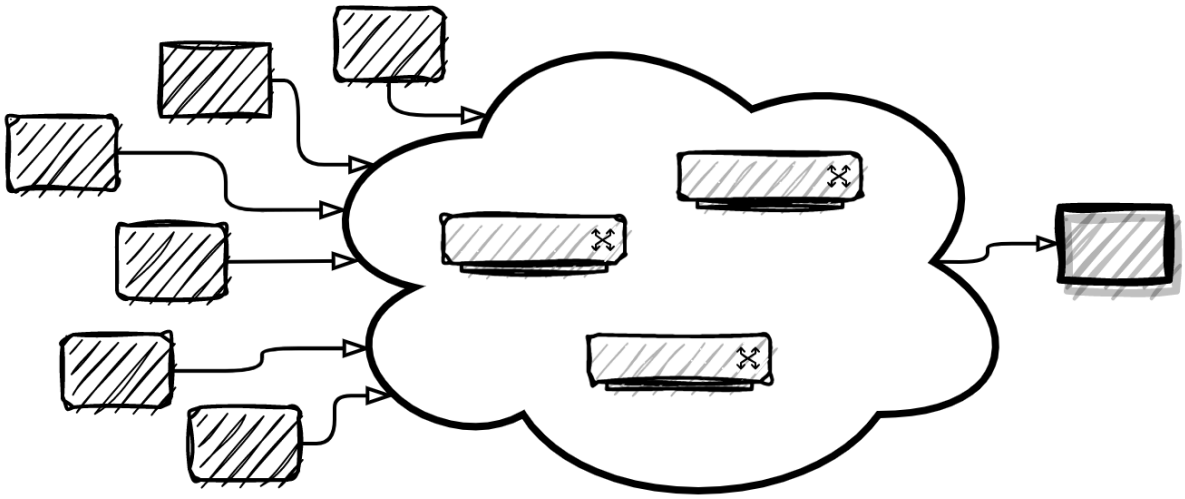

Dimensioning Packet Based Fronthaul for Radio Base Stations: Network Requirements and Traffic Scenarios

Ethernet Network Simulator

Master's thesis in Engineering Mathematics and Computational Science

JIMMY ABRAHAM

DEPARTMENT OF MATHEMATICAL SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2022
www.chalmers.se

Master's thesis 2022

Dimensioning Packet Based Fronthaul for Radio Base Stations: Network Requirements and Traffic Scenarios

Ethernet Network Simulator

JIMMY ABRAHAM

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Division of Applied Mathematics and Statistics
Chalmers University of Technology
Gothenburg, Sweden 2022

Dimensioning Packet Based Fronthaul for Radio Base Stations: Network Requirements and Traffic Scenarios
Ethernet Network Simulator
JIMMY ABRAHAM

© JIMMY ABRAHAM, 2022.

Supervisors: Krister Bergh & Peter R. Karlsson, Ericsson AB
Examiner: Ottmar Cronie, Department of Mathematical Sciences

Master's Thesis 2022
Department of Mathematical Sciences
Division of Applied Mathematics and Statistics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Authors figure.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2022

Dimensioning Packet Based Fronthaul for Radio Base Stations: Network Requirements and Traffic Scenarios
Ethernet Network Simulator
JIMMY ABRAHAM
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

A radio base station is a wireless communication station which is used for wireless telephone systems such as GSM (2G), WCDMA (3G), LTE (4G) and NR (5G). Today, radio base stations use circuit switched fronthaul and will continue to do so for a foreseeable future, however packet based fronthaul will grow as 5G development does as well. A radio base station for packet based communication consists of three main units: radio, baseband and switch. The aim of the thesis was to implement a simulator for packet based fronthaul and determine optimal network configurations for a given traffic load through statistical methods.
First the radio base stations were defined mathematically as a small network through graph theory. Then traffic generation was characterized as a two state Markov chain. Algorithms were designed for both network and traffic generation. Packets inside ethernet networks follow protocols which were simplified and adapted for radio base stations. An algorithm was designed for simulating the packets movement in the network. The simulator was implemented in Python and was evaluated through statistical methods to determine the amount of time steps and iterations that is required to have consistent simulation results. Two-sided one sample t-test showed that five iterations for each simulation was enough to generate samples close enough to the expected mean and 10000 time steps was shown to be enough for a low standard deviation.
Finally, data was generated with the simulator for selected combinations of network properties. Selected properties were number of radio units, buffer size, number of switches, and traffic load. A Poisson regression model was used to correlate properties of a given network with the amount of packet loss as response variable. Results show that the D^2 score is about 0.8 and prove that the model works with a reasonably good score. This also shows that with more time and development a more advanced analysis can be made, and better machine learning models can be made with the help of the Ethernet Simulator.

Keywords: Packets, Ethernet Frames, eCPRI, Telecommunication, Radio Base Station, RAN, 5G, Machine learning, Poisson Regression GLM, Fronthaul.

Acknowledgements

Firstly, I would like to express my gratitude and utmost thanks to my two supervisors Krister Bergh and Peter Karlsson at Ericsson for all their help, guidance, and continued support. They have been integral for the work of this thesis and their comments have helped immensely to improve the quality of this thesis. I would also like to thank Ericsson for giving me the opportunity and trust to pursue this thesis and providing the necessary tools needed.

Secondly, I would like to thank my examiner Ottmar Cronie who took upon this thesis work and who gave valuable inputs to make sure the thesis was up to standard.

Jimmy Abraham, Gothenburg, June 2022

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

BBU	Base Band Unit
CPRI	Common Public Radio Interface
C&M	Control and Management
eCPRI	enhanced Common Public Radio Interface
GSM	Global System for Mobile Communication, 2G
GLM	Generalised Linear Method
LTE	Long Term Evolution, 4G
NR	New Radio, 5G
PDF	Probability Density Function
UE	User Equipment
RAN	Radio Access Network
RBS	Radio Base Station
RU	Radio Unit
SFP	Small Form-factor Pluggable
WCDMA	Wideband Code-Division Multiple Access, 3G

Contents

List of Figures

List of Tables

1
Introduction

In the following sections the background, aim, limitations and outline are described. In the background section, fronthaul and radio base stations are explained as well as what role packet based communication has in this context. The aim explains what the thesis intended outcome will be and limitations show what won't be considered. Lastly, the outline of the thesis is presented.

1.1 Background

Ericsson is one of the leading providers of Information and Communication Technology to service providers. One department at Ericsson, Lindholmen has its focus to develop the software for configuration and supervision of Radio base stations (RBS). An RBS is a wireless communication station used for wireless telephone systems such as GSM (2G), WCDMA (3G), LTE (4G) and NR (5G). The smallest RBS installation consists of one Baseband Unit (BBU) and one or a few Radio Units (RU). In larger forms an RBS can contain several of both units where the size and complexity can vary greatly for different RBS installations. The user data transferred between user equipment and backhaul[1] (as well as the reverse direction) is encoded in a carrier transmitted in the physical sector where the user equipment is located at a specific time. BBUs and RUs communicate with each other through two different types of Fronthaul: Common Public Radio Interface (CPRI) or enhanced CPRI (eCPRI)[2].

1.1.1 CPRI

CPRI [3] is a circuit switched communication standard where the required amount of resources (bandwidth on the physical links) are static for a carrier. During configuration of the carriers, it is possible to determine if there is enough available capacity on each physical link, the capacity requirement is based on each networks peak utilization. There are pros and cons with static capacity allocation: When configuring new systems, it can be determined if there is enough resource available straight away as well as that the new carrier will not affect anything already configured in the system. It also allows very high constant throughput of data. However, it requires the same capacity whether or not there is a payload at any time, e.g. if

[1] The backbone of core network, what connects it to the rest of the world

[2] There are other types of fronthaul as well but these two are the ones supported by Ericsson and in the scope of this thesis.

there are three RUs in an RBS then all three will be allocating the same amount of resources even though they do not utilize it all at the same time. In Figure 1.1 an RBS with a CPRI system is shown with the allocated resources in the graph.

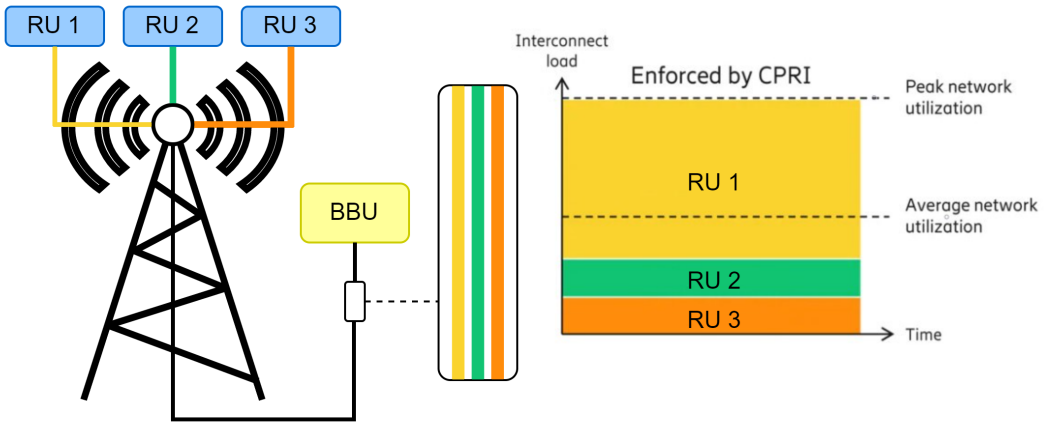

Figure 1.1: A CPRI network with one BBU and three RUs. Here each link to each RU is displayed to have constant utilization. Figure from [3]

1.1.2 eCPRI

eCPRI [3] is a packet switched communication standard. Here the communication is based on ethernet packets between the units in the RBS. In this case the required interconnect capacity for a carrier depends on the actual traffic load (amount of user data) at a specific time. This means that the required capacity is much smaller in low traffic scenarios compared to high traffic scenarios. This can be exploited to reduce the dimensioning of the physical link capacity by for instance using statistical multiplexing if several radios are sharing a link. Peak capacity can occur on rare occasions so using statistical multiplexing means that the total allocation is less than peak and would therefore reduce total cost of the system. This means that statistically there is a low probability that all capacity in the system is utilized at any given time. A packet based system would therefore allow for a more dynamic and cheaper system, here one RU could use all capacity at certain points in time while still allowing three RU's to be used simultaneously at a different time if they don't use peak capacity. In Figure 1.2 an RBS with an eCPRI system is shown, here the allocated resources are dynamic and differs greatly over time depending on throughput.

The trend is toward eCPRI and its dynamic capacity based packet system even though CPRI will be around for many years. This is confirmed in a whitepaper produced by Ericsson where eCPRI is referred to as "...the logical evolution of CPRI towards a new de-facto industry standard." [1].

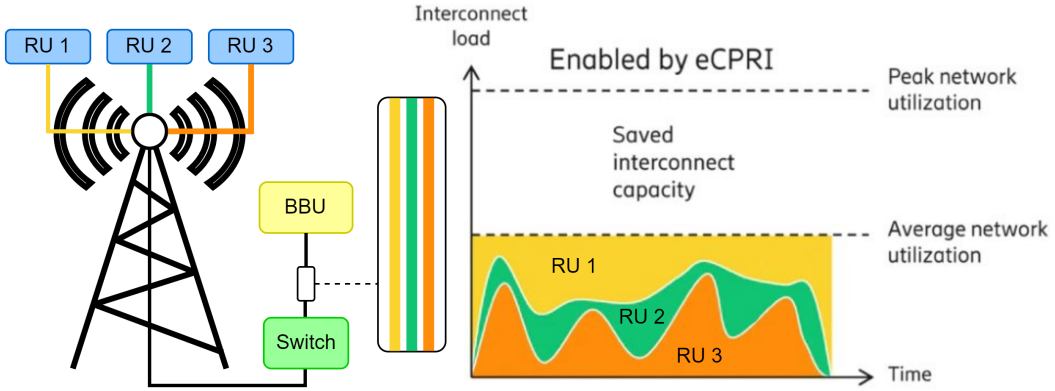

Figure 1.2: An eCPRI network with one BBU, one switch and three RUs. Here each link to each RU is shown to have variable utilization. Figure from [3]

1.2 Aim

The objective is to investigate how ethernet traffic is impacted by network configuration and traffic scenarios by implementing a simulator. Here a network configuration consists of 2 switches, up to 10 endpoints (an endpoint is either a BBU or a RU) and up to 50 traffic flows. A mathematical model will be formulated through statistics and machine learning which characterizes packet based communication.
The aim of the thesis is to investigate how the characteristics of packet based communication in an eCPRI based fronthaul depend on network properties as well as traffic scenarios. The thesis is divided in to three main parts:

1. Mathematically define a packet based network which fulfills telecommunication performance requirements.
2. Implement an ethernet network simulator where the behavior of small network configurations can be investigated for different traffic scenarios using a suitable open-source language.
3. Formulate and evaluate a model which aims to optimize a network for packet based communication through statistics and machine learning.

1.3 Limitations

To reduce the scope of the thesis and narrow down what will be investigated some limitations are set, the following list summarizes what will not be investigated or developed:

- Network topologies and traffic models will be defined as simple models, no consideration will be taken to complex internal hardware structure of any RUs or BBUs beside simple mathematical models.
- Traffic will be considered and formulated as either low, medium or high.
- No focus will be done on the user interface of the simulator.

- In an RBS there can be combinations of CPRI and eCPRI, however this thesis will only[3] be looking at an eCPRI fronthaul.
- Backhaul will not be considered.

