Discrete event traffic simulation
Laboratory Report 2

RTEL 2024/2025 **Class: 2MEEC_T01**

António Lopes, Pedro Duarte

1 Introduction

This report outlines the development of a Traffic Simulation model designed to replicate the behavior of a call-handling system through discrete events. It was created for the Telecommunications Networks course at FEUP, using Python as the programming language due to its simplicity and flexibility. The present report showcases the evolution of the work previously done in [1]. The main objective was to develop a packet-oriented waiting system, starting by developing a packet-loss system, and then introducing the waiting queue mechanism.

2 Methodology

As mentioned in [1], the developed traffic simulation model consists of a python class, *Simulation*, which has the following parameters, present in Table 1:

Table 1: Simulation class parameters.

λ	arrival rate (default: 200 calls/sec)
μ	service rate (default: 125 departure calls/sec)
max_resources	maximum number of system resources
queue_length	maximum queue length
max_delay	maximum acceptable delay (default: 0.5s)
T	simulation time period (default: 500s)

The class functions are used to implement an event-driven simulation, which generate call arrival and departure events, based on the simulation parameters λ and μ, respectively. This events are generated by an Exponential Distribution process.

The final system starts by generating an initial arrival event and process it with *call_arrival* function. This function is called every time an arrival event is processed and is responsible for generating the next arrival event and check the status of the simulation, based on the number of active calls. If the number of active calls is less than *max_resources*, the system generates the next departure event. Otherwise it will process the event based on *queue_length* (L) parameter, as follows:

- $L = 0$: The system will block the call (Erlang-B)
- $L = -1$: System assumes that L is infinite and appends the call to the waiting queue (Erlang-C)
- $L > 0$: System appends the call to the waiting queue if its length is less than L. Otherwise it will block the call.

For processing the departure events, the function *call_departure* is used. This function will free the calls from the system and check if there are any calls in the waiting queue. If this is true, the function generates the departure event associated with the first call in the queue. The simulation runs until the current time of the simulation exceeds the value of T.

In order to validate the designed system, numerous simulations were run with different input parameters and the results were compared with the theoretical values that can be obtained in [2] and [3]. Assuming A as the packet delay, the desired outputs of the simulations are:

a) B - the estimator of the blocking probability, for L=0.

b) P_d - the estimator of the probability that a packet is delayed, $P(A > 0)$.

c) A_m - the estimator of the average delay of all packets.

d) The histogram of the delay of the packets that suffer delay, $(A > 0)$.

e) P_x - the estimator of the probability that a packet is delayed more than A_x, $P(A > A_x)$, being A_x defined by *max_ delay* parameter.

f) L_1 - queue length which leads to a packet loss probability of 1%.

The source code of this simulation can be found in GitHub repository [4].

The next sections will present the results of these simulations and the main conclusions of this project.

3 Simulation Results

3.1 Loss system: Erlang-B distribution

The first simulations were run in order to obtain the estimator B, with $L = 0$. For this purpose, four scenarios where tested, using the default parameters of the system and ranging the *maximum_ resources* from 1 to 4. For every scenario, 30 simulations were run and the respective confidence intervals (95%) were calculated. Table 2 presents the results obtained for estimator B in each scenario.

Table 2: Estimator B simulation results.

Scenario	**Max Resources**	**Simulation results**	**Expected value**
1.A	1	61.52 ± 00.05 (%)	61.54 (%)
1.B	2	33.00 ± 00.06 (%)	32.99 (%)
1.C	3	14.92 ± 00.05 (%)	14.96 (%)
1.D	4	5.63 ± 00.04 (%)	5.65 (%)

3.2 Waiting system with infinite length queue: Erlang-C distribution

The second part of the work was focused in obtaining the estimators P_d, A_m, P_x and the histogram of the delay of the packets that suffer delay. The simulation parameters were changed as following: $\lambda = 3$, $\mu = 0.5$ and $T = 33333$, preserving the number of samples generated. Four different scenarios were tested, keeping $L = -1$ and ranging *maximum_ resources* value from $\lambda/\mu = 6$ to 9. As done before, each scenario was run 30 times in order to get their 95% confidence intervals. Considering *Service Level* $= 1 - P_x$, for $A_x = 0.5\ s$, the results of the estimators for P_d, A_m and *Service Level* are present in Tables 3, 4 and 5, respectively.

From the output tables 2, 3, 4 and 5, can be concluded that the simulation model produces expected results, being scenario 2.A the only one that deviates from the theoretical predictions. Even so, this behavior was also expected, as the probability formulas produce the same exact values if the number of servers in the system is less than λ/μ, meaning that the system is overloaded. This comportment was identified by testing multiple input combinations in [3].

Lastly, an example histogram was generated for each scenario, being the outputs presented in Figure 1. Again, it is possible to notice the difference in scenario 2.A, presenting an irregular

distribution when compared with the remaining histograms which tend to show an exponential function form that gets steeper as the estimator of A_m decreases.
The output results are available in *sim_ data.txt* file in [4].

Table 3: Estimator P_d simulation results.

Scenario	**Max Resources**	**Simulation results**	**Expected value**
2.A	6	98.98 ± 0.25 (%)	100 (%)
2.B	7	61.28 ± 0.37 (%)	61.38 (%)
2.C	8	35.67 ± 0.24 (%)	35.70 (%)
2.D	9	19.77 ± 0.18 (%)	19.60 (%)

Table 4: Estimator A_m simulation results.

