Case Study of $M/M/1$ Queues with Batch Arrivals

Luiz Gustavo Barros Guedes, Masoud khazaee and Aline Aires Teixeira

***Abstract*—This paper describes the execution of a simulation for an $M/M/1$ queue with batch arrivals. The model is specifically developed to assess the performance of a system when subjected to different levels of utilization, considering infinite and finite buffers with various sizes. The arrival and service times are exponentially distributed, while the batch sizes are decided by a Bernoulli process. The simulation produces inter-arrival and inter-service durations, mimicking the behavior of an $M/M/1$ queue where packets come in batches. The server's state and queue length are monitored to manage the occurrence of arrival and departure events, even when they happen simultaneously. The model modifies the pace at which arrivals occur by taking into account the intended level of utilization and the probability of occurrence associated with different batch sizes. The system's performance is evaluated based on the average number of packets in the system, the average duration packets spend in the system, and the likelihood of blocking incoming batches due to buffer overflow, in case of finite buffers.**

***Index Terms*—$M/M/1$ queues, batch arrivals, servers.**

I. Introduction

Queue systems can be observed in various scenarios such as bank queues, restaurant queues, supermarket lines, queues to enter events, among other cases. Generally, the reason for the existence of queues is simple: the demand for services is greater than the number of establishments or servers available to prevent the formation of queues.

In the context of telecommunications, pioneering studies were presented by Erlang in 1909 when he observed traffic congestion in telephone lines [1]. In his essays, Erlang observed that the stages of the telephone system could be modeled according to some known probability distribution curves. The system inputs followed a Poisson distribution with exponential waiting times for single or multiple servers. Erlang's work spread to various countries, leading to new contributions. In 1927, Molina presented his studies on the application of probability theory to address trunking problems in telephone lines [2].

With the recent advances in telecommunications networks, the modeling and analysis of queue systems become more imminent. Here, it is highlighted the scenario which considers batches of packets arriving in a queue system, especially in conditions where data is broadcast in bursts or bundled together in bigger units to improve efficiency.

This article provides a detailed simulation analysis of an $M/M/1$ queue, considering both infinite and finite buffer cases, specifically examining the patterns of batch arrivals and service procedures in a single-server setting. The system is designed to process groups or batches of data packets that arrive in accordance with a Poisson process. The time it takes to process each packet follows an exponential distribution. An analysis is conducted on the queuing behavior as follows: 1) Different utilization factors, ρ, which will be defined later; 2) Different buffer capacities, including both infinite, $M/M/1$, and finite buffers, $M/M/1/N$, where N is the buffer size. The performance metrics assessed are: the average time spent by packets in the whole system, considering the time spent in the queue and to be served; the blocking probability, associated with the batches that were disregarded because they could not fit into the system; and the average number of packets in the system.

The simulation includes the arrival of batches, differentiating between batches consisting of one or two packets, and assesses the effects on system performance of server status (occupied or available) and buffer overflow situations. This study offers useful insights into the performance and efficiency of queue management, providing data that can be used to optimize resource allocation in communication networks. The findings emphasize the importance of utilization factors and buffer size in ensuring system stability and reducing packet loss.

The remainder of the paper is structured as follows: Section II discourses about $M/M/1$ modeling, while Section III addresses a numerical analysis associated to the simulation process in order to assess the system's performance. Section IV concludes the work.

II. Modeling

A conventional $M/M/1$ queue comprises a queueing system whose arrival and service processes are Markovian, indicated by the letter M. In other words, the probability distributions used to model the arrivals and services are Poisson. The remaining number 1 in the simplified Kendall notation relates to the presence of a single server in the system. An $M/M/1$ queue, in this case, assumes the consideration of an infinite capacity buffer.

If one wishes to consider a finite buffer, limited to a amount of N elements in the system, the notation extends to $M/M/1/N$. The main parameters of an $M/M/1/N$ system with batch arrivals are:

- Batch arrival rate, λ_b: It represents the rate at which batches of packets enter the system, and it is commonly modeled using a Poisson process. The time intervals between arrivals are generated using an exponential distribution. The same characteristics can be used to describe the packet arrival rate, λ;
- Service rate, μ: It refers to the rate at which packets are handled by the server. It is also depicted as an exponential distribution for the time intervals between services;
- Batch size: The system differentiates between batches including a single packet and batches comprising two

packets. The batch size is determined by utilizing a Bernoulli distribution, where the probabilities p_1 and p_2 represent the likelihoods of selecting one and two packet batches, respectively;

- Utilization factor, ρ: It is given by the ratio of the packet arrival rate to the packet service rate, considering the possibilities related to the batch size, i.e., $\rho = \lambda/\mu$, which will be discussed properly later. This parameter can be varied to assess its impact on performance metrics within the system;
- Buffer size, N: It is the upper limit of a queueing system capacity, which can be set to a specified numerical value or an unbounded amount. The buffer size determines the upper limit on the number of packets that can be held by the system before packet or batch blocking takes place;
- Server state: The system tracks the server's condition, assessing whether it is currently in use (busy) or free (idle), which subsequently affects how incoming packets are managed;
- Blocking probability: It is the probability that a batch of packets can not be processed due to insufficient space in the buffer. Here, the probability is related to the batches blocked;
- Mean time in system: It is the average duration of time that a packet stays in the system, from when it arrives until it leaves;
- Mean packets in system: It is the mathematical average of the total number of packets present in the system, encompassing the packets in the queue and the one being processed by the server;
- Simulation time: It is the total amount of time of the simulation, ensuring that enough events are processed to produce statistically meaningful results.

These parameters lead to a arrival batch rate corresponding to

$$\lambda = \lambda_b \cdot \mathbb{E}[B]$$
$$\lambda_b = \frac{\lambda}{\mathbb{E}[B]}, \tag{1}$$

where $E[B] = \sum_k k \cdot p_k$ is the expected batch size value, denoting B as the random variable which models the batch size. The number of distinct batches containing different quantities of packets in their composition is given by k and p_k is the probability of occurrence regarded to the k-th batch.

A. Case study: two batch sizes with the same probability of occurrence

Consider an $M/M/1$ queue with batch arrivals (variable number of packets arrive at the server). The batches have different probabilities of arrival. For example, we can have a network with only two types of batches: a batch of size 1 packet, as in the $M/M/1$ case, and a batch of size 2 packets. In this case, the arrival probabilities of batches of sizes 1 and 2 can be equal. The arrival rate of the batches is λ. The service time has an average of $1/\mu$ (the service is performed for each packet, not for each batch).

For a system with two possible batch sizes, 1 and 2, with probabilities of occurrence p_1 and p_2, it follows that

$$\mathbb{E}[B] = 1 \cdot p_1 + 2 \cdot p_2. \tag{2}$$

Substituindo (2) em (1), tem-se

$$\lambda_b = \frac{\lambda}{1 \cdot p_1 + 2 \cdot p_2} = \frac{\rho\mu}{1 \cdot p_1 + 2 \cdot p_2}. \tag{3}$$

Figure 1 illustrates an example of a batch arriving in the referred system being build by two packets. In this case, $N = 13$ and there is only one server, as expected.

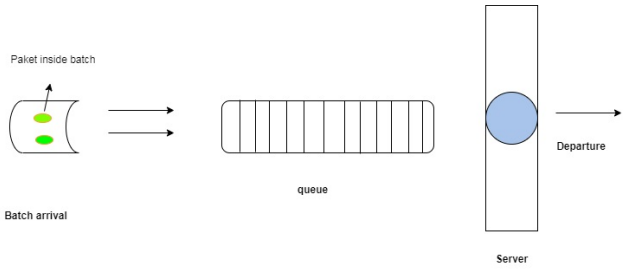

Fig. 1. $M/M/1$ queue with batch arrivals

B. Theoretical reference: conventional $M/M/1$

The $M/M/1$ queue theoretical model relies on the utilization of the following widely recognized formulas. Its noteworthy to mention that this theoretical reference considers infinite buffer, $p_1 = 1$ and $p_2 = 0$. The mean packets in system, $\mathbb{E}[q]$, is given by [3]

$$\mathbb{E}[q] = \frac{\rho}{1-\rho}. \tag{4}$$

The mean time in system, $\mathbb{E}[T_q]$, is given by

$$\mathbb{E}[T_q] = \frac{E[q]}{\lambda} = \frac{1}{\mu - \lambda}. \tag{5}$$

III. Numerical Analysis

In order to generate a flowchart for the provided code, we will delineate the sequential actions and conditional choices made throughout the code's execution. The flowchart will document the primary procedures, points of decision, and resulting consequences.

The algorithm is considered as follows:

1) **Start**
2) **Generate Arrivals and Services**
   - Generate inter-arrival times (*chegadas*) using an exponential distribution.
   - Generate inter-service times (*partidas*) using an exponential distribution.
   - Generate random variables for batch size determination (*b*) using a uniform distribution.
3) **Initialize Parameters**
   - Set the initial time *t* and *t_last* to 0.
   - Set the number of packets in the queue *lq* and the server state *ls* to 0.

- Initialize counters: *k1* for arrivals, *k2* for departures, *k* for completed departures, *blocked_batches* for blocked batches, and *L* for the total number of packets.
- Calculate *lambda1* for the arrival rate.
- Initialize arrival time *ta* and departure time *td*.

4) **Simulation Loop (until *t* reaches *sim_time*)**
5) **Calculate Time in System**
   - Update *L* based on the time difference and the number of packets in the system.
6) **Check for Arrival Event (*ta* < *td*)**
   - Determine batch size.
   - Increment arrival counter.
   - Generate next arrival time.
   - **If Server Busy**
     - **Check Buffer Overflow**
       - **If Not Overflow**
         - Increment queue length.
         - Store arrival times.
       - **If Overflow**
         - Increment blocked batches counter.
   - **If Server Free**
     - **If Batch Size = 1**
       - Occupy server.
       - Increment departure counter.
       - Generate next departure time.
       - Store arrival time.
     - **If Batch Size = 2**
       - **Check Vacant Position**
         - **If Vacant Position**
         - Occupy server.
         - Increment queue length.
         - Increment departure counter.
         - Generate next departure time.
         - Store arrival times.
         - **If No Vacant Position**
         - Increment blocked batches counter.
7) **Check for Simultaneous Arrival and Departure Event (*ta* == *td*)**
   - Determine batch size.
   - Increment arrival counter.
   - Generate next arrival time.
   - **Handle Departure**
     - Store departure time.
     - Calculate time in system.
     - Increment departure counter.
     - **If Packets in Queue**
       - Decrement queue length.
       - Occupy server.
       - Increment departure counter.
       - Generate next departure time.
     - **If No Packets in Queue**
       - Free the server.
       - Set next departure to infinity.
8) **Handle Arrival**
   - **If Batch Size = 1**
     - **If Server Busy**
       - **Check Buffer Overflow**
         - **If Not Overflow**
         - Increment queue length.
         - Store arrival time.
         - **If Overflow**
         - Increment blocked batches counter.
     - **If Server Free**
       - Occupy server.
       - Increment departure counter.
       - Generate next departure time.
       - Store arrival time.
   - **If Batch Size = 2**
     - **If Server Busy**
       - **Check Buffer Overflow**
         - **If Not Overflow**
         - Increment queue length.
         - Store arrival times.
         - **If Overflow**
         - Increment blocked batches counter.
     - **If Server Free**
       - Occupy server.
       - Increment queue length.
       - Increment departure counter.
       - Generate next departure time.
       - Store arrival times.
9) **Check for Departure Event (*ta* > *td*)**
   - Store departure time.
   - Calculate time in system.
   - Increment departure counter.
   - **If Packets in Queue**
     - Decrement queue length.
     - Occupy server.
     - Increment departure counter.
     - Generate next departure time.
   - **If No Packets in Queue**
     - Free the server.
     - Set next departure to infinity.
10) **Update Next Event Time**
    - Set *t* to the minimum of *ta* and *td*.
11) **Calculate Results**
    - Calculate the mean number of packets in the system.
    - Calculate the mean time in the system.
    - Calculate the blocking probability.
12) **Output Results**
    - Return the mean time in the system, blocking probability, and mean number of packets in the system.

This algorithm has been separated into two flowcharts: arrival event, as shown in Figure 2; departure event, as illustrates in Figure 3.

The simulation parameters are shown in Table I. These values has been chosen to be adequate to real world experiments.

TABLE I
SIMULATION PARAMETERS.

μ, packets/s	ρ	Simulation time, s
12	[0.2, 0.4, 0.6, 0.8, 0.9, 1.0]	5000

Python was used to evaluate the performance. We present a comparison between the theoretical and simulation outcomes of an $M/M/1$ queuing system. The objective is to verify the theoretical model by means of simulation and ensure a high degree of alignment between the obtained findings.

The simulation involves generating inter-arrival and service times that follow exponential distributions. The key performance metrics gathered include the average time spent in the system, the average number of packets in the system, and the blocking probability.

The initial results can be seen in Table II. At first, we have validated our code considering an $M/M/1$ queue. To do this, we set buffer size as infinite, $p_1 = 1$ and $p_2 = 0$, i.e., there is no chance of occurring arrivals of batches containing two packets. Only one packet can arrive. As expected, when the utilization factor increases, the mean packet time in system and the mean amount of packets in system increase either. Theoretical values reached by the application of (4) and (5) match to simulation results, as depicted in the last column of Table II. Besides of these, the consideration of infinite buffer leads to a null blocking probability. When the utilization factor is equal to 1, the system enters in an unstable and critical region of operation, revelead by the infinite approaches of the performance metrics, since the buffer is infinite.

TABLE II
PERFORMANCE COMPARISON USING DIFFERENT ρ VALUES IN INFINITE CASE, CONSIDERING PACKET ARRIVALS.

ρ	$\mathbb{E}[T_q]_s$, s	P_b	$\mathbb{E}[q]_s$, packets	$\mathbb{E}[T_q]_t$, $\mathbb{E}[q]_t$
0.2	0.1054	0.0000	0.2565	0.1042, 0.25
0.4	0.1407	0.0000	0.6809	0.1389, 0.6667
0.6	0.2049	0.0000	1.4643	0.2083, 1.5
0.8	0.3975	0.0000	3.7570	0.4167, 4
1	17.2847	0.0000	206.6625	∞, ∞

The results shown in Table III were retrieved only by simulation, since the theoretical expressions were not considered in this work due to the high mathematical complexity to reach them. Here, we consider $M/M/1$ queue with batch arrivals and infinite buffer size. The values yielded match with the ones seen in the Table II, considering packet arrivals. There are two reasons to explain this behavior. The first one is because the unitary element present in both systems is packet, so the performance metric are related to packets either. Another reason is because we have adjust the same λ to both systems. In other words, the effective packet arrival rate is the same in both systems.

TABLE III
PERFORMANCE COMPARISON USING DIFFERENT ρ VALUES IN INFINITE CASE, CONSIDERING BATCH ARRIVALS.

ρ	$\mathbb{E}[T_q]_s$, s	P_b	$\mathbb{E}[q]_s$, packets
0.2	0.1033	0.0000	0.2476
0.4	0.1410	0.0000	0.6818
0.6	0.2063	0.0000	1.4818
0.8	0.4272	0.0000	4.1141
1	12.9260	0.0000	155.2688

In Table IV, we present the results regarding to $M/M/1/5$ with batch arrivals. The buffer size is limited by 5 positions. As ρ increases, both $\mathbb{E}[T_q]$ and $\mathbb{E}[q]$ increase. Now, it is possible to see an non null ascending return fashion emergence of blocking probabilities as ρ increases. This happens because as buffer sizer is limited, if we keep a fixed packet service rate and ρ increases, it means that λ_b increases and the systems starts to be more populated, turning possible the block of batches.

TABLE IV
PERFORMANCE COMPARISON USING DIFFERENT ρ VALUES IN FINITE CASE ($N = 5$), CONSIDERING BATCH ARRIVALS.

ρ	$\mathbb{E}[T_q]_s$, s	P_b	$\mathbb{E}[q]_s$, packets
0.2	0.1343	0.0035	0.3212
0.4	0.1649	0.0273	0.7685
0.6	0.1926	0.0687	1.2727
0.8	0.2169	0.1281	1.7622
1	0.2441	0.2004	2.2482

In Table V, we have increases the buffer size to $N = 10$. Its behavior is similar to the one presented by the Table IV scenario, with the difference that as the buffer size increases (tends to infinite), the blocking probability values are smaller thans the ones reached by low buffer size values. This is expected, as we seen in the infinite buffer case. Something very interesting to notice is that when the system with finite buffer enters the critical operation region ($\rho = 1$), the mean amount of packets in the system becomes the half of the buffer size. This is justified by understanding that when $\rho = 1$, i.e., $\lambda = \mu$, the probability of having any integer value of packets from 0 to N is the same. We face a uniform distribution, with the mean equal to $N/2$.

TABLE V
PERFORMANCE COMPARISON USING DIFFERENT ρ VALUES IN FINITE CASE ($N = 10$), CONSIDERING BATCH ARRIVALS.

ρ	$\mathbb{E}[T_q]_s$, s	P_b	$\mathbb{E}[q]_s$, packets
0.2	0.1371	0.0000	0.3199
0.4	0.1777	0.0007	0.8546
0.6	0.2533	0.0102	1.8213
0.8	0.3426	0.0415	3.1384
1	0.4531	0.1073	4.7663

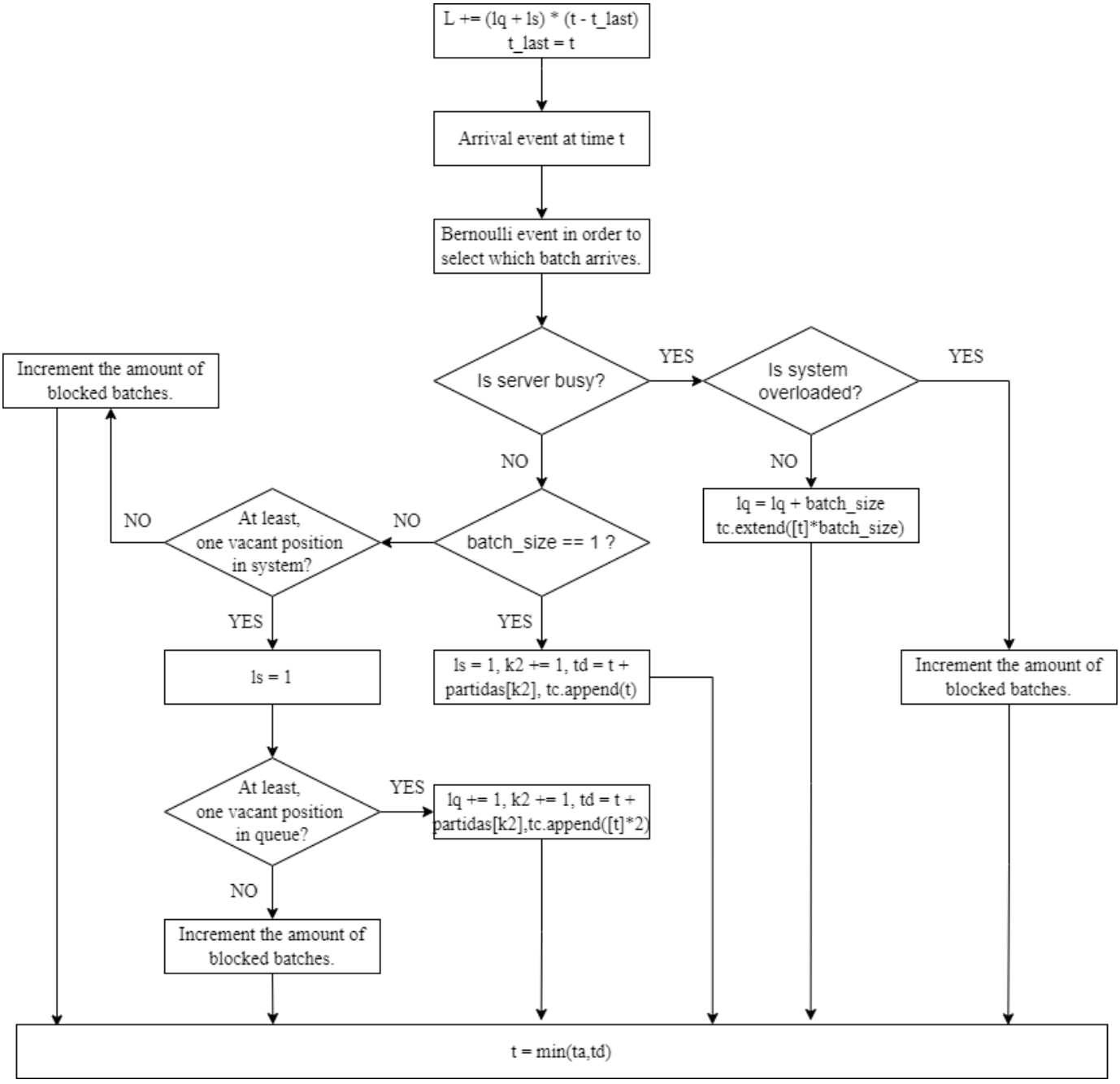

Fig. 2. $M/M/1$ queue flowchart with batch arrivals: Arrival event.

IV. Conclusions

This simulation investigates the influence of various utilization factors (ranging from 0.2 to 1.0) and buffer capacities on the performance of a $M/M/1$ queue with batch arrivals. The results offer valuable insights into the impact of adjusting these parameters on the efficiency and dependability of the system, especially in situations where the buffer size is limited compared to being unlimited. The comparison of the theoretical and simulation findings for the $M/M/1$ queuing system shows a significant agreement, confirming the precision of the theoretical model. Both methodologies yield consistent findings for the average time spent in the system and the average number of packets in the system. The assumptions of the theoretical conventional $M/M/1$ model and the conditions of the simulation are closely aligned, guaranteeing accurate forecasts for the performance of the queueing system.

The performance achieved by the $M/M/1$ queue, considering batch arrivals, is the same as that of the conventional system, which analyzes packet arrivals. Both the average time and the number of elements in the system consider packets, not batches, and this is the reason. Additionally, the effective

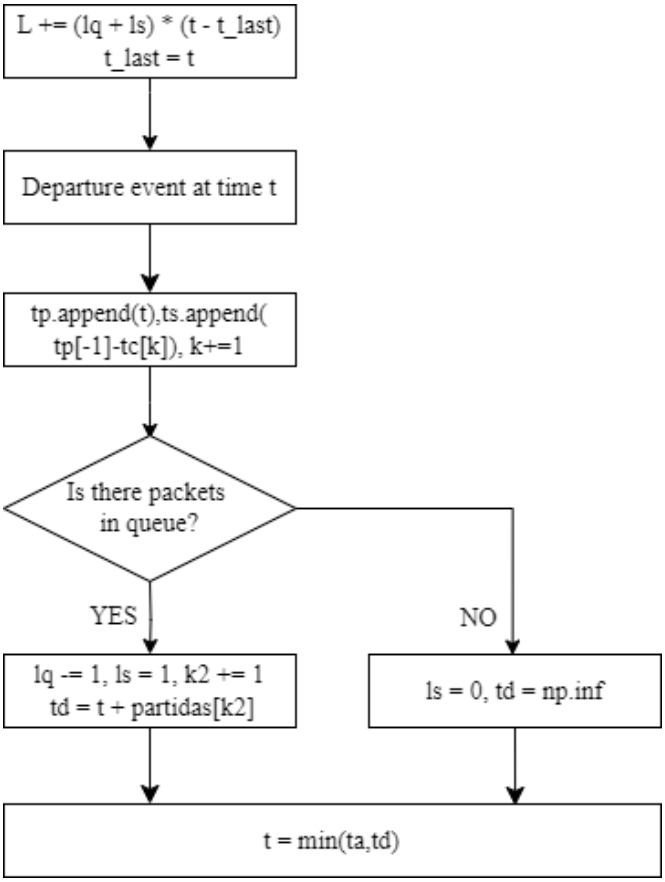

Fig. 3. $M/M/1$ queue flowchart with batch arrivals: Departure event.

arrival rates of packets are the same in both simulations.

Low utilization occurs when the service rate is significantly higher than the arrival rate. In this situation, the system can operate efficiently without rejecting batches or packets. However, if we reduce the buffer size, this condition may become constrained, increasing the likelihood of blocking even in low utilization scenarios.

As utilization increases, it is assumed that the arrival rate has increased while the service rate remains fixed. Therefore, a limited buffer will have a higher chance of rejecting batches, increasing the probability of blocking.

It was observed that, in the case of a finite buffer, when the utilization is close to 1, the number of packets in the system becomes approximately half of the total buffer capacity. This is explained by the equal probability of having any occupation in the system, from 0 to N. So, the mean value of this uniform distribution is $N/2$.

As future work, it is possible to verify theoretical expressions for performance analysis considering batch arrivals.

REFERENCES

[1] A. K. Erlang, "The theory of probabilities and telephone conversations," *Nyt. Tidsskr. Mat. Ser. B*, vol. 20, pp. 33–39, 1909.

[2] E. C. Molina, "Application of the theory of probability to telephone trunking problems," *The Bell System Technical Journal*, vol. 6, no. 3, pp. 461–494, 1927.

[3] N. S. Nafi, M. K. Hasan, and A. H. Abdallah, "Traffic flow model for vehicular network," in *2012 International Conference on Computer and Communication Engineering (ICCCE)*, 2012, pp. 738–743.