1.4 Outline

In section 2, the theory which is needed for the thesis will be explained. First, hardware and protocols for the different parts of an RBS is summarized and described in an abstract level for what is important when creating a simulator. Then the relevant parts of network flow, Graph theory and Markov chains are described. Algorithms, queuing theory, one sample t-test and finally Poisson regression are described as well.
In section 3, the method of each part is outlined, in total there are four main parts: Investigation, the traffic generator, network model and, the Ethernet simulator.

- In the investigation each part of the hardware is described and any necessary simplifications for the simulator are explained.
- The traffic generator first shows how it is based on Markov chains and the stationary distribution is also shown.
- Then the mathematical abstraction of an RBS is described for different types of networks, here graph theory and optimization is utilized. The different types of networks are defined and shown with figures.
- Lastly the simulation algorithm is shown and described, here a final summary of the programming language and some context for the implementation is shown.

In section 4, results and discussion is presented. It begins with evaluating different parts of the simulator. First a simple mathematical model which is calculated and compared to the simulator to confirm it functions properly. Then the traffic generator is used to determine the different number of iterations and time steps needed to generate trustworthy data. Data is generated, then a Poisson regression model is used to show linearly dependence for different networks switch buffer sizes. In each section a discussion is shown for results given.
In section 5, further research is suggested. Here suggestions and comments for what could have been investigated further if time had permitted is shown and a list of different concrete examples which can be further examined.
Finally, in section 6 a conclusion for all results is summarized and connected to the aim stated in the introduction.

[3]Since CPRI and eCPRI parts of the fronthaul are independent, the work is still applicable for RBSs having both types.

2
Theory

In this chapter the relevant theory and important specifications which are needed to give an understanding of how to interpret the results are shown. It begins with a short description of the hardware and specifications of the different parts of a radio base station, eCPRI is described and relevant parts of the specification is presented. Then network flow in graphs will be introduced as well as Markov time chains. After that a short summary of what algorithms and queuing theory is shown. One sample t-test is explained and finally the Poisson regression theory is presented.

2.1 Packet, Hardware and eCPRI

In this section network packets, the radio base station and relevant parts of the eCPRI specification are outlined.

2.1.1 Network Packet and Ethernet frame

A network packet is a segmented part of a larger file or message. When data is sent over a network it is divided into smaller parts, i.e., packets. Dividing files into packets makes it possible for several computers or devices to communicate at the same time. If the files wouldn't be divided, then for the duration of sending a file no other device could communicate during this time on the network. Each packet contain, information where it should go and during transmission the packets can be seamlessly interwoven with each other and utilize the network to its maximum. There are several different types of packets depending on which protocol is being used. In this thesis the Ethernet protocol is used which is a technology for computer networking where the most common use are for: local area networks (LAN), metropolitan area networks (MAN) and wide area networks (WAN) [2]. Communication for Ethernet is done with ethernet frames [12]. The main differences between different protocols are the sizes of the payload and how the packet is structured. Almost all protocol standards divide the packets into two parts, the header, and the payload. A standard ethernet frame contains the following components, listed below:

- Preamble
- Source address
- Destination address
- Payload size
  - Can vary between: 46-1500 bytes

Everything but the payload is 18 bytes large so the total size of a standard ethernet frame with its components is between 64-1518 bytes. Source and destination addresses are important for obvious reasons. From here on ethernet frames and packets will be used interchangeably. There are two important aspects which need to be considered, packet loss and packet priority:

1. Packet loss is what happens when a packet is lost, i.e., removed from the network. A packet can be removed depending on the following reasons:
   - It was too late to its destination (one-way delay).
   - It was corrupt (something wrong happened when sent or during transmission towards its final destination).
   - A link or node towards the destination can become full and therefore no space is left and it will be discarded.

   If any of the above incidents occur the packet is thrown, i.e., removed from the system. Link capacity, node buffers and one-way delay will be discussed further in chapter 2.1.2.1, 2.1.2.2 and 2.1.3 respectively.
2. Packet priority is how prioritized a packet is inside a node, i.e., if there are several packets arriving at the same time the priority is used to decide which one will be sent first. There are a multitude of different priorities in real-life applications however for this thesis only transmission time and arrival time will be considered.

2.1.1.1 Packet Switched Communication

Packet switching means that a packet can take any route from A (its source) to B (its destination). There are no pre-determined paths. However there will always be a shortest path (several[1] in some cases) to take from A to B. In case of high traffic, the shortest path can be fully utilized at the moment a packet is sent, the packet can then take another route to its destination. If the path is short enough the packet will arrive in time and not result in packet loss. Movement of packets over networks can also be called flow of packets over the network.

2.1.2 Radio Base Station

A radio network is the network which you connect your phone to through e.g., 3G or 4G, these networks are called RAN or Radio Access Network. A RAN is a large network which can be placed in a city for example. A RAN consists of many RBSes. The RBSes are units connected, the part studied in this thesis is one of these networks of interconnected units called the fronthaul. The basic hardware of an RBS is one Baseband unit, one switch, and one or several Radio units (an RBS can contain more than these components but for the sake of the thesis these other types of units are not important).
The following scenario describes and explains how an RBS is realized. It shows how traffic moves through the system of an RBS:

- A phone is connected to internet and starts an app which wants to connect to a webserver. Data-traffic is first sent through the air to the RU (this part

[1] There can be more than one path with same length and capacity.

is not interesting for the thesis). The RU then transmits the traffic to the switch which in turn switches the traffic to a BBU. The BBU then processes the traffic and sends it to the RANs core which in turn sends the traffic to the app's webserver. Any traffic sent from the webserver is then sent through the same steps in reverse. Sending traffic to the core is called uplink traffic and traffic going to the phone is called downlink traffic.

The up and downlink traffic inside an RBS can be seen as independent[2] of each other in the scope of this study. Uplink and downlink inside an RBS can therefore be seen as two separate systems. Furthermore, in this thesis a RU, switch or BBU can in most circumstances be regarded as equivalent and instead of writing all names explicitly they can be referred to as a node or several nodes.

An RBS can be simple and consist of just a few nodes or be a massive installation of up to hundreds of nodes. The topology of the network can vary depending on switches and connections. Different RBSs can also share nodes between each other, and they can share the switches of the fronthaul. This together ends up in a very challenging situation when dimensioning the fronthaul network.

For this thesis an RBS will reflect the topology of one or more BBUs, one or two switches and one or more RU's. Below in Figure 2.1 one simple radio base station is illustrated with one switch, one BBU and two radio units which are connected to a mobile phone each.

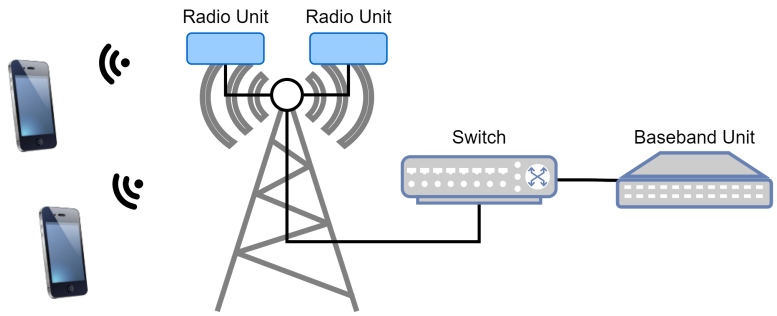

Figure 2.1: Radio base station shown with two RUs, one switch, and one BBU.

In real applications both CPRI and eCPRI can be present in the same system, however the scope of this thesis will only cover eCPRI networks.

2.1.2.1 Link speed

Between BBU, switch and RU there are links which can be either radio waves or physical (electrical or optical) cables. This thesis will only be describing and analyzing physical links, the word 'physical' will be omitted henceforth. A link contains three different parts:

- An electrical or optical transmitter.
- An electrical cable or optical fiber (the same cable or fiber is usually used for sending data in both directions).

[2]In the RAN software stack there is dependencies between the payload of uplink and downlink data, but this can be ignored when analyzing the fronthaul network as this thesis does.

- An electrical or optical receiver.

The transmitter and receiver are often mounted in a component called Small Form-factor Pluggable (SFP) which is connected to the BBU, switch and radio. The prices for SFPs are about the same in the 10-50Gbps range however the price for a 100Gbps is about five times [15] the amount of the other slower ones. The required link speed is dependent on the RU it is connected to. Having higher link speed than what the RU can handle makes the link unnecessarily expensive or any lower link speed is not making full use of the RU. Which leads to the following conclusion: the RU and connected link have the same speed capacity.
The capacity of an RU also has higher cost for higher throughput. But a simplification for this thesis have been done and generalized that higher speeds are much more costly than lower speeds of the links. Only a few different speeds are considered, these are listed in the Table 2.1 below.

Link speed [Gbps]
10
25
50
100

Table 2.1: List of cables for different speeds.

The links considered in this thesis are optical and here the speed of light travels at 2/3 of its speed in vacuum.

2.1.2.2 Switch Buffers

A buffer is defined as something to reduce shock of fluctuation, here it is against the possibility of having to throw away packets. Every RU, switch and BBU in the system have the possibility to buffer packets in case the outgoing link does not have the capacity to transmit all packets at once. Mainly it is the switch which will buffer packets, however the possibility exists that the RU or BBU before a switch will have to buffer packets as well. The buffer size is similarly as link capacity, a question of price, the bigger the buffer the more costly it is. The size of the buffer is however not the only issue, storing packets for too long will result in packet loss discussed in 2.1.1. There are two different types of traffic:

1. Best effort. In this case, the arrival time of the traffic is of small importance. The user can accept that it takes some time to download a large file.
2. Real time traffic. In this case, it is important that the data arrives in a timely manner. Watching a movie where the picture is freezing all the time is annoying. However, there are even more critical situations like if you are remotely controlling a machine or vehicle. Here it can be a question of life and death that the information is received at the correct time.

When it comes to buffer sizes of the different nodes (RU, BBU and switch) according to ESnet [13]: "The general rule of thumb is that you need 50ms of line-rate output queue buffer, so for for a 10G switch, there should be around 60MB of buffer. This

is particularly important if you have a 10G host sending to a 1G host across the WAN". Which translates to approximately:

$$\frac{60MB}{1518B} = \frac{60 * 10^6}{1518} \approx 39525\, packets$$

However, looking at a few different switches [14] the range is anything between 8MB to 128MB. Which translates to approximately 5270 to 84321 packets. Note that these numbers are viable for a general computer network. A general computer network is traditionally used for best effort traffic, moving non-timing critical data between different nodes. So, for best effort traffic high delay is acceptable, however in real time traffic low delay can be critical depending on the application. Delay times over networks will be discussed in next chapter 2.1.3.

2.1.3 Enhanced Common Public Radio Interface

eCPRI [3] enables the packets to take any route to its destination in similarity to the Ethernet protocol. The packets can also utilize the full capacity of links compared to CPRI. There are some requirements in eCPRI which needs to be adhered to, for this thesis the main one is the delay from sender to receiver called a One-Way delay measurement and is defined as:

$$t_D = (t_2 - t_{CV2}) - (t_1 + t_{CV1}) \tag{2.1}$$

Here t_2 is the arrival time at its destination and t_1 is the time it was sent from its source. As an abstraction t_{CV1} and t_{CV2} can be seen as the internal delay time it takes for data to move inside the receiving and sending node respectively. The values for some latencies for the one-way delay measurement from high to low is defined in Table 2.2. High also has several of its own classes which are seen in the same table classified as high25-high500.

Table 2.2: Maximum One-Way delay definitions for different use cases [3].

Latency	Max One-way delay	Use case
High25	25 μs	Ultra-low
High100	100 μs	NR performance
High200	200 μs	Fiber lengths of 40 km
High500	500 μs	Large latency installations
Medium	1 ms	User case (slow) C & M plane (fast)
Low	100 ms	C & M plane

The larger delays in 2.2 above (medium and low) are viable for a general computer network using best effort traffic, they are kept here as a point of reference.

2.1.3.1 ON/OFF traffic

The traffic in eCPRI follows an ON/OFF pattern, it either sends all traffic inside a time frame and then nothing is sent until next ON/OFF period. In the following

Figure 2.2 the ON/OFF traffic pattern is displayed.

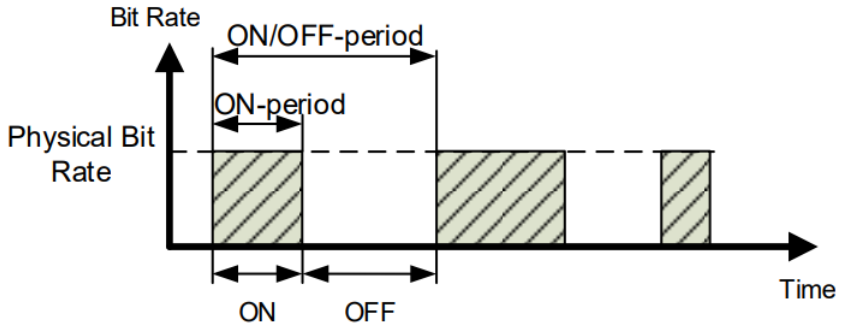

Figure 2.2: ON/OFF traffic for eCPRI. Image comes from CPRI.info [3]

2.1.3.2 First In First Out

There are different ways of prioritizing packets in an eCPRI system, the most common and simplest one is First In First Out (FIFO). It works by always prioritizing the first arrival to be sent next, i.e., the packet which arrived first is sent first.

