SHORTEST PROCESSING TIME FIRST OMNET++ PROJECT

TRAFFIC THEORY 2022/23

Draghini Chiara - 10823602
Cagliani Luigi - 10562871

Introduction

Queueing systems are fundamental models used to understand and analyze a wide range of real-world scenarios, from network traffic management to customer service operations. Theys are critical for optimizing resource allocation and ensuring efficient utilization in various domains. In this report, we present the implementation on OMNET++ of an M/G/1 queuing system with a Shortest Processing Time First (SPTF) discipline.

The M/G/1 queueing system is a classic model characterized by three key components:

- M: exponential service times
- G: general distribution for inter-arrival times
- 1: single server in the system

In our specific case, the inter-arrival times follow a “uniform(0, L)” distribution, adding an element of randomicity to the system that reflects the unpredictability often encountered in real-world scenarios. The SPTF discipline introduces a scheduling rule where jobs are selected for service based on their processing times: it aims to minimize the time jobs spend in the queue by prioritizing those with shorter processing times.

Objectives

The primary objective of this project is to implement the M/G/1 queuing system with the SPTF discipline in the OMNeT++ simulation environment, by doing so, we aim to achieve the following goals:

- Simulate the behavior of the queuing system under the defined conditions.
- Collect statistics on various performance metrics, including average general and conditional queuing times, average response time, server utilization factor and queue length over time.
- Conduct a comparative analysis between the experimental results obtained through simulation and the theoretical values derived from queuing theory formulas.

Methodology

Upon thorough consultation of the OMNET++ documentation, we identified the optimal approach for achieving our objectives. This involved implementing a queue sorting mechanism using a

specific command, accompanied by the creation of a dedicated comparison function. This function effectively evaluates the processing time of messages within the queue, enabling the prioritization of messages with the shortest processing time at the forefront. Detailed implementation of this logic can be located within the “Queue.cc” file.

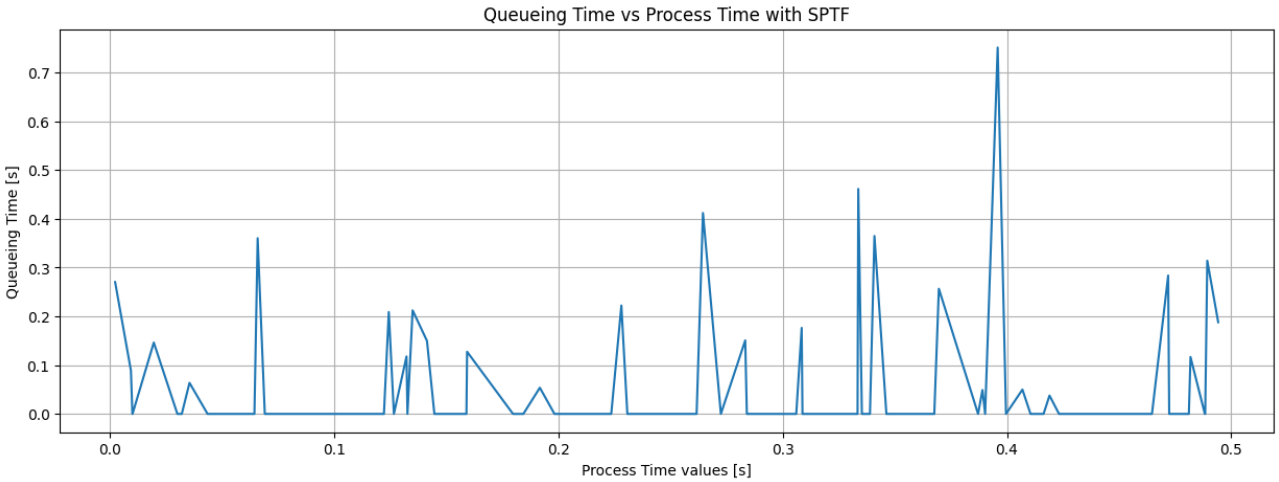
However, to accomplish this, we encountered a limitation with the existing “cMessage” type: consequently, we crafted a new message type with a dedicated processing time field. This addition allowed for the association of a value with each message, thereby enabling the sorting process. This structure is located within the “NewFieldsMessage.msg” file. After its creation, it became apparent that for seamless integration and to avoid memory allocation errors, it was best to separate the headers “.h” from the “.cc” files.

Upon achieving a successful build, we proceeded to execute the simulation to its completion, thus obtaining the results.

