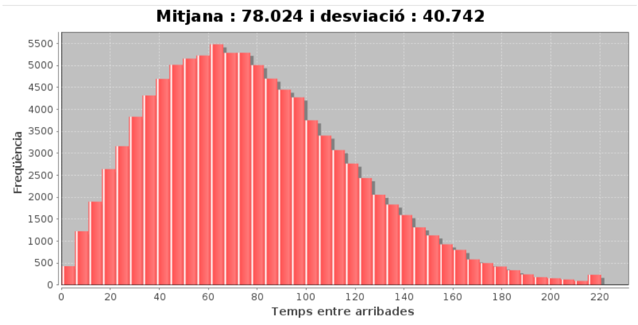
	ρ	**Shape, a**	**Scale, b**
1	0.4	0.5425	17.93
2	0.7	0.5425	31.39
3	0.8	0.5425	35.87
4	925	0.5425	41.48

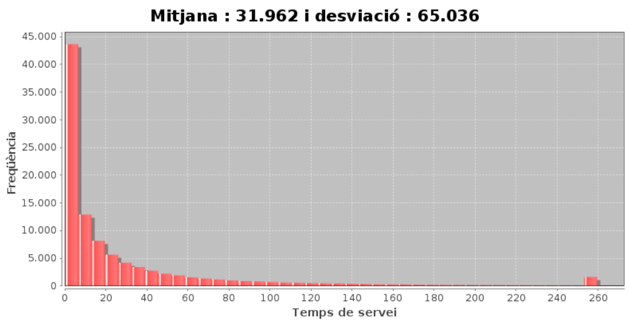
3.2.1 Rho 0.4

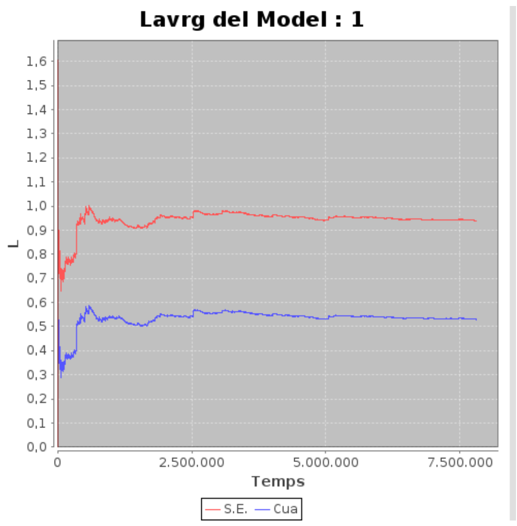
For a loading factor of 0.4, the results obtained have been:

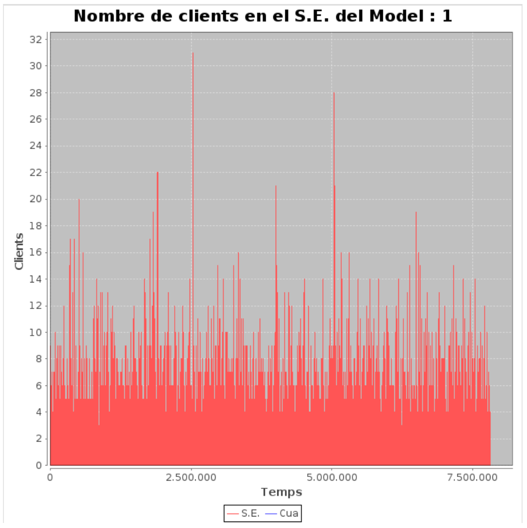






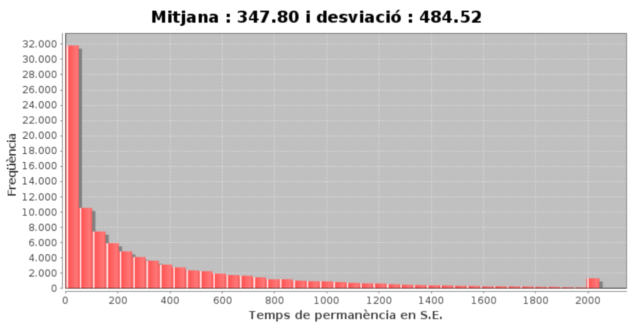


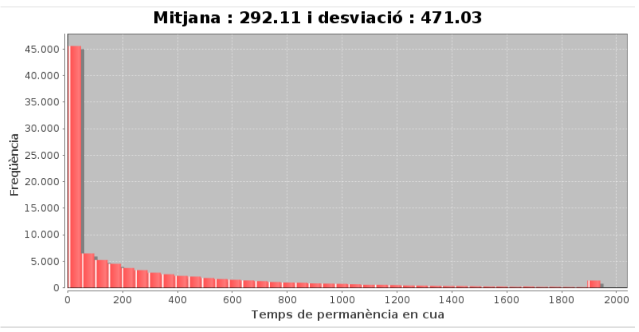


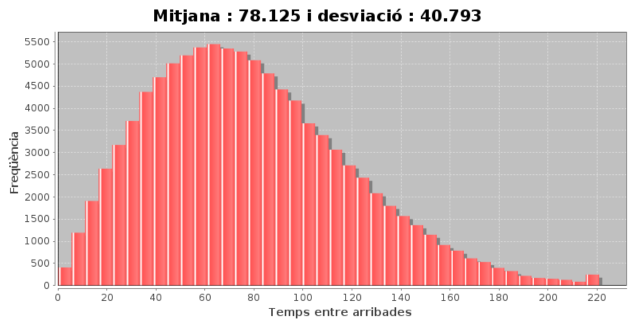


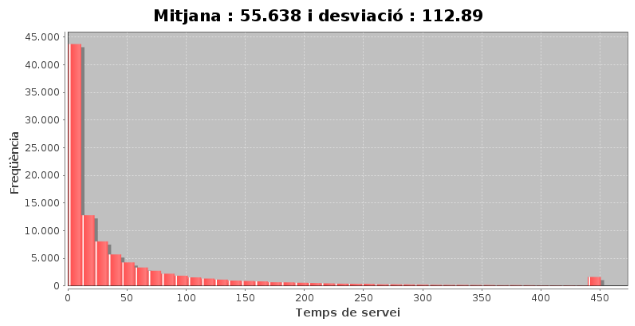
3.2.2 Rho 0.7

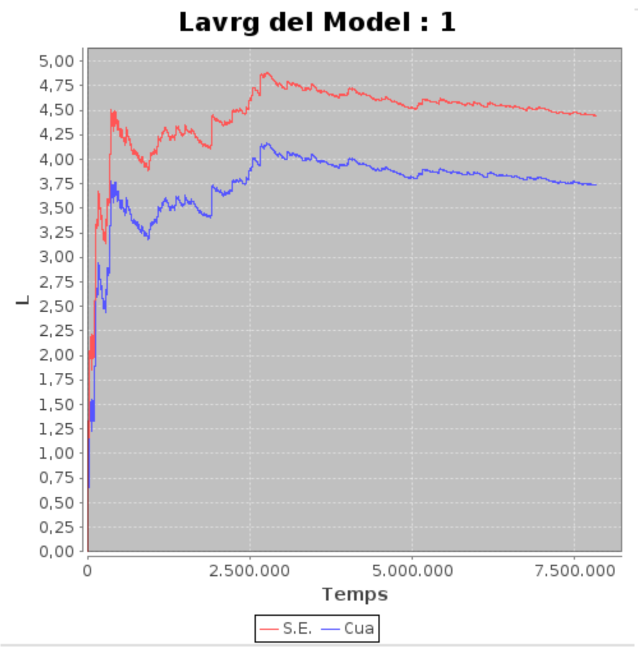
For a loading factor of 0.7, the results obtained have been:

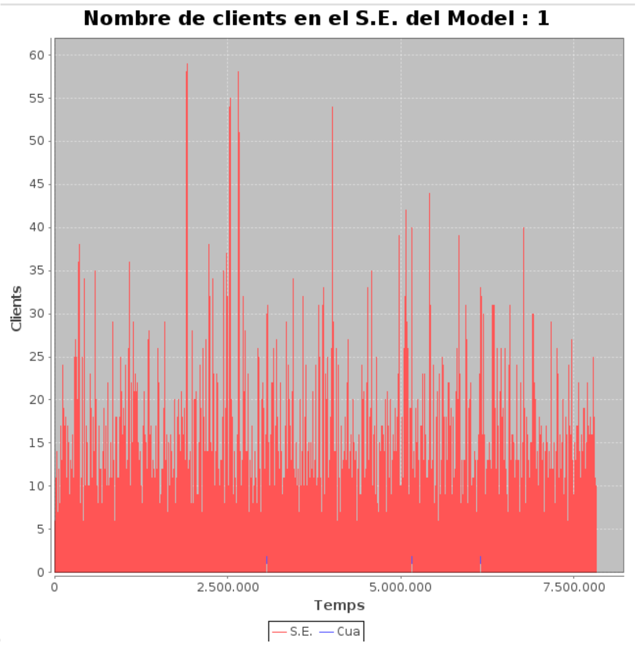






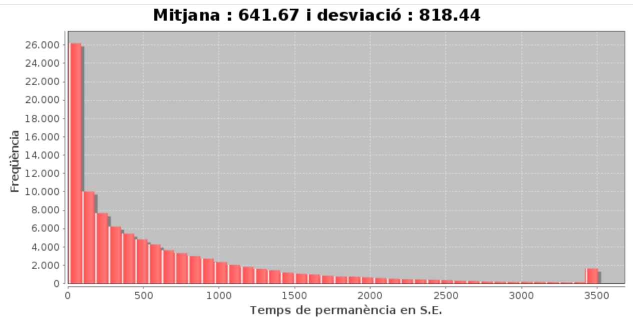


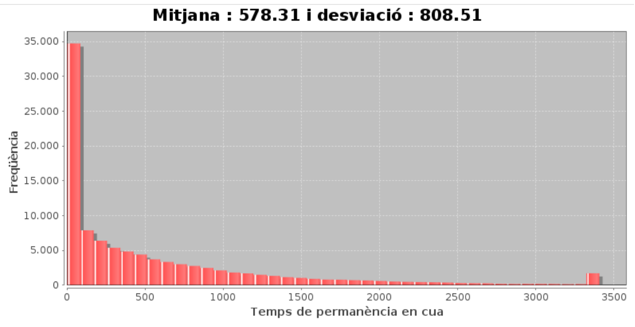


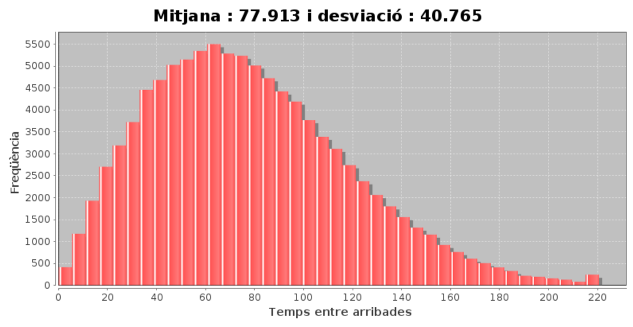


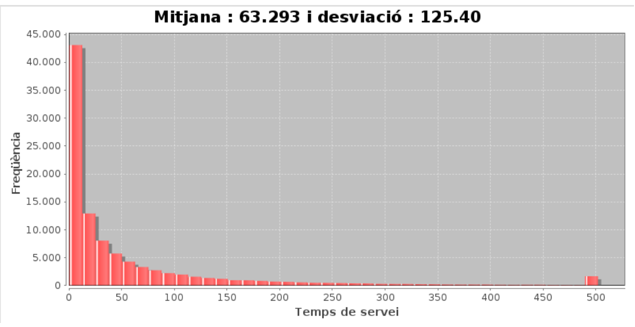
3.2.3 Rho 0.8

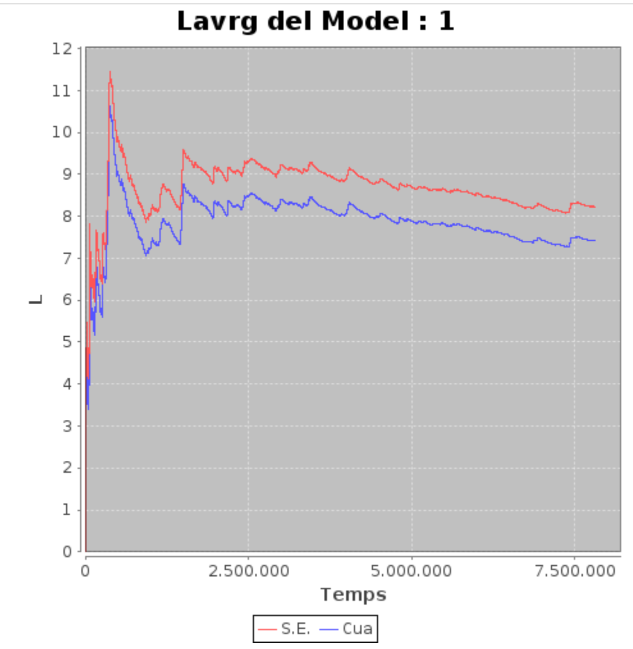
For a loading factor of 0.8, the results obtained have been:

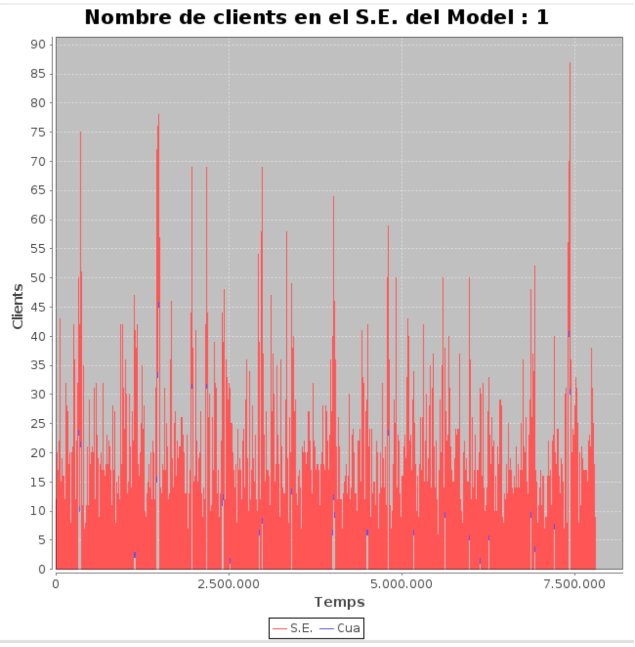






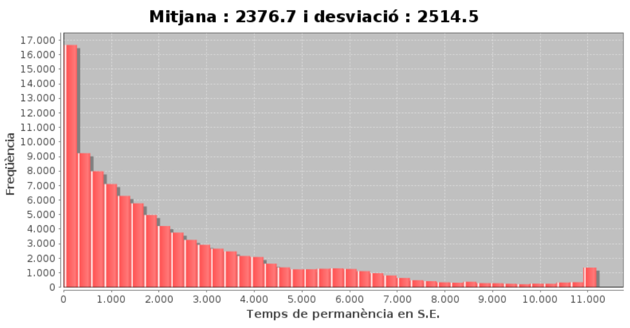


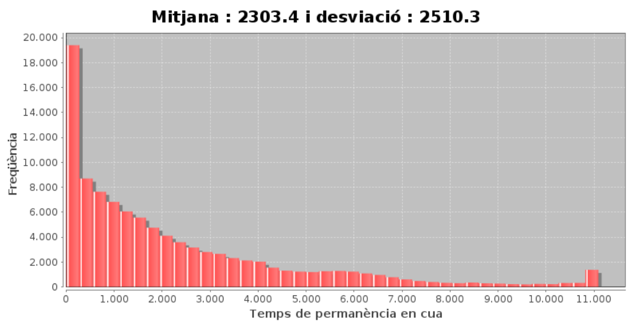


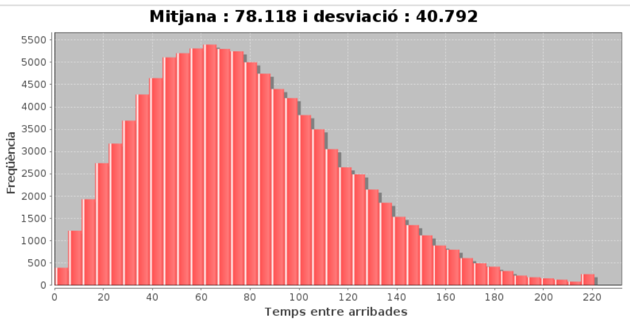


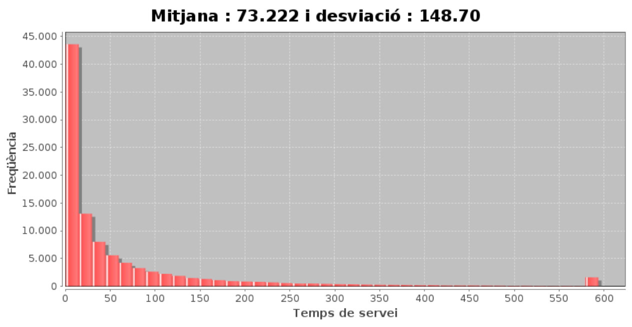
3.2.4 Rho 0.925

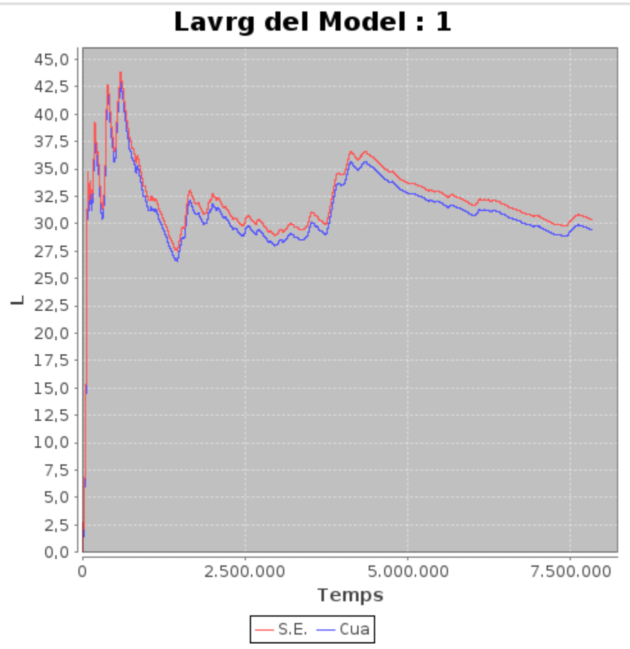
For a loading factor of 0.925, the results obtained have been:

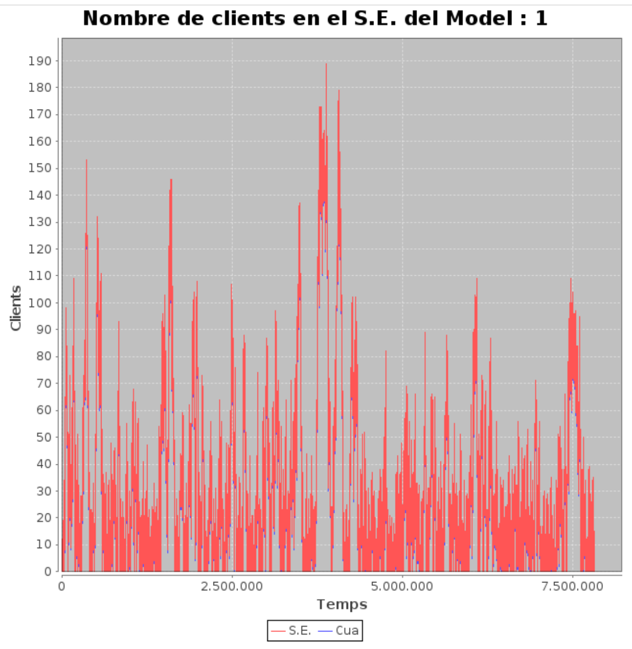












3.2.5 Java simulator executions summary

Taking into account that the results may vary on the seed, we performed 10 different executions for each ρ with different seeds and we computed the confidence interval of W_q and L_q, resulting in the following table:

Wq, n=100k				
Run	**0.4**	**0.7**	**0.8**	**0.925**
1	41.314	275.27	534.35	2257.5
2	40.235	282.39	520.83	1764.9
3	41.321	254.15	519.74	1809.4
4	38.279	272.69	475.14	1504.2
5	36.089	258.13	491.34	2190.7
6	38.135	257.61	535.69	1853
7	37.157	260.4	488.23	1682.3
8	38.842	266.15	549.26	2066.4
9	39.944	237.73	541.76	1581.9
10	37.896	266.84	468.8	1966.2
Mean	**38.9212**	**263.136**	**512.514**	**1867.65**
Std. dev	±1.75	±12.57	±29.22	±251.18

Lq, n=100k				
Run	**0.4**	**0.7**	**0.8**	**0.925**
1	0.53	3.53	6.85	28.94
2	0.52	3.62	6.68	22.63
3	0.53	3.26	6.66	23.20
4	0.49	3.50	6.09	19.28
5	0.46	3.31	6.30	28.09
6	0.49	3.30	6.87	23.76
7	0.48	3.34	6.26	21.56
8	0.50	3.41	7.04	26.49
9	0.51	3.05	6.94	20.28
10	0.49	3.42	6.01	25.21
Mean	**0.50**	**3.37**	**6.57**	**23.94**
Std. dev	±0.02	±0.16	±0.37	±3.22

Comparing all graphics shown and results calculated, we can say that as biggest the loading factor is:

- the time between arrivals has not been affected because it is defined by the arrival Weibull distribution.
- the sojourn time on W.S will increase for more individuals, although many individuals will continue being 0 seconds on it.
- the sojourn time on queue will also increase for more individuals, although, many individuals will continue being 0 seconds on it.
- the expected individuals waiting on queue increases considerably, being 0.5 when ρ is 0.4 and increasing to almost 24 when ρ is 0.925.
- the expected individuals on W.S increases considerably.

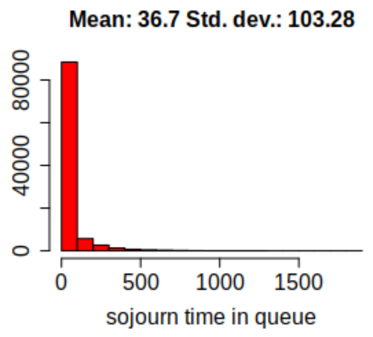
- the service time follows the expected density function for the given loading factor.
- the number of clients served at the same time does not change. It can be just one because the server just has one server. What changes is the probability distribution with which the service is working.
- The difference between the length of the system and the waiting queue is more or less the same in all values of ρ, in the Lavrg chart and they stabilize at the same time. However, the values reached increase greatly the greater the ρ.

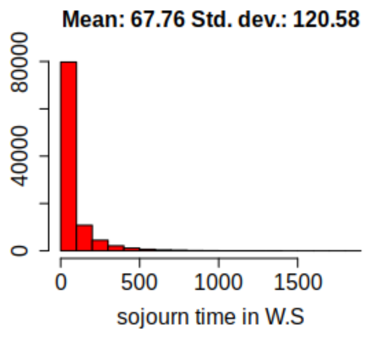
3.3 Manual implementation

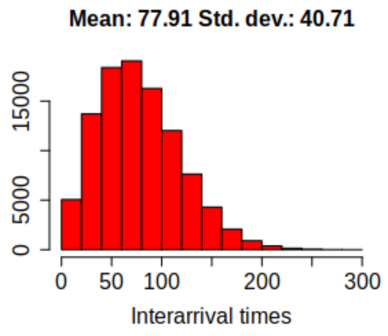
To implement the simulation of the system we used R and the pseudocode given with the instructions of the assignment. The code can be found attached to this report.

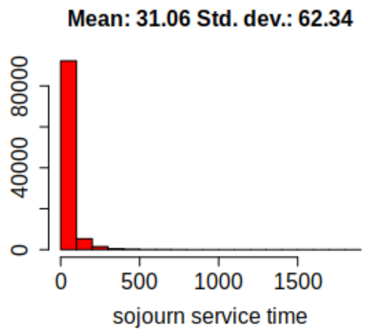
For each rho we have obtained the following graphics:

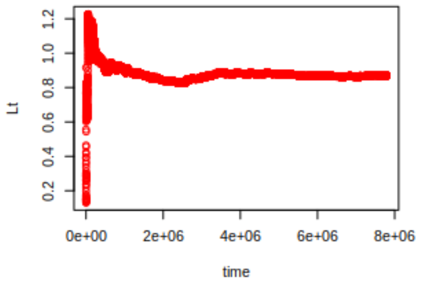
3.3.1 Rho 0.4



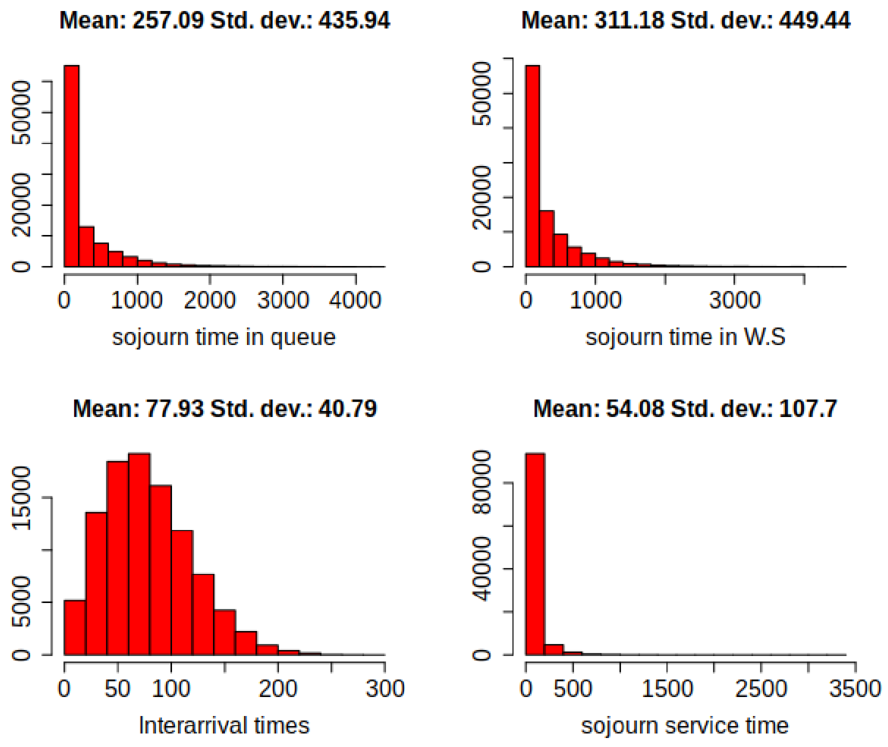




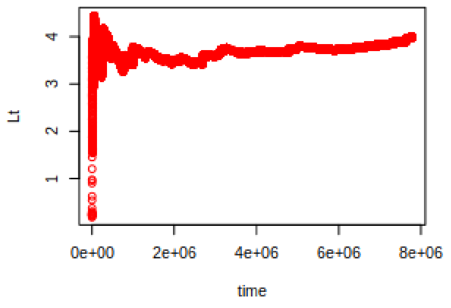




3.3.2 Rho 0.7

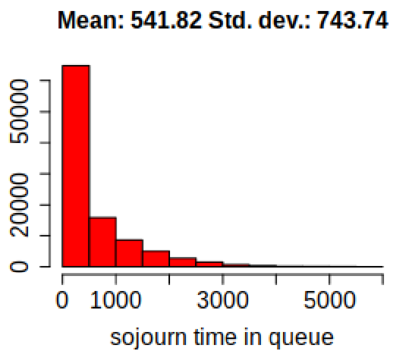


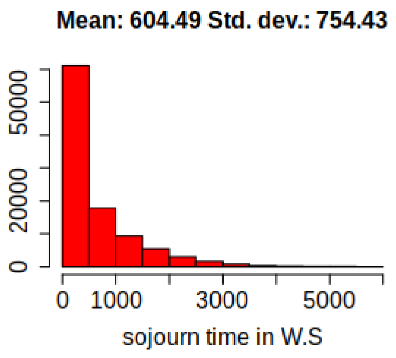
Mean: 257.09 Std. dev.: 435.94
sojourn time in queue
Mean: 311.18 Std. dev.: 449.44
sojourn time in W.S
Mean: 77.93 Std. dev.: 40.79
Interarrival times
Mean: 54.08 Std. dev.: 107.7
sojourn service time

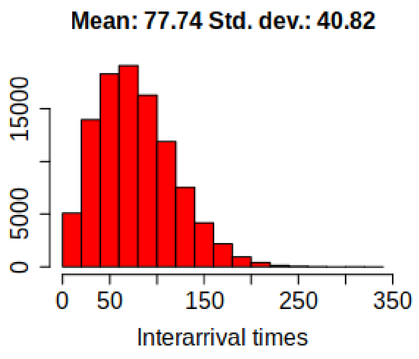


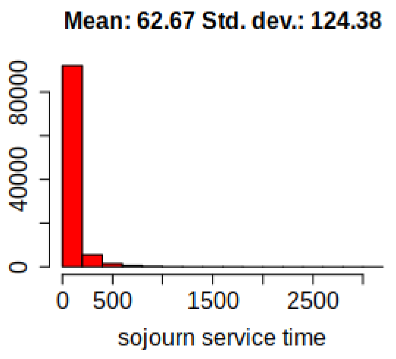
Lt
time

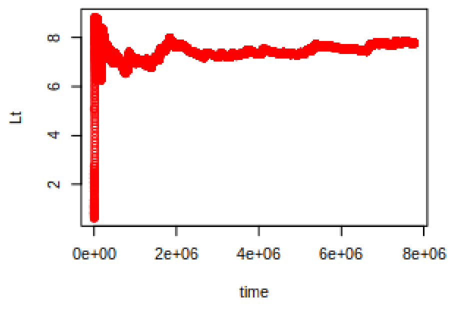
3.3.3 Rho 0.8


Mean: 541.82 Std. dev.: 743.74
0
20000
50000
0
1000
3000
5000
sojourn time in queue

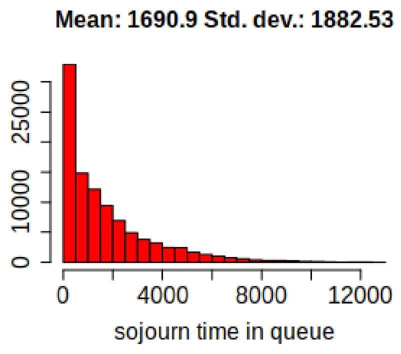

Mean: 604.49 Std. dev.: 754.43
0
20000
50000
0
1000
3000
5000
sojourn time in W.S

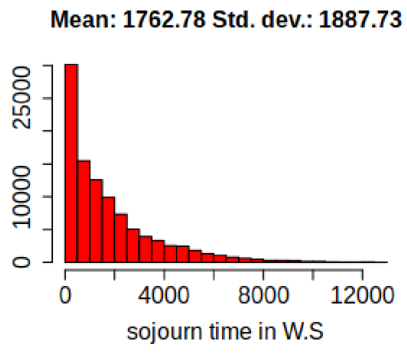

Mean: 77.74 Std. dev.: 40.82
0
5000
15000
0
50
150
250
350
Interarrival times

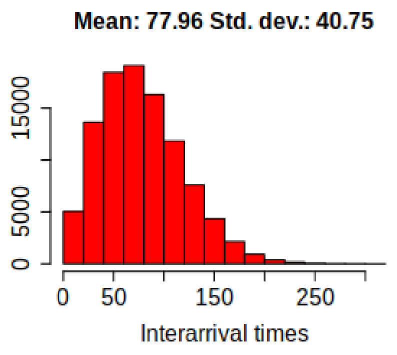

Mean: 62.67 Std. dev.: 124.38
0
40000
80000
0
500
1500
2500
sojourn service time

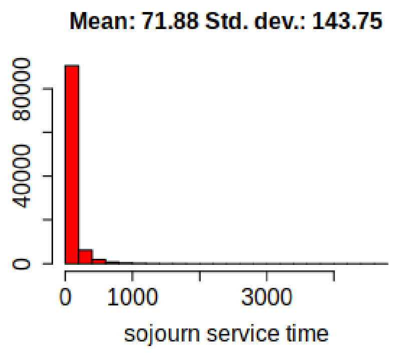

Lt
2
4
6
8
0e+00
2e+06
4e+06
6e+06
8e+06
time

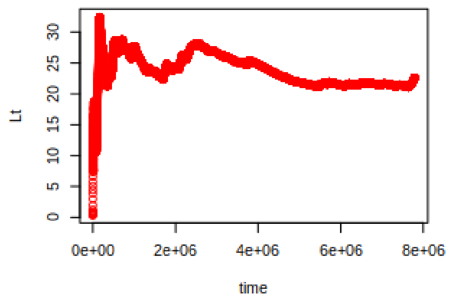
3.3.4 Rho 0.925











3.3.5 Own simulator executions summary

Here, also 10 executions have been performed in order to test the system developed more deeply. The results obtained for confidence interval of W_q and L_q are:

Wq, n=100k				
Run	**0.4**	**0.7**	**0.8**	**0.925**
1	36.754	261.20	505.908	1697.407
2	39.243	256.739	534.246	2052.343
3	37.944	267.326	534.649	1666.148
4	38.922	270.064	511.451	1525.336
5	36.429	263.395	492.026	1987.588
6	39.905	263.932	522.180	2090.817
7	38.557	275.440	520.473	1918.263
8	36.443	258.218	537.597	2146.12
9	37.084	252.038	558.679	1628.191
10	35.742	268.088	533.351	2191.31
Mean	**37.702**	**263.645**	**525.056**	**1890.35**
Std. dev	±1.407	±6.924	±18.870	±240.858

Lq, n=100k				
Run	**0.4**	**0.7**	**0.8**	**0.925**
1	0.471	3.346	6.486	21.884
2	0.503	3.295	6.864	26.293
3	0.486	3.425	6.864	21.379
4	0.500	3.462	6.535	19.591
5	0.467	3.383	6.304	25.488
6	0.511	3.393	6.709	26.836
7	0.494	3.533	6.681	24.583
8	0.467	3.312	6.888	27.561
9	0.474	3.229	7.171	20.915
10	0.458	3.438	6.841	28.171
Mean	**0.483**	**3.382**	**6.734**	**24.270**
Std. dev	±0.018	±0.089	±0.247	±3.082

Comparing all graphics shown and results calculated, the same conclusions as with Java simulator are obtained. So we can say again that as biggest the loading factor is:

- the time between arrivals has not been affected because it is defined by the arrival Weibull distribution.
- the sojourn time on W.S will increase for more individuals, although many individuals will continue being 0 seconds on it.
- the sojourn time on queue will also increase for more individuals, although, many individuals will continue being 0 seconds on it.
- the expected individuals waiting on queue increases considerably.
- the expected individuals on W.S increases considerably.
- the service time follows the expected density function for the given loading factor.
- the number of clients served at the same time does not change. It can be just one because the server just have one server, what it changes is the probability distribution with which the service is working.

4 Results comparison

In the following table we can see clearly the different results obtained in the three cases researched.

Remember that in the case of the java simulator and in our implementation, the final W_q is the mean of 10 executions with different seeds (in the case of Java simulator) and with a population of 100.000 individuals.

Wq - Results Comparison			
rho	**Allen-Cuneen aprox.**	**Java simulator (10x)**	**Manual impl. (10x)**
0.4000	44.4790	38.9212	37.702
0.7000	272.4390	263.1360	263.645
0.8000	533.7596	512.5140	525.056
0.9250	1902.9080	1867.6500	1890.35

The same applies to L_q where the results have been the following ones:

Lq - Results Comparison			
rho	**Allen-Cuneen aprox.**	**Java simulator (10x)**	**Manual impl. (10x)**
0.4000	0.5700	0.4990	0.483
0.7000	3.4920	3.3733	3.3822
0.8000	6.8430	6.5706	6.734
0.9250	24.3960	23.9441	24.270

The following images show the reviewed data of both tables in a more graphical way:

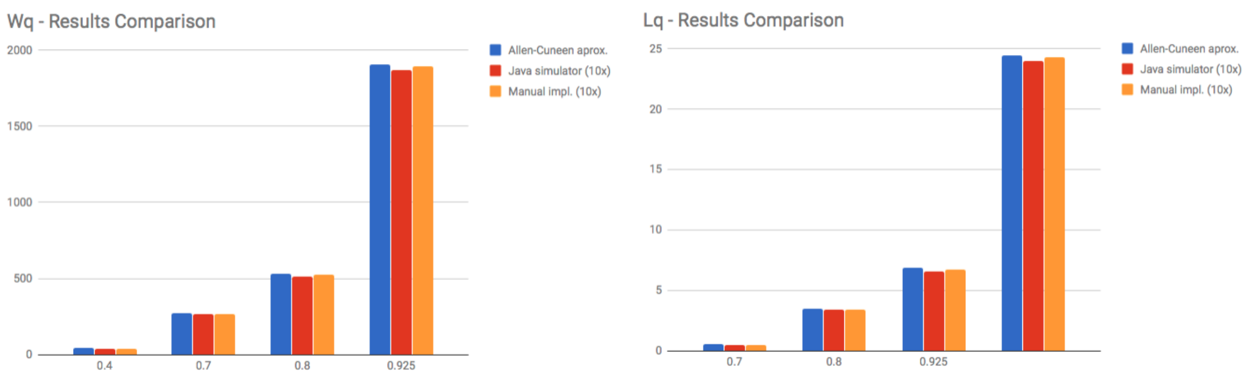


Figure 2: Wq and Lq results comparison of all implemented methods

We can observe that the results do not differ greatly neither in the W_q results nor the L_q. But, Allen Cuneen estimated in all cases a greater value of ρ than the ones obtained in Java or our implementation in R. And we observe that there is a proportional relation between W_q and L_q.

So we could say that Allen-Cuneen is a good approximating for calculating queue systems that have a long-tailed distributions and just one server.

We can also say that we have developed a great simulator, comparing it to Java one. In the following graphics we can see this in more clearly way. The four graphics show the confidence interval of values W_q and L_q for both Java and manual implementations.

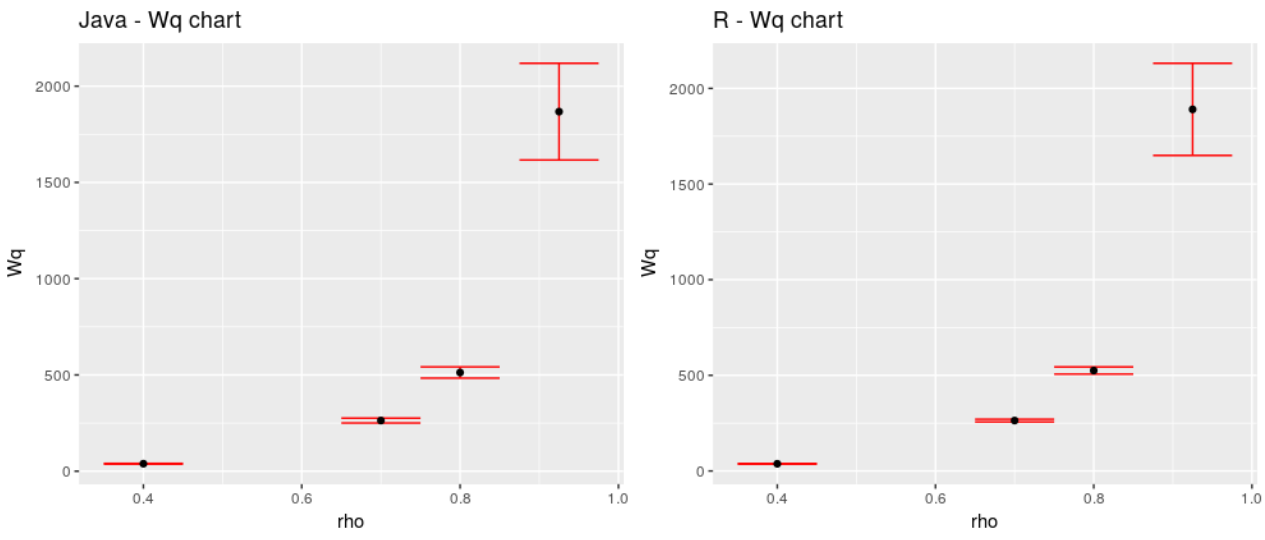


Figure 3: Java sim. (left) and R implementation (right) Wq comparison

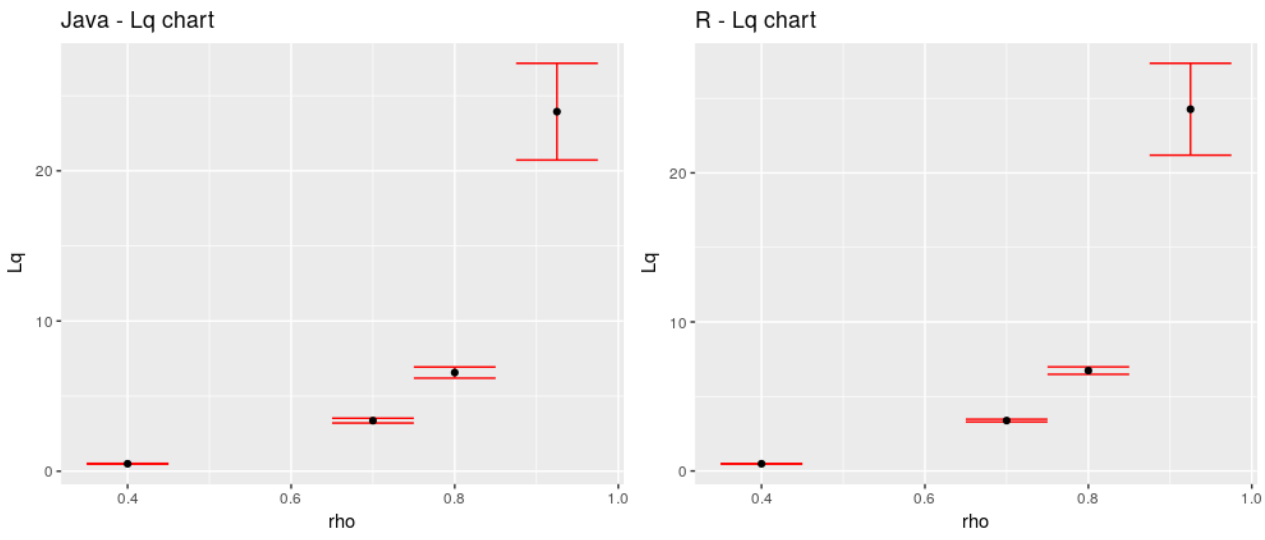


Figure 4: Java sim. (left) and R implementation (right) Lq comparison

As we can see with the graphics shown above, the confidence intervals between both simulations do not change much, and they show that the bigger the loading factor is, the greater the values and standard deviations are for W_q and L_q.

So, to sum up, we can say again that as biggest the loading factor is:

- the time between arrivals is not affected by the loading factor ρ because it is defined by the arrival distribution, which in this case is a Weibull distribution.

- the sojourn time on W.S will increase alongside the increase of ρ, although many individuals will continue being 0 seconds on it regardless of ρ value.

- the sojourn time on queue will also increase for more individuals, although, many individuals will continue being 0 seconds on it.
- the expected individuals waiting on queue increases considerably alongside ρ increase.
- the expected individuals on W.S increases considerably.
- the service time follows the expected density function for the given loading factor.
- the number of clients served at the same time does not change. Only one can be served because the system only has one server. What changes is the probability distribution with which the service is working.