2.2 Network flow in graphs

To model an RBS mathematically as a network, graph theory is utilized. Graphs are a way to mathematically model pairwise relations between objects. Connections between nodes in a graph can either be directed or undirected, direction is usually denoted by an arrow when visualizing graphs. Below in Figure 2.3 the system is visualized, here the nodes i and j are connected with edge $e_{i,j}$ where the index corresponds to $e_{from,to}$. There is also an associated variable $c_{i,j}$ for each edge, which can be anything correlating to its edge, some examples are: cost, capacity or price.

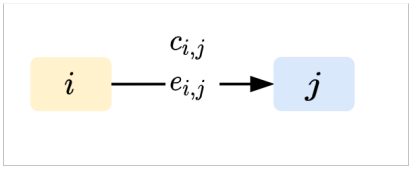

Figure 2.3: A directed graph with two nodes and one edge $e_{i,j}$ with an associated cost $c_{i,j}$.

Network flow [11] over graphs is mathematically explained in three types of constraints: Capacity constraint, conservation conditions and non-negativity. However the end node or sink would be excluded from the conservation since it is the end node and everything entering that node will stay there[3]. In a similar fashion the entering node will have an ingoing flow which won't be coming from a node but

[3]In real applications the end node would in turn send the packets to either the backhaul or UE.

is the generated traffic $g(t)$, more on this in chapter 3.2. Below each constraint is explained and displayed mathematically:

Let $V = (N, E)$ be a network with N nodes and E directed edges. Here $s, t \in N$ are the source and the sink of V respectively, each edge has an associated cost or capacity $c_{i,j}$. The flow $f_{i,j}$ can be any number as long as the following constraints hold.

1. Capacity: The flow over an edge cannot exceed a value $c_{i,j}$.

$$f_{i,j} \leq c_{i,j} \tag{2.2}$$

2. Conservation: Sum of flows going into a node must leave that node. Excluding the sink t as leaving node and source s as entering node.

$$\sum_{i \in V \setminus t} f_{i,j} = \sum_{k \in V \setminus s} f_{j,k} \tag{2.3}$$

3. Non-negativity: The flow cannot be negative.

$$f_{ij} \geq 0 \tag{2.4}$$

For this thesis the flow will be packets which can be described as an integer value of 1 for each packet in a flow. There is also a time aspect t which is discrete. For each time step a certain number of packets is sent to a source node and then put in to the flow of the network. Hence the time t is introduced as a variable on the flow as follows: $f_{i,j}(t)$. Since for each time step there can be accumulation of packets and there can be different amounts in different time steps the buffer $B_i(t)$ is introduced. Each node i has its own buffer B_i. Each buffer B_i cannot be larger than the given buffer size of b_i for each buffer.

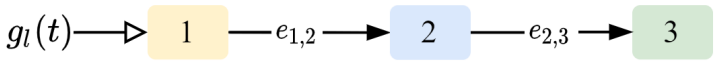

Figure 2.4: A directed graph with three nodes: 1, 2 and 3, two edges: $e_{1,2}$ and $e_{2,3}$. The traffic $g_1(t)$ enters node 1.

Using the graph in Figure 2.4 an example of the system in use is shown below in equation 2.5. Here the traffic $g_1(t)$ sends one packet each time step (Equation 2.5e) however the simulator utilizes a Markov chain, more on this in the next chapter 2.3.

$$B_1(t) = B_1(t-1) + g_1(t) - f_{1,2}(t), \tag{2.5a}$$
$$B_2(t) = B_2(t-1) + f_{1,2}(t) - f_{2,3}(t), \tag{2.5b}$$
$$B_3(t) = B_3(t-1) + f_{2,3}(t), \tag{2.5c}$$
$$B_i \leq b_i, \quad i = 1, 2, 3, \tag{2.5d}$$
$$g_1(t) = 1, \tag{2.5e}$$
$$f_{i,j}, B_i \geq 0, \quad \forall\, i, j = 1, 2, 3 \tag{2.5f}$$

Equation 2.5a is the amount of packets in the buffer in current time step, adding arriving packets and subtracting leaving packets from the buffer. This pattern repeats for Equations 2.5b and 2.5c, beside there is no leaving packets in the last Equation. Equation 2.5e shows how many packets are sent in each time step, this can be any positive integer. Equation 2.5f states the non-negativity restraint for all flows and buffers.

2.3 Markov Chains

A stochastic processes [4] is the collection of some random variables which follows a pattern and where the variables change within the state space, usually one variable is time. State space is the mathematical values of which the stochastic process can take, this can be all real values, just integers or some restricted binary space. A common and simple stochastic process is the Markov chain which is a stochastic process that describes events which depend on each other in a sequence.
A Markov chain [6] can be described as the weather for two different conditions: Raining and Not raining[4]. So, the weather today gives a certain probability for what the weather will be tomorrow, this is sequence which is only dependent on the former state. This can then be seen as a state machine where each state has a certain probability of moving to a different state or staying in the same state. Below in Figure 2.5 a state machine with two states are shown.

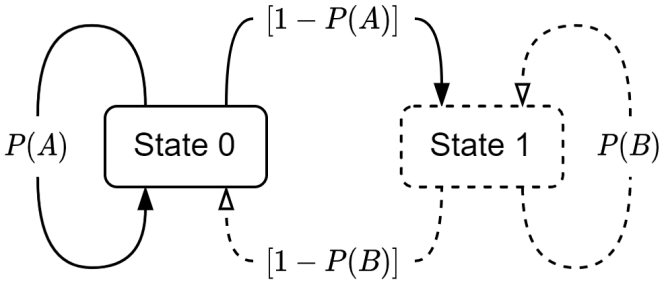

Figure 2.5: A state machine with two different states.

The definition for a general Markov Chain is the following: The stochastic process $\{X_n, n = 0, 1, \ldots\}$ with state space I is called a discrete-time Markov chain if, for each $n = 0, 1, \ldots,$

$$P\{X_{n+1} = i_{n+1} \mid X_0 = i_0, \ldots, X_n = i_n\} = P\{X_{n+1} = i_{n+1} \mid X_n = i_n\} \tag{2.6}$$

where the Markov property is satisfied which states that the next state depends of the current state. However, looking at time-homogeneous transition probabilities, the following assumption is taken:

[4]Commonly known as regular weather in Gothenburg

$$P\{X_{n+1} = j \mid X_n = i\} = p_{ij}, \quad i, j \in I \tag{2.7}$$

Here p_{ij} is called the one-step transition probabilities and satisfies equation 2.8.

$$p_{ij} \geq 0, \quad i, j \in I, \quad \text{and} \quad \sum_{j \in I} p_{ij} = 1, \quad i \in I. \tag{2.8}$$

Below in equation 2.9 is an example of an one-step transition matrix with two states and $I =: \{1, 2\}$, using the example from Figure 2.5 then $P(A) = 0.70$ and $P(B) = 0.55$.

$$\mathbf{P} = \begin{pmatrix} 0.70 & 0.30 \\ 0.45 & 0.55 \end{pmatrix} \tag{2.9}$$

2.3.1 Stationary distribution

The stationary distribution of a Markov chain is the distribution which it goes towards and says how much time the chain spends in one state overall, which also could be interpreted as the total intensity of one state. Below in equation 2.10 the general distribution is shown.

$$\pi^{(n)} = \begin{bmatrix} P(X_n = 0) & P(X_n = 1) & \cdots \end{bmatrix} \tag{2.10}$$

Considering a finite Markov chain $\{X_n, n = 0, 1, 2, \ldots\}$ where $X_n \in S = \{0, 1, 2, \cdots, r\}$, assuming the chain is irreducible and aperiodic, then Equation 2.11 below has a unique solution.

$$\begin{aligned} \pi &= \pi P \\ \sum_{j \in S} \pi_j &= 1 \end{aligned} \tag{2.11}$$

The limiting distribution of Equation 2.11 is the following:

$$\pi_j = \lim_{n \to \infty} P(X_n = j \mid X_0 = i), for\, all\, i, j \in S$$

Given a two state transition matrix P as:

$$P = \begin{bmatrix} 1 - a & a \\ b & 1 - b \end{bmatrix}$$

then the limiting distribution for any two state chain is shown below in equation 2.12.

$$\pi = \begin{bmatrix} \pi_0 & \pi_1 \end{bmatrix} = \begin{bmatrix} \frac{b}{a+b} & \frac{a}{a+b} \end{bmatrix} \tag{2.12}$$

2.4 Algorithms

An algorithm is a well-defined procedure which follows a set of rules usually given an input to solve a problem and then returning an output [5]. Algorithms can be viewed as tools for solving computational problems which are well-defined. Algorithms are not dependent on programming language and a common way of conveying an algorithm is by pseudo-code. Below in algorithm 1 an example pseudo-code is shown, here the input x is used to randomize a list of x numbers and then square them and returns the squared list.

Algorithm 1 Example pseudo-code

```
Input: n
X := Set of n randomized numbers
for x ∈ X do
    Add x^2 to Y
end for
return Y
```

Pseudo-code should only convey the method of an algorithm, an abstraction of the idea which can be applied to any suitable way of solving it.

2.5 Queuing Theory

Queuing theory [6] is the study of mathematically defining waiting lines and queues. Using customers and servers to explain the basics: Customers arrive at a server with an intensity λ, i.e., the average amount of customer per time unit. When a customer arrive at a server they either need to wait in line or leave the system. A customer is served in a timely manner, in average $b = 1/\mu$ time units. Finally, the customer leaves the server. There are multiple different ways of defining queues and therefore there are different stochastic processes to define them. To define a queuing network can be complex and therefore there is a notation defining them, the Kendall notation.

2.5.1 Kendall notation

Kendall notation [10] is a standard way of describing different queuing models. The notation is written as: `A/S/c/K/N/D` here the different parts stand for:

- A: The arrival process
- S: The service time distribution
- c: Number of servers
- K: Number of places in the queue
- N: The calling population
- D: The queue's discipline

Most common use of the system only includes the first three and one common queue is `M/M/1` where M is for Markov chain for both arrivals and service time.

2.6 One sample t-test

One sample t-test [9] is used for small samples and under the assumption that the population distribution is normal. The t-distribution is a distribution which depends on the sample size given, below in Equation 2.13 the density function of the t-distribution is shown

$$f(x) = \frac{\Gamma\left(\frac{k+1}{2}\right)}{\sqrt{k\pi}\Gamma\left(\frac{k}{2}\right)}\left(1+\frac{x^2}{k}\right)^{-\frac{k+1}{2}}, \; k \geq 1 \tag{2.13}$$

Here the parameter k, is the number of degrees of freedom which reflects the sample size - 1 and $\Gamma(x)$ the gamma function. The cumulative distribution function (cdf) is given in Equation 2.14 below.

$$F(t) = \int_{-\infty}^{t} f(x)dx \tag{2.14}$$

The one sample t-test is normally used to test a hypothesis for the population mean. Hypothesis testing is a way of trying two hypotheses which are designed to match the given data properly. We consider two different hypotheses: null-hypothesis H_0 and the alternative hypothesis H_1. For a two-tailed one sample t-test the hypothesis test becomes:

$$H_0 : \mu = \mu_0$$
$$H_1 : \mu \neq \mu_0$$

where μ_0 is the null hypothesized population mean of the population and μ is the true population mean. Below in equation 2.15 the t-test statistic is shown.

$$t_{obt} = \frac{\bar{x} - \mu_0}{s_{\bar{x}}} \tag{2.15}$$

Here $\bar{x}$ is the sample mean and $s_{\bar{x}}$ is the estimated standard error. Below in equation 2.16 the estimated standard error is shown.

$$s_{\bar{x}} = \frac{s}{\sqrt{n}} \tag{2.16}$$

Here s^2 is the sample variance and, n the sample size where s^2 is shown below in equation 2.17.

$$s^2 = \frac{1}{n-1}\sum\left(x_i - \bar{x}\right)^2 \tag{2.17}$$

The null hypothesis H_0, is then tested using the p-value which is defined as the "Probability of obtaining a sample more extreme than the ones observed in the data, assuming H_0 is true". Thus, the null hypothesis is rejected if the p-value is less than the given α. Here α is the significance level and says how big the rejection region is; usually, α is represented as $100(1-\alpha)\%$. α usually is 5% or lower depending on how accurate the test needs to be. In Figure 2.6 below the two-sided rejection region

for $\alpha = 5\%$ and t-distribution with $k = 4$ is visualized. Here an obtained value of $t_{obt} = 1.5$ is shown as well, for which the null-hypothesis would not be rejected.

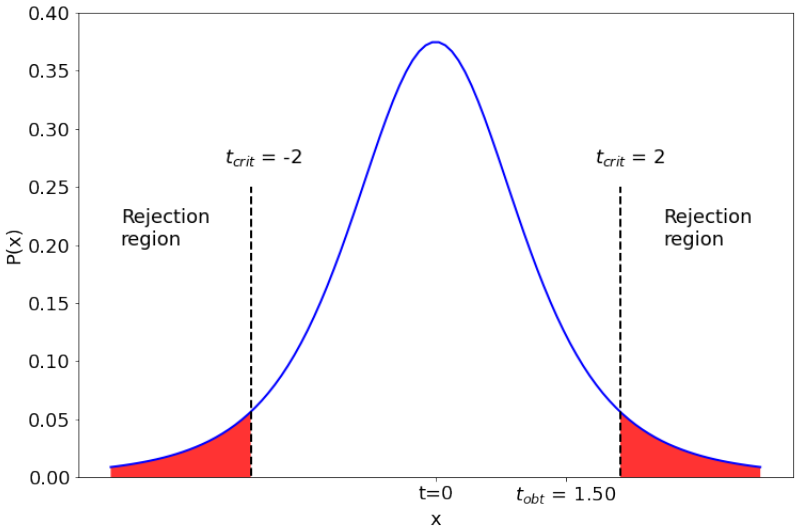

Figure 2.6: The t-distribution with $k = 4$ and a two-tailed rejection region for $\alpha = 5\%$. Here an obtained value $t_{obt} = 1.5$ is also visualized.