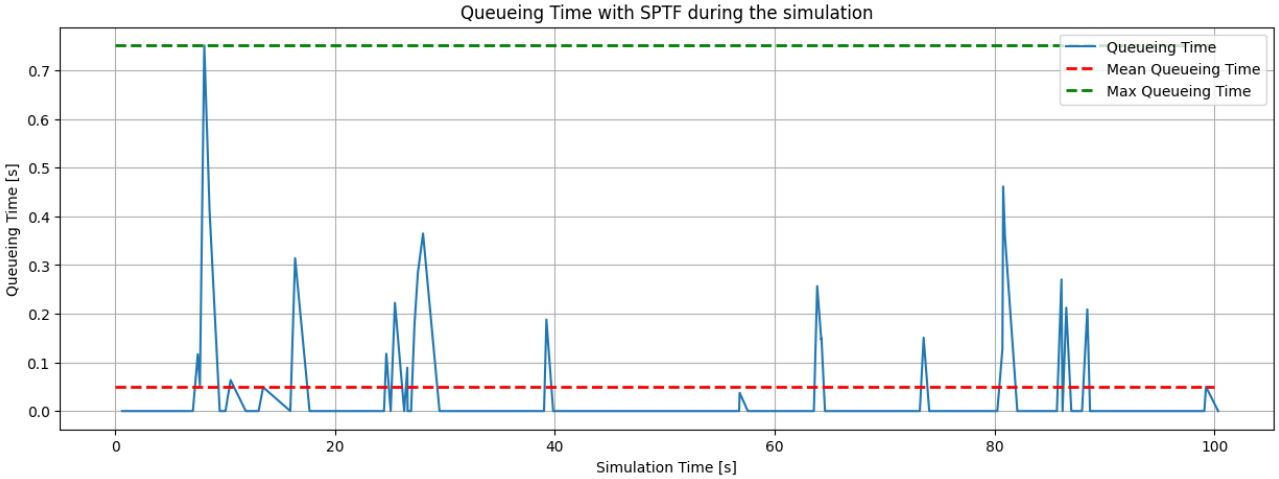
Parameters and Final Results

The results of the project are included as “.csv” files in the ”result” folder of the project. We were able to produce the plots below leveraging the data with the python code “TrafficTheoryPlots.ipynb”.

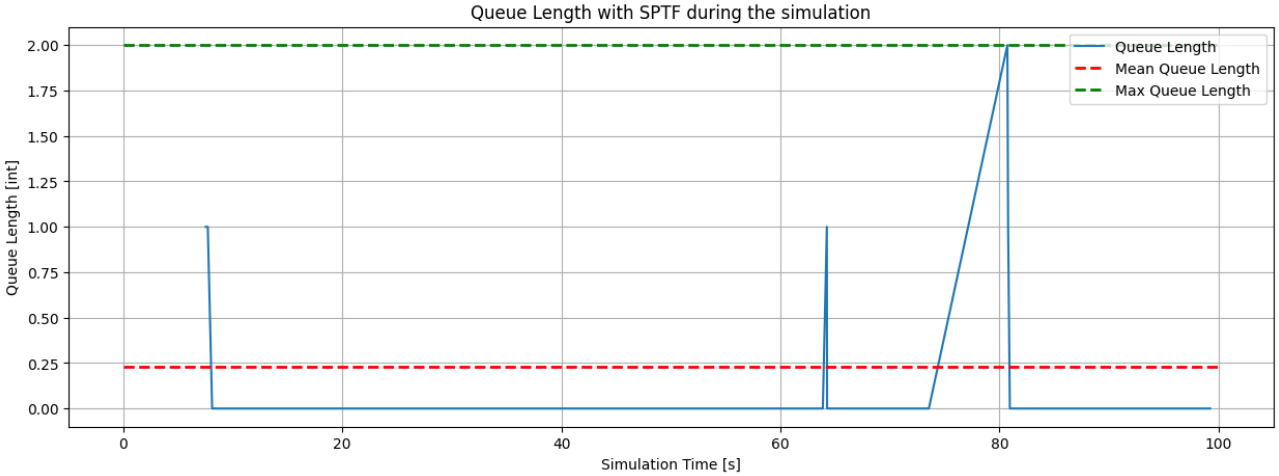
In this plot, we observe how the queueing time changed according to the processing time of the packet. When the queueing time is at zero, it is because the server was found idle: meaning the job was processed immediately upon arrival. It is noticeable that there are queueing time peaks where the processing time was higher.



