Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

Nora Lien Røneid

Model-Based Estimation for Latency Guarantees

Master's thesis in Communication Technology and Digital Security
Supervisor: Yuming Jiang
June 2023

NTNU
Norwegian University of
Science and Technology

Nora Lien Røneid

Model-Based Estimation for Latency Guarantees

Master's thesis in Communication Technology and Digital Security
Supervisor: Yuming Jiang
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

NTNU
Norwegian University of
Science and Technology

Title: Model-Based Estimation for Latency Guarantees

Student: Nora Lien Røneid

Problem description:

According to ITU-T, one key requirement for communication networks in the year 2030 is the guarantee and assurance of services, where timeliness of data delivery will be central to enabling time-sensitive applications such as extended reality (XR) and Tactile Internet (TI). The character of such time-sensitive applications is defined by latency guarantees, which, for example, could be delivered within 1ms with a probability of 99.99%. To ensure this, methods that can be used to estimate guarantees are needed. For instance, when a new service request is received, for admission control decision, the system needs a method to estimate if its latency guarantee requirement can be met and if the existing customers' such guarantees can still be ensured.

The objective of this project is to review analytical models for latency guarantee estimation and investigate their validity or applicability. Examples of such models include those from the classical queueing theory under steady states. Through analysis of data collected from simulations, the validity or applicability of the reviewed models will be investigated. Insights will also be generated. Different models will be considered. The considered setups include Poisson and periodic arrivals for arrival processes, exponentially distributed, deterministic and upper-bounded service times for service processes, and FIFO and priority for queueing.

The specific tasks include: (1) Review of analytical models that have been proposed for latency guarantee estimation. (2) Construct a simulation platform that enables the collection of related data for latency guarantee analysis, and conduct the analysis. (3) Compare model-based results with simulation results and discuss the findings.

Approved on: 2023-01-23

Supervisor: Professor Yuming Jiang, IIK (NTNU)

Abstract

Many new technologies within communication networks have strict requirements when it comes to the latency of the communication within the network. There are latency requirements that have to be guaranteed to make sure that the technologies can work as intended. This is important to be able to avoid situations such as self-driving cars ending up in a crash, having remote surgeries go wrong, and having video calls being experienced as choppy and slow. This Master's thesis investigates whether or not analytical models can be used for guaranteeing and predicting the latency within a network through comparisons to results generated by a simulation of the network. There has not been a lot of research in recent years on whether or not these models are still applicable.

There is a need to do a literature study to be able to investigate if the analytical models can be used for guaranteeing latency requirements or not. A simulation platform also has to be developed to be able to create results that the model's results can be compared to. Lastly, graphical tools are needed to be able to compare the results of the two.

Through this study, it was discovered that most of the results of the analytical models compare well to the results generated by the simulator. Some models deviate more from the simulated results than others and appear too optimistic about how much latency will be present in the system. The models can consist of both exact and approximated results, but in general this is not what decides how well the results are lining up. It is necessary to be aware of if the load level on the system is changing over time as this will affect the results, especially in the phase between two load levels.

Sammendrag

Mange nye teknologier innen kommuniserende nettverk setter strenge krav til forsinkelsen på kommunikasjonen. Det kreves garantier på at forsinkelseskravene kan holdes for at teknologiene skal kunne fungere slik de er tenkt. Dette er viktig for å kunne unngå situasjoner som at selvkjørende biler krasjer, at fjernkirurgi går galt og at videosamtaler oppleves hakkete og trege. Denne masteravhandlingen undersøker hvorvidt analytiske modeller kan benyttes for å garantere og forutse forsinkelsene i et nettverk gjennom sammenlikning med resultater generert ved hjelp av simuleringer av nettverket. I nyere tid er det gjort lite forskning på om disse modellene fremdeles er gjeldende.

For å kunne undersøke om de analytiske modellene kan brukes for å garantere forsinkelseskrav må først en litteraturstudie av modellene bli gjort. Videre må en simuleringsplattform utvikles for å kunne generere resultatene modellenes resultater skal sammenliknes med. Til slutt må resultatene sammenliknes ved hjelp av grafiske verktøy.

Gjennom denne studien er det funnet ut at de fleste resultater fra de analytiske modellene stemmer godt overens med resultatene generert av simulatoren. Noen modeller avviker mer fra de simulerte resultatene enn andre og er for optimistiske om hvor mye forsinkelser som vil oppstå i systemet. Modellene kan bestå av både eksakte og tilnærmede resultater, men det er generelt ikke dette som avgjør hvor godt resultatene stemmer overens. Det er nødvendig å være obs på om belastningsnivået på systemet endres over tid da dette vil påvirke resultatene, spesielt i overgangsfaser mellom ulike belastningsnivåer.

Preface

This thesis is written as the final contribution to the 5-year Master of Science in Communication Technology and Digital Security at the Norwegian University of Science and Technology (NTNU). The supervisor for the thesis has been Professor Yuming Jiang at the Department of Information Security and Communication Technology.

Acknowledgements

I would like to thank my supervisor Professor Yuming Jiang for all the help and guidance throughout the writing of this thesis and also the specialisation project last semester. I would also like to give a big thank you to my family and friends for all the help and support throughout my five years here in Trondheim.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1
Introduction

This chapter provides an introduction to the thesis. It first motivates the task by mentioning application fields where the results of this task could be useful. Then some related work is presented. The objectives and the research questions of the thesis are presented and lastly, an outline of the rest of this thesis is given.

1.1 Motivation

The communication networks of the future will contain many time-sensitive applications. There are many fields where these applications can be taken advantage of, some include industry, robotics and telepresence, virtual reality and healthcare. For the industry case, sensors and automation are very important. The sensor controller has to communicate with the sensors with low latency and reliable data transfers. As for robotics and telepresence, the same type of remote-controlled behaviour is a prerequisite for most scenarios. For virtual reality, having multiple users communicate while using the technology, low latency communication is required as well. Tele-surgery is an application within the healthcare field that also requires extremely low latency [1]. All these time-sensitive applications need a way to guarantee that their latency requirements can be kept.

For the 5th generation of mobile networks (5G), one of the three main identified usage scenarios is Ultra Reliable Low Latency Communication (URLLC). There are strict latency requirements in URLLC scenarios and it is among other things required that the end-to-end latency in the network is less than one millisecond [2]. Some specific use case scenarios in need of URLLC are industry automation, mission-critical applications and self-driving cars [3]. To have industry automation run smoothly, the time factor is significant. This means it is dependent on time-engineered communication where on-time delivery of information is a necessity, hence keeping the latency at a given time no matter what else is happening [4]. For mission-critical applications, the latency requirements are important to be kept to

ensure feedback is received fast enough [5]. For the last case mentioned before, with self-driving cars, to keep the passengers of the vehicles safe, there is a need to respond to unpredictable events in the span of a few milliseconds [4]. In all of these cases, there is once again a need to guarantee that the latency requirements are kept.

Another area of technology where there is a lot of progress happening is in the field of Augmented Reality and Virtual reality (AR/VR), or Extended Reality (XR), this field includes holographic media and AR glasses. Holographic-Type Communications (HTC) is a technology where a person in one location can be shown as a hologram in another location with the information being sent through the network. One set of extensions to this is to have the hologram be able to be "touched" and this requires ultra-low delay to work (have the touch feedback feel accurate) [4]. Apple's AR glasses have been expected for some years and recently Apple Vision Pro was announced[1]. AR glasses add a layer of sight to the user, this could be seeing a city through the eyes of an architect, discovering the history of a place or creating more interactive learning experiences for students [6]. Both AR glasses and other AR technologies require low latency between doing an action and seeing it happen on the AR display [7].

Consider an example scenario where a customer is using an access point as their entry point to the network. They are having a video call and require low latency to be satisfied with the service. Another user is wanting to start a video call using this same access point, meaning the traffic going through this point would increase. There needs to be made an admission control decision to decide whether or not this user can be admitted to the access point for their video call. The decision will be made based on whether or not the latency guarantees can be held with this added traffic through the access point. This means there has to be a way to predict what the new latency with the two users would be. One way to make this prediction could be to use analytical models of the system, which is why this thesis focuses on reviewing such models.

In newer times, analysis of these analytical models has not been discovered through the work on this thesis. That is hence why this Masters's thesis aims to review analytical models for latency guarantee estimation and also investigate their validity or applicability to tomorrow's communication networks.

1.1.1 Related work

There is not a lot of related work for this that has been discovered while researching this topic. There are for instance results validating the queueing model $M/M/1$

[1]Apple press release: "Introducing Apple Vision Pro: Apple's first spatial computer", link: https://www.apple.com/newsroom/2023/06/introducing-apple-vision-pro/ (last visited: Jun. 9, 2023)

dating over 100 years back and other models have also been verified throughout the years, discovering all of this work goes beyond the scope of this thesis.

In [8], approximated results for the tail in a multiclass system are proposed. Simulation is then used to decide how accurate the approximations are. This thesis will use simulation in a similar way by evaluating the applicability of analytical models.

1.2 Objectives

The objective of this Master's thesis is to review analytical models for latency guarantee estimation and investigate their validity or applicability. The investigation of the analytical models is done by comparing them to simulation results. This provides insights into whether such models are applicable or what may be needed to make them applicable. The specific models that will be investigated are $M/M/1$, $M/G/1$, $M/D/1$, $G/G/1$, $D/D/1$, $G/G/1 - priority$ and $M/G/1 - priority$. The research question for this thesis is:

RQ: *How are analytical models applicable for latency prediction and guarantees, if at all?*

1.3 Outline

Chapter 2 gives the background information needed for the rest of the thesis. This is done by presenting the formulas needed for finding the results later. Chapter 3 provides the methodology used to create the results needed for answering the research questions. Chapter 4 presents the results obtained throughout the work with the objective that is needed for answering the research question. Chapter 5 discusses the results presented earlier and how these results answer the research question. Chapter 6 gives a conclusion to the results and what they say and gives future work to be done on this topic.

Chapter 2
Background

The following chapter contains an introduction to different queueing models investigated in the thesis.

2.1 System model and notation

2.1.1 System model

All queueing models described in this chapter are considered single-server systems with an infinite queue size. The queue is either consisting of a single class or more classes with different priority levels. If there is only a single class, packets are handled in a First In First Out (FIFO) manner to decide which packet to serve next. Figure 2.1 provides an overview of the system described. The cloud shows where the packets are being generated from with rate λ, the packets arrive in the queue/buffer before being processed with rate μ [9]. The model time is normalised with the unit being the expected service time of a packet in a system. The packet size is normalised with the unit being the expected packet size in bits.

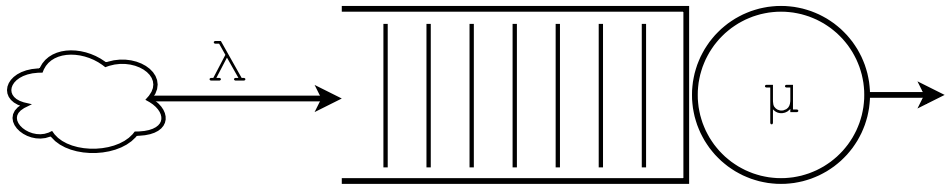


Figure 2.1: System outline with one source

If there are multiple classes and hence multiple priority levels, a non-preemptive scheduling algorithm like strict priority is used to decide the next packet to be served. Figure 2.2 shows how this system will look with multiple clouds generating packets with rate λ_i instead of just one and the strict priority being shown by having the

different classes enter different queues where the highest priority, λ_1, always jumps the queue by being handled before the packets arriving in the other queues. The processing part of the system is still the same.