The p-value is obtained through getting the total area left of $-|t_{obt}|$ and right of $|t_{obt}|$ value under the curve in the t-distribution. The two-sided p-value is obtained using Equation 2.18 below where the t-distributions cdf is utilized:

$$\text{p-value} = 2 * (1 - F(|t_{obt}|)) \tag{2.18}$$

Equation 2.18 is valid when the distribution used for the p-value is symmetric around zero, which the t-distribution is.

2.7 Poisson Regression

A Poisson regression model is a generalized linear regression model (GLM). A GLM works by letting the response variable y be a function of a linear combination of the covariates/predictors and we refer to the function in question as a link function. Depending on how the response data looks some different distributions can be chosen. One of these is the Poisson distribution which uses the log function as the link function. There are three components to a GLM and for a Poisson regression model they are the following:

1. Distribution of Y: Poisson
2. Link function: $log(\mu)$
3. Linear predictors: $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots$

Below in Equation 2.19 the probability density function (pdf) for the Poisson distribution is shown. Here μ_i is the expected mean and variance for the ith response.

$$P\left(Y_i = k_i\right) = \frac{e^{-\mu_i}\mu_i^{k_i}}{k_i!}, \quad k_i = 0, 1, \ldots, \tag{2.19}$$

Hence the linear predictor becomes the following Equation 2.20. Note that μ_i can have different values for different values of X. Taking the logarithm of Equation 2.20a it becomes 2.20b, below both Equations are shown.

$$log(\mu_i) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n \tag{2.20a}$$

$$\mu_i = e^{(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n)} \tag{2.20b}$$

The response variable μ_i, can be a count or a rate with explanatory variables $X = (x_1, x_2, \ldots, x_n)$ which can be categorical or continuous. The aim is to estimate the parameters/coefficients β_i, $i = 1, 2, \ldots, n$ to obtain a fitted model for the given data. This is done by an iterative re-weighted least square method through the following steps:

- First the working dependent variable:

$$z_i = \eta_i + \frac{y_i - \mu_i}{\mu_i} \tag{2.21}$$

  Here $\eta_i = \log\left(\mu_i\right)$ is the link function.
- Then the iterative weights:

$$w_i = 1/\left[\mu_i \frac{1}{\mu_i^2}\right] = \mu_i \tag{2.22}$$

- Then finally β is estimated through Equation 2.23. Here X is the model matrix (from data), W the diagonal matrix of weights, and z the working dependent variable given in Equation 2.21:

$$\hat{\beta} = \left(X'WX\right)^{-1} X'Wz \tag{2.23}$$

- These steps are then repeated until β converge.

When β has converged the Poisson regression model becomes Equation 2.24 with the expected mean μ for the set of explanatory variables X.

$$\mu = e^{(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n)} \tag{2.24}$$

2.7.1 Performance score

Scoring the Poisson regression model is done using the D^2 score [8] which is the fraction of deviance explained, it is a form of skill score and is a generalization of R^2. It is computed according to the following Equation 2.25:

$$D^2(y, \hat{y}) = 1 - \frac{\text{dev}(y, \hat{y})}{\text{dev}\,(y, y_{\text{null}}\,)} \tag{2.25}$$

Where $\hat{y}$ is the predicted value of given input y, y_{null} is the optimal predicted value. *dev* is the deviance which is calculated as the following:

$$\text{dev}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 2\left(y_i \log\left(y_i/\hat{y}_i\right) + \hat{y}_i - y_i\right)$$

Where:$y_i \in y, i = 1, 2, ..., n_{\text{samples}}$ and $\hat{y}_i \in \hat{y}, i = 1, 2, ..., n_{\text{samples}}$.

Here n_{samples} is the number of data points for which the deviance is calculated, i.e., the number of test samples for which the score is to be calculated with. The maximum score is 1 where the model would predict correctly on all given inputs and the lowest score can be -1, i.e., it can be equally bad as good. If the model would predict y_{null} while disregarding the input features the D^2 score would become 0.

3
Methods

The thesis work was divided in to four main parts: Investigation, the traffic generator, network model, and the Ethernet simulator. Investigation clarifies all different relevant parts of the RBS and the protocol used for packets. Latency magnitudes are defined, and the nodes buffer sizes and internal latency are defined. The traffic generator is described as a Markov chain and its stationary distribution is also shown. The network model is defined with graph theory. Finally, the ethernet simulator is briefly described and the algorithm which was implemented is described, some programming implementation is shown here as well.

3.1 Investigation

In this section all the parts which was discussed in section 2.1 will be formalized and modified for use in the simulator. First a packet structure fitting the simulator will be defined, then link speeds will be used to define the length of a time frame inside the simulator. Then the buffer sizes will be defined for the simulator, lastly delay times are briefly summarized.

3.1.1 Packet

Ethernet frames is the basis for the simulator packets, it will contain the three parts mentioned in 2.1.1: Source and destination address and payload. Two additions will be made: the transmission and arrival time for each packet. These will enable the simulator to determine how long it takes for each packet to reach its destination. If the transmission time is too long it will be useless for the RBS and will then be discarded see 2.1.1. The packet will then contain the following five components:

- Source address.
- Destination address.
- Packet size; here each packet will be normalized as size 1.
- Transmission time.
- Arrival time.

3.1.2 Links

For simplicity's sake a link will only contain one packet per time frame, however since there are different speeds of links a base line needs to be set. This base line will be the fastest link of 100Gbps (see 2.1) and the length of the link is therefore

something that reflects that. So given the 2/3 of speed of light for a 100Gbps link with one packet of 1518B and $v = 2/3 * 299792458$ the length of the link becomes:

$$L = \frac{Packet\,size][Byte]}{[Linerate][Bbps]/v[m/s]} = \frac{1518[Byte]}{100(10^9/8)[Byte/s]/v[m/s]} = \frac{1518}{62.54...}[m] \approx 24.3m$$

and length of a time step, i.e., simulation step, is the following:

$$t = \frac{L[m]}{v[m/s]} \approx 0.12\mu s$$

So, for each link with one packet the length will be 24.3 meters and the time frame for this length is 0.12 μs. Hence for the speed of the other links (50, 25, 10) a *time taken* property is introduced. Which will increase the time it takes for a packet to be sent from one node until it is received on the other end of the link. This can in turn be specified as a simple function:

$$\text{Time taken} = \frac{100}{Speed}[t]$$

So, for 50, 25 and 10Gbps links the time taken will be 2, 4 and 10 respectively. There will also be the possibility of adding different length for links as well using a different principle, since adding length to a link will enable it to buffer more packets, so if a link is $2L$ then that link will have a buffer size of 2. Essentially buffer sizes for the links are a function with the multiple x of length $L = 24.3$:

$$\text{Link buffer size} = x[L]$$

3.1.3 Packet latency

The latency is calculated according to Equation 2.1 and with the given latency length for t_{CV1} and t_{CV2} as 1 the equation becomes:

$$t_D = (t_2 - t_{CV2}) - (t_1 + t_{CV1}) = (t_2 - 1) - (t_1 + 1) \tag{3.1}$$

There were several different delay times in Table 2.2 however only a few are important for the simulator, for this thesis the important ones are between $25\mu s$ and $200\mu s$ from Ultra-low to High200. As NR (5G) requirement is a one-way delay of $100\mu s$ is the most important. The possibility of adding longer fiber lengths for the network $200\mu s$ is also considered. Different simulations can be setup using these different constraints depending on which sort of network is tested.

3.1.4 Node latency and buffer size

The node latency t_{CV1} and t_{CV2} in Equation 2.1 from chapter 2.1.3 is as mentioned before the internal latency of a node. For the thesis node latency is approximated to one time frame. This is based on the fact that it takes some time to process data inside the nodes. As described earlier the internal behavior on the switches will not be modeled in detail, for this thesis one time frame is a good approximation.

Stated in the section 2.1.2.2 the buffer sizes of switches can range from 5270 - 84321 packets for each buffer. The rule of thumb stated that the accumulated amount for 50 ms should be utilized which is approximately 39525 packets. However, since this simulator is designed with a NR network in mind, for which the maximum One-Way delay is between 25 and 200 μs a 50 ms buffer would not make any sense. If the buffer is 50 ms large for a network with a maximum One-Way delay of 200 μs then, any packets added after 200 μs would have to be thrown when they arrive. Since the delay would be at least 200 μs or greater than what is allowed for the One-Way delay. The three important buffer sizes are shown below:

$$25ms \Rightarrow 25/0.12 \approx 208\, packets \tag{3.2}$$
$$100ms \Rightarrow 100/0.12 \approx 830\, packets \tag{3.3}$$
$$200ms \Rightarrow 200/0.12 \approx 1660\, packets \tag{3.4}$$

3.2 The Traffic Generator

The traffic generator is a state machine with on/off traffic. The traffic generator will either send all- or not send all packets at current time frame with a given probability, hence the traffic generator is a state machine with two states: `Send` and `not send`. Below in Figure 3.1 the state machine is visualized for the two states with probabilities P(A) and P(B).

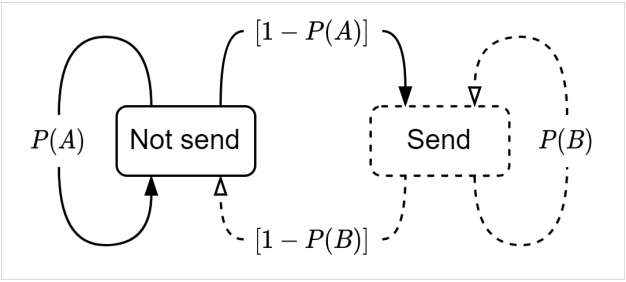

Figure 3.1: The state machine for the traffic generator.

The different probabilities are divided in to four possible outcomes given the two states:

$P(A)$: Not send, traffic was not sent in previous time frame.
$1 - P(A)$: Send, traffic was not sent in previous time frame.
$P(B)$: Send, traffic was sent in previous time frame.
$1 - P(B)$: Not send, traffic was sent in previous time frame.

Given the state machine the traffic generated has different load depending on the probabilities of sending and not sending in the two states. There are three different traffic loads: low, medium, and high. So for each given intensity there is a given probability of sending or not sending p packets at the current time frame. The probabilities $P(A_i)$ and $P(B_i)$ can be seen in Table 3.1 below and are based on measurements on real networks.

Table 3.1: The different probabilities for low, medium and high intensity of traffic.

$P(X)$	Low	Medium	High
$P(A_i)$	0.96	0.92	0.68
$P(B_i)$	0.84	0.83	0.91

Using the different probabilities for traffic loads the stationary probabilities can then be calculated using Equation 2.12. Calculations for the low traffic distribution is shown below.

$$\pi_1^{Low} = \frac{a}{a+b} = \frac{1-0.96}{1-0.96+1-0.84} = 0.2$$

Since the equality $\pi_0^{Low} + \pi_1^{Low} = 1$ holds, the second distribution is calculated as: $\pi_0^{Low} = 1 - 0.2 = 0.8$. In Table 3.2 below the different distributions for all traffic loads have been calculated using the values from Table 3.1. In the table the indices are changed to: $\pi_0 = \pi_{Not\,send}^{Traffic\,load}$ and $\pi_1 = \pi_{Send}^{Traffic\,load}$.

Table 3.2: Stationary distributions for all traffic loads.

State	i=Low	i=Medium	i=High
$\pi_{Not\,send}^{i}$	0.8	0.68	0.22
π_{Send}^{i}	0.2	0.32	0.78

3.2.1 Traffic generator algorithm

Since the state machine will be implemented in a program an algorithm was designed. Below in algorithm 2 the pseudo-code is outlined.

Algorithm 2 The traffic generator algorithm

```
procedure TRAFFIC-GENERATOR(T, P)   ▷ T: time steps, P: State probabilities
    G := List of zeros, T indices long
    State = Not send
    for i = 1 to T do
        R = random float ∈ [0, 1]
        if State = Not send then
            if R ≤ P(Send|Not send) then
                Add Packet to G[i]
                State = Send
            else
                Do nothing                          ▷ R ≤ P(Not send|Not send)
                State = Not send
            end if
        else if State = Send then
            if R ≤ P(Send|Send) then
                Add Packet to G[i]
                State = Send
            else
                Do nothing                          ▷ R ≤ P(Not send|Send)
                State = Not send
            end if
        end if
    end for
    Return: G                                       ▷ G is the generated traffic
end procedure
```

3.3 Network Model

In Radio Base Stations there are endless possibilities for different configurations and in this chapter the goal is to define the notation of and to make a clear definition of the different topologies that will be considered in this thesis.