Scenario	**Max Resources**	**Simulation results**	**Expected value**
2.A	6	88.488 ± 18.71 (s)	N/A
2.B	7	1.233 ± 0.027 (s)	1.228 (s)
2.C	8	0.360 ± 0.005 (s)	0.357 (s)
2.D	9	0.132 ± 0.002 (s)	0.131 (s)

Table 5: Estimator of *Service Level* (1-P_x) for $A_x = 0.5\ s$ simulation results.

Scenario	**Max Resources**	**Simulation results**	**Expected value**
2.A	6	2.03 ± 0.32 (%)	0.00 (%)
2.B	7	52.25 ± 0.43 (%)	52.19 (%)
2.C	8	78.31 ± 0.20 (%)	78.35 (%)
2.D	9	90.61 ± 0.14 (%)	90.74 (%)

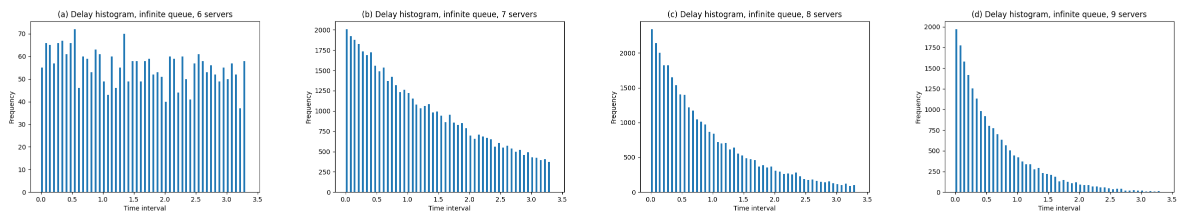

Figure 1: Histograms of delayed packets, (A $>$ 0), for scenarios 2.A (a), 2.B (b), 2.C (c) and 2.D (d).

3.3 General case – waiting system with finite length queue

With the previous steps validated, the final objective was to find the queue length L_1 which leads to a packet loss probability of 1%. Completing this final step assures that all the expected features of the system were well developed and integrated. In order to discover L_1, the simulation was setup with the initial parameters of scenario 2.B ($max\_resources = 7$). Making use of [3], a initial $L = 30$ was chosen, decrementing this value in each scenario until a loss probability of 1% was achieved. For each queue length, 30 simulations were run. In Figure 2, a graph with the blocking probabilities for each queue length is presented. The dots in red ($L = 14$, $L = 15$) show the queue lengths for which the blocking probability is closer to 1%, being $L_1 = 14$, with

$B = 0.0109 \pm 0.0003$. To validate the discovered length, the estimator of P_d was compared with the expected results from [3], which are shown in Table 6.
The full output results are available in *block_prob_finder.txt* file in [4].

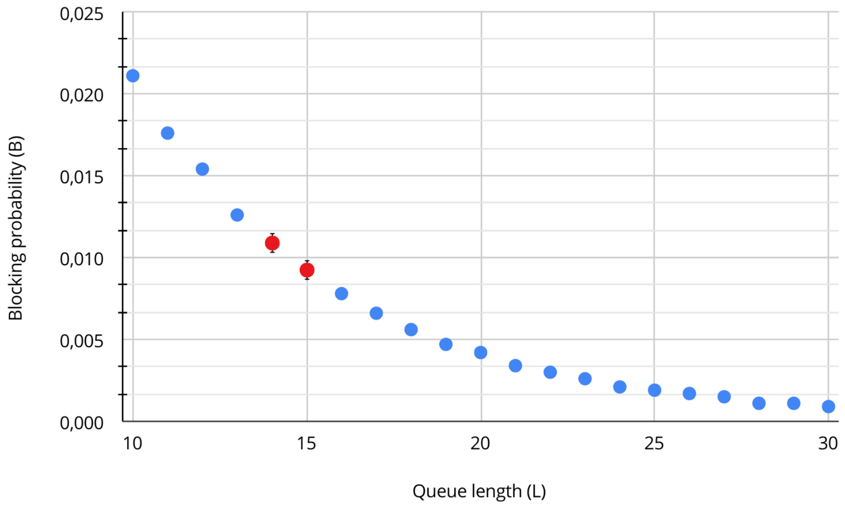

Figure 2: Blocking probability chart for different queue lengths (scenario 2.B).

Table 6: Estimator P_d results, for finding L_1.

Scenario	Queue Length	Simulation results	Expected value
3.A	13	58.42 ± 0.30 (%)	58.44 (%)
3.B	14	59.09 ± 0.29 (%)	58.88 (%)
3.C	15	59.27 ± 0.21 (%)	59.26 (%)
3.D	16	59.55 ± 0.25 (%)	59.58 (%)

4 Conclusions

In conclusion, all simulation outputs aligned well with the theoretical values, confirming the accuracy of the model. The results demonstrate that the implemented system operates as expected, meaning that the final outcome is a fully functional traffic simulator for a waiting system with a finite queue, capable of accurately representing the system's performance under various scenarios. This success provides a solid foundation for future enhancements and further developments on the system's complexity.

References

[1] P. Duarte, A. Lopes. Discrete event traffic simulation - Laboratory Report 1. October 2024

[2] Java Call Center Optimization Library. Erlang B Formula Calculator: `https://erlang.chwyean.com/erlang/erlangB.html` . Last accessed: October 2024.

[3] Java Call Center Optimization Library. Erlang C Formula Calculator: `https://erlang.chwyean.com/erlang/erlangC.html` . Last accessed: October 2024.

[4] P. Duarte, A. Lopes. GitHub Repository: `https://github.com/Pektro/RTEL` . Last accessed: October 2024.