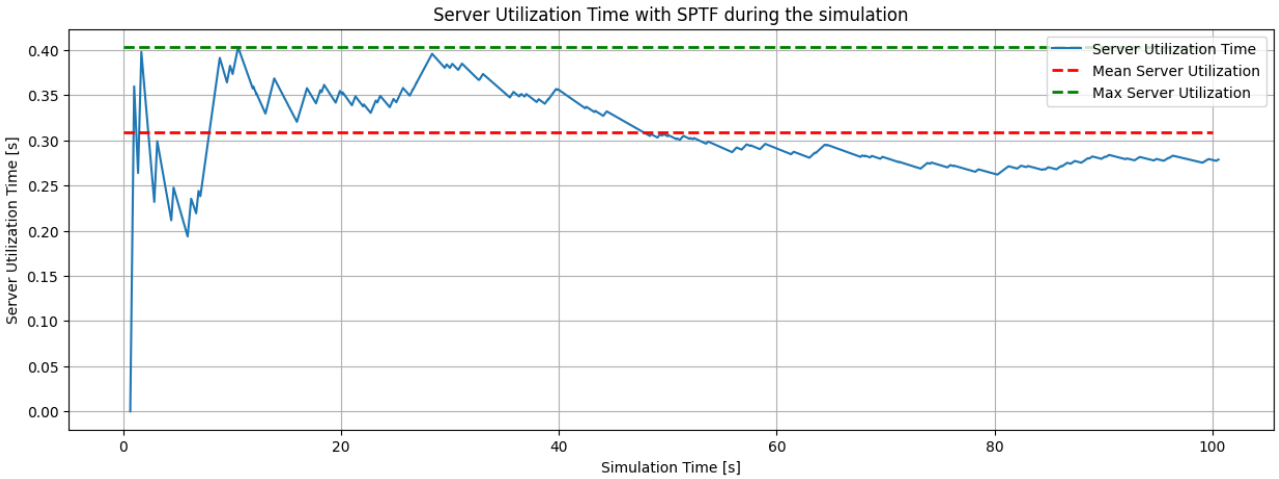
In the next one, we have the overall queueing time over the simulation.



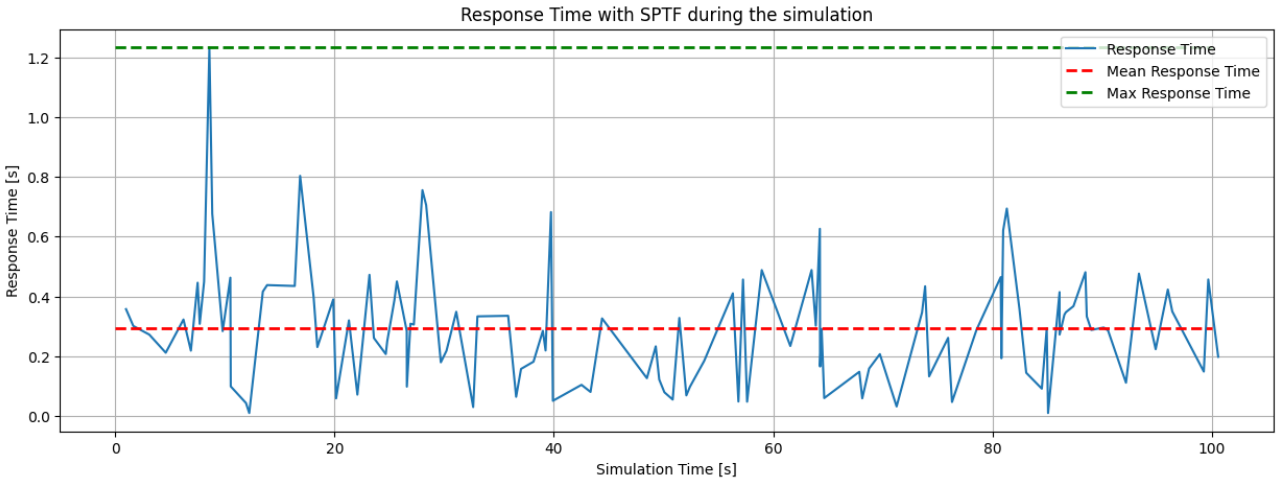
Here, we can see the evolution of the queueing length. We only ever reach 2 waiting messages in the queue: this could be because the “avgInterArrivalTime” value wasn’t small enough to create longer queues and the server was working faster.



The graph below shows the activity time of the server.



The last plot is the response time over the simulation.



The simulation had the following parameters:

Arrival Time: $1 / \lambda = 0.8$ [s]

Average Service Rate: $1/\mu = 0.2436$ [s^{-1}]

L = 0.5 for Service Rate = uniform(0,L)

Number of customers: N = 115.

	THEORETICAL	**OMNET++**
AVG UTILIZATION FACTOR	$\rho = \lambda E[S] = 0.3045$	0.31071231656334
AVG RESPONSE TIME	$W_{SPTF} = Wq + E[S] = \lambda E[S^2] + L/2 = 0.30934$	0.29263301012997
AVG QUEUING TIME	$W_{q,SPTF} = \lambda E[S^2]/ [2(1- \rho) S^2] = 0.076672$	0.048974026579496

Referring to the above table, we want to note that since the arranging the queue happens prior to the message handling process and given that the sequencing of packets was *conditioned* on the process time parameter: in our scenario, we interpreted the queueing time and conditional waiting time in OMNET++ as the same.

Overall, the theoretical values are coherent with the simulation values as we expected. However, they do exhibit minor discrepancies: we wonder if these variations could potentially stem from the limited number of jobs generated, which were only 115, or simply because the simulation has its own delays within the within the OMNET++ environment. To these points, one could consider extending the experimentation and delving deeper into the research to explore additional possibilities.