3.3.1 Network Introduction and Notation

Below in Figure 3.2 a simple network is illustrated. 3.2a is shown with traffic going both from and to backhaul on the left hand side and User Equipment (UE) on the right hand side. Since uplink and downlink traffic are independent of each other two different networks can be constructed as can also be seen as networks 3.2b and 3.2c. These two networks together are equivalent to network 3.2a. Here the nodes: Source, Intermediate and Destination are introduced. Links will be called edges for the abstraction of networks.
The edges in network 3.2b and 3.2c will be referred to as $e_{i,j}$ where the indices i and j are defined as from node and to node respectively; i.e., $e_{from,to}$. The edges are directed and each have a capacity $c_{i,j}$ associated with it. Furthermore, the Source,

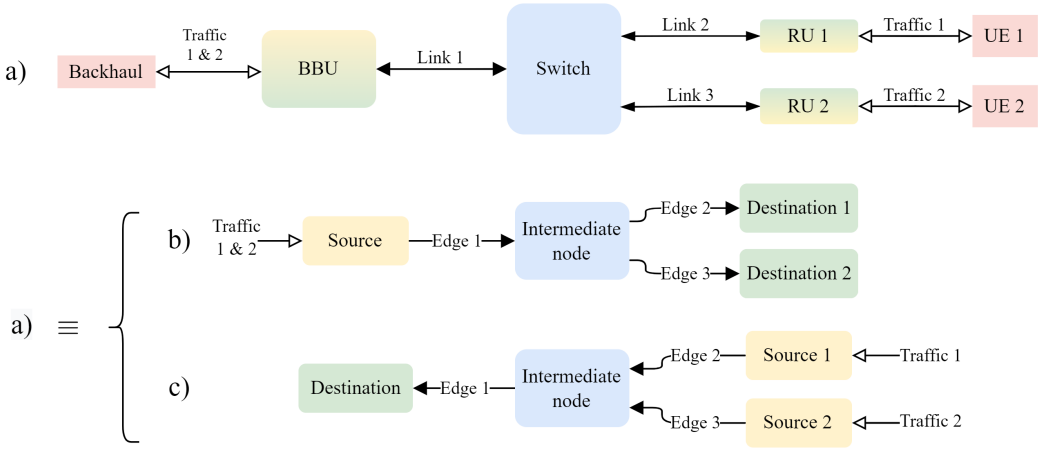

Figure 3.2: a) A network with one BBU, one switch and two RU's. Traffic going both ways. **b)** Traffic going to the RUs, downlink network. **c)** Traffic going to the BBU, uplink network.

Intermediate and Destination nodes will be referred to as s_i, w_j and d_k respectively. The indices i, j and k are independent non-negative integers, more on this in section 3.3.2. See Figure 3.3 below for reference.

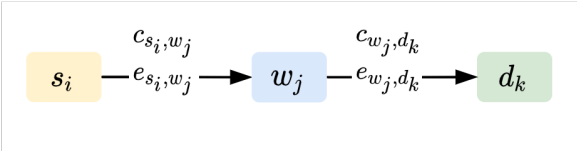

Figure 3.3: Network notation with indices of edges, capacity and nodes.

There are two cases with different characteristics of topologies which will be focused on in this thesis. Case one is defined as a *one edge network* where each source and destination node only has one edge. Similarly case two is defined as a *two edge network* where each source and destination node has two edges. One and two edge only refers to the source and destination nodes, in between these two nodes one (or several) intermediate node is positioned and doesn't necessarily only have one or two edges, this will be discussed further in 3.3.3.1 and 3.3.3.2.

3.3.2 Mathematical Definitions of the Network

First the sets which will be used are defined in Table 3.3 below. The constants m, n and h are parameters which describe the configuration and can be seen as input variables, l is defined in Equation 3.6 below.

Table 3.3: The different sets.

$G := \{g_1, \ldots, g_l\}$	Set of l traffic generators
$M := \{s_1, \ldots, s_m\}$	Set of m source nodes
$N := \{d_1, \ldots, d_n\}$	Set of n destination nodes
$H := \{w_1, \ldots, w_h\}$	Set of h intermediate nodes
$P := \{p_1, \ldots, p_l\}$	Amount of packets to be sent for each generator
$B := \{b_1, \ldots, b_{m+n+h}\}$	Buffer size for each node in the network
	l, m, n and $h \in \mathbb{Z}^+$

To define a network set either n for a downlink (3.2b) or m for an uplink network (3.2c) as ≥ 2. Then either m or n will be 1 (whichever wasn't set). In Equation 3.5 below the definition is shown:

$$\begin{cases} if\, m \geq 2, \quad n = 1, \\ if\, n \geq 2, \quad m = 1 \end{cases} \tag{3.5}$$

Number of traffic generators l are defined as the maximum of either m or n:

$$l = max\{m, n\} \tag{3.6}$$

The entire system aims to generalize a network for an RBS. First the traffic which is generated for $g_i(t)$ will belong to certain source and destination nodes. If $l = m$ then traffic $g_i(t)$ will be sent from node s_i and the sum of all traffic $g_i(t)$ will be arriving to node d_1. However if $l = n$ then it is reversed, i.e., the sum of traffic $g_i(t)$ is sent from source s_1 and each traffic $g_i(t)$ is sent to a destination node d_i. The buffer $B_i(t)$ is defined for each node as the incoming traffic $g_i(t)$ or $f_{j,i}(t)$ (depending if source or intermediate) plus what was kept from previous time step $B_i(t-1)$ minus what is leaving for the current time step $f_{i,j}(t)$. Below in the equation system 3.7 and 3.8 network $l = m$ and $l = n$ are defined mathematically. Below each equation system a description is listed for each row of equations in respective system.

$$l = m : \begin{cases} B_i(t) = B_i(t-1) + g_i(t) - f_{i,j}(t), \quad i \in M, j \in H, & \text{(3.7a)} \\ B_i(t) = B_i(t-1) + \sum_{i \in H} f_{j,i}(t) - f_{i,d}(t), \quad i \in H, j \in M, & \text{(3.7b)} \\ B_d(t) = B_d(t-1) + f_{j,d}(t), \quad j \in H, & \text{(3.7c)} \\ B_i \leq b_i, \quad i \in M \cup N \cup H & \text{(3.7d)} \\ g_i(t) = p_i, \quad i \in M & \text{(3.7e)} \\ f_{i,j}, B_i \geq 0, \quad \forall\, i, j \in M \cup H \cup N & \text{(3.7f)} \end{cases}$$

- 3.7a describes the buffer for each source node i in the network.
- 3.7b describes the buffer for all intermediate nodes.
- 3.7c describes the buffer for the singular destination node.
- 3.7d limits the size of the buffers.
- 3.7e is the p amount of packets coming into the system which.
- Equation 3.7f is the non-negativity requirement.

For an $l = n$ network the first three equation differs, the equations for this network is listed below in Equation system 3.8.

$$l = n : \begin{cases} B_s(t) = B_s(t-1) + \sum_{k \in N} g_k(t) - f_{s,j}(t), \quad j \in H, & \text{(3.8a)} \\ B_i(t) = B_i(t-1) + f_{s,i}(t) - f_{i,j}(t), \quad i \in H, j \in N, & \text{(3.8b)} \\ B_i(t) = B_i(t-1) + f_{i,j}(t), \quad i \in N, j \in H & \text{(3.8c)} \end{cases}$$

The difference between 3.7 and 3.8 is small beside for the first equation, 3.7a. The differences are described below.

- 3.7a is the sum of all traffic going in to the buffer for the singular source node.
- 3.8b here the indexes changes slightly to accommodate for the differences for the other two equations.
- 3.8c describes all the different destination nodes i instead of a singular destination node in the former equation.

3.3.3 Networks

Here the different one edge networks which will be used are introduced. In an attempt to make referencing the different networks easier a system is introduced. This system is based on Kendall's notation and uses a `a/b/c` notation. Here it will be defined as `ae/bi` where `a` and `b` are defined as:

- a: The number of edges going from source and to destination nodes.
- b: The number of intermediate nodes, some special cases will be noted:
  - c: for connected intermediate nodes

Example of the system in use:
Using the system for a "one edge, one intermediate node" network, its name would be: `1e/1i` network. This exact network can be seen in Figure 3.4 in subsubsection 3.3.3.1.

3.3.3.1 One Edge Networks

Below in Figure 3.4 the basic case is shown with one intermediate node and m or n amount of source nodes or destination nodes respectively. This network will be referred to as `1e/1i`.

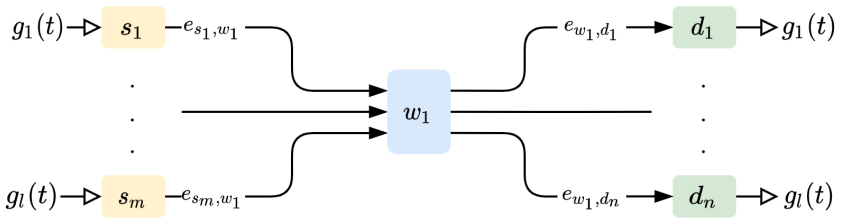

Figure 3.4: `1e/1i`. A one edge network with one intermediate node.

In figure 3.5 below the network has an extra intermediate node in series. This network is interesting when there's exceedingly long distances between RUs and the BBU. The capacity on the edge e_{w_1,w_2} is usually much higher than the other edges in the network. This network will be referred to as `1e/2ic`.

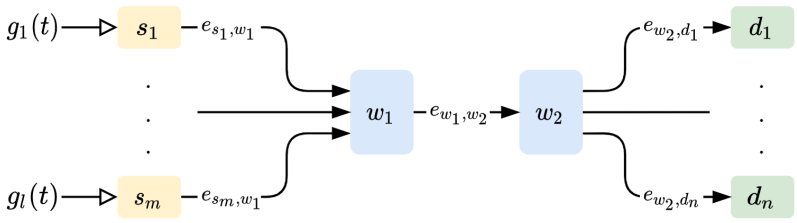

Figure 3.5: `1e/2ic`. One edge network with two intermediate nodes in series.

3.3.3.2 Two edge Networks

Here the two edge networks which will be used are introduced. Below in Figure 3.6 a network with two intermediate nodes is shown, here the traffic has two possible routes to take to its destination node. The added intermediate node w_2 enables redundancy in the network i.e., the capacity can be exceeded for one edge without losing packets between source and destination node. This network will be referred to as `2e/2i`.

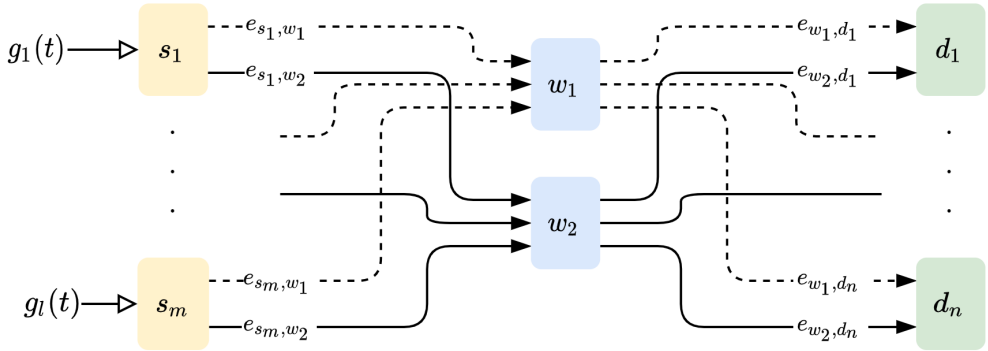

Figure 3.6: `2e/2i`. Two edge network with two intermediate nodes.

In Figure 3.7 a network with two intermediate nodes and a connection in between them is shown, here the traffic has four possible routes to take to its destination node. Similarly to the `2e/2i` network, this network enables further redundancy within the intermediate nodes. This network will be referred to as `2e/2ic`.

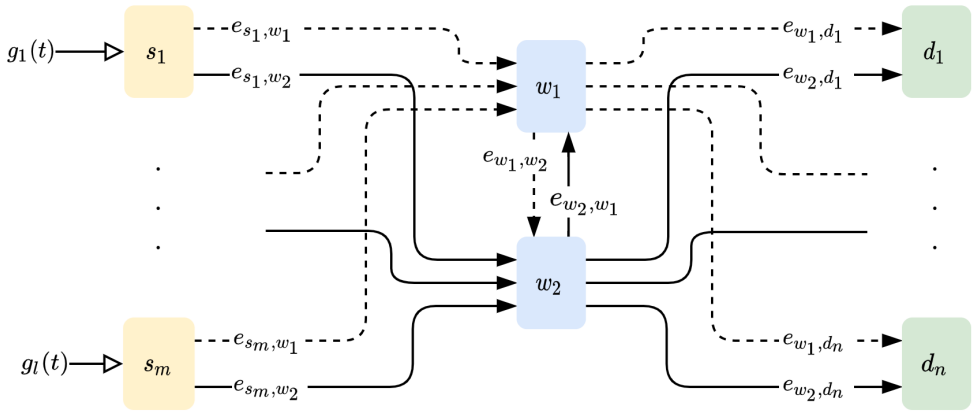

Figure 3.7: `2e/2ic`. Two edge network with two intermediate nodes which are also connected.

3.3.4 Network creation algorithm

The network is general and the only requirement here is that it returns edges given a template. The template is one of the intermediate connection models given in chapter 3.3.3. A node connection is defined as whichever node is directly connected to another one. The general formulation can be seen below in algorithm 3. M, N and H in the algorithm below are all defined as sets in Table 3.3 above.

Algorithm 3 Network creation algorithm

Input: M, N, H, template ▷ Template: Intermediate template
M := Set of m source nodes
N := Set of n destination nodes
H := Set of h intermediate nodes
for i = 1 to max{M, N}, given template **do**
    Create edge
    Append to V
end for
return V

3.4 The Ethernet Simulator

First the execution algorithm is presented which is the final step for the simulator. Then an overview of the program structure is shown. The interaction and design of the three main parts of the simulator is explained as well. Finally, a brief overview of how it was implemented in Python.