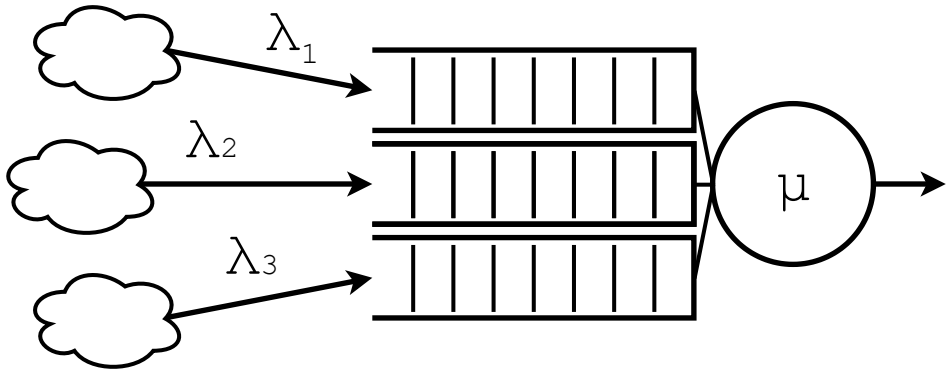


Figure 2.2: System outline with multiple classes

A third system model outline is shown in figure 2.3. This model is showing what happens if there are multiple sources generating packets for the queue instead of just one. This case can happen with priority queueing added as well.

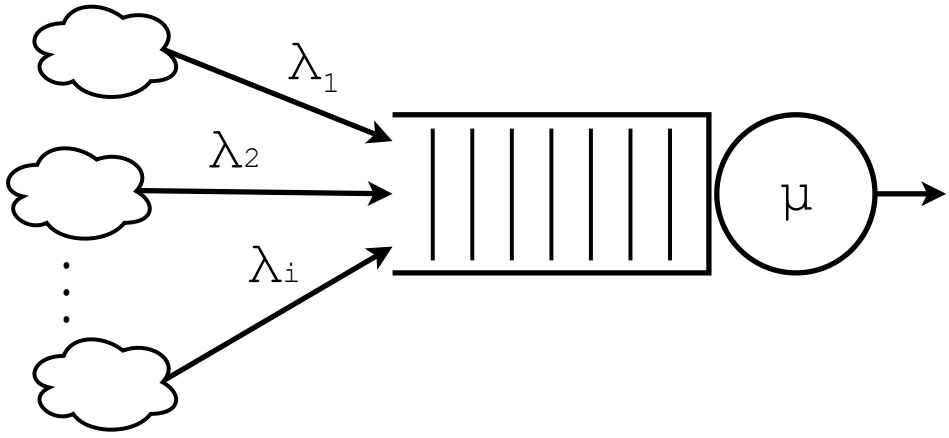


Figure 2.3: System outline with multiple sources

For all models, the stability condition $\lambda < \mu$ has to be fulfilled for the system to be able to reach steady state. This means the arrival rate of packets has to be lower than the service rate of the packets. The formulas presented in this chapter are only valid as long as the system is stable.

Table 2.1: Notation

Symbol	Description
λ	Average arrival rate
σ_a^2	Variance of interarrival time
μ	Average service rate
σ_s^2	Variance of service time
ρ	Server utilization, $\rho = \frac{\lambda}{\mu}$
δ	Average service time
W	Waiting time in queue
$\overline{W}$	Expected waiting time in queue
D	System delay
$\overline{D}$	Expected system delay

2.1.2 Notation

Table 2.1 shows the notation used for the single queue single server system. When there are multiple classes present, a corresponding subscript is added.

2.2 Single class

With all packets belonging to a single class there is only one priority level with packets arriving based on the same process and their size is also based on the same process.

2.2.1 $M/M/1$

An $M/M/1$ model means the inter-arrival time of packets follows a Poisson process, hence are independent and exponentially distributed with parameter λ (memoryless). The packet sizes are also independent and exponentially distributed meaning service time is exponentially distributed with parameter μ. This gives the system utilization of $\rho = \frac{\lambda}{\mu}$ [10].

The $M/M/1$ model has many exact results from classical queueing theory. The expected waiting time is shown in equation 2.1, the waiting time distribution, the Complementary Cumulative Distribution Function (CCDF), is shown in equation 2.2, expected system delay is shown in equation 2.3 and the system delay CCDF is shown in equation 2.4, all from [11].

$$\overline{W} = \frac{\rho}{\mu - \lambda} \tag{2.1}$$

$$P\{W > t\} = \rho e^{-(\mu-\lambda)t} \tag{2.2}$$

$$\overline{D} = \frac{1}{\mu - \lambda} \tag{2.3}$$

$$P\{D > t\} = e^{-(\mu-\lambda)t} \tag{2.4}$$

2.2.2 $M/G/1$

With $M/G/1$, packet inter-arrival time is exponentially distributed with parameter λ. Packets size, and hence service time, does not follow a specified pattern and is generalised. Parameters are μ and σ_s^2. Service time could for instance be uniformally distributed. Formulas for $M/G/1$ are hence more generic than for $M/M/1$. The expected waiting time shown in [11] is reproduced in equation 2.5.

$$\overline{W} = \frac{\rho^2 + \lambda^2\sigma_s^2}{2\lambda(1-\rho)} \tag{2.5}$$

2.2.3 $M/D/1$

An $M/D/1$ model means the packet's inter-arrival time follows a Poisson process, hence is exponentially distributed with parameter λ. The packet sizes are deterministic, hence is service time deterministic with parameter μ and service time $\delta = \frac{1}{\mu}$ is constant [11]. Expected waiting time and system delay are given in [11] and shown in equation 2.6 and 2.7. The expected waiting time distribution is given in equation 2.8.

$$\overline{W} = \frac{\rho}{2(\mu - \lambda)} \tag{2.6}$$

$$\overline{D} = \frac{\rho}{2(\mu - \lambda)} + \frac{1}{\mu} \tag{2.7}$$

$$P\{W > t\} = 1 - (1-\rho)\sum_{k=0}^{\lfloor t \rfloor} \frac{(\rho k - \lambda t)^k}{k!} e^{-(\rho k - \lambda t)} \tag{2.8}$$

2.2.4 $G/G/1$

With $G/G/1$, no specific pattern is followed for packet inter-arrival time with parameters λ and σ_a^2. Packets size, and hence service time, does not follow a specified pattern either, parameters μ and σ_s^2. Both processes are hence generalised. [11]

shows an approximation for the expected waiting time, an upper bound for the expected waiting time and an approximation for the waiting time distributions CCDF, reproduced in equation 2.9, 2.10 and 2.11.

$$\overline{W} \approx \frac{\lambda(\rho^2 + \lambda^2\sigma_s^2)(\sigma_a^2 + \sigma_s^2)}{2(1-\rho)(1+\lambda^2\sigma_s^2)} \tag{2.9}$$

$$\overline{W} \leq \frac{\lambda(\sigma_a^2 + \sigma_s^2)}{2(1-\rho)} \tag{2.10}$$

$$P\{W > t\} \approx \rho e^{-\rho t/\overline{W}} \tag{2.11}$$

2.2.5 $D/D/1$

With a D/D/1 model, the inter-arrival time of packets is deterministic, this could be periodic or another pre-determined pattern. The service time is also deterministic; fixed or other. Parameters used are λ for inter-arrival time and μ for service time. The system is stable as long as $\lambda < \mu$ [10]. Network calculus can be used to find an upper bound for the system delay, D. In [11] it is shown how this can be done in general. Below follows the specific bounds for one source and multiple sources.

When there is one source generating packets for the queue with all packets having packet size L and the server rate is constant and equal to R the service rate μ is equal to $\frac{R}{L}$. This means the upper bound for the system delay is given by

$$D \leq \frac{L}{R}.$$

When there are multiple sources generating packets for the system where all packet sizes are equal to L. The number of sources generating packets is equal to I and the server rate is still constant and equal to R. The upper bound for the system delay is given by

$$D \leq \frac{I \cdot L}{R}.$$

2.3 Multiple classes

Each class i can have its own arrival process, packet size distribution and priority level i. The highest priority level is 1 and the class number i is always equal to the priority level of the class i. There are few exact results [11].

2.3.1 $G/G/1 - priority$

For $G/G/1 - priority$, with the assumption that the expected waiting time for each class i, $\overline{W}_i$, is known, the following waiting time distribution from [11] follows:

$$P\{\overline{W}_i > t\} \quad \approx \quad \rho e^{-\rho t/\overline{W}_i} \tag{2.12}$$

where the system utilization $\rho = \sum_i \rho_i$.

2.3.2 $M/G/1 - priority$

With $M/G/1 - priority$ all classes have packets arrive by a Poisson process and inter-arrival times of each class i are exponentially distributed with parameter λ_i, hence the combined arrival process is also an exponential distribution. Classes will have different priority levels, meaning the queueing time for a packet is dependent on its priority class. Each class service time could be exponentially distributed or follow another distribution like the uniform distribution, parameters μ_i and $\sigma^2_{s,j}$. The expected waiting time for each class shown in [11] is:

$$\overline{W}_i \quad = \quad \frac{\overline{R}}{(1 - \sum_{j=1}^{i-1} \rho_j)(1 - \sum_{j=1}^{i} \rho_j)} \tag{2.13}$$

with

$$\overline{R} = \sum_j \frac{\rho_j^2 + \lambda_j^2 \sigma_{s,j}^2}{2\lambda_j}$$

To obtain the waiting time distribution for $M/G/1 - priority$, equation 2.12 can be used, but now without the assumption that $\overline{W}_i$ is known. The equation will hence look the same:

$$P\{\overline{W}_i > t\} \quad \approx \quad \rho e^{-\rho t/\overline{W}_i} \tag{2.14}$$

Chapter 3
Methodology

The methodology chapter provides an overview of the methods and procedures used in this project. It is split into multiple parts and starts with the literature study. The simulation process is then described and how the simulator works. Lastly, it is shown how both the theoretical and simulated results can be plotted.

3.1 Literature study

A literature study was done to find theoretical formulas for use and comparison with simulated results. The result of the literature study is shown in chapter 2.

3.2 Simulation

Creating, using and verifying the simulator is a big part of this thesis. Below follows the steps made in this process. Firstly, the setup of the simulator is presented. Then, how to generate a dataset from simple inputs and use this for verification purposes is shown. Extensions of the simulator are then presented. Lastly, algorithms needed for the generation of different needed inputs for the simulator are described.

The full code for the simulator created for this project can be found on GitHub[1]. It consists of files used for generating input to the simulator, running the simulator and plotting the results of the simulator. It also has files for running specific use cases with corresponding folders where inputs and results are saved.

3.2.1 Simulator setup

The simulator is implemented in Python using SimPy [12] for simulation-specific purposes and NumPy [13] for randomized number pulling. The simulator is set up in such a way that it can handle many different cases based on what inputs are given to it.

[1]Full GitHub project: https://github.com/noralir/TTM4905

Figure 3.1 provides an overview of how the simulator is working. When the simulator is started a generator starts as many packet generators as sources wanted for the simulation. The packet generator then creates packets with the inter-arrival time of the generation of the packets based on what distribution it is following. The generated packets then go through the system before saving their data.

The simulator takes multiple inputs to decide how it will work for specific runs. To keep track of the specific parameters and values used for each simulation, a JSON file is used. This file is added as an input variable to the simulator as `input_variables`. How this JSON file can look is shown in section 3.2.2, 3.2.3 and 3.2.4. Another field that is added to the simulator is the `filename_data` field. It can either take the boolean value `False` if the full output dataset of the simulator should not be saved, or, if it is going to be saved, `filename_data` takes the name of the CSV file the data should be saved in as input. The input field `folder_nth` follows a similar pattern as it decides if packet #n (#0, #10, #100, #1000 and #10000) should be saved by either getting the boolean `False` or the folder where the data should be saved.

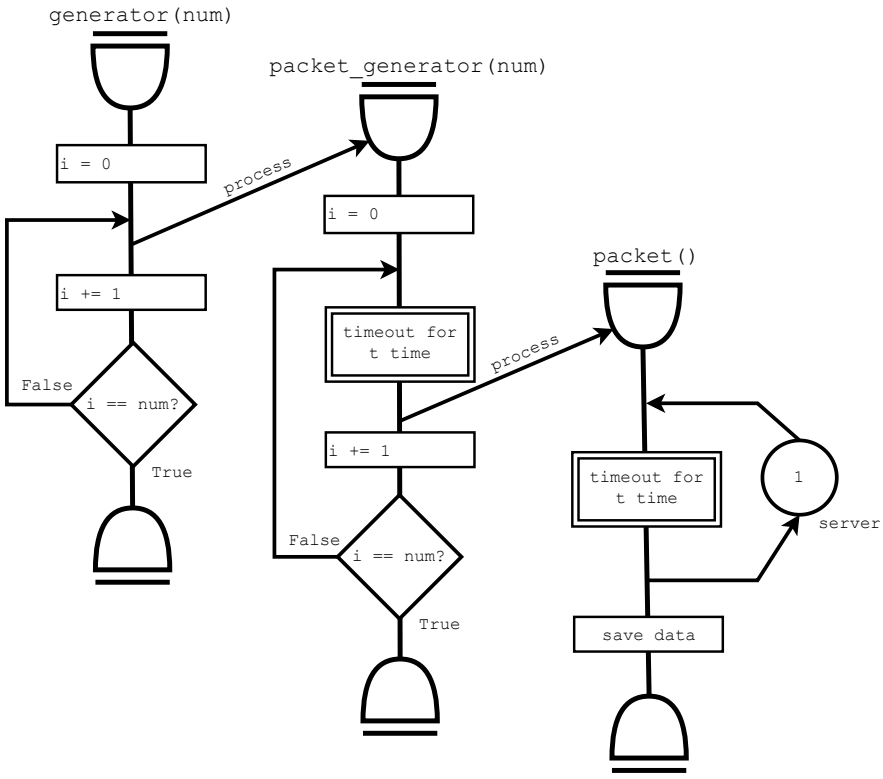


Figure 3.1: Activity diagram for simulator

In addition to just running the simulator as is, there is also the option to decide how many times it should be run by using another function on top. For this function, the same inputs as for the simulator needs to be provided as well as a number for how many times it should be run. This function is especially needed if there is a want to collect packet $\#n$ from each run, as one run would only provide one packet for this collection.

Further in this section follows a more in-depth description of how the inputs and outputs of the simulator look and are created. First a simple simulator, then some extensions to the simulator and lastly some helper algorithms to be used for input variable generation.

3.2.2 Simple simulator inputs

The simulator generates packets based on inputs given to the function via a JSON file. Below are the inputs used by the simulator with example values. The λ for the system becomes the inverse of the average inter-arrival time, `avg_pkt_ia_time`. Note that when `capacity` is set to 1, the expected service time equals the packet size `avg_pkt_len_bits`, meaning μ becomes the inverse of the packet size. The example input values would generate a simulation following an $M/M/1$ distribution with $\lambda = 1/15$ and $\mu = 1/10$.

```
----------------------------------------------------------------------
{
"avg_pkt_ia_time": 15,       # Average inter-arrival time for packets
"dist_type_pkt_ia_time": "M",# Arrival process distribution
"avg_pkt_len_bits": 10,      # Average packet size
"dist_type_pkt_len": "M",    # Packet size distribution
"capacity": 1,               # Capacity of server
"num_pkts": 1000             # Number of packets to be generated
}
----------------------------------------------------------------------
```

The `dist_type_pkt_ia_time` field determines the arrival process distribution. The value can be `"M"`, `"D"` or `"G_uniform"`, resulting in respectively exponential, deterministic (periodic) or uniform distribution of packet arrivals. Similarly, the field `dist_type_pkt_len` takes input values `"M"`, `"D"` or `"G_uniform"` resulting in respectively exponential, deterministic (fixed) or uniform distribution of packet sizes and hence service time.

Verification of simple simulator

The initial simple simulator has to be verified. After verifying that it is working as intended, it can be extended to handle more complex data and hence create more complex and interesting datasets to compare theoretical results with. The simulator can be verified by comparing it with theoretical queueing models lining up with what should be generated.

3.2.3 Simulate with multiple parts

If there is a want to change variables over time, the values of the initial inputs shown in section 3.2.2 can be extended to be lists instead. The different indexes of the list will then represent the different parts of the run. Below is an example of the extended inputs, having a run with two parts.

```
----------------------------------------------------------------------
{
"avg_pkt_ia_time": [15, 10],        # Change inter-arrival time
"dist_type_pkt_ia_time": ["M", "M"],# Keep same arrival process
"avg_pkt_len_bits": [10, 10],       # Keep same packet size
"dist_type_pkt_len": ["M", "D"],    # Change packet size distribution
"capacity": 1,                      # Capacity is always constant
"num_pkts": [1000, 1000]            # Number of packets per part
}
----------------------------------------------------------------------
```

When simulating with multiple parts, the output of interest is packet $\#n$ after changing the parameters. The resulting plots are hence a statistical distribution for packet $\#n$.

3.2.4 Simulate with multiple sources

Both of the above examples show when there is only one source generating packets. If there is a want for multiple sources generating packets per part, the values at the different indexes are swapped for lists and a variable stating the different sources' priority level is added; `num_sources`. If all sources have the same priority level (e.g. `[1, 1]`), there will still be only one class of packets present in the simulation. If the sources have different priority levels (e.g. `[1, 2, 3]`), there will be multiple classes of packets present in the simulation. It is still possible to add more parts to the run by adding more indexes.

```
-----------------------------------------------------------------------
{
"avg_pkt_ia_time": [[15, 25]],        # Different inter-arrival time
"dist_type_pkt_ia_time": [["M", "D"]],# Different arrival processes
"avg_pkt_len_bits": [[10, 10]],       # Same packet size
"dist_type_pkt_len": [["M", "M"]],    # Same packet size distribution
"capacity": 1,                        # Capacity is constant
"num_pkts": [[1000, 1000]],           # Number of packets per source
"num_sources": [[1, 1]]               # Same priority
}
-----------------------------------------------------------------------
```

When generating a dataset based on a simulation with multiple sources and no priority, the wanted dataset is the combined output of all sources. If there are multiple priorities, and hence classes, i, present, the dataset should be split with one dataset per class i, meaning sources with the same priority level should still be combined.

Generation of inter-arrival times and packet sizes

When multiple sources are present, there is sometimes a need to generate different expected inter-arrival times and packet sizes to ensure sources are not identical. Especially if the arrival process is deterministic. Below follows the algorithms used for achieving this in this project.

For the inter-arrival times, a list with different values is needed. Algorithm 3.1 shows pseudo-code of how a list of unique inter-arrival times is generated, based on given wanted average `avg` and the number of sources `n`. To keep the average of the list equal to `avg`, the average of the generated inter-arrival times has to be equal to $\texttt{avg} \cdot \texttt{n}$. This implementation assumes `n` is an even number.

By having $\texttt{n} = 4$ and $\texttt{avg} = 20$, the list becomes `[70, 90, 50, 110]`. This list is often sorted in ascending order before being used in simulation.

Note that `avg` will not be equal to the expected inter-arrival time of the system. The expected inter-arrival time will be the weighted average value of the generated list. This is because the lower inter-arrival times will generate more packets than the highest ones in any given longer time period. For exact calculations of λ, see section 3.3.1.

For the packet sizes, another list with different values is needed. Algorithm 3.2 shows pseudo-code of how a list with packet sizes is generated, based on the given expected average `avg` and the number of sources `n`.

Algorithm 3.1 Generation of inter-arrival times

```
function generate_inter_arrival_times(n, avg):
    average = avg * n
    increment_number = avg
    number1 = integer(average - increment_number/2)
    number2 = integer(average + increment_number/2)
    inter_arrival_times = [number1, number2]
    current_increment = increment_number
    while the length of inter_arrival_times is less than n:
        append (number1 - current_increment) to inter_arrival_times
        append (number2 + current_increment) to inter_arrival_times
        set current_increment to current_increment + increment_number
    return inter_arrival_times
```

Algorithm 3.2 Generation of packet sizes

```
function generate_packet_sizes(n, avg):
    average = avg
    start = average-4.5
    packet_sizes = [integer(start+1), integer(start+2), ...,
                    integer(start+10)]
    if n is equal to 10:
        return packet_sizes
    else if n is equal to 100:
        update packet_sizes to repeat the original list 10 times
        return packet_sizes
    else if n is equal to 1000:
        update packet_sizes to repeat the original list 100 times
        return packet_sizes
```

3.2.5 Output of simulator

The simulator produces a CSV file as output. The file contains headers and one row for each packet with values gathered during the run. An example of how the file can look follows below:

```
-----------------------------------------------------------------------
number, t_generated, t_buffer, t_processing, pkt_size_bits, n_in_queue
001, 0, 0, 3, 3, 0
023, 1, 3, 2, 2, 0
012, 2, 1, 1, 1, 1
101, 5, 1, 1, 1, 0
...
-----------------------------------------------------------------------
```

The file's headers are set and always the same. The `number` field gives each packet a number, where the last digit gives the packet's priority level and the second to last digit gives which source generated the packet and the remaining digits give the packet number from this source. The number `001` gives packet 0 for source 0 with priority 1, and `1232` gives packet 12 from source 3 with priority 2. `t_generated` is the time the packet was generated, `t_buffer` is the time the packet spent in the queue before being processed. `t_processing` is the service time of the packet, `pkt_size_bits` is the packet size and `n_in_queue` is the number of packets observed in the queue as this packet arrived in the system.

3.3 Plotting of theoretical results

For plotting the theoretical results of the formulas given in chapter 2 with values based on the simulation input parameters, Python with Matplotlib [14] is used.

3.3.1 Calculation of theoretical parameters

For plotting, there is a need to calculate the expected λ and μ from the inputs given to the simulation. When there is only one source, this is straightforward, but when there are multiple sources j belonging to the same class i, there is a bit more calculation needed.

Calculation of λ

Where λ_j is the inverse of the expected inter-arrival time of source j and λ_i is the expected inter-arrival rate of packets of class i. And λ is the expected inter-arrival

time of all packets.

$$\lambda_i = \sum_j \lambda_j$$
$$\lambda = \sum_i \lambda_i$$

Calculation of σ_a^2

Some theoretical formulas require the calculation of σ_a^2. This is done based on what distribution type is used. See table 3.1 with variance values as given in [9].

Table 3.1: Calculation of σ_a^2

Distribution type	σ_a^2
Exponential	$\frac{1}{\lambda^2}$
Uniform	$\frac{1}{12} \cdot (\frac{2}{\lambda})^2$
Deterministic	0

Calculation of μ

μ_j is the expected packet size of class j. μ_i is the expected packet size of class i.

$$\mu_i = \frac{\lambda_i}{\sum_j \frac{\lambda_j}{\mu_j}}$$

Calculation of σ_s^2

Some theoretical formulas require the calculation of σ_s^2. This is done based on what distribution type is used. See table 3.2 [9].

Table 3.2: Calculation of σ_s^2

Distribution type	σ_a^2
Exponential	$\frac{1}{\mu^2}$
Uniform	$\frac{1}{12} \cdot (\frac{2}{\mu})^2$
Deterministic	0

3.4 Plotting of simulated results

The plotting of the simulated results is also done in Python using Matplotlib [14]. This assures that comparing the theoretical and simulated results is a simple procedure.

There is a need to remove parts of the dataset before plotting if there are multiple sources with different arrival intensities generating packets. This is because there, in general, is said that each source should generate the same amount of packets, the highest intensity sources will hence be done generating packets before the lower intensity sources are done. The tail of the dataset must then be removed to ensure the correct parts are kept. This can be done by identifying the last packet of the source with the highest intensity and disregarding packets that were generated after this point.

There is some post-processing needed to obtain the results wanted for the plotting. The simulator produces CSV files with one line of information per packet as output. This cannot be used directly to create a CCDF of the generated dataset.

For finding the simulation's CCDF for the waiting time, the values for the `t_buffer` field are plotted as a histogram. By having the histogram ranges short enough, the resulting plot will take the form of a CCDF and hence the same form as the theoretical results. The CCDF for the system time can be found similarly, but using the sum of the `t_buffer` and `t_processing` fields as the input values for creating the histogram.

Chapter 4
Results

This chapter contains the results of the simulated use cases with the theoretical distributions that go along with them, using appropriate numbers for the parameters to have them line up as well as possible.

There are three use cases that could be investigated for the different analytical models; single source, single source with multiple parts and multiple sources. The single source cases are the most straightforward cases with one source generating traffic. This results in one plot to compare to the theoretical distribution. This use case is needed to provide a basic comparison between theoretical and simulated results. The next use case still has one source, but multiple parts where the input parameters are changing. This is interesting to look at as the load on a network can vary over time. The results of interest are packet $\#n$ after changing the parameters and all these packet $\#n$ distributions are compared to the theoretically corresponding plot. The last use case has multiple sources generating traffic. The reason for looking into this use case is as there can be multiple users using the same access point and the traffic will hence come from different sources. It is interesting to see how the combined output of the sources compares to the theoretical plots.

The chapter first shows the results for the single-class cases; $M/M/1$, $M/G/1$, $G/G/1$ and $D/D/1$. The queue discipline is FIFO. The first results are from cases relating to the $M/M/1$ model, beginning with one source, then one source with multiple parts and lastly multiple sources. Similarly follows the cases related to the $M/G/1$ model. Then some cases related to the $G/G/1$ and $D/D/1$ models are shown.

The results for the cases containing multiple classes and priorities are then presented. The two specific cases are $G/G/1 - priority$ and $M/G/1 - priority$. The queueing discipline implemented is strict priority. For $G/G/1 - priority$, the investigated cases are with one source per class and multiple classes per class. For $M/G/1 - priority$, a case with one source per class is investigated.

4.1 Summary of results

All use cases shown in this chapter are summarised in table 4.1. The table states in what figure or table the results of the specific use case can be found, what analytical model was investigated in the use case (theoretical distribution) and the different values that were used as inputs to the simulation and for the theoretical formulas.

Table 4.1: Summary of all use cases shown

Resulting figure	**Theoretical distribution**	**Priority i**	**# sources**	**Arrival distribution of source**	λ_i	**Service time distribution of source**	μ_i
Figure 4.1	$M/M/1$	1	1	Exponential	0.067	Exponential	0.1
Figure 4.4	$M/M/1$	1	1	Exponential	0.015, 0.025, 0.045, 0.015, 0.025	Exponential	0.083, 0.083, 0.083, 0.083, 0.083
Figure 4.5	$M/M/1$	1	1	Exponential	0.015, 0.025, 0.045, 0.015, 0.025	Exponential	0.167, 0.1, 0.063, 0.167, 0.1
Figure 4.6a	$M/M/1$	1	10	Exponential	0.073	Exponential	0.1
Figure 4.6b	$M/M/1$	1	100	Exponential	0.073	Exponential	0.1
Figure 4.6c	$M/M/1$	1	10	Deterministic (periodic)	0.073	Exponential	0.1
Figure 4.6d	$M/M/1$	1	100	Deterministic (periodic)	0.073	Exponential	0.1
Figure 4.7	$M/G/1$	1	1	Exponential	0.05	Uniform	0.083

Figure 4.8	$M/D/1$	1	1	Exponential	0.067	Deterministic (fixed)	0.1
Figure 4.9a	$M/D/1$	1	10	Exponential	0.073	Deterministic (fixed)	0.1
Figure 4.9b	$M/D/1$	1	100	Exponential	0.073	Deterministic (fixed)	0.1
Figure 4.9c	$M/D/1$	1	10	Deterministic (periodic)	0.073	Deterministic (fixed)	0.1
Figure 4.9d	$M/D/1$	1	100	Deterministic (periodic)	0.073	Deterministic (fixed)	0.1
Figure 4.7	$G/G/1$	1	1	Uniform	0.1	Uniform	0.167
Table 4.6	$D/D/1$	1	1	Deterministic (periodic)	0.1	Deterministic (fixed)	0.1
Table 4.6	$D/D/1$	1	10	Deterministic (periodic)	0.073	Deterministic (fixed)	0.1
Figure 4.12	$G/G/1-$ *priority*	1	1	Deterministic (periodic)	0.033	Deterministic (fixed)	0.100
		2	1	Exponential	0.033	Exponential	0.200
		3	1	Exponential	0.033	Exponential	0.143
Figure 4.13a	$G/G/1-$ *priority*	1	10	Deterministic (periodic)	0.0137	Exponential	0.0571
		2	10	Exponential	0.00685	Exponential	0.0571
		3	10	Uniform	0.00914	Exponential	0.0571
Figure 4.13b	$G/G/1-$ *priority*	1	100	Deterministic (periodic)	0.0137	Exponential	0.0571
		2	100	Exponential	0.00686	Exponential	0.0571
		3	100	Uniform	0.00915	Exponential	0.0571
Figure 4.14a	$G/G/1-$ *priority*	1	10	Deterministic (periodic)	0.0137	Deterministic (periodic)	0.0571
		2	10	Exponential	0.00685	Exponential	0.0571
		3	10	Uniform	0.00914	Uniform	0.0571

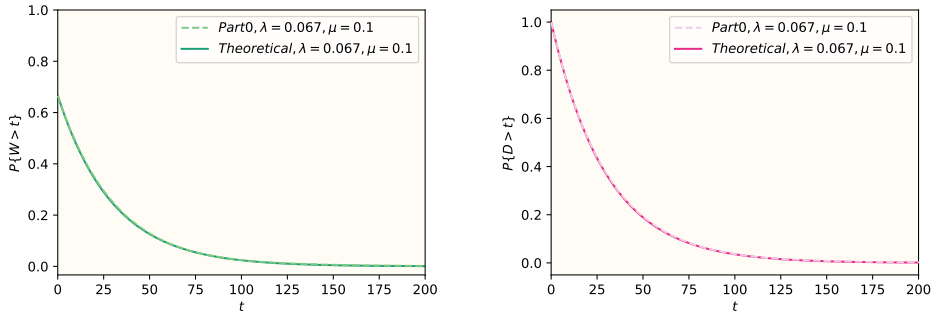
Figure 4.14b	$G/G/1-$ *priority*	1	100	Deterministic (periodic)	0.0137	Deterministic (periodic)	0.0571
		2	100	Exponential	0.00686	Exponential	0.0571
		3	100	Uniform	0.00915	Uniform	0.0571
Figure 4.12	$M/G/1-$ *priority*	1	1	Exponential	0.043	Deterministic (fixed)	0.100
		2	1	Exponential	0.043	Uniform	0.200
		3	1	Exponential	0.043	Exponential	0.143

4.2 $M/M/1$

An $M/M/1$ model has an exponential arrival process and exponentially distributed service times. Below follows the results of the simulation with $M/M/1$ parameters with comparisons to the theoretical results of the same model. First a case with a single source, then two cases with a single source with multiple parts and lastly cases with multiple sources.

4.2.1 Single source

The first case that was investigated was an $M/M/1$ model with $\lambda = 0.067$ and $\mu = 0.1$. The capacity of the server was 1 and 100 000 packets were generated. The resulting CCDF for the waiting time and the sojourn time (system delay) of the simulation and the theoretical formula are shown in figure 4.1.

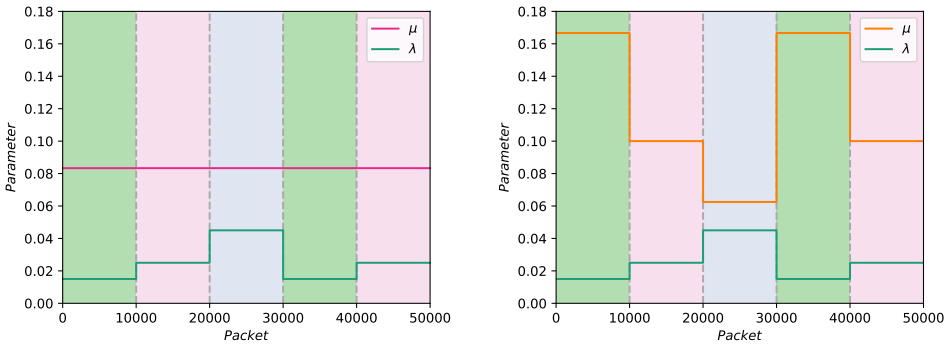


(a) CCDF for waiting time **(b)** CCDF for sojourn time

Figure 4.1: $M/M/1$ model with one source

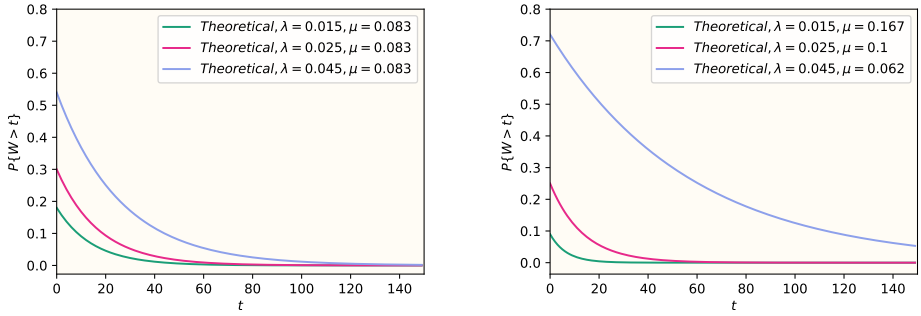
4.2.2 Single source with multiple parts

Consider an $M/M/1$ model with parameters changing over time, it is then interesting to investigate packet $\#n$ after changing parameters.



(a) Changing λ **(b)** Changing λ and μ

Figure 4.2: $M/M/1$ model with changning parameters for each part of a run

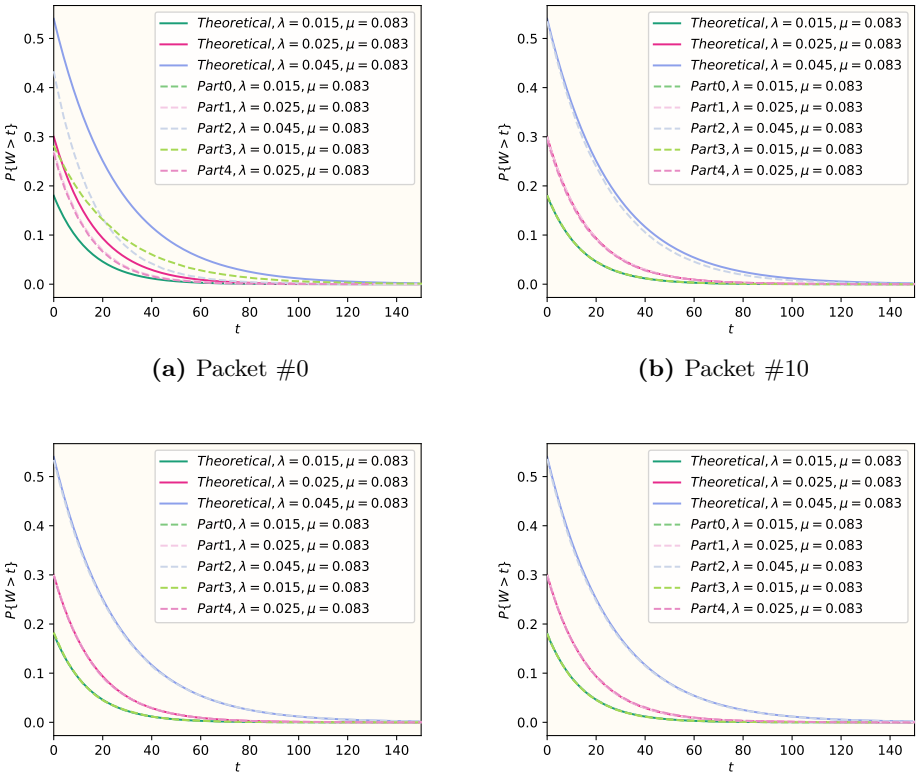


(a) Changing λ **(b)** Changing λ and μ

Figure 4.3: Theoretical $M/M/1$ model with changing parameters

First, consider changing λ after 10000 packets, having 5 parts in the run. λ is changing between 0.015, 0.025 and 0.045 while μ is kept constant at 0.083. Figure 4.2a shows this and how the parameters vary for this case. This results in the system starting out with a low load, going up to a medium load, then a high load, and back down to a low load before ending with a medium load. This means there are three theoretical curves forming for the five parts, the CCDF for the waiting time of these are shown in figure 4.3a. While only changing λ, the simulator was run 48 101 times with the 5 parts with 10 000 packets per part while collecting packet #0, #10, #100,

#1000 and #10000 of each part of each run. The resulting CCDF for the waiting time of packet #n is shown in figure 4.4. Note that in figure 4.4a, the very first simulated line is not visible as every first packet generated has 0 buffer time. Packet #10000 of each part is not shown in the figure.

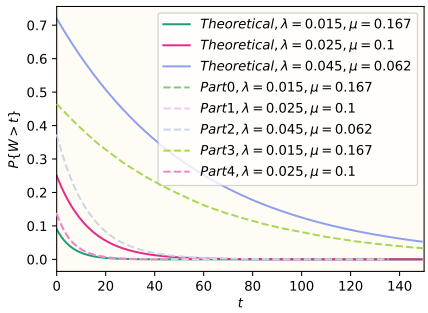


(a) Packet #0 **(b)** Packet #10

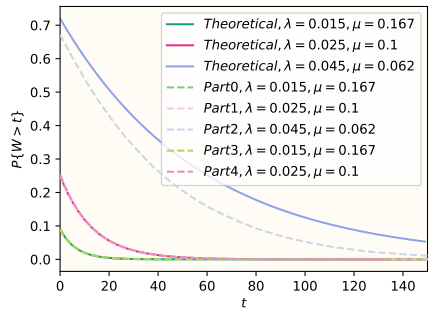
(c) Packet #100 **(d)** Packet #1000

Figure 4.4: $M/M/1$ model with changing λ

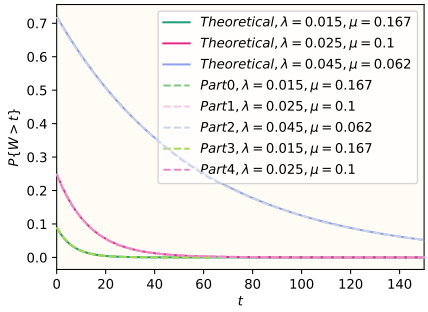
The other investigated case for an $M/M/1$ model with multiple parts is changing both λ and μ as shown in figure 4.2a. λ is still varying between 0.015, 0.025 and 0.045, μ now varies between 0.167, 0.1 and 0.063. The system still has the load varying as low-medium-high-low-medium, but by changing μ as well, the differences in the loads are greater than for the latter case. The theoretical plots are shown in figure 4.3b. For this case, the simulator was run 46 601 times with 20 000 packets per part per run while recording packet #0, #10, #100, #1000 and #10000 of each part of each run. The resulting CCDF for the waiting time of packet #n is shown in figure 4.5, again is packet #10000 of each part recorded but not shown.



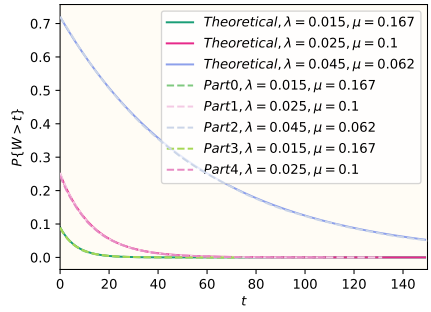
(a) Packet #0



(b) Packet #10



(c) Packet #100



(d) Packet #1000

Figure 4.5: $M/M/1$ model with changing λ and μ

4.2.3 Multiple sources

It can be interesting to see what happens when multiple independent sources generate packets. The reason for this is that for one access point, there is most likely more than one user. It is then important to understand how the traffic is acting if there are multiple sources and if it will still be comparable to analytical models.

Table 4.2: $M/M/1$ with multiple sources simulation cases

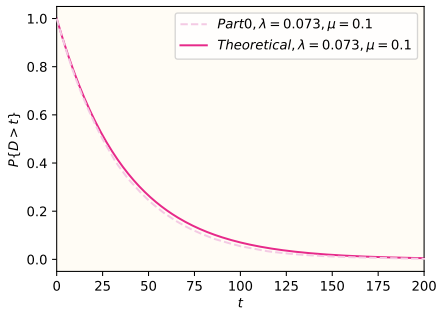
#sources	Inter-arrival time distribution per source	Service time distribution per source
10	Exponential	Exponential
100	Exponential	Exponential
10	Deterministic (Periodic)	Exponential
100	Deterministic (Periodic)	Exponential

Multiple cases were investigated and are summarised in table 4.2. The first two cases all have sources that have an exponential arrival distribution and an exponential service time distribution. Firstly this combination is simulated with 10 sources and then with 100 sources. The last two cases have all sources have a deterministic (periodic) arrival process and an exponential service time distribution. Once again, it is first simulated with 10 sources and then with 100 sources. Algorithm 3.1 with `n` equal to 10 or 100 depending on the case and `avg` equal to 15 was used for generating different arrival intensities for the different sources. Table 4.3 summarises the results of this algorithm. The resulting λ for the system is 0.071. The packet size for all sources is 10 resulting in the μ for the system becoming 0.1. When there are 10 sources generating packets, there are 10000 packets per source, while when there are 100 sources, there are 1000 packets per source, resulting in 1 000 000 packets generated per simulation.

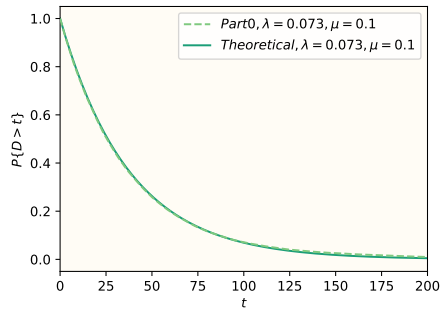
Table 4.3: Input and output for algorithm 3.1 for cases with multiple sources

#sources	`n`	`avg`	`generate_inter_arrival_times(n, avg)`
10	`10`	`15`	`[82, 97, 112, 127, 142, 157, 172, 187, 202, 217]`
100	`100`	`15`	`[757, 772, 787, 802, 817, ... 2182, 2197, 2212, 2227, 2242]`

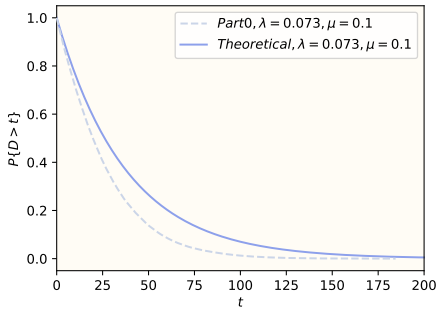
All sources generate packets arriving in the same system and queue, the dataset is the output of all sources aggregated to one. The resulting CCDFs for the waiting times of the different cases are shown in figure 4.6.



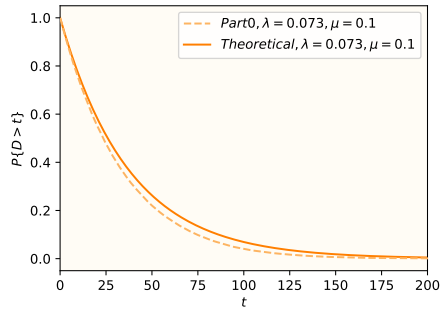
(a) 10 sources with exponential arrival process and exponential service time



(b) 100 sources with exponential arrival process and exponential service time



(c) 10 sources with deterministic (periodic) arrival process and exponential service time



(d) 100 sources with deterministic (periodic) arrival process and exponential service time

Figure 4.6: Waiting time CCDFs for $M/M/1$ cases with multiple sources

4.3 $M/G/1$

$M/G/1$ models have an exponential arrival process, but generalized service time. The case investigated for this model has an exponential arrival process with $\lambda = 0.05$ and a uniform distribution of service time with $\mu = 0.083$ and $\sigma_s^2 = 48$. The simulation was run with 1 000 000 packets. Table 4.4 gives the resulting theoretical expected waiting time and the average waiting time of the simulation. Figure 4.7 gives the CCDF of the simulation.

Table 4.4: $M/G/1$ case with one source

λ	μ	σ_s^2	$\overline{W}$	**Simulated average waiting time**
0.05	0.083	48	12.00	12.10

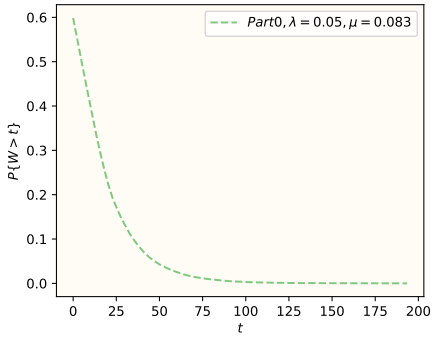


Figure 4.7: Simulated $M/G/1$ waiting time distribution

4.4 $M/D/1$

For the $M/D/1$ model, arrivals are exponentially distributed while service time is deterministic and fixed. Below follows the results for the cases with a single source and with multiple sources.

4.4.1 Single source

This case with one source has packet arrival rate λ equal to 0.067 and service time μ equal to 0.1. The simulation was run with generating 100 000 packets. Figure 4.8 shows the simulations waiting time CCDF and corresponding theoretical distribution using equation 2.8.

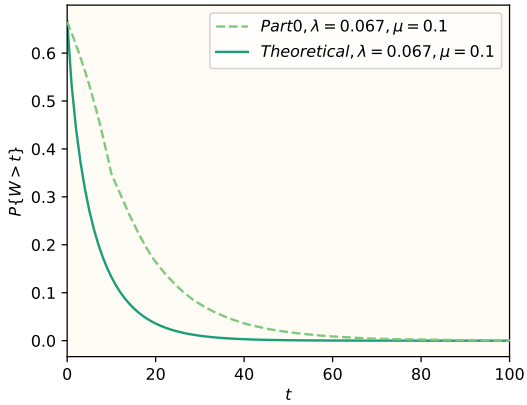
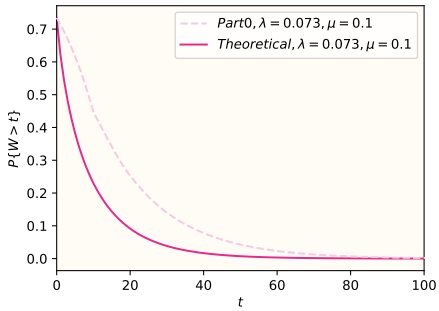


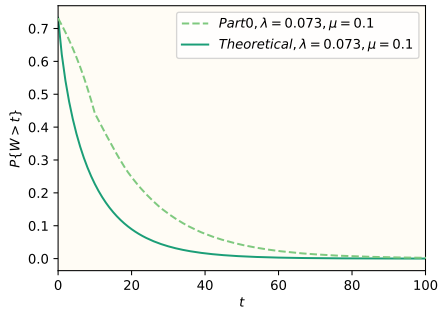
Figure 4.8: $M/D/1$ single source waiting time CCDF

4.4.2 Multiple sources

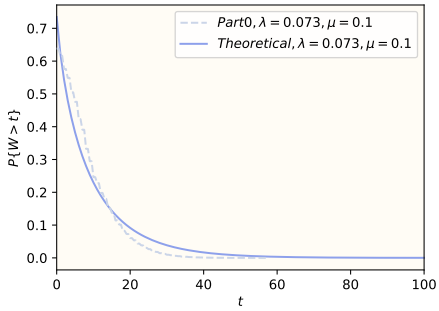
Four different cases with multiple sources generating packets were investigated. Table 4.5 summarises the cases. For the first two sources, the arrival process is exponential while for the latter two, it is deterministic and periodic. Algorithm 3.1 was used for generating the different rates with `n` equal to 10 or 100 depending on the case and `avg` equal to 15, table 4.3 once again shows the output of these cases. This means λ is equal to 0.073. The service time is deterministic and fixed for all cases with $\mu = 0.1$. All cases were run with a total of 1 000 000 packets. The resulting plots are shown in figure 4.9.



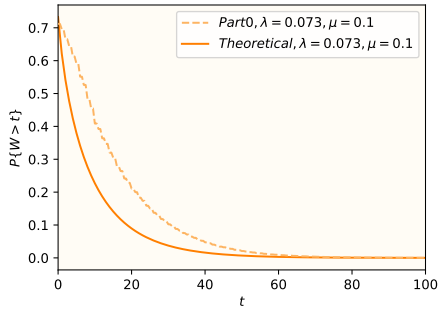
(a) 10 sources with exponential arrival process and deterministic (fixed) service time



(b) 100 sources with exponential arrival process and deterministic (fixed) service time



(c) 10 sources with deterministic (periodic) arrival process and deterministic (fixed) service time



(d) 100 sources with deterministic (periodic) arrival process and deterministic (fixed) service time

Figure 4.9: Waiting time CCDF for $M/D/1$ cases with multiple sources

Table 4.5: $M/D/1$ with multiple sources simulation cases

#sources	Inter-arrival time distribution per source	Service time distribution per source
10	Exponential	Deterministic (Fixed)
100	Exponential	Deterministic (Fixed)
10	Deterministic (Periodic)	Deterministic (Fixed)
100	Deterministic (Periodic)	Deterministic (Fixed)

4.5 $G/G/1$

For $G/G/1$, the arrival process is generalised and so is the service time. Below follows the investigated case for the $G/G/1$ model.

4.5.1 Uniform arrival process and uniform service time distribution with one source

Simulating with $\lambda = 0.1$, $\sigma_a^2 = 33.333$, $\mu = 0.167$, $\sigma_s^2 = 12$ and 100 000 packets with uniform arrival process and uniform service time distribution results in the approximated CCDF for the waiting time shown in figure 4.10.

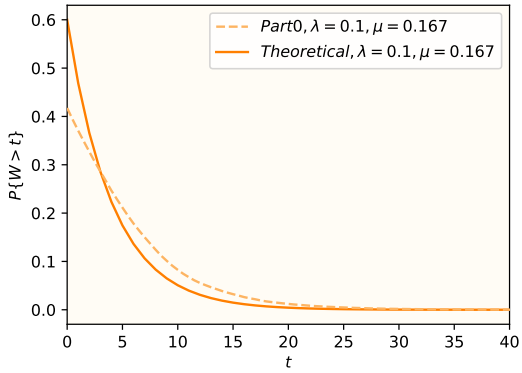


Figure 4.10: $G/G/1$ approximated waiting time CCDF with one source

4.6 $D/D/1$

The next cases that were investigated were ones that can be compared to the network calculus bounds of $D/D/1$ models. The first case was with one source and the other

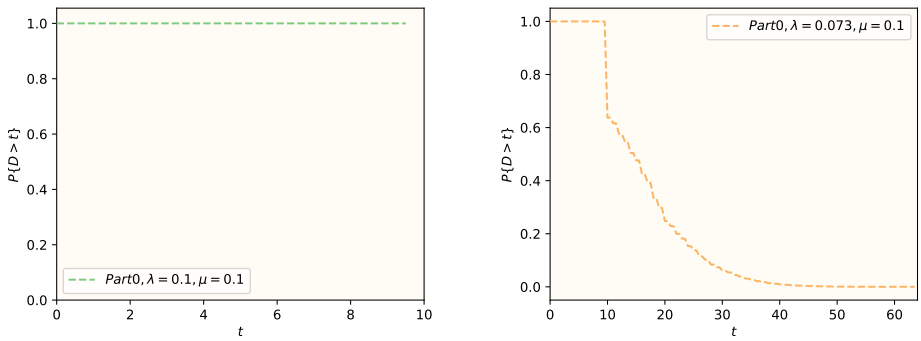
case was with 10 sources. The $D/D/1$ systems have a deterministic (periodic) arrival process and deterministic (fixed) service time. Table 4.6 summarises the results.

Table 4.6: $D/D/1$ simulation cases

Case	**Network calculus bound**	**Theoretical maximum delay value**	**Maximum measured delay**
Single source	$\frac{L}{R}$	10	10
Multiple sources	$\frac{I \cdot L}{R}$	100	63

When simulating with one source, the inter-arrival time used was 10, meaning $\lambda = 0.1$. The packet size was $L = 10$ and server rate was $R = 1$, and the service time μ is equal to 0.1. The simulation was run with 100 000 packets and the max measured delay was 10. The simulated delay time distribution for this case is shown in figure 4.11a.

When the simulation was run with 10 sources, the inter-arrival times were generated using algorithm 3.1, `generate_inter_arrival_times(n, avg)`, with `n = 10` and `avg = 15`. These inputs result in the inter-arrival times of the sources becoming `[82, 97, 112, 127, 142, 157, 172, 187, 202, 217]`. The λ then becomes 0.073. Packet size L is still 10 and server rate R is still 1, μ is then still 0.1. The simulation was run with 10 000 packets per source, meaning 100 000 packets in total. The max measured delay of the system was 64. The simulated delay time distribution for this case is shown in figure 4.11b.



(a) 1 source **(b)** 10 sources

Figure 4.11: Simulated delay CCDF for $D/D/1$

4.7 $G/G/1 - priority$

For the $G/G/1 - priority$ model, both the arrival process and service time are generalised processes. There are multiple classes i present, for the following cases there will be three classes; 1, 2 and 3, present. First, a case with one source per class is investigated and then cases with multiple sources per class after this.

For $G/G/1 - priority$, there is no formula shown for finding $\overline{W}_i$ and it was assumed known in section 2.3.1. To be able to use the waiting time distribution introduced in equation 2.12, the average waiting time for each class is decided by the resulting average waiting time of each class i from the simulation.

4.7.1 One source per class

The first case investigated was with three classes and one source per class in the system. Table 4.7 shows the specific parameters per class. The packet sizes vary, the highest priority class consists of small packets, the next highest priority has big packets while the lowest priority level has a medium size. This type of priority could be implemented in a network by having small control message packets have the highest priority, big video call packets with latency requirements have the second highest priority and the remaining packets have the lowest priority.

Table 4.7: $G/G/1 - priority$ parameters for single source per class case

Class i	**Inter-arrival time distribution**	λ_i	**Service time distribution**	μ_i
1	Deterministic (Periodic)	0.033	Deterministic (Fixed)	0.100
2	Exponential	0.033	Exponential	0.200
3	Exponential	0.033	Exponential	0.143

The simulation was done with 1 000 000 packets per source, resulting in 3 000 000 packets. Simulating with the values in table 4.7 give the plot in figure 4.12 for the waiting time CCDF. The simulation calculated the following values for $\overline{W}_i$:

$$\overline{W}_1 = 2.23$$

$$\overline{W}_2 = 8.36$$

$$\overline{W}_3 = 19.97.$$

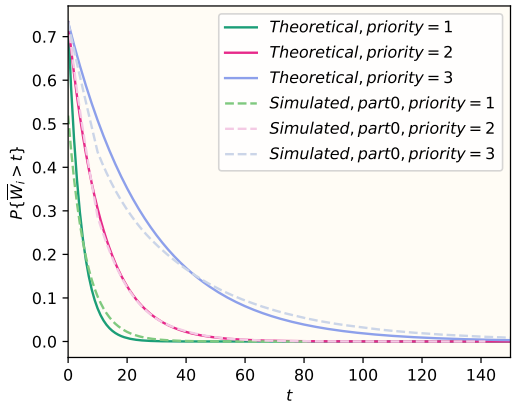


Figure 4.12: $G/G/1 - priority$ single source waiting time CCDF

4.7.2 Multiple sources per class

There are four different cases investigated for $G/G/1 - priority$ with multiple sources j per class i. All cases have three classes. All sources j for class i have the same arrival process and service time distribution, but parameters λ_j and μ_j may vary. There is a need to generate input parameters for both arrival and service time using algorithm 3.1 and 3.2. Table 4.8, 4.9, 4.10 and 4.11 give the inputs and the outputs for the cases investigated. The resulting lists of these tables are used as inputs to the simulator. The theoretical intensities can then be found using the formulas presented in section 3.3.1. The resulting intesities for all four cases are presented in table 4.12 (λ_i intensities) and 4.13 (μ_i intensities) with the arrival and service time distribution for the given class as well.

Table 4.8: Input and output for algorithm 3.1 with 10 sources

Class	`n`	`avg`	`generate_inter_arrival_times(n, avg)`
1	10	80	`[440, 520, 600, 680, 760, 840, 920, 1000, 1080, 1160]`
2	10	160	`[880, 1040, 1200, 1360, 1520, 1680, 1840, 2000, 2160, 2320]`
3	10	120	`[660, 780, 900, 1020, 1140, 1260, 1380, 1500, 1620, 1740]`

Table 4.9: Input and output for algorithm 3.2 with 10 sources

Class	`n`	`avg`	`generate_packet_sizes(n, avg)`
1	10	17.5	[13, 14, 15, 16, 17, 18, 19, 20, 21, 22]
2	10	17.5	[13, 14, 15, 16, 17, 18, 19, 20, 21, 22]
3	10	17.5	[13, 14, 15, 16, 17, 18, 19, 20, 21, 22]

Table 4.10: Input and output for algorithm 3.1 with 100 sources

Class	`n`	`avg`	`generate_inter_arrival_times(n, avg)`
1	100	80	[4040, 4120, 4200, 4280, 4360, ..., 11640, 11720, 11800, 11880, 11960]
2	100	160	[8080, 8240, 8400, 8560, 8720, ..., 23280, 23440, 23600, 23760, 23920]
3	100	120	[6060, 6180, 6300, 6420, 6540, ..., 17460, 17580, 17700, 17820, 17940]

Table 4.11: Input and output for algorithm 3.2 with 100 sources

Class	`n`	`avg`	`generate_packet_sizes(n, avg)`
1	100	17.5	[13, 14, ..., 21, 22, 13, 14, ..., 22, 13, ...]
2	100	17.5	[13, 14, ..., 21, 22, 13, 14, ..., 22, 13, ...]
3	100	17.5	[13, 14, ..., 21, 22, 13, 14, ..., 22, 13, ...]

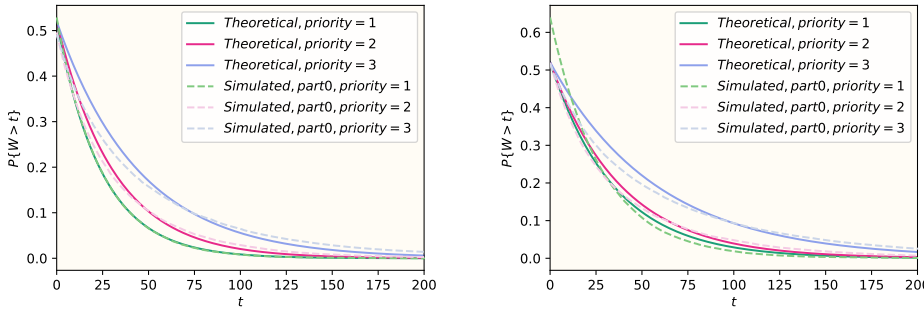
Table 4.12: λ_i intensities for $G/G/1-priority$ cases with multiple sources per class

Case	**Class 1**	λ_1	**Class 2**	λ_2	**Class 3**	λ_3
1	Deterministic (Periodic)	0.0137	Exponential	0.00685	Uniform	0.00914
2	Deterministic (Periodic)	0.0137	Exponential	0.00686	Uniform	0.00915
3	Deterministic (Periodic)	0.0137	Exponential	0.00685	Uniform	0.00914
4	Deterministic (Periodic)	0.0137	Exponential	0.00686	Uniform	0.00915

Table 4.13: μ_i intensities for $G/G/1 - priority$ cases with multiple sources per class

Case	**Class 1**	μ_1	**Class 2**	μ_2	**Class 3**	μ_3
1	Exponential	0.0571	Exponential	0.0571	Exponential	0.0571
2	Exponential	0.0571	Exponential	0.0571	Exponential	0.0571
3	Deterministic (Periodic)	0.0571	Exponential	0.0571	Uniform	0.0571
4	Deterministic (Periodic)	0.0571	Exponential	0.0571	Uniform	0.0571

The resulting plots of the simulations are given in figure 4.13 and 4.14. The simulations were run with 10 000 packets per source when there was 10 sources per class and 1000 packets per source when there was 100 sources per class. The calculated $\overline{W}_i$ for all classes of all cases are listed in table 4.14.

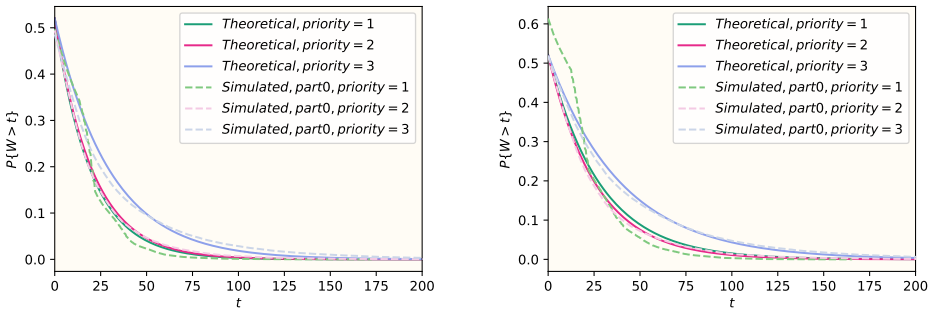


(a) 10 sources per class **(b)** 100 sources per class

Figure 4.13: Waiting time CCDF for $G/G/1 - priority$ cases with multiple sources with generalised arrival process and exponentially distributed service time

Table 4.14: Calculated $\overline{W}_i$ for $G/G/1 - priority$ cases

Case	$\overline{W}_1$	$\overline{W}_2$	$\overline{W}_3$
1	12.74	15.54	22.84
2	17.92	19.35	30.41
3	10.06	10.56	15.49
4	14.63	13.51	21.14



(a) 10 sources per class **(b)** 100 sources per class

Figure 4.14: Waiting time CCDF for $G/G/1-priority$ cases with multiple sources with generalized arrival process and generalised distributed service time

4.8 $M/G/1-priority$

The $M/G/1-priority$ model has an exponential arrival process and a generalised service time process. A case with one source per class and three classes in total is presented.

4.8.1 One source per class

The $M/G/1-priority$ case that was investigated was with one source per class and three classes. Table 4.15 summarises the information about the different classes and they roughly compare to the classes used for the case in section 4.7.1 with the same argument. Figure 4.15 shows the resulting CCDF for the waiting time distribution for this case.

Table 4.15: $M/G/1-priority$

Class i	**Inter-arrival time distribution**	λ_i	**Service time distribution**	μ_i	$\sigma^2_{s,i}$
1	Exponential	0.043	Deterministic (Fixed)	0.100	0
2	Exponential	0.043	Uniform	0.200	8.333
3	Exponential	0.043	Exponential	0.143	49

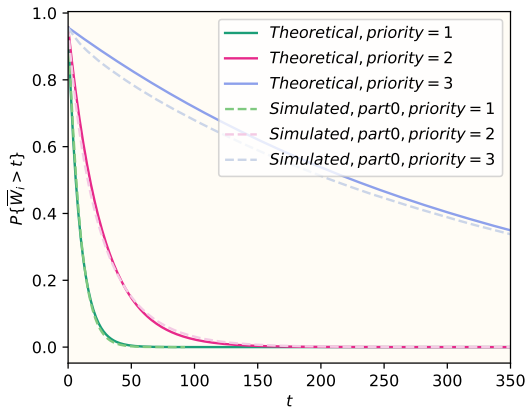


Figure 4.15: $M/G/1 - priority$ single source waiting time CCDF

Chapter 5
Discussion

The discussion chapter goes through the results of the models presented earlier and tries to discuss what the results are saying and how they can be used to decide the model's applicability to latency prediction.

5.1 Summary of findings

In general, there are two main findings to highlight from the results. The first is that when the input parameters to the system are changing, it takes some time before the simulated distributions line up with the theoretical distributions. The other main finding is that when there are multiple sources generating packets, the simulated distributions still line up with the theoretical ones.

5.2 $M/M/1$

The first cases that were investigated were the ones related to the $M/M/1$ model. When looking at the single source case seen in figure 4.1, there is an almost perfect alignment between the theoretical curve and the simulated resulting curve. This verifies that the simulator is working as intended as it is lining up with one of the most known theoretical models there are.

When adding multiple parts to the run and comparing packet $\#n$ of each run with the theoretical distribution for this, shown in figures 4.4 and 4.5, it is very clear that it takes some time before the simulated distribution line up with the theoretical distribution.

When only changing λ, the simulated curves are lining up with the theoretical ones at packet #100, but already very close at packet #10. The curve of part 2 of the run - when the system went from a medium load to a high load, is the curve that has the hardest time adjusting to the new theoretical distribution. However, for the

distributions of packet #0, no curves are lining up with the theoretical distributions. The curves are either "on their way up or down" to their supposed distribution.

When changing both λ and μ, the results in general appear similar to the results when only changing λ. By packet #100, the results are lining up well and part 2 of the run is taking the longest to adjust to the new load. Part 3 of the packet #0 distribution is also showcasing that when going from a high load to a low load, the packet distribution is closer to that of the high load than the low load. Both part 1 and part 2 are also much closer to the previous lower distribution than the new higher one.

To summarise what is observed when there are multiple parts to a run, the initial state of the system after changing parameters does not line up with any of the theoretical distributions. This is because reaching the steady state for the system takes some time.

The last investigated cases were having multiple sources generate packets, results shown in figure 4.6. For the first two cases with both exponentially distributed arrival and service time and results of the simulations are in both cases very similar in shape to the theoretical distribution. For the latter two, with deterministic (periodic) arrival processes for all sources and exponential service time, the results of the simulations are lining up less with the theoretical results than with exponential arrival processes for the sources. The simulation curve lines up better with 100 sources than with 10. The reason 10 sources are not lining up perfectly could be that 10 sources are not enough to create a Poisoon process and hence making the arrival process exponentially distributed. With 100 sources, the curves are lining up even better and the deterministic sources' arrival process can be approximated as a Poisson process with the same distribution as the theoretical $M/M/1$ models distribution.

Considering the theoretical formulas for the $M/M/1$ model presented in section 2.2.1 are exact results and not approximations there is an expectation that the results should line up well with simulated results and others, as have been seen in this work. The exception seen in this work is when the system still has not reached its steady state, which is reasonable based on what is known about the initial states of a system.

If there is reason to believe a system can be looked at as an $M/M/1$ system and there is a way to find the value for λ and μ, the $M/M/1$ model could be used for latency prediction.

5.3 $M/G/1$

There is only one theoretical formula related to the $M/G/1$ model presented in section 2.2.2. Since this is only the formula for the expected waiting time and

no distribution, the comparison that can be made between theoretical results and simulated results is limited to one number per case. This gives less ground to discuss upon. Table 4.4 shows that the theoretical expected waiting time is equal to 12.00 while the simulated average waiting time is 12.10. This means the results are lining up quite well with almost the same expected waiting time, but the theoretical result is a bit optimistic.

Since there is no theoretical curve to compare the simulation's waiting time distribution to (shown in figure 4.7), there is no real point in discussing its shape. It does however follow a similar curve as all other simulated plots presented which at least verifies the simulation is most likely working as intended in this case as well.

5.4 $M/D/1$

For the $M/D/1$ model, the investigated cases were having one source and multiple sources. For the single source case (figure 4.8), the theoretical curve is more optimistic than the simulated results. This is also the case when multiple sources are present in the system (figure 4.9). The best correlating results are surprisingly when there are 10 sources with deterministic arrival processes (figure 4.9c). As was seen from the $M/M/1$ case with 10 sources and deterministic arrival, this is not acting as a Poisson process (not enough sources) and should then in theory not be a good match.

The waiting time distribution function for $M/D/1$ (equation 2.8) is too optimistic compared to the simulated results, meaning the theoretical results predict less waiting time in the system than what is actually appearing in the simulated system.

Considering the equations used for the theoretical curve, which is not an approximated result, it is surprising that the results are not closer in correlation. As the same difference in results is seen in all but one result, it is fair to assume that with the given values used for this, the correlation is not as well as would have been needed to be able to use this model for latency prediction.

5.5 $G/G/1$

For the case related to the $G/G/1$ model shown in figure 4.10, the results of the simulation and the theoretical formula are lining up to some extent. The curves of the two are shaped differently. Considering that both the arrival process and the service time distribution are considered generalised, the correlation is better than what could have been feared since there are close to no rules for what the simulator could use as inputs. Another case could give a worse correlation between the two.

The theoretical formulas for the $G/G/1$ model presented in section 2.2.4 are approximated results. This means there should not be an expectation that the results correlate perfectly with actual or simulated results. The reason for the simulated results correlating as well as they do in figure 4.10 could be as both the arrival and service time distribution are uniformly distributed. This is a well-known distribution and quite a predictable one. Perhaps changing the arrival process and service time distribution to other distributions or changing the distribution per packet or after a given period in the simulator would give a more unpredictable behaviour. This would also make it harder to know the theoretical values for λ, σ_a^2, μ and σ_s^2 and hence find the theoretical approximations for the system for comparison.

5.6 $D/D/1$

There were two cases that were investigated for the $D/D/1$ model; single source and multiple sources. The results were shown in table 4.6. The bounds of both cases are held, the single source hitting the max while the multiple sources case did not. The only way the max of the multiple sources bound could have been hit is if all sources generated a packet at the exact same time, meaning the last packet would have had to wait for the max time possible. Since algorithm 3.1 was used to create different inter-arrival times, this was not the case for this case.

As discussed earlier, when there are enough deterministic (periodic) arrival processes, the arrival process may be approximated as a Poisson process with exponentially distributed inter-arrival time. This means that the case with multiple sources could also be compared to that of the $M/D/1$ model's theoretical results.

5.7 $G/G/1 - priority$

The first cases with multiple classes present that were investigated were the cases that line up with the $G/G/1 - priority$ formula. Firstly the case with one source per class was looked into with the results in figure 4.12. The resulting curves of the simulation line up well with the theoretical results with some slight curve differences, especially for the lowest priority class.

The two next cases investigated were with multiple sources per class with generalised arrival and exponentially distributed service times seen in figure 4.13. The curves of the results are lining up well with some slight differences in shape.

Moving on to the last two cases with generalised arrival and service times seen in figure 4.14, the results are lining up less than for the latter cases. This could be because the processes are even more generalised in these cases as the service time distributions also differ between the classes.

Considering that equation 2.12 assumes $\overline{W}_i$ to be known and this value is pulled from the simulation, there should be a reason to believe the results should line up. This also means that the theoretical results cannot be viewed as fully theoretical. It is also an approximated result. Using this value directly from the simulation should increase the odds of getting two plots that look similar, as they in general do.

The $G/G/1 - priority$ model seems to be applicable for latency guarantee estimation as long as there is a possibility to gather the theoretical values.

5.8 $M/G/1 - priority$

For the single source per class case of the $M/G/1 - priority$ model (with results shown in 4.15), the results are lining up well with similar curves. The two highest-priority classes are lining up better than the lowest-priority one. The results are lining up better than for the $G/G/1 - priority$ cases. A reason for this could be that there are stricter rules when it comes to the arrival process and hence this is easier to predict than for the previous case.

Equation 2.14 gave an approximation for the waiting time distribution. Since it is an approximation and not an exact result for the theoretical results, the results will never be perfect. The theoretical formula is the same as for $G/G/1 - priority$ other than calculating $\overline{W}_i$ based on parameters instead of pulling the values from the simulation. This means that if this model was presumed to be a good fit for a system, more parameters would have to be known to find the theoretical distribution of the system.

In general, the $M/G/1 - priority$ system is providing results that makes it seem applicable for latency guarantee estimation.

5.9 Limitations and finding variables of real systems

There are some limitations related to the results of this thesis. Since the simulator was verified using some of the formulas from the analytical models and no other form of verification, there is no way to fully prove the simulator is working as a real part of a network. Simulated results are in general good and close to real results, but will never catch every side and everything that could happen in a real system.

Because of this, the results and comparisons can never be used completely by themselves for deciding if the models are applicable for latency guarantee estimation. Some other form of verification of these results should be added. Like testing on an actual network and on even more versions of the models. Section 6.2 gives some further comments on this.

A problem that has been touched upon in earlier sections of this chapter is the fact that there is a need to find the parameters of the real systems where the latency guarantee is needed. As this thesis has been using simulation results, the expected values of the parameters for the theoretical results have not been a problem to determine. It will however in other more realistic cases become a problem and if there is no way of finding the values of the parameters, there is no point in using analytical models for latency prediction and for ensuring guarantees are kept.

Chapter 6
Conclusion and future work

This chapter provides a conclusion to the thesis and presents further work that could be done in the future to provide more insights on this topic.

6.1 Conclusion

This thesis first reviewed analytical models through a literature study to gather exact and approximated results that would later be compared to the results of the constructed simulator.

The constructed simulator and how it is set up were then presented. The focus of the presentation was on showing what types of inputs the simulator can handle and how the resulting output of the simulator is constructed.

There are two main findings of this work. The first is that when the load on the system is changing there is a need to be careful with using analytical models for latency predictions as it will take some time before the system is stable and hence act as the theoretical model is predicting. The second finding is that when there are multiple sources generating packets, the arrival process may be approximated as a Poisson process with exponentially distributed inter-arrival time. This means it is still able to be compared to the analytical models as long as there are enough sources creating packets if the sources themselves do not have an exponential arrival process.

A lot of the analytical models investigated provides results that are comparing well to the results of the simulations. This gives the impression that analytical models can be used for latency prediction and guarantees and are still applicable. There are however some analytical models that are giving too optimistic results; $M/D/1$ and $G/G/1$. Based on the cases studied in this thesis, these would not be applicable for providing latency prediction and guarantees.

Even though the other analytical models have results that line up well, and hence

they are usable for latency prediction and guarantee, there is still a need to act with caution as the arrival rates and packet sizes may change over time. If the parameters suddenly have a big shift, neither the latter nor new intensities provide a correct comparison. It is hard to decide the distribution of the waiting time and delay when the system is in this transient stage. The bigger the differences in load, the longer time it takes before the new distribution of the waiting time and delay are valid.

When there is priority queueing happening in the system, there are multiple plots that all have to match up well for the models to be considered applicable for latency prediction. Both priority models investigated are doing well for all classes of the system meaning they are applicable for latency prediction and guarantees. When there are some differences, the models appear a bit pessimistic, so if these were to hold the system should still keep the requirements.

This thesis aimed to review analytical models for latency guarantee estimation and compare these results to the result of a constructed simulator. It concludes that most models investigated are applicable for latency guarantee estimation.

6.2 Future work

When it comes to future work, multiple groups of use cases could be investigated. All use cases are applicable for all analytical models investigated in this thesis and others that were not considered in this thesis, for instance, other analytical models based on network calculus, queueing theory or others.

One case that could be investigated is to look into non-stationary results (transient solution) and if approximations of these can be used when the system is in a state of changing the load.

It could also be interesting to compare analytical models to other results like real data instead of simulated data. This could also help confirm if the simulator results are good and further provide another argument for why analytical models can or cannot be used for latency guarantee estimation.

Another use case that can be investigated could be having multiple sources per run and the run having multiple parts. This could lead to other results that can either confirm what is already discovered or highlight either the good or bad sides of the models.

This thesis has not considered having multiple parts to a run with priority queueing, this use case would be useful to see if the classes are affected in the same or different way. The intensities could either all change at the same time or just for one class or source at a time.

It could also be interesting to see if the cases with multiple classes would be affected by the number of classes present in the system. Investigating with for instance 5, 10 and 20 sources could be interesting to see if a pattern is occurring.

For most use cases, there has only been one set of input parameters used. Another use case that would be interesting to see the results of is if the amount of load on the system will affect how well the results are lining up.

References

[1] ITU, «The Tactile Internet. ITU-T Technology Watch Report», Aug. 2014.

[2] R. Ali, Y. B. Zikria, *et al.*, «URLLC for 5G and Beyond: Requirements, Enabling Incumbent Technologies and Network Intelligence», *IEEE Access*, vol. 9, pp. 67 064–67 095, 2021.

[3] J. Navarro-Ortiz, P. Romero-Diaz, *et al.*, «A Survey on 5G Usage Scenarios and Traffic Models», *IEEE Communications Surveys Tutorials*, vol. 22, no. 2, pp. 905–929, 2020.

[4] FG-NET-2030, «White Paper: Network 2030: A Blueprint of Technology, Applications, and Market Drivers Toward the Year 2030», Nov. 2019.

[5] Q. Zhang and F. H. P. Fitzek, «Mission Critical IoT Communication in 5G», in *Future Access Enablers for Ubiquitous and Intelligent Infrastructures*, V. Atanasovski and A. Leon-Garcia, Eds., Cham: Springer International Publishing, 2015, pp. 35–41.

[6] T. S. Perry, «Look Out for Apple's AR Glasses: With head-up displays, cameras, inertial sensors, and lidar on board, Apple's augmented-reality glasses could redefine wearables», *IEEE Spectrum*, vol. 58, no. 1, pp. 26–54, 2021.

[7] K. Kim, M. Billinghurst, *et al.*, «Revisiting Trends in Augmented Reality Research: A Review of the 2nd Decade of ISMAR (2008–2017)», *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 11, pp. 2947–2962, 2018.

[8] Y. Jiang, C.-K. Tham, and C.-C. Ko, «An approximation for waiting time tail probabilities in multiclass systems», *IEEE Communications Letters*, vol. 5, no. 4, pp. 175–177, 2001.

[9] P. J. Emstad, P. E. Heegaard, *et al.*, *Dependability and performance in information and communication systems. Fundamentals*. Department of Telematics, 2016.

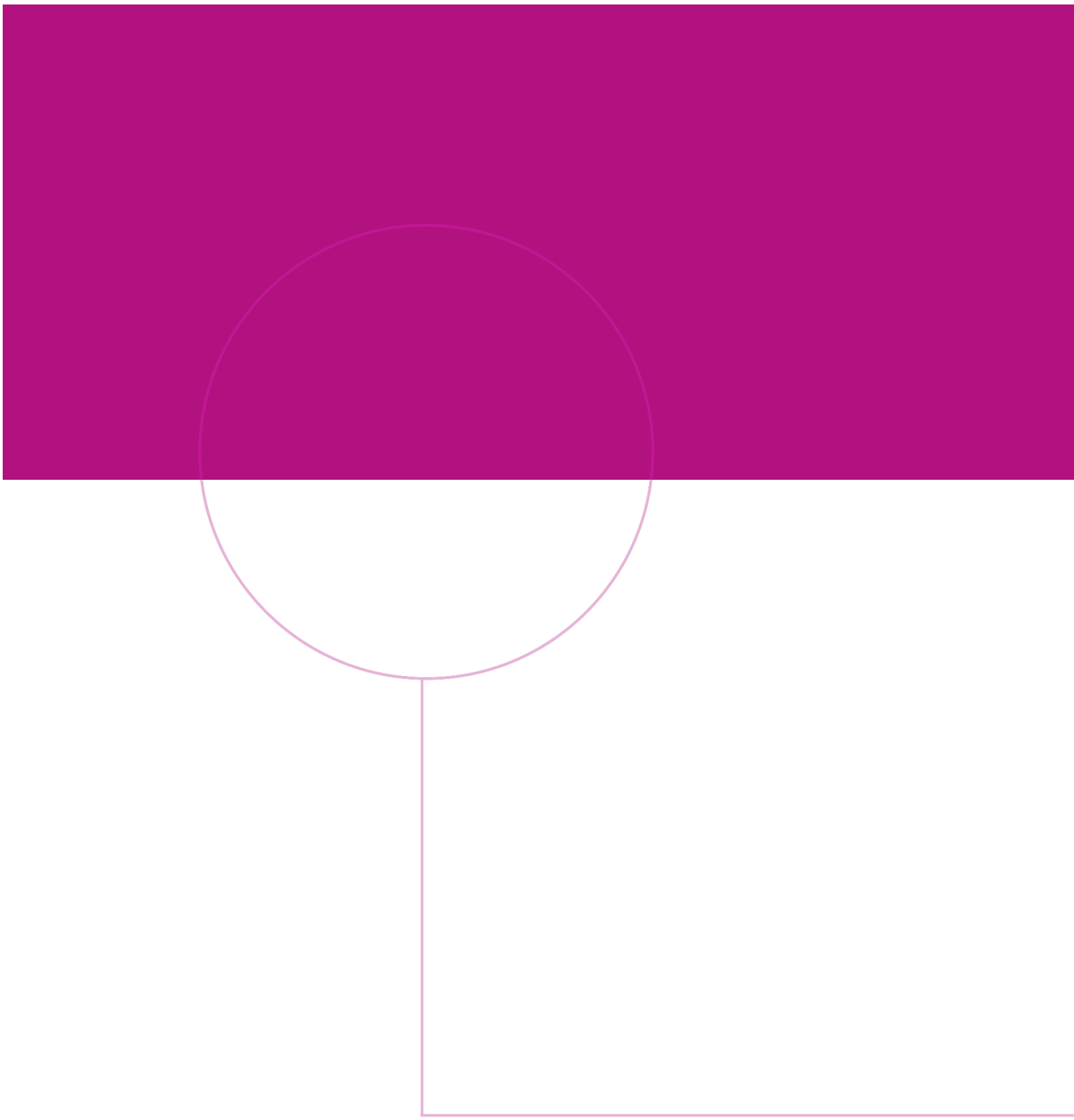
[10] M. Zukerman, *Introduction to Queueing Theory and Stochastic Teletraffic Models*, 2020.

[11] Y. Jiang, «Queueing Delay Models», May 2023.

[12] Team SimPy. «SimPy, Discrete event simulation for Python». (2020), [Online]. Available: https://simpy.readthedocs.io (last visited: Jun. 7, 2023).

[13] NumPy. «NumPy, The fundamental package for scientific computing with Python». (2023), [Online]. Available: https://numpy.org/ (last visited: Jun. 7, 2023).

[14] The Matplotlib development team. «Matplotlib, Visualization with Python». (2023), [Online]. Available: https://matplotlib.org/ (last visited: Jun. 7, 2023).



NTNU

Norwegian University of
Science and Technology