3.4.1 Execution algorithm

The algorithm was designed using the information in the investigation, chapter 3.1. Below in algorithm 4 the pseudo-code is shown for the execution of a packet based fronthaul. The network creation and traffic generator are both inputs for the execution algorithm and are noted as V and G in the algorithm respectively.

Algorithm 4 Execution algorithm

```
Input: V, N, T, L      ▷ V: Network, N: generators, T: timesteps, L: Traffic-load
A := Empty list                          ▷ Nodes and edges with ≥ [one packet]
G := traffic-generator(N, T, L)                                ▷ Generated traffic
t := 0                                                          ▷ Time step t
while t ≤ T or A ≠ empty do
    A ← Get active nodes & edges ∈ V
    Sort A in descending order, with destination nodes first
    for i=0 to length(A) do
        P ← First packet in A[i]
        C ← Get connection for packet                              ▷ Node or edge
        if C ≠ full then
            Move P from A[i] to E
            Set C to active
            if C is P's destination node then
                K ← Get [time of arrival] - [time of sending]          ▷ Latency
                if K ≥ [Latency req.] then
                    Discard P                  ▷ Packet loss, latency to high
                end if
            end if
        else if A[i] = Node then                          ▷ Keep in node buffer
            Do nothing
        else if A[i] = Edge then
            Discard P                                  ▷ Packet loss, buffer full
        end if
        if A[i] is empty then
            set A[i] to inactive
        end if
    end for
    if G[t] ≠ empty then          ▷ Initiate traffic from traffic-generator
        for i=0 to length(G[t]) do
            P_G ← Get packet i from G[t]
            Move P_G from G[t] to P_G source node
            Set source node to active
        end for
    end if
    t += 1
end while
```

3.4.2 Program structure

The program has an object-oriented design. Each major part was split into its own separate class. The classes are network creation, traffic generator and execution. Each of them is combined in the simulator for which everything is run through. Below in Figure 3.8 an illustration of the structure is shown.

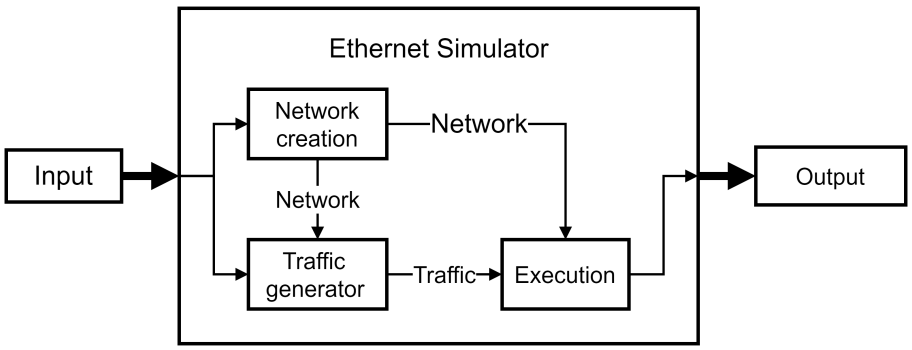

Figure 3.8: Structure of the ethernet simulator.

3.4.3 Program implementation

The ethernet simulator was implemented using Python and manipulating dictionaries for both the network and packet generation. The reasoning for using Python was several:

- Easy to implement something that will work fast.
- Former knowledge as well as supervisors at Ericsson having knowledge of the language.
- Several modules are available that enables easy addition to the simulator (such as numpy, sci-kit, matplotlib etc.).

Python has a data structure called dictionary, which is an array with indexing for each value. Where an example would be the following for two keys: `dict={"key 1": value, "key 2": value}`. The use of dictionary made it easy to create both network and packets, so each node was defined as a key with some different properties and space for its current packets. In a similar fashion, each packet had its property for addresses as well as the other properties discussed in chapter 3.1.1. So, the end result will be easy to read as well as easy to get information about content in whatever node buffer at any given time frame.
The simulator was designed as three different classes: Network, traffic generator and execution. Each of these classes run together through the simulator class and ends with saving all relevant data to a csv file.

4
Results

First an evaluation of the ethernet simulator is shown, using equations and output from the simulator. Then some tests are conducted to confirm a good enough number of iterations and number of time steps for generation of data, afterwards the findings are summarized. After that a regression analysis is done, first it is determined and motivated what sort of data is to be generated. Then a Poisson regression model is used to fit the data and its effectiveness is shown.

4.1 Evaluation of the Simulator

In this section the ethernet simulator is first evaluated with a simple model which can be verifiable by hand calculations. Then to determine how many iterations are needed for the data to be within its average given from stationary Markov chain the one sample t-test is utilized. Finally, the amount time steps needed to reach a low standard deviation is investigated. The main goal of the evaluation is to minimize the number of iterations and time-steps for each simulation run while making sure that the simulation is accurate.

4.1.1 Validating the simulator for a small example

Using the example from Figure 2.4 which is a network with one of each source, intermediate and, destination node and the 1e/1i topology. Then sending packets with 100% probability for 2 time steps, then Equation 4.1 shows how the packets should move over the different buffers.

$$B_1(t) = B_1(t-1) + g_1(t) - f_{1,2}(t), \tag{4.1a}$$
$$B_2(t) = B_2(t-1) + f_{1,2}(t) - f_{2,3}(t), \tag{4.1b}$$
$$B_3(t) = B_3(t-1) + f_{2,3}(t), \tag{4.1c}$$

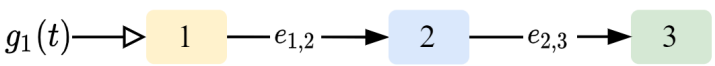

Figure 4.1: A directed graph with three nodes: 1, 2 and 3, two edges: $e_{1,2}$ and $e_{2,3}$.

Instead of using integers, here each packet will be denoted as p_1 and p_2 for clarity for the calculations in Equation 4.2. Beside it a table with the printout is shown, the data in Table 4.1 is generated from the simulator from which a screenshot was taken which can be seen in Appendix A. The simulator runs until all packets have reached its destination.

$$
\begin{aligned}
&B_1(0) = 0 + 0 - 0,\\
&B_2(0) = 0 + 0 - 0,\\
&B_3(0) = 0 + 0,\\
\\
&B_1(1) = 0 + p_1 - 0,\\
&B_2(1) = 0 + 0 - 0,\\
&B_3(1) = 0 + 0,\\
\\
&B_1(2) = p_1 + p_2 - p_1,\\
&B_2(2) = 0 + p_1 - 0,\\
&B_3(2) = 0 + 0,\\
\\
&B_1(3) = p_2 + 0 - p_2,\\
&B_2(3) = p_1 + p_2 - p_1,\\
&B_3(3) = 0 + p_1,\\
\\
&B_1(4) = 0 + 0 - 0,\\
&B_2(4) = p_2 + 0 - p_2,\\
&B_3(4) = p_1 + p_2,\\
\\
&B_1(5) = 0 + 0 - 0,\\
&B_2(5) = 0 + 0 - 0,\\
&B_3(5) = [p_1 + p_2] + 0,
\end{aligned}
\tag{4.2}
$$

Table 4.1: Data from a printout taken of the ethernet simulator, see Appendix A

B1(0)	0	0	0
B2(0)	0	0	0
B3(0)	0	0	-
B1(1)	0	1	0
B2(1)	0	0	0
B3(1)	0	0	-
B1(2)	1	1	-1
B2(2)	0	1	0
B3(2)	0	0	-
B1(3)	1	0	-1
B2(3)	1	1	-1
B3(3)	0	1	-
B1(4)	0	0	0
B2(4)	1	0	-1
B3(4)	1	1	-
B1(5)	0	0	0
B2(5)	0	0	0
B2(5)	2	0	-

4.1.2 Number of iterations

There are three main classes implemented for the ethernet simulator: Network, Traffic generator and Execution. The simulator is deterministic, if the same number of packets are sent at the same time steps for two different simulations it will return the same result. See calculations in Equation 4.2 and Table 4.1 above.
The randomness only comes from the Traffic generator; hence number of samples and time steps can be evaluated on that class alone. Since the Traffic generator returns a random number of packets for given time steps an average needs to be taken to remove the effect of extreme cases for different generation of traffic. Below is the result from two different executions of the traffic generator. Here samples

will be the number of iterations which are executed with the same inputs for each iteration.

Using the stationary distribution as the mean traffic intensity a comparable mean is obtained. For low traffic it was calculated to 0.2 according to Table 3.2. This means that the overall intensity of the traffic is 20% of a full traffic load. Full traffic meaning that traffic is sent in every time step, compare to the example in chapter 4.1 above. Generating low traffic with 10 000 time steps means that the average load should be the following:

$$Number\,of\,generators * time\,steps * \pi_{send}^{Low} = 1 * 10\,000 * 0.2 \approx 2000$$

Below in Figure 4.2 1000 samples have been generated and plotted in a histogram.

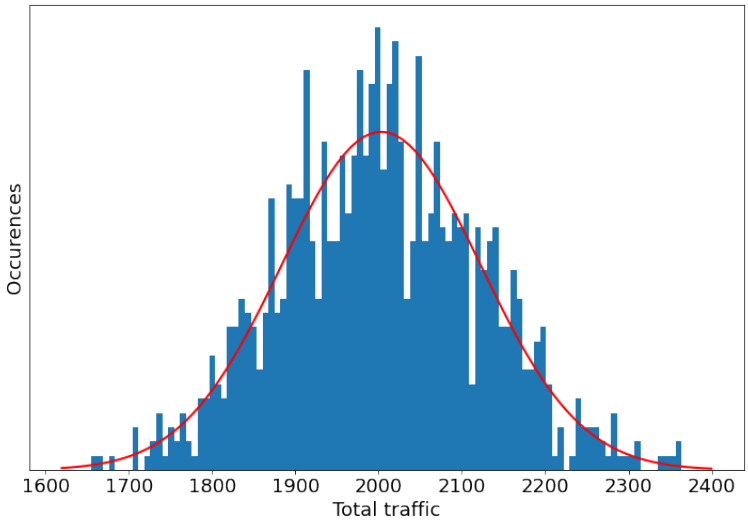

Figure 4.2: Histogram of $n = 1000$ samples with 10 000 time steps for low traffic. The red line shows the normal distribution of the data.

As can be seen the figure shows the samples being approximately normal distributed with the mean being around 2000. In Figure 4.3 below another generation of traffic is shown for $n = 10\,000$ samples and 1000 time steps.

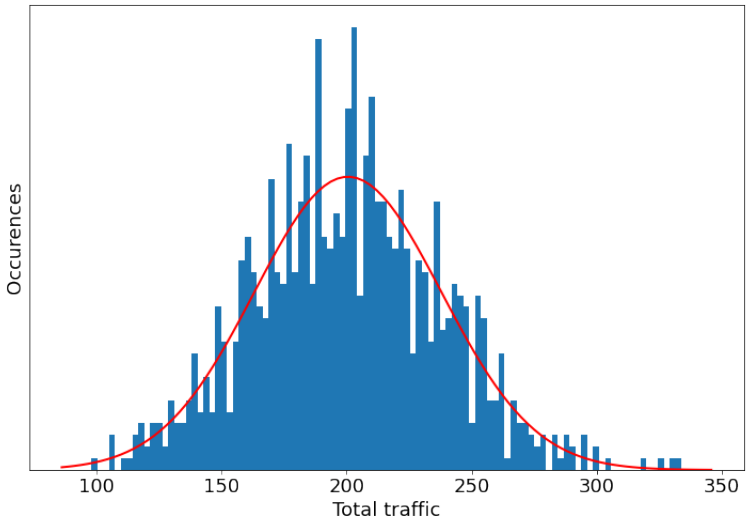

Figure 4.3: Histogram of $n = 10\,000$ samples with 1000 time steps for low traffic. The red line shows the normal distribution of the data.

As can be seen both Figure 4.2 and 4.3 show a normal distribution of data with a mean around the intensity of 20% of the total traffic generated. The results above confirms that the data is normally distributed, and that the intensity is close to the total traffic generated. Next step is to determine how many iterations each execution of the simulator needs to get an average close to the stationary distribution for: $\pi^i_{send}, i = Low, Medium, High$ which are $0.2, 0.32, 0.78$ respectively.

Continuing with low traffic and performing a one sample t-test for different number of iterations and time steps. Stating the null hypothesis as no difference from the mean, i.e., $H_0 : \mu_1 = \mu_n$ with a significance level of at least 5%. Traffic is generated for 1000 time steps and between $5 - 200$ iterations to see if the samples diverge to much from the mean. In Figure 4.4 below the data is shown.

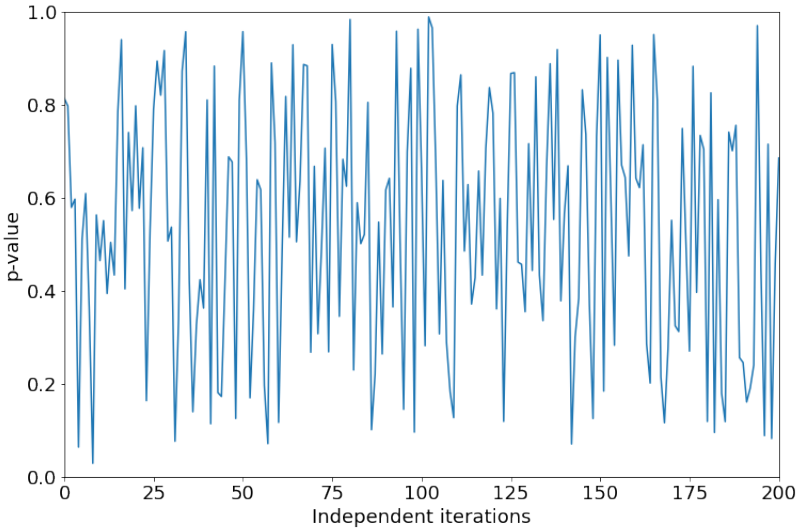

Figure 4.4: P-values generated for 1000 time steps for samples between 5 and 200 taken.

Here the most extreme value is: $p = 0.06$ at 4 iterations. Since $p > 0.05$ none of the iterations reject the null hypothesis. To see if there is a difference between 1000 and 10 000 time steps another test with the same amount of samples as before of t-tests are done, below in Figure 4.5 the result are shown.

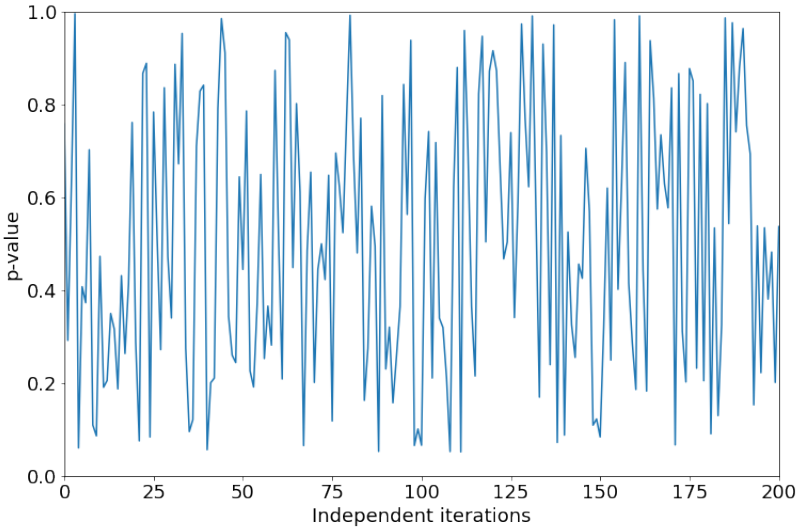

Figure 4.5: P-values generated for 10 000 time steps for samples between 5 and 200 taken.

Similarly, to 1000 time steps the most extreme value was: $p = 0.052$ at 116 iterations. The null hypothesis is therefore not rejected at 5% significance level.

Given that the null hypothesis was not rejected for any number of iterations shows that none of them are outside the mean. Any number of iterations is therefore a reasonable number to do for each run, since the run time of the simulator increases for each iteration a smaller amount is preferable. Therefore, the number of iterations needed does not need to be higher than 5 for the data to be within an average mean of the traffic load for any generation of data.

4.1.3 Number of time steps

Given the number of iterations per run is set to 5 then a number of time steps for each simulation needs to be decided as well. The goal is to look at what amount of time steps gives a standard deviation good enough to trust the result of the simulator. Low traffic was generated with the traffic generator with time steps between $1000-500\,000$. For each simulation 5 iterations were done, and the standard deviation is calculated and taken as a percentage of the amount it deviates. Below in Figure 4.6 the result is shown.

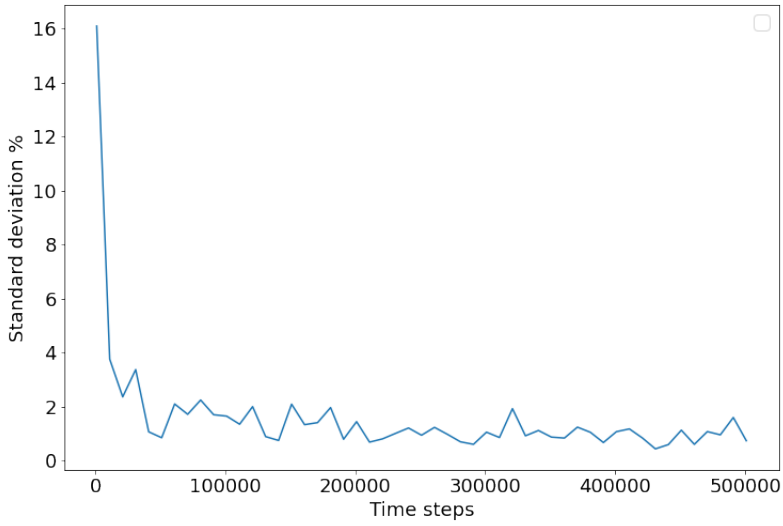

Figure 4.6: Standard deviation plotted for simulations with time steps between 1000 - 500000.

Although the data is fluctuating it seems to even out somewhere below 2% for time steps at about 50 000 and above. Looking closer for an interval between $20\,000 - 150\,000$ below in Figure 4.7 the result can be seen.

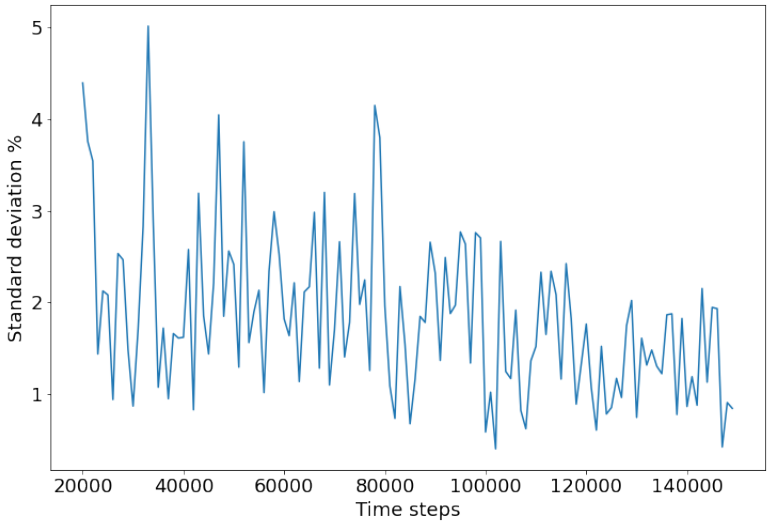

Figure 4.7: Standard deviation plotted for simulations with time steps between $20\,000 - 150\,000$.

The data shows that at above 80000 time steps the standard deviation seems to be stagnant, therefore choosing time steps at 100 000 will be enough to ensure a standard deviation of $\approx 2\%$. A simulation with 100 000 time steps shows that a run takes about 3 seconds with a standard laptop.

4.1.4 Summary of Evaluation

Here the settings of the Ethernet simulator are listed:

- **Five iterations per simulation run** with the same settings and inputs. For the five different iterations the only thing changing will be the traffic generation. The average of all iterations will be close to the average of given intensity for given traffic load (low, medium, or high).
- **At least** 100 000 **time steps** is needed for a standard deviation around 2 % per iteration.
- Each iteration takes $\approx 3s$, so each simulation takes $\approx 15s$.

4.2 Regression Analysis

First a short investigation on how the data for buffer size is depending on packet loss is presented. After that the data generation is shown. Then the Poisson regression model is trained and used in a performance metric to determine how well it did.

4.2.1 Intermediate buffer size

To get an understanding for how the buffer size depend on packet loss and to motivate a machine learning model a simple simulation was ran with 100 time steps.

The simulation ran until the packet loss was below 5% and for $3-6$ source and destination nodes. Below in Table 4.2 the data can be seen.

Table 4.2: Simulation run performed with 100 time steps, topology 1e/1i, High traffic load and packet loss in percentage cutoff at 5%.

No. Source	No. Destination	Buffer size	Packet loss %
3	1	115	5.0
4	1	189	2.0
5	1	263	5.0
6	1	332	5.0
1	3	1	0.0
1	4	1	0.0
1	5	1	0.0
1	6	1	0.0

Below in Figure 4.8 the data is plotted.

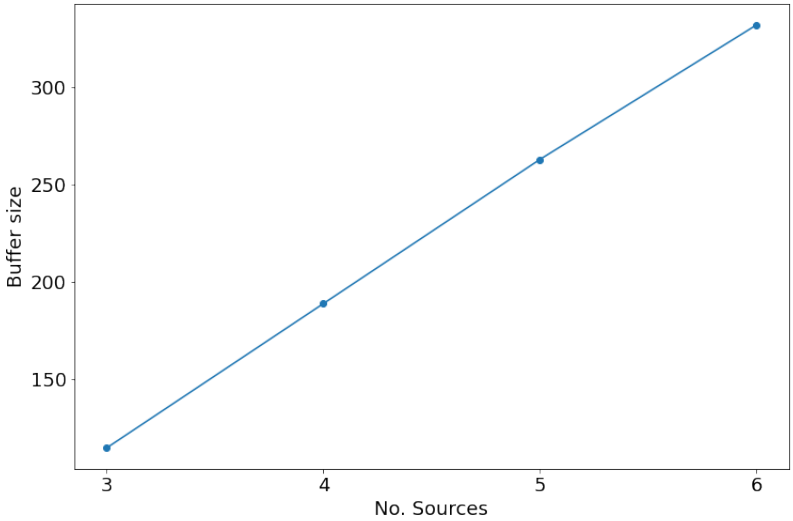

Figure 4.8: Buffer size plotted as a function of number of sources.

Two conclusions can be drawn from this simulation run:

- Do not need to simulate downlink network for these type of networks, i.e., only need to change number of sources. Number of destination nodes can be kept at one.
- Buffer size is linearly dependent on the amount of packets being sent.

The linearly dependency shows that a linear regression model would suit this data well, the GLM model Poisson Regression is chosen to try to optimize a network for a given traffic load.

4.2.2 Generating data

The goal is to optimize the network for a given traffic load. First which variables to generate data for is chosen. Packet loss will be the response variable for the regression model. Number of sources are ranged between 3 to 6 since this amount will cover many normal RBS installations today. Two different topologies are chosen: 1e/1i and 1e/2ic. The other two topologies (2e/1i and 2e/2ic) were not implemented properly because of time constraints. Three different traffic loads. Finally a range of different buffer sizes with a middle point of 830 stated as relevant size of buffer in Chapter 3.1.4. The range is defined as: 20 numbers which ranges between 83 - 1660 increases with 5% (rounded to nearest integer), i.e., Buffer sizes:= $\{1660 * 0.05, 1660 * 0.10, 1660 * 0.15, \ldots, 1660 * 1.00\}$. The different inputs are listed below:

- Number of sources: 3-6
- Topology: 1e/1i, 1e/2ic
- Traffic load: Low, medium and High
- Buffer sizes: $\{83, 166, 249, \ldots, 1660\}$.

The total amount of runs for all variables is simply the amount of each multiplied together:

$$[\texttt{No.sources}] * [\texttt{No.topologies}] * [\texttt{No.traffic loads}] * [\texttt{No.buffer sizes}] = 4 * 2 * 3 * 20 = 480$$

Each simulation takes approximately 15 seconds to run so for a total of 480 runs. This means that with each simulation taking 15 seconds, given in 4.1.4, the total data generation takes about 2 hours.
Below in Figure 4.9 one result is shown for a `1e/1i` network with low traffic. Here the packet loss can clearly be seen as linearly dependent on buffer size.

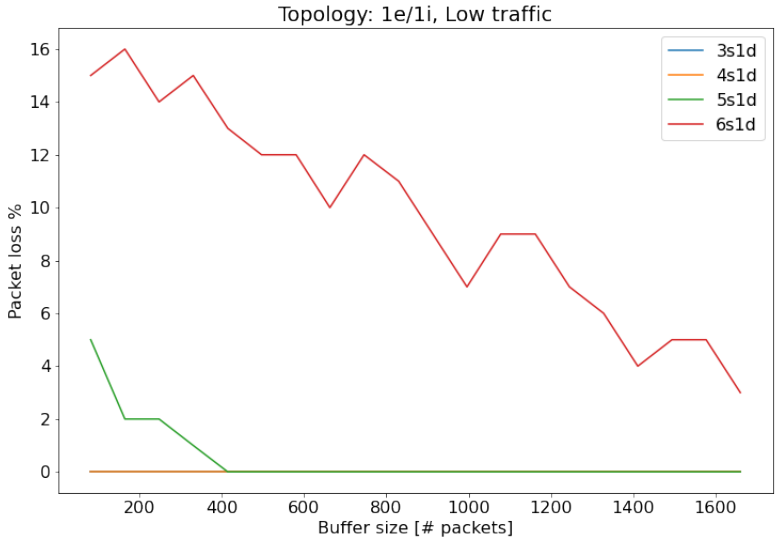

Figure 4.9: Low traffic with a `1e/1i` network with packet loss in percentage as a function of buffer size.

The rest of the figures for the generated data with different topologies and traffic load can be seen in Appendix B.

4.2.3 Poisson Regression Model

Implementation of the Poisson regression model is done through Scikit-learn in Python. First the data is normalized through the method `StandardScalar()` which standardizes the inputs with the function: $z = (x - u)/s$ where u is the mean of the input, s the standard deviation and x the sample input and z the standardized value. Then data is split in to 60% training and 40% testing data randomly. Then the data is fitted for the model. The explanatory variables are:

$$
\begin{aligned}
x_1 &= Traffic load \\
x_2 &= Sources \\
x_3 &= Topology \\
x_4 &= Buffer size
\end{aligned}
$$

The resulting coefficients for each of the inputs can be seen in 4.3 below:

$$
\begin{aligned}
\beta_0 &= 2.954 \\
\beta_1 &= 1.021 \\
\beta_2 &= 0.308 \\
\beta_3 &= -0.009 \\
\beta_4 &= -0.063
\end{aligned}
\tag{4.3}
$$

Then plugging that in to Equation 2.24, the model becomes the following Equation 4.4 (here the logarithm has been taken for readability):

$$
log(p_{loss}) = 2.954 + 1.021x_1 + 0.308x_2 - 0.009x_3 - 0.063x_4 \tag{4.4}
$$

The model was then tested using the D^2 score described in chapter 2.7.1. The remainder of the data which wasn't used to fit the model is tested. With a D^2 score of 0.83, the model is shown to be an effective estimate of packet loss for the four explanatory variables.

5
Further Research

Here things are listed that could be further improved upon or investigated separately but closely related to this thesis work. Some of the suggestions have been discussed through theory and method, but not yet implemented.

- Delay, link speed and two topologies (2e/2i and 2e/2ic) were not added because of time constraint. Both theory and implementation has been prepared for these however some things are incomplete or missing for them to work properly.
- Different traffic loads on different nodes, currently only one traffic load for the whole network is considered. Being able to change any node to a given traffic load would be more correlated to reality and therefore also beneficial for future design of RBSs.
- Generate more data, i.e., more buffer sizes and larger amount of source nodes to accommodate larger RBSs. Increasing the data amount would be more beneficial for any regression analysis made.
- Analyze the data with other regression models, e.g. General Additive Model for Poisson Regression.
- More traffic models which could fill the gap between low, medium and high could be added to further mimic real-life applications.
- Combining several RBSs with both eCPRI and CPRI in the simulator.
- Using the simulator with an implementation of a deep reinforcement learning.

6
Conclusion

It has been shown within this thesis that it is possible to define an RBS mathematically which fulfills telecommunication performance requirements as a small network through graph and optimization theory. It has been proven that a two state Markov chain was appropriate for generating traffic and that average intensity of traffic is given by the stationary Markov chain.
It has been proven that an implementation of the ethernet simulator in Python works. The ethernet simulator was proven valid for small time steps through comparisons between hand calculations and output data. The ethernet simulator was also proven to be efficient with five iterations per run with 100 000 time steps for each iteration.
It has been shown that a Poisson regression (GLM) was a fitting machine learning model for the data generated by the ethernet simulator. The ingoing variables was four: Number of sources, topology, traffic load and buffer size. The response variable was the packet loss in percentage. The model was validated with a D^2 score of ≈ 0.8.

Bibliography

[1] E. Trojer et al., "Packet fronthaul – design choices towards versatile RAN deployments", Ericsson.com. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/white-papers/packet-fronthaul-design-choices. [Accessed: 18- Apr- 2022].

[2] Ralph Santitoro (2003). "Metro Ethernet Services – A Technical Overview" (PDF). mef.net. https://web.archive.org/web/20181222184046/http://www.mef.net/Assets/White_Papers/Metro-Ethernet-Services.pdf archived on December 22, 2018. [Accessed January 19 May 2022].

[3] "Common Public Radio Interface", Cpri.info, 2022. [Online]. Available: http://www.cpri.info/spec.html. [Accessed: 18- Apr- 2022].

[4] H. C. Tijms, A First Course in Stochastic Models. Amsterdam: John Wiley & Sons Ltd, 2003.

[5] Cormen, T., Leiserson, C., Rivest, R. and Stein, C., 2009. Introduction to algorithms. 3rd ed. Cambridge, Massachusetts. London, England.: The MIT Press.

[6] Enger, J. and Grandell, J., 2022. Markov Kompendium. Stockholm: Kungliga Tekniska Högskolan.

[7] Pishro-Nik, H., 2014. Introduction to Probability, Statistics, and Random Processes. 1st ed. Massachusetts: Kappa Research.

[8] scikit-learn. 2022. 3.3. Metrics and scoring: quantifying the quality of predictions. [online] Available at: https://scikit-learn.org/stable/modules/model_evaluation.html#regression-metrics> [Accessed 29 May 2022].

[9] Rice, J., 2021. Mathematical statistics and data analysis. New Delhi: Cengage Learning.

[10] N. Daigle, J., 2005. Queueing Theory with Applications to Packet Telecommunication. 1st ed. Mississippi: Springer.

[11] S. Keshav, Mathematical Foundations of Computer Networking. Waterloo: Addison-Wesley, 2012.

[12] "Ethernet frame: definition and variants of the frame format", IONOS Digitalguide, 2022. [Online]. Available: https://www.ionos.com/digitalguide/server/know-how/ethernet-frame/. [Accessed: 03- May- 2022].

[13] "Router/Switch Buffer Size Issues", fasterdata.es.net, 2022. [Online]. Available: https://fasterdata.es.net/network-tuning/router-switch-buffer-size-issues/. [Accessed: 03- May- 2022].

[14] "packet buffers", People.ucsc.edu, 2022. [Online]. Available: https://people.ucsc.edu/~warner/buffer.html. [Accessed: 03- May- 2022].

[15] Limited, R., 2022. SFP Price - Cisco Global Price List. [online] Itprice.com. Available at: https://itprice.com/cisco-gpl/sfp [Accessed 12 May 2022].

A

Appendix

```
B1(0)  0  0   0
B2(0)  0  0   0
B3(0)  0  0   -

B1(1)  0  1   0
B2(1)  0  0   0
B3(1)  0  0   -

B1(2)  1  1  -1
B2(2)  0  1   0
B3(2)  0  0   -

B1(3)  1  0  -1
B2(3)  1  1  -1
B3(3)  0  1   -

B1(4)  0  0   0
B2(4)  1  0  -1
B3(4)  1  1   -

B1(5)  0  0   0
B2(5)  0  0   0
B3(5)  2  0   -
```

Figure A.1: A printout from the Ethernet Simulator.

B

Appendix

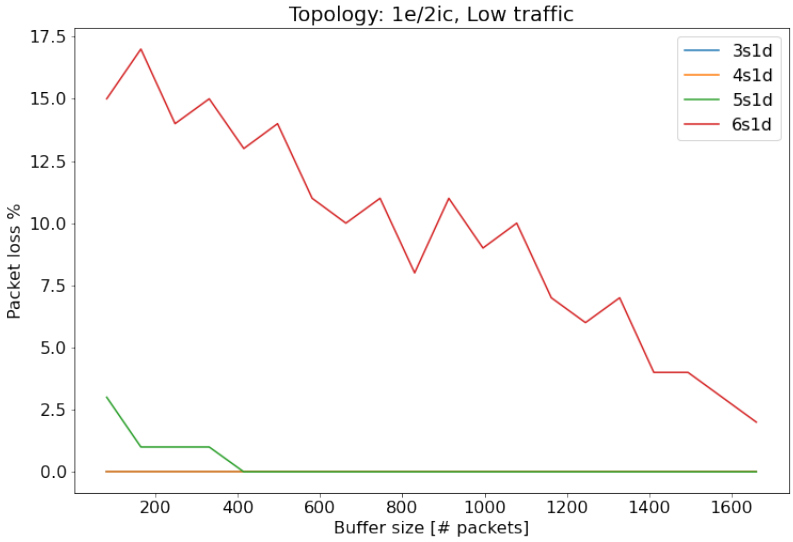

Figure B.1: Low traffic with a `1e/2ic` network with packet loss in percentage as a function of buffer size.

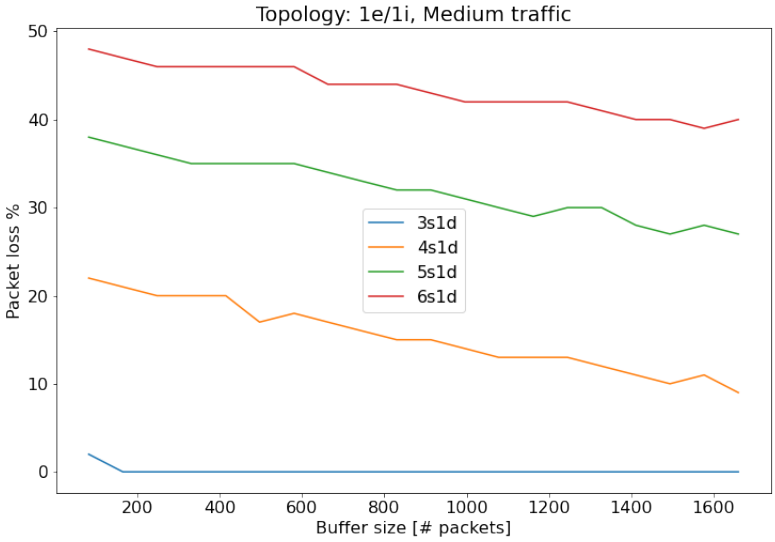

Figure B.2: Medium traffic with a `1e/1i` network with packet loss in percentage as a function of buffer size.

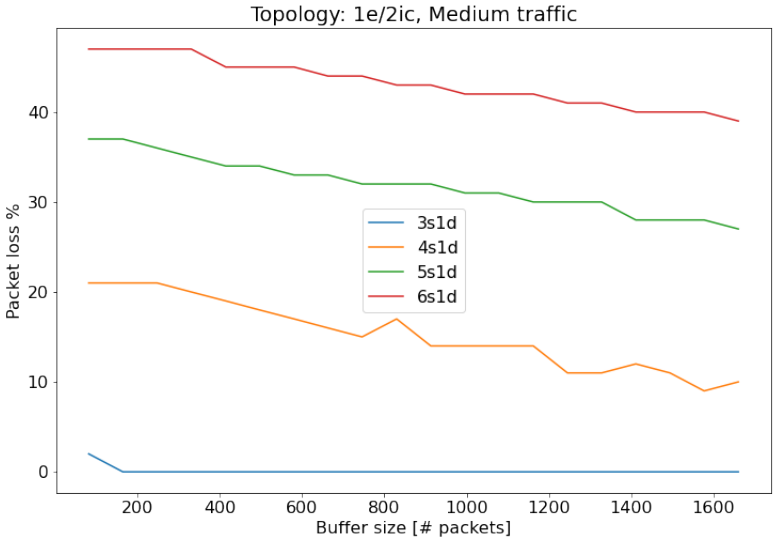

Figure B.3: Medium traffic with a `1e/2ic` network with packet loss in percentage as a function of buffer size.

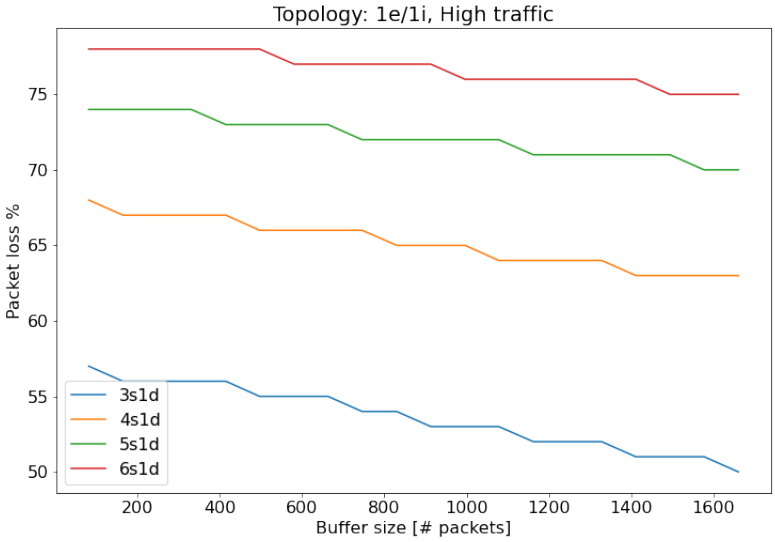

Figure B.4: High traffic with a `1e/1i` network with packet loss in percentage as a function of buffer size.

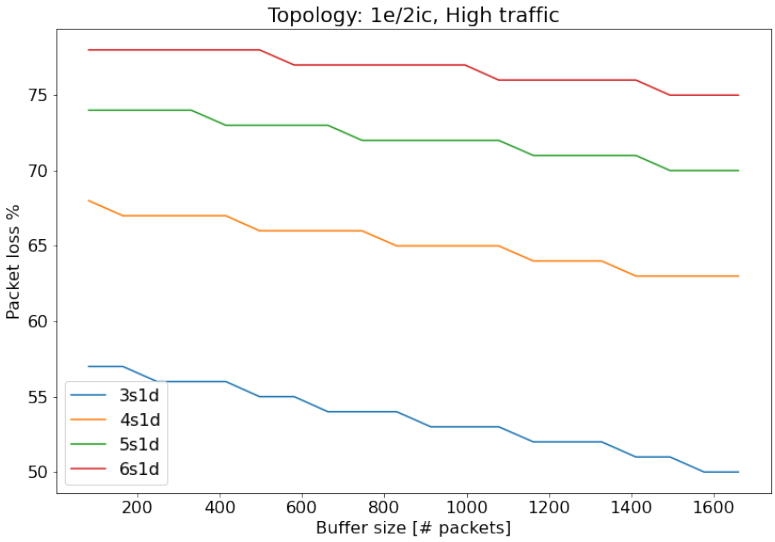

Figure B.5: High traffic with a `1e/2ic` network with packet loss in percentage as a function of buffer size.

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY