



Hochschule für Technik  
und Wirtschaft Berlin

University of Applied Sciences

---

# Analysis of Communication Networks Using Simulators OMNET++ and NS3

## Master Thesis

from

**Majd Rekik**

Student Id: s0560679

Department 1 –Energy and Information –  
of the University of Applied Sciences Berlin

of the requirement for the degree of  
**Master of Engineering (M. Eng.)**

in the study programme

**INFORMATION AND COMMUNICATION TECHNOLOGY**

Date of submission: 2022-10-04

First Supervisor: Prof. Dr.-Ing. Christoph Lange

Second Supervisor: Dipl.-Ing. Lutz Molle

---



# Contents

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Introduction</b>                                  | <b>1</b>  |
| 1.1      | Motivation . . . . .                                 | 1         |
| 1.2      | Research Aims and Objectives . . . . .               | 3         |
| 1.2.1    | Research Aim . . . . .                               | 3         |
| 1.2.2    | Research Objectives . . . . .                        | 4         |
| 1.3      | Thesis Structure . . . . .                           | 4         |
| <b>2</b> | <b>Background of The Study</b>                       | <b>5</b>  |
| 2.1      | Computer Networks . . . . .                          | 5         |
| 2.1.1    | Types of Nodes . . . . .                             | 5         |
| 2.1.2    | Network Classification . . . . .                     | 6         |
| 2.1.3    | Network Topologies . . . . .                         | 7         |
| 2.1.4    | Protocols and Layers . . . . .                       | 11        |
| 2.2      | Reference Models . . . . .                           | 12        |
| 2.2.1    | Physical Layer . . . . .                             | 13        |
| 2.2.2    | Data link layer . . . . .                            | 13        |
| 2.2.3    | Network Layer . . . . .                              | 15        |
| 2.2.4    | Transport Layer . . . . .                            | 15        |
| 2.2.5    | Session layer . . . . .                              | 17        |
| 2.2.6    | Presentation Layer . . . . .                         | 17        |
| 2.2.7    | Application layer . . . . .                          | 17        |
| 2.3      | Network Elements . . . . .                           | 18        |
| 2.4      | Network Analysis Criteria . . . . .                  | 19        |
| 2.4.1    | Packet Loss Ratio . . . . .                          | 20        |
| 2.4.2    | End-to-end delay . . . . .                           | 20        |
| 2.4.3    | Jitter . . . . .                                     | 22        |
| 2.4.4    | Throughput . . . . .                                 | 22        |
| <b>3</b> | <b>Network Simulation Tools</b>                      | <b>23</b> |
| 3.1      | Fundamental concepts in network simulators . . . . . | 23        |
| 3.1.1    | Network Simulation Tools . . . . .                   | 23        |

|          |  |           |
|----------|--|-----------|
| 3.1.2    | Simulation and Emulation . . . . .                         | 24        |
| 3.2      | Network Simulators Type . . . . .                          | 24        |
| 3.2.1    | Simple and complex . . . . .                               | 24        |
| 3.2.2    | Free and Commercial . . . . .                              | 24        |
| 3.3      | Presentation of NS-3 Simulator . . . . .                   | 25        |
| 3.3.1    | NS-3 Description . . . . .                                 | 25        |
| 3.3.2    | NS-3 Simulator Modules . . . . .                           | 25        |
| 3.3.3    | NS-3 Terms . . . . .                                       | 27        |
| 3.4      | Presentation of OMNET++ Simulator . . . . .                | 28        |
| 3.4.1    | OMNET++ Description . . . . .                              | 28        |
| 3.4.2    | OMNET++ Architecture . . . . .                             | 28        |
| 3.4.3    | OMNET++ Main Files . . . . .                               | 29        |
| 3.4.4    | OMNET++ Framework . . . . .                                | 30        |
| 3.5      | Comparison of OMNeT++ and NS3 Network Simulators . . . . . | 30        |
| <b>4</b> | <b>Experiments and analysis</b>                            | <b>35</b> |
| 4.1      | Network Model . . . . .                                    | 35        |
| 4.2      | Results of The Simulation in OMNET++ and NS3 . . . . .     | 38        |
| 4.2.1    | Packet Loss Ratio . . . . .                                | 38        |
| 4.2.2    | End-to-end Delay . . . . .                                 | 40        |
| 4.2.3    | Jitter . . . . .   | 41        |
| 4.2.4    | Network Collisions . . . . .                               | 42        |
| 4.2.5    | Network Throughput . . . . .                               | 43        |
| 4.2.6    | Number of Simulation Events . . . . .                      | 46        |
| 4.2.7    | Simulation Length . . . . .                                | 47        |
| 4.3      | Discussion . . . . .                                       | 48        |
| <b>5</b> | <b>Conclusion and Future Work</b>                          | <b>51</b> |
| 5.1      | Conclusion . . . . .                                       | 51        |
| 5.2      | Future Work . . . . .                                      | 52        |
| <b>A</b> | <b>NS3</b>   | <b>53</b> |
| A.1      | NS3 Installation . . . . .                                 | 53        |
| <b>B</b> | <b>OMNET++</b>   | <b>61</b> |
| B.1      | OMNET++ Installation . . . . .                             | 61        |
|          | <b>Abbreviations</b>                                       | <b>65</b> |
|          | <b>List of Figures</b>                                     | <b>67</b> |

|  |           |
|--|-----------|
| <b>List of Tables</b>                  | <b>69</b> |
| <b>Listings</b>                        | <b>71</b> |
| <b>Bibliography</b>                    | <b>73</b> |
| <b>Statement of Academic Integrity</b> | <b>77</b> |

## Abstract

Telecommunication networks are one of the foundations of modern world technology. Therefore, studying the theoretical foundations of this field helps propose more efficient network topologies and protocols. For this purpose, network simulators are helpful in the communication system field. They provide a complete and cost-effective environment for testing, developing, and evaluating network topologies and protocols.

Simulators may have similarities and differences in cost, design, programming languages, performance, customization, accuracy, and implementation details. Awareness of these differences is essential as they can be critical in choosing which simulator to use.

In this paper, communication networks are analyzed using NS3 and OMNET++. The two simulators are studied and compared based on network performance criteria such as packet loss rate, throughput, end-to-end delay, jitter, collisions, number of events, and simulation time.

The study consists of two main parts. The theoretical part provides information about computer networks and the mathematical understanding of the various performance criteria used to analyze computer networks. In addition, the technical specifications of the two simulators are presented, including architecture, programming languages, and programming libraries.

The last section provides the details of the experiment. Based on the simulation results, the behavior of the two simulators is compared and analyzed. Finally, the results are summarized, and future perspectives for the study are given.

**Keywords:** NS3, OMNET++, communication networks, computer networks, TCP/IP, PPP protocol, CSMA/CD protocol, packet loss rate, throughput, end-to-end delay, jitter, collisions, number of events, and simulation time.

# Chapter 1: Introduction

In this chapter, the motivation and objectives of our research are presented. Section 1.1 first presents the motivation for this master thesis. Then, the research objectives are described in Section 1.2. Finally, Section 1.3 concludes this chapter by presenting the organization of the rest of the paper.

## 1.1 Motivation

Research in communications and computer networks has increased significantly due to the constant advancement of computer technology. As a result, computer networks have become a focus for researchers and network specialists who continuously develop and explore new and old network topologies, architectures, protocols, and technologies.

When setting up a novel computer network or protocol, efficiency, reliability, and performance are often the primary concerns, especially for large-scale applications such as cloud infrastructures or low-end devices where resources are limited. Therefore, it is necessary to test and study the proposed setup in multiple scenarios to verify the quality of the connections and the speed of data transmission.

A model is helpful because its efficiency could be studied without building the network. In addition, if the model is simple, it may be possible to use analytical methods to obtain relatively authentic values for performance metrics. However, using analytical approaches with a sophisticated model is not feasible due to the complexity and dynamics of multiple scenarios. Thus, simulation is considered a fast, cost-effective, and more flexible methodology for exploring new network protocols, algorithms, and improvements to the old [46].

Simulating communication networks may also have other reasons, such as validating networks for malware behavior, testing the performance of anti-malware software, or its impact on the network[30]. For these purposes, the accuracy of network simulators

is of great importance. Moreover, we are entering an era of high-bandwidth networks, internet speeds are increasing worldwide, and the Internet of Things is rising, where simple devices communicate in real time and require high bandwidth. They use a variety of specific protocols [47].

This means that the landscape of communication networks is changing, and technical specifications are also changing. In response, major efforts are being made to develop new protocols or improve old ones. Therefore, network simulators that are more than a decade old should be continuously improved and adapted to the new requirements. Moreover, continuous criticism can help the development of network simulators to go in the right direction.

In all the above cases, using test environments instead of simulations is always an option but often not practical because test environments have significant drawbacks. For example, they are often less reproducible, challenging to configure, and very hard to use, even though they are potentially more realistic. Therefore, simulations are considered better methodology because they are more reproducible, easier to configure, and easier to implement, even though they may have limited accuracy[46].

The use of network simulators is usually cyclical. Figure 1.1 illustrates The cycle of network simulation.

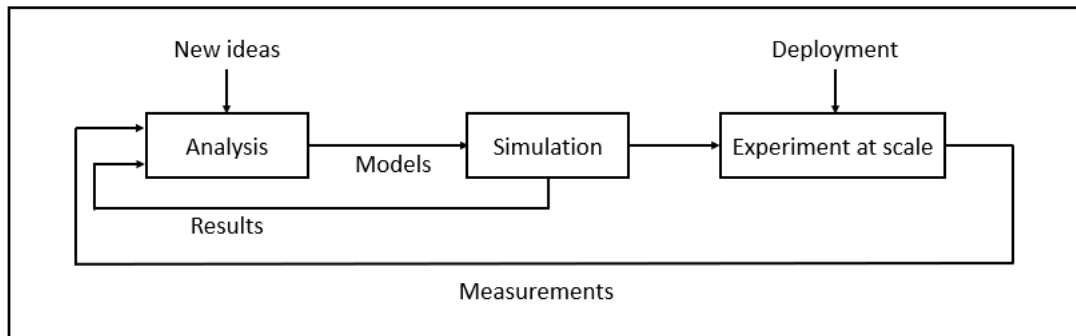


Figure 1.1: Simulation cycle [26]

It starts with an analysis phase, during which ideas are collected, and brainstorming takes place. These ideas are then converted into models, and the models are used in the simulation. After the simulation of the proposed model, the results are collected. These results can then be used for a new brainstorming session. The process can keep looping until the development team decides to proceed with the deployment. The previously simulated model can now be implemented in test environments, and new measurements can be made. Those measurements can be helpful for further analysis



and brainstorming, providing new ideas for improving the model. The whole process can continue in a loop, and the network can achieve better performance[26].

Choosing an appropriate network simulation program for a particular task can be very critical and confusing, as there are a variety of network simulators on the market. There are commercial simulators and open source simulators. The most popular simulators are NS2, NS3, OPNET, OMNET++, NetSim, REAL, QualNet, and J-Sim [25].

In addition, network simulators can use different programming languages for network design, protocol implementation, and flow control handling. The choice of network simulators depends on several factors, such as the performance and memory consumption of network simulators for large networks. Other factors include scalability, reliability, and debugging for different network models and protocols, programming languages, available support for different network components, and documentation [46].

## **1.2 Research Aims and Objectives**

### **1.2.1 Research Aim**

The thesis's main objective is to analyze communication networks using different network simulators. The network simulators used in this thesis are OMNET++ (Objective Modular Network Testbed in C++) and NS3 (Network Simulator version 3). In this work both simulator are investigated and compared theoretically and practically. In the experimental part, those simulators will be used to model a computer network and to calculate packet loss ratio, throughput, end-to-end delay, jitter, collisions, number of events, and simulation time. In the end, the findings will be collated, and then both simulators will be compared.

## 1.2.2 Research Objectives

The thesis has the following objectives:

- Development of the necessary knowledge of the communication network includes understanding the network layer model, basic network topologies and protocols, and traffic models.
- Explaining the steps of Installation of the software OMNET++ and NS3 and their theoretical analysis.
- Measure quantifiable metrics from the simulated model , packet loss ratio, throughput, end-to-end delay, jitter, collisions, number of events, and simulation time.
- Analysis and discussion of the extracted results.
- Create a simulation environment that can be used for future studies.

## 1.3 Thesis Structure

This report will be divided into five chapters, including the introduction. This introductory chapter explains the motivations that led to this work, followed by the thesis's objectives and structure.

The **second chapter** presents the background of our work. The necessary terms such as computer networks standards, technologies, parameters and performance criteria used when dealing with computer networks computer network analysis are presented in more details.

The **third chapter** deals with OMNET++ and NS-3. Both simulators are separately presented in terms of architecture, specification, properties and installation steps. This chapter is concluded with comprehensive comparison between the two simulators.

Next comes **chapter four** deals with evaluation , analysis and discussion of the of the results obtained from the simulation.

The **chapter five** the results of this work are finally summarized and an outlook on possible extensions is given.

# Chapter 2: Background of The Study

This work aims to examine and compare two simulation tools. However, before looking at these tools, it is necessary to clarify what computer networks mean and what standards, technologies, and parameters must be considered when dealing with them. It is also necessary to decide which criteria will be used for this comparison and analysis. All this will be explained in the following chapter.

## 2.1 Computer Networks

Computer network is a collection of interconnected hardware devices called nodes that exchange data. This data is transmitted over a channel, which is either a wired channel that transmits an electrical signal or a wireless channel that transmits electromagnetic waves. In order to exchange data between nodes, a set of rules called protocols are required for all nodes involved [35].

### 2.1.1 Types of Nodes

The devices in a network, called nodes, can send, receive, redirect, or perform all of the above functions over the network. In a network, some nodes may have a specific role in connecting other nodes, such as switch, repeater, bridge, hub, and router. They serve only to assemble the network and control its flow. Hosts are computers that use the network and send and/or receive data when needed. They have an integrated card specifically designed to connect to a network. This card is called a network interface card (NIC)[35].

### 2.1.2 Network Classification

Computer networks have been classified according to the Network's range of operations. Connected nodes in a few meters form a personal area network (PAN) . The local area network (LAN) is within a range of a thousand meters. Connecting several local area networks form a metropolitan area network (MAN). Furthermore, networks ranging over a thousand kilometers are considered Wide Area Networks (WAN).

#### • Personal Area Network(PAN)

Personal devices connected within a few meters are referred to as a personal area network (PAN). A PAN allows communication among personal devices such as computers, phones, peripheral devices, and video game consoles and for connectivity to higher-level networks such as the Internet [35].

Personal networks can be wired, using computer buses such as USB to connect printers, computer mouse, or keyboards to a computer, or wireless, such as a wireless PAN (WPAN), using radio technologies such as Bluetooth[35].

#### • Local Area Network (LAN)

A local area network (LAN) is located within a 1 km radius, such as in a building, home, office, or university [10]. The LAN can take a wired form using technologies such as Token Ring, Ethernet, and Fast Ethernet [10]te, or a wireless form such as IEEE 802.11a, 802.11b, and 802.11g [44]. LANs allow the exchange of resources such as files or hardware equipment that multiple users may need.

#### • Metropolitan Area Network (MAN)

The Metropolitan Area Network (MAN) is a more extensive computer network than the LAN, which covers an area of 5 to 50 km, such as large campuses and cities. MAN's examples: Network of a telephone company that offers its customers a high-speed DSL and cable TV network. That requires particular technologies such as ATM, FDDI, and SMDB to transmit video, voice, and data [35] [12].

- **Wide Area Network (WAN)**

A WAN covers a large geographic area, such as communication lines that extend beyond the borders of large cities, regions, or countries. The Internet itself is the ultimate example of a WAN. Multiple LANs and MANs can be connected using equipment such as bridges, routers, or gateways, allowing them to share and exchange data easily [13]. WANs can be established to interconnect multiple private networks within an organization at different locations [35] [12].

### 2.1.3 Network Topologies

Given a number  $N$  of nodes, these can be connected arbitrarily to form a network. However, the resulting networks can be unreliable, inefficient, and very complex, leading to the development of different forms of options with different strengths and weaknesses. These options are called topologies, in which the establishment of network nodes and traffic flows are defined. The following section reviews the most typically used network topologies.

- **Point to point topology**

Point-to-point topology is the most basic network topology. It consists of two nodes connected directly via a single medium, as shown in figure 2.1 [15].

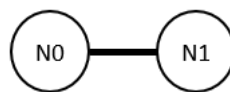


Figure 2.1: Point to point topology [27]

This topology can be expanded to a line topology network or a daisy chain topology where several nodes are connected in a line topology. When sending data from one node to another, the data must pass through all the nodes until it arrives at the destination. This is very risky if a node fails, as there is no alternative path. as shown in figure 2.2 [15] [32].

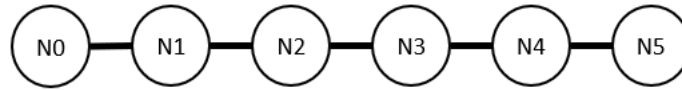


Figure 2.2: Line topology [27]

### • Bus topology

The bus topology is the first type of wired network in which all nodes connected to the same medium over which data is transmitted. In the bus topology, only one node can send data. If two nodes attempt to send data simultaneously, collisions occur, resulting in data loss. The advantages of this topology are the simplicity and low cost of connecting a new element or node. In addition, if a node fails, the entire network is not damaged, but if a link fails, the network can smash because there is no alternative path, as shown in figure 2.3 [15] [32].

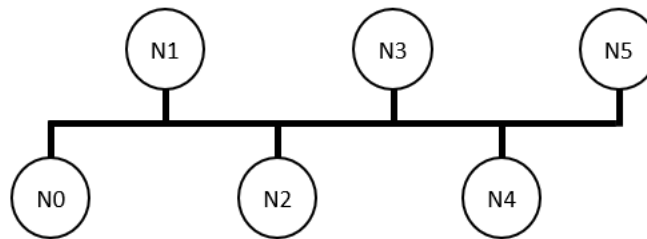


Figure 2.3: Bus topology [27]

### • Star topology

The star topology consists of different nodes connected to a central node, usually called a hub or switch. This central node is responsible for receiving data from the source node and forwarding it to the destination node. The point-to-point configuration ensures fault tolerance because the failure of one link isolates the node connected to it [15] [32]. However, if the core node fails, the entire network collapses, requiring troubleshooting or replacement. However, using a cable between the core node and every other node can result in extensive cabling, which increases installation costs, as shown in figure 2.4 [15].

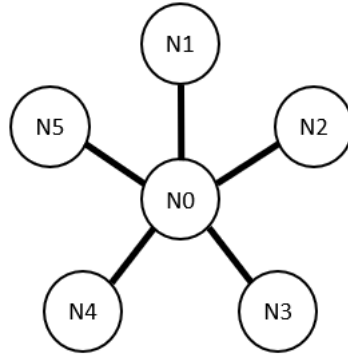


Figure 2.4: Star topology [27]

- **Ring topology**

Each node must be connected to two other nodes in a ring topology to form a closed loop or ring as shown in figure 2.5 . There is no central node to transmit data, so data is transmitted in a specific direction through all the nodes to find the destination node, which is inefficient. In a ring network, the failure of one node can interrupt data transmission [15] [32]. However, it is less sensitive than a bus network because this problem can be solved using a dual ring topology where each node will have four connected branches. However, while this makes the topology more robust against failures, it also increases the cost [15] [32].

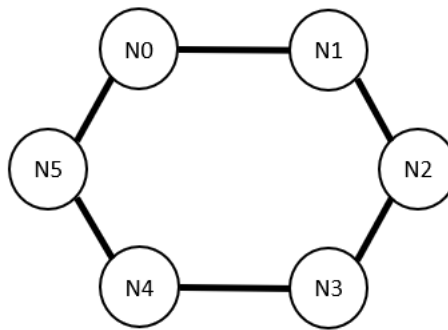


Figure 2.5: Ring topology [27]

- **Mesh topology**

In the mesh topology, each node is linked to all other nodes, which leads to a fully meshed topology, as shown in figure 2.6 , or a partially meshed topology see

figure 2.7 [15]. In the full mesh topology, no collisions occur since all nodes are directly linked to all other nodes in the network. This arrangement results in a robust network, as traffic can bypass a broken link. However, this requires a large amount of cable. Using a partially connected mesh network reduces the number of cables and the number of interfaces needed for each node, although it is not as resilient to a broken link [15].

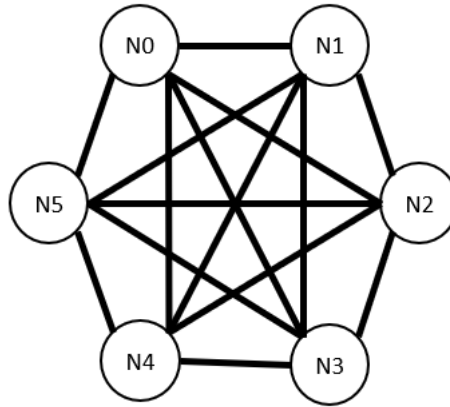


Figure 2.6: Fully mesh topology [27]

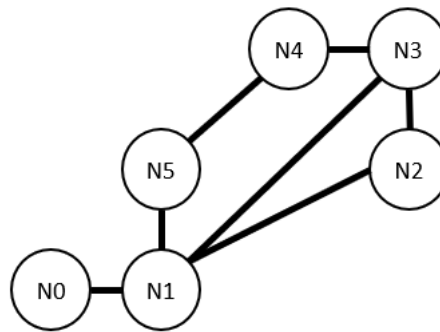


Figure 2.7: partially meshed topology [27]

### • Tree topology

A tree network is the best choice for a widely spread network with many branches. It can combine two or more star and bus networks. Every star network is considered a separate LAN where all nodes are connected to a central computer or server. This later is connected to the primary link, the bus, as shown in figure 2.8 [15] [32].



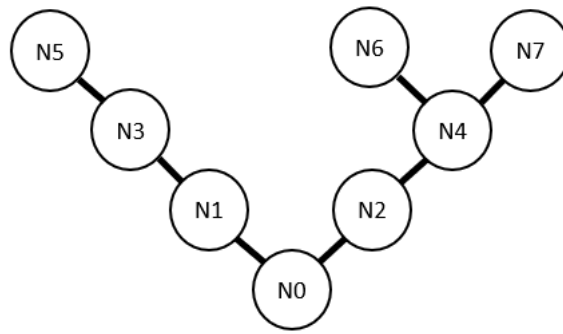


Figure 2.8: Tree topology [27]

### 2.1.4 Protocols and Layers

Communication between two nodes via a medium can be a complex process and is only confirmed when all nodes share a set of rules between them in order to exchange data. Those rules are called protocols. However, no single protocol can transmit data from one node to another [35]. Instead, different protocols must be combined together and cooperate to form a protocol suite. This suite is like a super-protocol, consisting of a set of protocols used at different levels. Therefore, the entire process is divided into multiple layers to overcome this limitation. Those layers are arranged on a vertical hierarchy stack. Each layer of this stack is superimposed on the layer below it, where each layer operates independently of the other (see Fig. 2.9).

The objective of each layer is to provide an interface for the upper and lower layer to use their services and operations. Protocols are responsible for controlling such services and operations. Each node in a network must follow a minimum set of layers to join the network [35].

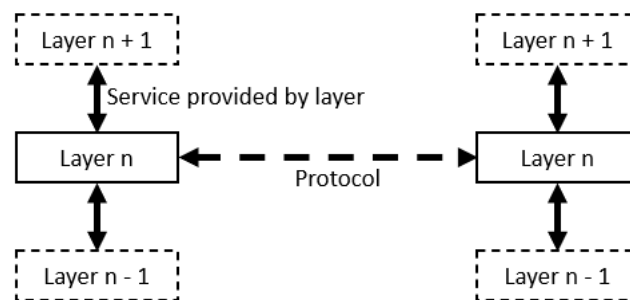


Figure 2.9: Relationship between a service and a protocol [35]

Protocols and layers solve an essential set of problems namely:

- Establish universal standards about how two hosts can communicate and how data is forwarded.
- Defining connection rules and conventions about how to react to specific events and what is expected from a node.
- Error detection and recovery mechanisms to tolerate physical noise that interferes with signals and possible hardware failures.

## 2.2 Reference Models

Computer networks operate under several protocols with different level of architecture. There are Two widely adapted network models which are the the OSI (Open System Interconnection) reference model and the TCP/IP reference model The OSI model is general and theoretical and is not used as standard manufacturers follow. On the other hand, the TCP/IP model protocols are widely used in practice[35].

The "Open System Interconnection Model" (OSI model) breaks down a network architecture into seven different layers, while the TCP/IP model consists of four layers. The figure 2.9 illustrates the OSI model and TCP/IP model. In addition, sample protocol examples for each layer were listed [35].

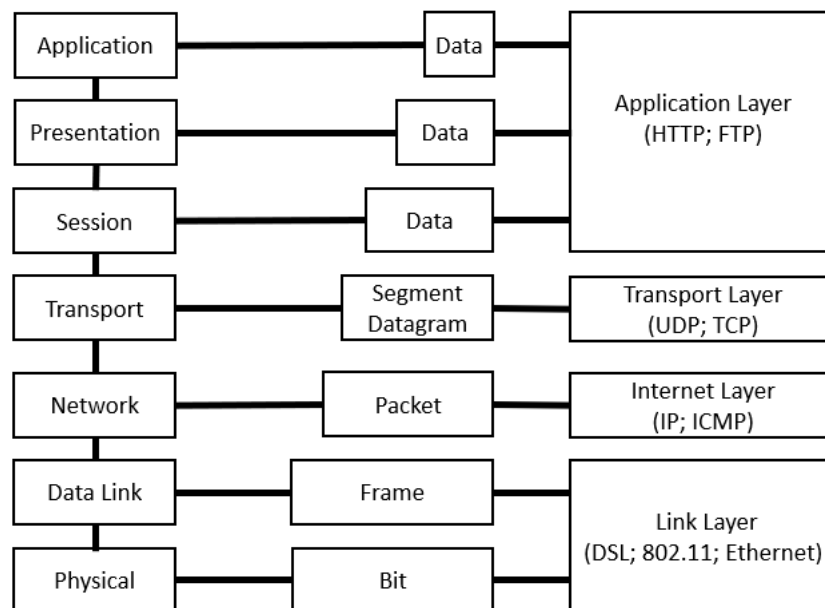


Figure 2.10: OSI and TCP/IP reference model [35]

### 2.2.1 Physical Layer

The Physical layer forms the infrastructure on which networks can be established. Its main task is to transmit digital information on a transmission medium. For example, the so-called "twisted pair cable" transmits electromagnetic signals via four twisted copper wires in wired networks [35] [12]. An alternative physical medium, the fiber-optic cable, transmits information by sending light pulses. The Feature of fiber optic cables is that they have less signal attenuation, allowing longer cables to be used. Besides, optical fiber cables are lighter, thinner, and not sensitive to electromagnetic interference. Therefore, cable types, lengths, and connections must be defined in this layer. Simultaneously it defines whether transmission can only take place in one direction (simplex), in two directions but only one at a time (half-duplex), or in both directions simultaneously (full-duplex) [35] [12].

The physical layer also includes the mechanical, electrical, functionality, and technical requirements for activating, maintaining, and deactivating physical connections for bit exchange between entities. It is designed to support the data link layer with specific services [35]. Some services of the physical layer are listed as follows:

- Specify physical interface requirements, including cables mechanical specification of the required speed, connectors specification like pin type, size, number, and configuration, and electrical properties such as the type of the signal, duration, and bandwidth.
- Controlling the quality of service metrics like the error rate, transmission rate, and transmission delay.
- Check the transmission flow and detect errors before sending it to the upper layer.

### 2.2.2 Data link layer

The data link layer uses the services described above, so it does not have to be concerned with the physical representation of the bits [35].

The data link layer's main task is to split the bits that flow from the physical layer to frames and detect and correct transmission errors if needed. In addition, there is data flow control to ensure that slower receivers are not affected by faster senders.

The Frames are given a checksum and sequence number. With the aid of this checksum, error correction can be easily integrated, and successively received frames can be placed back into the correct order. In addition, there is the regulation of the collision-free use of a shared communication medium [35] [12].

### • CSMA/CD protocol

CSMA/CD is a protocol that manages a network that uses a shared half duplex channel. Its name stands for Carrier Sense Multiple Access with collision detection [5]. In this protocol, every node senses the channel before transmitting the data. If the channel is idle, it begins sharing information. If not, it waits. While transmitting, the node keeps listening to the channel; if it detects that another node is also using the channel, it aborts the transmission and emits a jam signal, telling all nodes to stop transmission and apply the back-off algorithm [5]. figure 2.11 illustrates the flow diagram for the CSMA/CD protocol.

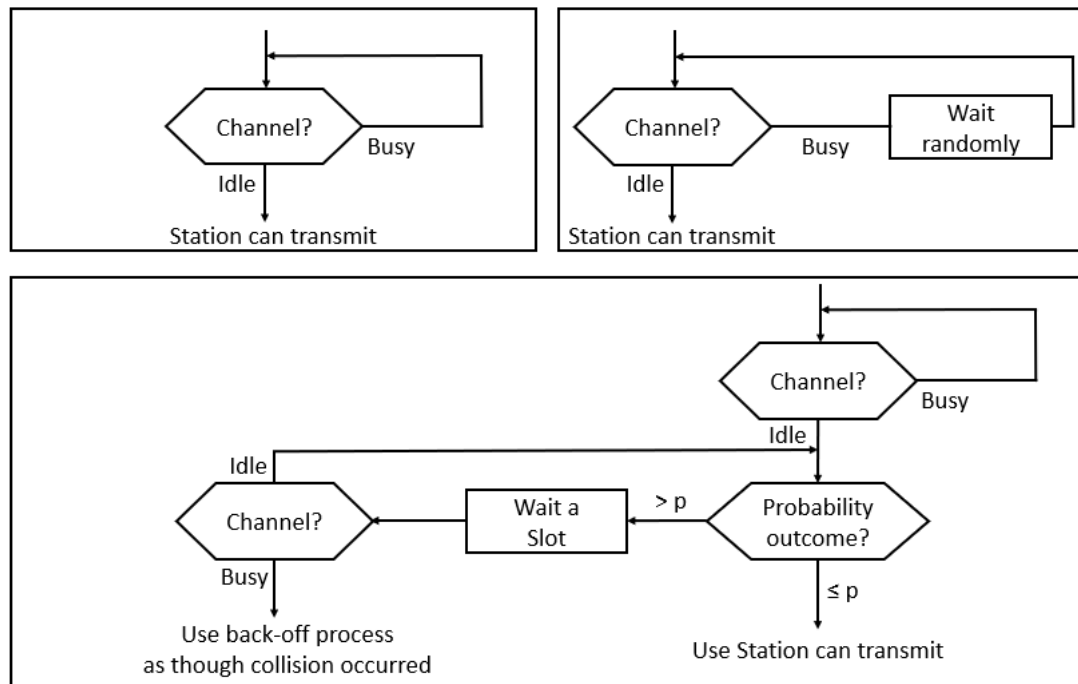


Figure 2.11: flow diagram for the CSMA/CD protocol [5]

- **Point to point protocol**

Point-to-Point Protocol (PPP) is a data link layer protocol between two nodes. It works only for peer-to-peer connections on full-duplex and cannot run on a shared channel since full-duplex channels cannot be shared between more than two machines [15].

### 2.2.3 Network Layer

Frames sent from the datalink layer are transmitted as packets from source to destination. The logical addressing of the individual components in the network takes place in the network layer [35] [12].

Unlike frames carried through the wire from one end to the other, packets must cross through several stations to their destination. The network topology has to be known to determine the appropriate path of the packets through the network. Thus this layer is responsible for routing, which includes tasks such as setting up and updating routing tables. The best-known protocol of this layer is the Internet Protocol (IP) [12].

### 2.2.4 Transport Layer

The transport layer provides complete communication between two participants. A transport service is available here, primarily for connection setup and disconnection. This ensures that the correct bits are transmitted. It is essential to maintain the sequence and not to send sequences twice [35].

Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are examples of transport layer protocols [35].

The Transmission Control Protocol (TCP) is connection-oriented and stands for reliable transmission between two applications; using TCP. One should act as a server and the other as a client. TCP features are auto error detection and correction. It also has unique algorithms to prevent overloads and control the data flow [35].

To be able to exchange data, a TCP connection must be set up first between the client's socket, which is the sender, and the receiver, which is the server. Sockets are platform-independent interfaces provided by the operating system between the

operating system's network protocol implementation and the application software. As a result, they are uniquely recognizable by an IP address and port number [35].

The client first sends a "synchronization-packet" to the server, which waits at the destination address. If the server receives this, it returns an "acknowledgment-packet". At this point, the client assumes the state "Connection-Established" and sends an "Acknowledgment-packet" to the server. Once this reaches the server, it switches to the "Connection-Established" state. The data packets to be exchanged are sent immediately after establishing the connection. The receipt of the data is confirmed with an acknowledgment packet; if the acknowledgment is not received, a new transmission is required. This guarantees that each data packet reaches its destination . Figure 2.12 illustrate TCP Connection Establishment [35] [28].

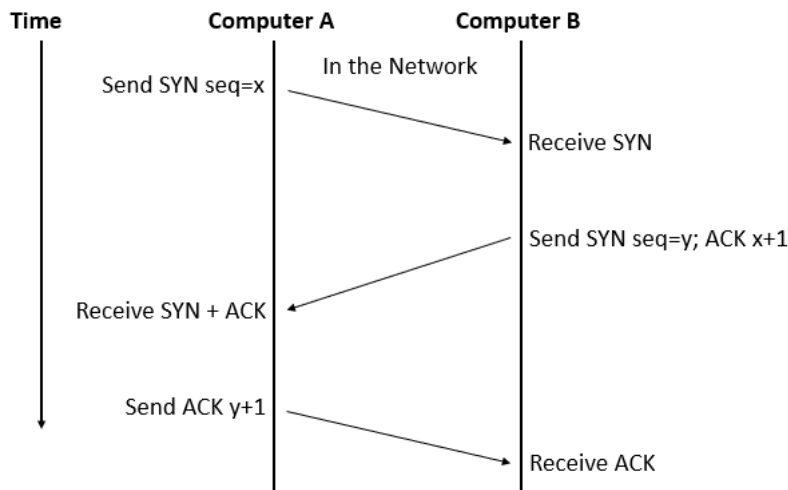


Figure 2.12: TCP Connection Establishment [28]

The closing of the connection starts with the client sending the "FIN-packet" to the server. The server acknowledges its receipt with the "acknowledgment-packet" and enters the "CLOSE-WAIT" state. Then it sends a "FIN-packet" back and waits for a final "Acknowledgment-packet" to be sent by the client. When this is sent, it closes the connection permanently, while the client will typically wait for another 30 seconds before closing its socket. Figure 2.13 illustrate TCP Connection Establishment [28] [35] .

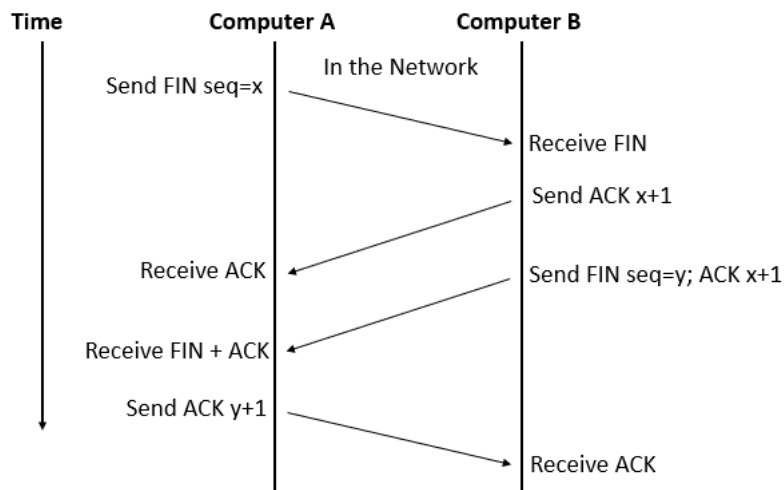


Figure 2.13: TCP Connection Termination [28]

### 2.2.5 Session layer

The session layer permits computers to use sessions when communicating. Among these are, for example, dialog control flow of the data streams and protection against simultaneous operation execution [35].

### 2.2.6 Presentation Layer

The Presentation layer sets the syntactic and semantic representation of the bits transmitted to guarantee that the data is securely transmitted in the network. This layer also performs encryption. Thus, it is not possible for unauthorized persons reading external data [35].

### 2.2.7 Application layer

Application layer is the final layer within the OSI model and is the one that the user interacts with directly through software application. A list of the services offered by this layer given as follows: determination of the data to be transmitted and Identification of the communication partners. Synchronization process of the communication. In this layer there is a wide range of protocols. Most well-known

are HyperText Transfer Protocol(HTTP), File Transfer Protocol (FTP), Domain Name System Protocol(DNS)[35].

### 2.3 Network Elements

Different kinds of network devices are required for communication among several computers in a computer network, in addition to physical media. Each device thereby acts on different layers [35] [18].

On the lowest OSI layer, the physical layer, hubs, and repeaters, for example, act to bring all connected devices together electrically. Each incoming signal is transmitted to all the output channels. Hubs and repeaters do not have logic and operate at the first OSI layer. The case with the use of hubs and repeaters is that they make the network topology a bus system; therefore, a collision will occur if two computers are transmitting simultaneously. In large computer networks, the bandwidth will be affected [35] [18].

A better efficient approach would be for the network device to pass each input signal only to the output channel where the target device would be served. This demands that network devices understand the frames and are qualified to decode the header information. Devices that are developed to do this are known as bridges or switches. These devices operate at the second OSI layer and have a different configuration from hubs. When different computers connected to a switch decide to communicate with each other, they can communicate simultaneously and with no fear of collision.

The routers, which process the (IP) packets, operate at the third OSI layer. The router software is only concerned with the packet headers and may not know how the packet arrived at it. A packet route is selected based on the information provided in the header and the ability to determine the fastest path between two network computers.

Communication between one computer and another in a computer network may involve many different network devices, and the actual data must travel through several layers in each case. Not every network device has to support all possible OSI layers. figure 2.14 gives an example of communication between two hosts via switch and router. Both hosts have all OSI layers implemented in a Simplified view, although switch, and router only operates on the second, and third lower OSI layers [35] [18].



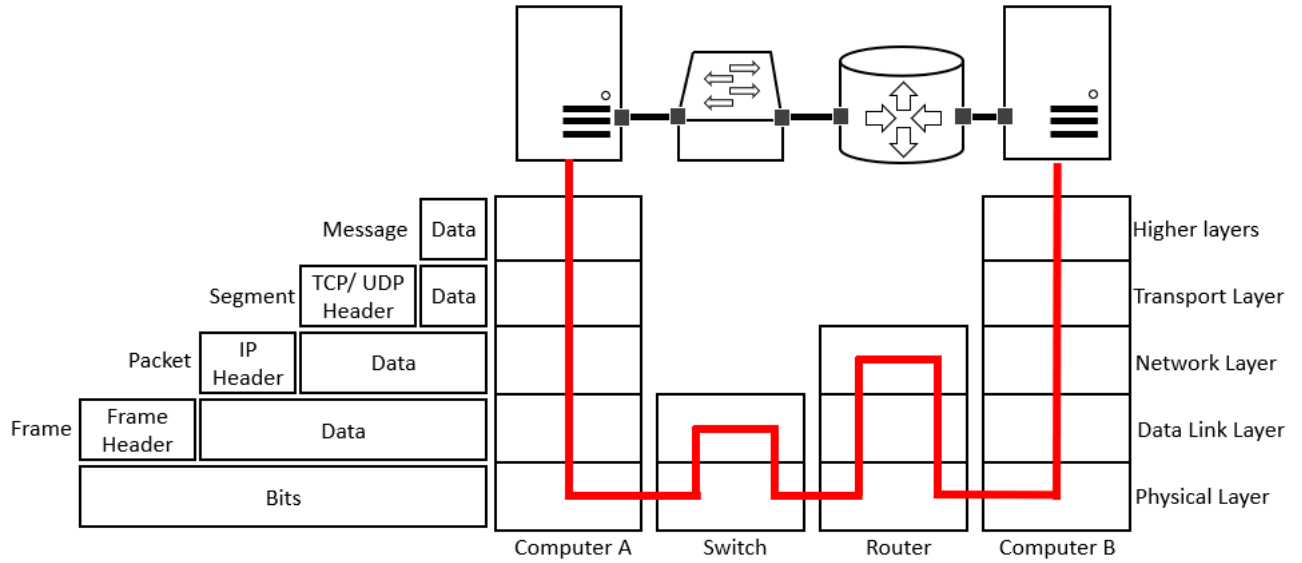


Figure 2.14: Example of data flow for different network components based on simplified OSI model [18]

Data sent from a computer A must pass through all the layers until it comes across the physical layer. Each layer hooks a specific header to the data (see figure 2.1). Once it reaches the physical layer of computer one, the data hence consists of the headers of the transport layer (TCP/UDP header), network layer (IP header), and data link layer (frame header). The switch can identify the output port of the frame and transmit it appropriately using the attached data link layer header. For the router to forward the packet to computer B, it must handle the network layer header to put the packet on the correct path. The data travels via all the layers of the switch and router until it arrives at computer B. At this point, it passes through all OSI layers computer B one more time, where the layer headers are discarded.

## 2.4 Network Analysis Criteria

Data flow between two nodes is sensitive and requires evaluation to ensure efficient productivity and responsiveness of network usage. According to the authors [22], the most commonly used performance criteria are packet loss ratio, end-to-end delay, jitter, and throughput. In the following subsections, a brief description of the different criteria is presented [29].

### 2.4.1 Packet Loss Ratio

Whenever a sender generates a traffic flow in the form of a sequence of data packets, there is the risk that some of them may never arrive at the source. As a result, the information received is less than it initially sent, known as packet loss. The loss of packets is the most significant indicator that something in the network goes wrong for many reasons, such as transmission error and Node load exceeding the buffer's capacity, leading to buffer overflow and device failure [16] [34]. The formula determines the packet loss:

$$Packet\ loss = \frac{\sum(Packets\ Sent - Packets\ Received)}{\sum Packets\ Sent} \times 100 \quad (2.1)$$

### 2.4.2 End-to-end delay

Delay, in general terms, describes an amount of time that elapses. A network's delay represents the time it requires for data to pass from one node to the destination node over the network. Network delays can lead to packet loss but cannot be totally eliminated. Processing delay, queuing delay, transmission delay, and propagation delay are some of the delays that can affect the system noticeably [16] [34].

- **Processing Delay**

The processing delay is the time required to inspect the IP packet header and identify the packet's final destination. There is no equation for the processing delay since it depends on the processing speed of the processor.

- **Transmission delay**

The transmission delay is how long it took to put the entire packet into the wire see figure 2.15 .



Figure 2.15: Transmission delay

It is affected by the transmission rate and the packet length. The following formula is used to measure:

$$DT = \frac{N}{R} \quad (2.2)$$

DT represents the transmission delay, N represents the packet length in bits, and R represents the transmission rate in bits/second.

#### • Propagation delay

Propagation delay refers to the time required to transmit packets over the cable see figure . This time depends on the medium used and the distance to be covered see figure 2.15 .

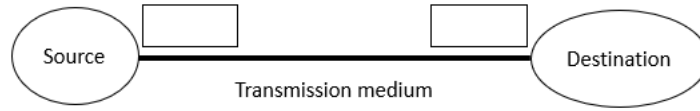


Figure 2.16: Propagation delay

The propagation delay is calculated as the distance between the sender and the receiver divided by the transmission speed. The formula for calculating propagation delay is:

$$DP = \frac{D}{S} \quad (2.3)$$

DP represents the Propagation delay, where D stands for the distance and S for the propagation speed of the medium.

#### • Queuing delay

It will occur if the link is occupied and there are other packets in transmission. The queuing time is based on the number of packets queued for transmission.

The delay in the queue depends on the following factors: If there is no packet in the queue waiting to be transmitted, the waiting time is very short compared to the case where packets are in the queue waiting to be transmitted. On the other hand, the more packets arrive, the longer the waiting time. Furthermore, packets might be sent at random time intervals. There is no simple formula for the queuing delay DQ.

In this work, the network delay is the sum of all the above measurements.

$$end - to - end - Delay = DP + DT + DQ \quad (2.4)$$

### 2.4.3 Jitter

With inconsistent delays and high packet loss, the arrival time of single packets is volatile. The jitter refers to the variation in delays over the network. Big changes in packet delay mean the network is not stable, and low jitter means the network is stable [16] [34].

$$Jitter = \frac{\sum variation\ delay}{\sum Packet\ received} \quad (2.5)$$

### 2.4.4 Throughput

The throughput is the measurement showing the speed at which data is transmitted through the network from the sender to the destination in a given period. The throughput is measured in bits per second (bps). The throughput measurement refers to the number of packets transmitted in one unit through the network. Having a network with high throughput is highly preferable [16] [34].

$$Throughput = \frac{\sum Received\ Packet\ Size}{\sum start\ time - \sum stop\ time} \quad (2.6)$$

# Chapter 3: Network Simulation Tools

In this chapter, the basic concepts of network simulation are presented. First, the main features, the programming languages, characteristics, and properties of network simulators OMNET++ and NS3 are presented. Then both simulators are compared in terms of platforms, API, user interface, and supported network types.

## 3.1 Fundamental concepts in network simulators

It is essential to understand the basic concepts of network simulators so they can exploit their benefits and produce results that are close to reality. The following subsection introduces the basic concepts [24].

In this section, basic concepts are presented.

### 3.1.1 Network Simulation Tools

A network simulator is a simulation tool that provides a user interface to users for defining a model that uses various network elements. This user interface can be either a command line or a graphical user interface (GUI). In the case of the command line interface, solid programming knowledge is needed, while the GUI is for new users with basic programming knowledge. Network simulators allow the users to model any real-world model in which the users can modify several network settings and then analyze the output. However, in practice, the network simulators are not ideal, and models may not match real-world behavior because of natural circumstances and technical issues. Nevertheless, network simulators can generate close results that provide the user with a helpful overview of the network being investigated and how parameter changes may affect its behavior [25].

#### 3.1.2 Simulation and Emulation

Simulation is a procedure through which a researcher can model real-world samples by implementing various network components, including nodes, routers, switches, physical links, and packets, and applying mathematical formulas for the evaluation. Simulation experiences can be carried out online or offline in a controlled environment, which can be monitored using a combination of different parameters and configurations [24]. The emulation is an extension of the simulations where the end devices like computers could be linked to the simulation models via emulators to behave as if they would be attached to the real network [24].

### 3.2 Network Simulators Type

Network simulators can be categorized by criteria such as simple or complex. and free or commercial tools

#### 3.2.1 Simple and complex

There are several diverse network simulators whose functionality differs from simple to more complex.

With the basic functionalities of simple simulators, the user can create simple topologies consisting of nodes and connections between these nodes, as well as the traffic generation in a network. Furthermore, the graphical user interface makes the work easy for the user, allowing them to visualize the simulation and use the existing network models with the aid of drag-and-drop facilities [24].

On the other hand, more complex network simulators offer users more freedom to explore and interact within the core of the simulation tools as they allow them to access the network protocols program where only experienced developers can effectively interact with them [24].

#### 3.2.2 Free and Commercial

The advantage of such free simulators is the fact that their source code is accessible to everyone. Users can explore the software core and improve its function or implement new features depending on their needs. In such a type of network simulator,

there is no united development control where different organizations and researchers contribute to developing new technologies and continuous changes to the software core. This leads to huge diversity, instabilities, and a lack of complete documentation that can result in serious problems. Free and open-source network simulators are NS2, NS3, OMNET++ and J-Sim [24] [6].

However, commercial network simulators are personal software whose source code is not shared with the public. Any implementation of new features or functions and source code managing is done only by the company that holds the simulators. The user must pay a certain amount of money for the software licenses. The benefit of commercial network simulators is that they are delivered with organized, structured, and up-to-date support documentation for the user. A team of experts from the company is constantly updating the software [24] [6].

## **3.3 Presentation of NS-3 Simulator**

### **3.3.1 NS-3 Description**

NS3 network simulator version 3 is an open-source discrete event simulator [31]. NS2, GTNets, and the YANS simulator inspired NS3 [9]. The development of NS3 started in 2006 by NSF CISE and INRIA, and the first official version was released in 2008[9]. The primary purpose of NS3 was to deliver a different software platform written in C++ and a python script interface to improve simulation performance[17]. Different organizations and researchers are continuously working on providing new telecommunication models and network protocol implementations, data link layers functionality, tracing features, and analytical methodologies. NS3 is widely used as a research tool for studying and evaluating protocol design, interactions, and large-scale systems[9]. Version 3.34 of NS-3 is installed, see Appendix A for complete installation.

### **3.3.2 NS-3 Simulator Modules**

NS-3 has several modules which can be modified and actually used to build a model of the communication system [9]. The structure of the NS-3 software is shown in Figure 3.1.

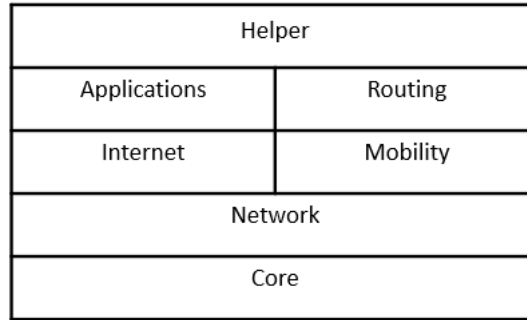


Figure 3.1: Software organization of NS3 [9]

- **Core Module**

Basic module, it is independent, it allows the creation of a hierarchical class structure coming from the basic Object class. This hierarchy has several functionalities specially designed to meet the needs of the simulation.

- **Network Module**

The network module is used to model the nodes, channels and network devices. It can contain several node and each node can include more than one NetDevice which is attached to a channel through which it sends and receives packets. A node can contain several protocol handlers to accept the packets received by the NetDevice. This module is the most important module in the simulator.

- **Mobility Module**

This module allows to define the position of the nodes and associates a movement model to the simulation.

- **Internet Module**

The classes in this module define the protocols UDP and TCP for internet layer of TCP/IP reference model.

- **Routing Module**

The classes in this module define the routing protocols that exist in NS-3.

- **Application Module**

Some applications are integrated and provided by NS-3. They are installed in nodes and can be started/stopped at specific times during the simulation.

- **Helper Module**

This model can be considered as a high level packaging. It facilitates the complex simulation scenarios and the installation of different modules in different modules in different agents.



### 3.3.3 NS-3 Terms

It is important to understand the meaning of the terms used in the simulator and the abstractions made. NS-3 uses terms that are widely used in networking but may have special meaning in NS-3 usage [9]. The architecture of NS-3 Node is shown in Figure 3.2.

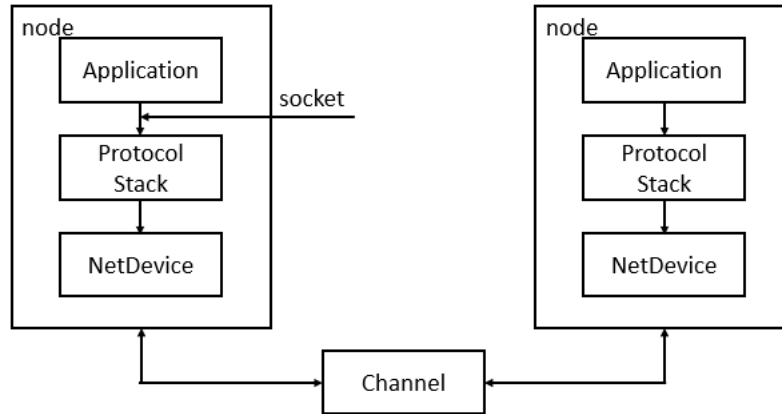


Figure 3.2: NS3 Node architecture [9]

The following are the main ones:

- **Node** represents any network element by the class Node. The configuration of a Node can be managed by adding components, protocols, applications to define the different network element such as host machine, router, switch from each other.
- **Application** object is needed to run the simulation. The exchange of packets during a simulation requires for example the description of the application within the participating nodes. For example in order to manage TCP communication we can use `TcpEchoClientApplication` and `TcpEchoServerApplication`.
- **Channel** describes the link that connects nodes, or more precisely the NET-devices installed in the nodes. Various class options are defined, such as `PointToPointChannel` to model an Ethernet link and `WifiChannel` to model a wireless link.
- **NetDevice** ensure the connection between the Nodes and the Channel.

- **Topology Helpers** is need to connect a large number of nodes to each other, this process can be very complicated. NS-3 provides TopologyHelpers to make such tasks easier.

## 3.4 Presentation of OMNET++ Simulator

### 3.4.1 OMNET++ Description

OMNET++ (Objective Modular Network Testbed in C++) is a modular, component-based C++ simulation library and framework, primarily for building network simulators [11]. It is generally used for research and development of communication networks. OMNET++ can also be useful in the simulation of an IT system, a queueing network and of hardware architectures [43]. This means that OMNET++ is not an independent simulator. It is considered as an infrastructure that allow creating a simulator. Unlike NS-3, OMNET++ has an IDE (Integrated development environment) and a GUI (Graphical User Interface) based on Eclipse to facilitate graphical network editing and animation. The graphical user interface and modular architecture make it easy to use OMNET++ for network simulation [7].

The Version 5.6.2 of OMNET++ is installed , see Appendix B for complete installation.

### 3.4.2 OMNET++ Architecture

The architecture of the OMNET++ model is composed of several hierarchically connected modules, as shown in the figure 3.3.

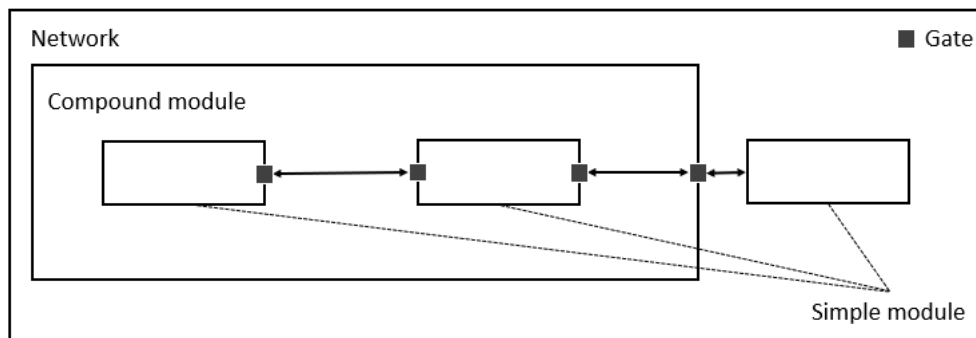


Figure 3.3: OMNET++ Architecture [42]

- **Simple module** programmed in C++ and used to define the real behavior of a system. For each simple module there is a .cc file and an .h file.
- **Compound module** composed of one or more simple modules or compound modules linked together. The parameters, ports and modules of each compound module are specified in a .ned file.

The communication between the different modules is done through message exchanges. The messages can represent packets, frames of a computer network. Messages are sent and received through ports called also gates which represent the input and output interfaces for each module [42].

The design of a network is done in the network description file (.ned file) and the different parameters of each module are specified in the configuration file (.ini file). At the end of each simulation, OMNET++ generates two new files omnet.vec and omnet.sca which allow for the analysis of simulation results [43].

### 3.4.3 OMNET++ Main Files

The OMNET++ program is based on a collection of modules, their parameters, their connections and the simple module behavior, are divided into different files to ensure the separation of topology and functionality. For every simulation program a NED file and a INI file is needed

- **File (.Ned)**  
OMNeT++ offers its unique DSL (Domain Specific Language) called NED to describe the simulation structure and topology. A NED file is necessary for each simulation setup to specify parameters and gates of simple module and Compound module and the connections between them. The NED file can be used in two modes: graphic mode or text mode.
- **File (.ini)**  
Is closely linked with the NED file. Allows the user to initialize the parameters of the modules . It has several sections: Like the general parameter which provides the names of the application, the limit of the simulation time , the source and destination addresses.

### 3.4.4 OMNET++ Framework

OMNET++ follows a framework approach. It does not directly offer components for simulation, but supports the user with tools to create simulation components [20].

Frameworks like the INET framework add components from special application areas to OMNET++. These frameworks are developed independently. The INET framework uses the same structures as OMNET++. It is based on modules that communicate via messages. Network devices such as hosts, switches and routers are implemented as compound modules. The INET framework is an open-source model library for the OMNET++ simulation environment that makes it easier to develop simulations in the area of computer networks by providing protocols, agents and other models for the user. It consists of a set of models that provides the functionality of various elements from the physical layer to the application layer of the OSI model [20]. Some of the INET framework features are listed below:

- OSI layers implemented (physical, link-layer, network, transport, application)
- IPv4/IPv6 network stack
- Transport layer protocols: TCP, UDP
- Routing protocols
- Wired/wireless interfaces
- Visualization support

## 3.5 Comparison of OMNeT++ and NS3 Network Simulators

In this section, OMNET++ and NS3 are compared in terms of programming languages, platforms and visualization tools. Furthermore, the difference in simulator API and algorithms implemented for each software for wired network are described.

The core libraries of both network simulators are written in C++. Moreover, each network simulator has different programming languages for designing and implementing the network topology. Figure 3.4 illustrates the shared and unique scripting languages that are used in these simulators.

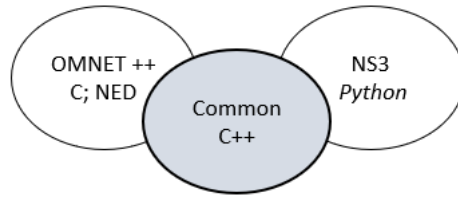


Figure 3.4: Scripting languages shared and unique for OMNeT++ and NS3

Besides the programming languages, NS3 offers only a command line interface, while OMNeT++ has a graphical user interface that allows the user to drag and drop several network elements in order to create the network topology.

Both network simulators NS3 and OMNeT++ support different operating system platforms. Figure 3.5 shows the platform supported by OMNeT++ and NS3.

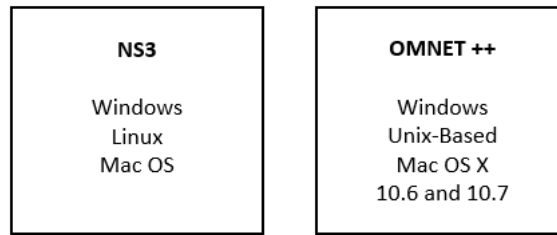


Figure 3.5: NS3 and OMNeT++ platforms

Visualization analysis of the simulation is a necessary operation in a computer network. NS3 does not provide any integrated GUI (Graphical User Interface) for visualization and analysis of the network. External tools are needed, such as NetAnim, pyviz, ns3-viz [40] [8]. On the other hand, OMNeT++ IDE provides users with integrated support for the simulation analysis and visualization [19].

Both simulators support different network models and technologies protocols used for wired and wireless networks. The development team implements such models and technologies protocols by converting the stander algorithms into script code written in c++ and integrating it in the software's source code.

They usually should ideally follow all stated specifications of the standard. However, that is not always the case. A simplified version of the protocol can be put in place of

the complete version. One of the reasons for this limitation is that not all hardware behavior can be modeled in software.

Currently, the two simulators include a set of implementations for the most common protocols through the application programmer interface (API) [33]. API can be defined as the function and classes that the network simulator provides for the user or researcher to use those implemented API to define protocols, models, and technologies in their simulation code.

In the following, we will explore the difference between the APIs provided by the two simulators when using wired communication protocols. The protocols used in this thesis are the PPP protocol and CSMA/CD protocol.

OMNET++ and NS3 differ considerably in the API they provide for wired communication [2][3] [23][19].

Omnet ++ supports an Ethernet interface, which operates as a full-duplex point-to-point connection where the data can be transmitted in both directions simultaneously with no collision [39].

This Ethernet interface contains two different Mac modules: The EthernetMac and EthernetCSMAMac [39].

The EthernetMac operates only in full duplex mode, which does not support CSMA/CD protocol. This means there is no collision detection, no retransmission with exponential backoff algorithms, no carrier expansion, and no frame bursting.

On the other hand, the EthernetCSMAMac operates with a full-duplex and a half-duplex mode, where the full-duplex mode acts as the EthernetMac. On the other hand, the half-duplex activates collision detection by sending jam signals and retransmits frames in case of collisions using the exponential backoff algorithm.

OMNET++ states full support for IEEE 811.3 and jam signal occurs , the EtherMac module for OMNET++ is complete except for a few tiny configuration details[21]

According to the User Guide of OMNET ++ [19], the PPP protocol in the INET framework is a minimal implementation of the PPP protocol standard. It only comes with encapsulation and decapsulation of data into frames.

NS 3 includes only two types of connections the PointToPoint model and the CSMA Model [1] [4].

The PointToPoint model connects two NetDevice using PointToPointChannel, which can be described as a full-duplex [4]. In contrast to the OMNET++ simulator, the data are not framed.

NS3 does not support an Ethernet interface using CSMA/CD for collision detection in a shared medium. Instead, it only supports a Model with the name CSMA Model, a simplified Ethernet CSMA/CD interface standard. It supports a light version of the Carrier-Sense function and allows for Multiple Access to a shared medium. The state of the medium is instantaneously visible to all devices. As a result, there is no need for collision detection in this model. Therefore, there is no implementation for it or any transmission in progress being "jammed." In NS3, the access to the channel is based on priority, and the node that comes first is served first, So it is safe to say that NS3's CSMA/CD model does not follow the standard and uses a light version of the CSMA/CD protocol [2][3].





# Chapter 4: Experiments and analysis

This chapter explains the experiments and analyses performed with the OMNET++ and NS3 network simulators. The network model for the experiments is explained in detail in the first section of the chapter. The second section provides details about all the results of the simulation and analysis conducted for OMNET++ and NS3 network simulators, whereas the third section offers a details discussion of all the experiments results and analysis.

## 4.1 Network Model

To study the essential differences between the two simulators, we started by reading the literature, the documentation of the two software, and studying previous research papers that were done in this field. Much work was done in comparing the differences between OMNET++ and NS3 network simulators in terms of general instruction or description of the software. On the other hand, fewer papers are comparing the two simulators regarding simulating legacy telecommunication technologies such as CSMA/CD and PPP protocols. Simultaneously, interest is high in relatively new protocols such as Wi-Fi.

Going further in this direction, we could find more detailed resources in the literature and the open source community about legacy protocols such as CSMA/CD and PPP and their implementation compared to fewer resources available for newer protocols.

So one can expect that stimulating legacy protocols gives more accurate results. However, interestingly, when we started this work, we noticed a significant contrast in the implementation of both protocols in ns3 and omnet++.

Therefore comparing the two simulators regarding their support for legacy protocols has important implications because NS3 and OMNET++ were first released after protocols like CSMA/CD and PPP had matured and were finalized. While relatively newer protocols like Wi-Fi are still changing, newer versions are continually being released.

It is therefore reasonable to assume that the simulators are proficient in the older protocols, especially since they are the starting point for any course on computer networks and the starting point for the documentation of both simulators.

From there, we chose to focus on NS3's and OMNET++ support for CSMA/CD and PPP protocols in depth.

In order to focus our study on the specification of these protocols, we set up the following network model :

We will simulate a wired peer-to-peer network containing a client and a server running a single TCP application. The cable is configured with a data rate of 100Mbps and a delay of 0.4 microseconds.

The client establishes a TCP connection with the server. Then, the client sends a preset number of packets, fixed at 1000 packets with a packet size of 512 bytes, to the server periodically following a configurable data rate from 512Kbps to 512Mbps.

Depending on the configuration of each packet the server receives, it can respond with an echo packet whose size is equal to that of the received packet multiplied by an echo factor. The echo can be immediate or after a configurable delay.

The time at which each application starts/stops and the time at which data transmission begins are configurable and are specified for each simulation session we run.

After all, packets are sent, the client application closes the TCP connection, and both applications exit. For the simulation time, we set 100 seconds as the time limit to allow the execution until the end.

In this experiment, the setup described above is simulated with two different cable types: in full-duplex mode with PPP protocol, which ensures data transmission in both directions simultaneously, and in half-duplex mode with CSMA/CD protocol, which ensures data transmission in both directions, but only in one direction at a time.

The following Table lists details for Simulation Parameters used for the simulation experiment.

Table 4.1: Simulation Parameters used for the simulation experiment

| Parameters              | Description   |
|-------------------------|---|
| peer-to-peer network    | client and server   |
| Simulation environment  | OMNeT++/NS-3  |
| Simulation time         | 100 second  |
| Transport protocol      | TCP   |
| number of packets       | 1000 packets  |
| TCP Packet size         | 512 bytes   |
| Queue                   | DropTail  |
| Queue                   | default   |
| Channel connection type | <ul style="list-style-type: none"> <li>• full-duplex mode running PPP protocol</li> <li>• half-duplex mode running CS-MA/CD protocol</li> </ul> |
| Channel Data Rate       | 100Mbps   |
| Channel Delay           | 0,4 microseconds  |
| Client Data Rate        | 0.5Mbps ,1Mbps ,2Mbps ,4Mbps ,8Mbps ,16Mbps ,32Mbps ,64Mbps ,128Mbps ,256Mbps ,512Mbps  |
| server echo factor      | 1   |
| server echo delay       | 0   |
| server start time       | 0 second  |
| client start time       | 0 second  |

## 4.2 Results of The Simulation in OMNET++ and NS3

The results of all simulation experiments with corresponding analysis are presented in the coming subsections using the OMNET++ and NS3 network simulators. All these findings are obtained based on a mathematical, qualitative, and comparative analysis performed using different communication standards, programming algorithms, and mathematical equations.

### 4.2.1 Packet Loss Ratio

During the experiments, the packet loss ratio is computed in OMNET++ and NS3.

Figure 4.1 below shows the packet loss ratio during the simulation of peer-to-peer network topology in in full-duplex mode running PPP protocol and half-duplex mode running CSMA/CD protocol for the client side and server side inin OMNET++ and NS3.

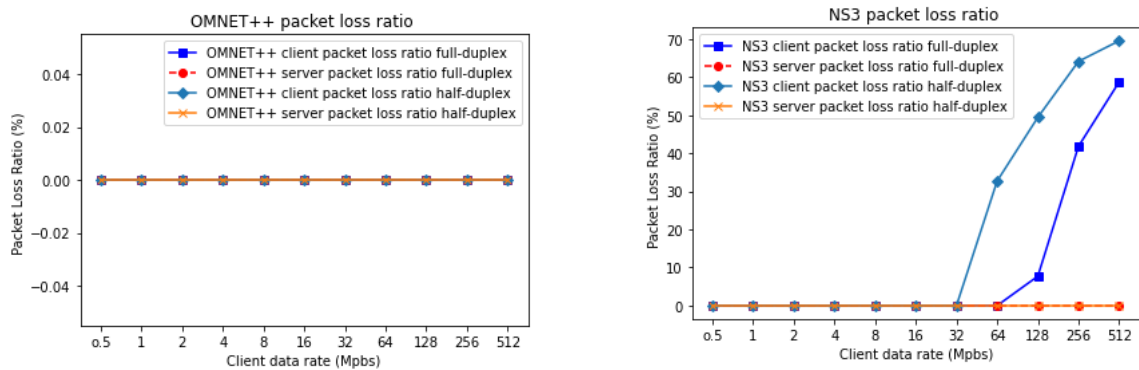


Figure 4.1: Comparison of the packet loss ratio during execution of the simulation in OMNET++ and NS3

The diagram clearly shows no packet loss in OMNET++ in both half-duplex and full-duplex modes on the client and server sides, even at high client data rates.

On the other hand, the packet loss rate of the simulation in NS3 has an increasing trend for both modes only on the client side. However, the packet loss ratio for both

modes is the same for the client data rate from 0, 5Mbps to 32 Mbps, which is 0 % loss . The packet loss ratio in the half-duplex modes using the CSMA/CD protocol shows a sharp increase from a client data rate of 32 Mbps. In contrast, a sharp increase in the packet loss ratio in the full-duplex modes that operate with the PPP protocol can be observed from a client data rate of 64 Mbps onwards.

It can be concluded that packet loss occurs when the packets flow from the client side to the server side and that the full-duplex mode causes less packet loss than the half-duplex mode.

When we read the percentage of the packet loss ratio in the figure above, we cannot be sure where the packet loss occurs and whether every packet sent by the client has reached the channel.

Therefore, the total number of bytes transmitted from the client side into the channel and the number of bytes transmitted through the channel to the server side is calculated in both modes see figure 4.2.

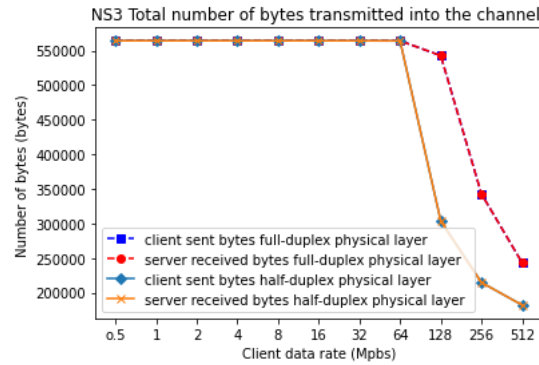


Figure 4.2: Number of bytes transmitted into the channel

As seen in figure 4.2 , for both full duplex and half duplex modes, the channel successfully transmits all bytes received from the physical layer of the client to the server and vice versa.

On the client side, the number of bytes sent and received from the application layer to the physical layer is calculated see Figure 4.3 .

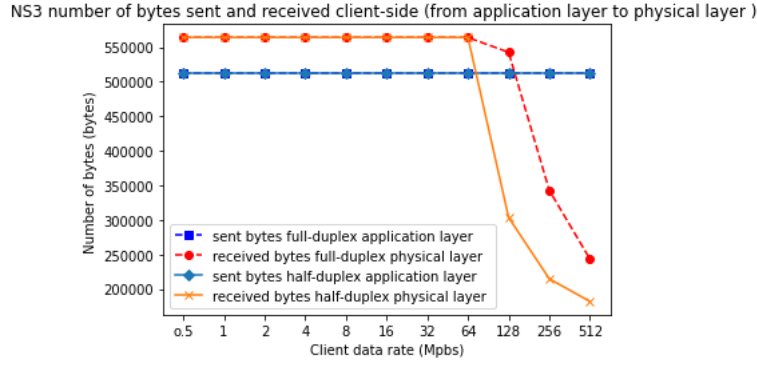


Figure 4.3: Number of bytes sent and received client side from application layer to the physical layer

On the other hand, the number of bytes sent by the application layer remains the same for both full-duplex and half-duplex modes at 512000 bytes, which means that the 1000 packets with a packet size of 512 bytes are all sent.

However, from the figure, we can see that not all the bytes have reached the physical layer of the client machine. Therefore, we can confidently say that the packet loss occurs on the client machine, not the channel.

Note that the physical layer may have a higher number of bytes than the application layer due to additional headers.

### 4.2.2 End-to-end Delay

During the experiments, the average time that elapses between sending and receiving packets is computed in OMNET++ and NS3.

Figure 4.4 below shows the end-to-end delay during the simulation of peer-to-peer network topology in full-duplex mode running PPP protocol and half-duplex mode running CSMA/CD protocol in OMNET++ and NS3.

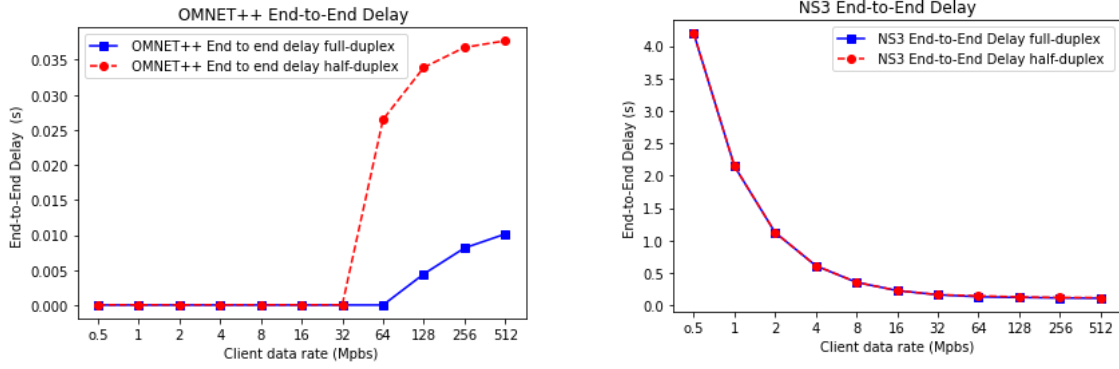


Figure 4.4: Comparison of the end-to-end delay during execution of the simulation in OMNET++ and NS3

The results show an opposite behavior between the simulators OMNET++ in both modes. NS3 has a decreasing trend. As the client data rate increases, we get less end-to-end delay. In OMNET++, on the other hand, the end-to-end delay remains unchanged until the client data rate reaches 32 Mbps for half-duplex mode and 64 Mbps for full-duplex mode, at which point the time consumed by TCP packets increases significantly.

In comparison, NS3 shows a decreasing trend, and OMNET++ has an increasing trend when the value of the client data rate increases. Therefore, the first impression is that NS3 has a better end-to-end delay for a higher client data rate than Omnet++.

However, this is not the case, as the highest end-to-end delay in OMNET++ was almost 0.035 seconds at a data rate of 512 Mbps. Meanwhile, the end-to-end delay of NS3 at a low data rate was almost 4 seconds.

From the figure 4.4 , it can also be noticed that the NS3 end-to-end delay in both modes is the same. On the other hand, TCP packets in OMNET++ consume more time in half-duplex mode than in the full-duplex mode.

### 4.2.3 Jitter

During the experiments, delays between packets from the client to the server and vice versa are computed in OMNET++ and NS3.

Figure 4.5 below shows jitter during the simulation of the peer-to-peer the network topology in full-duplex mode running PPP protocol and half-duplex mode running CSMA/CD protocol for the client and server sides in OMNET++ and NS3.

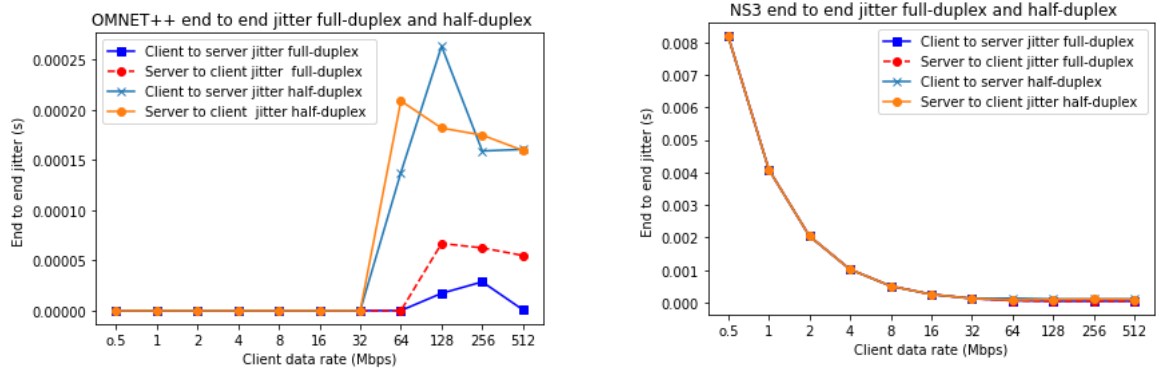


Figure 4.5: Comparison of the jitter during execution of the simulation in OMNET++ and NS3

As seen in the graph above, the jitter in NS3 for both modes decreases as the client data rate increases. On the other hand, OMNET++ remains at significantly low values, which is 0 s, until the data rate of the client surpasses the value of 32 Mbps in half-duplex mode and the value of 64 Mbps in full-duplex mode.

This increase is more significant in half-duplex mode due to packet collisions than in PPP full-duplex mode, which is caused by traffic load fluctuations due to the increase in packet error rate.

NS3 starts with the least stable flow rate among all simulations, indicated by the highest jitter value of 0.008 seconds at the beginning for a data rate of 0,5Mbps.

In OMNET++, the client-server traffic flow in half-duplex mode is the least stable with the highest jitter value of 0.00028s at a data rate of 128 Mbps.

#### 4.2.4 Network Collisions

During the experiments, network collisions occur when the client and server try to send the packets simultaneously are calculated in OMNET++ and NS3.

Figure 4.6 below shows network collisions during the peer-to-peer network topology simulation in half-duplex mode running CSMA/CD protocol for the client and server sides in OMNET++ and NS3.



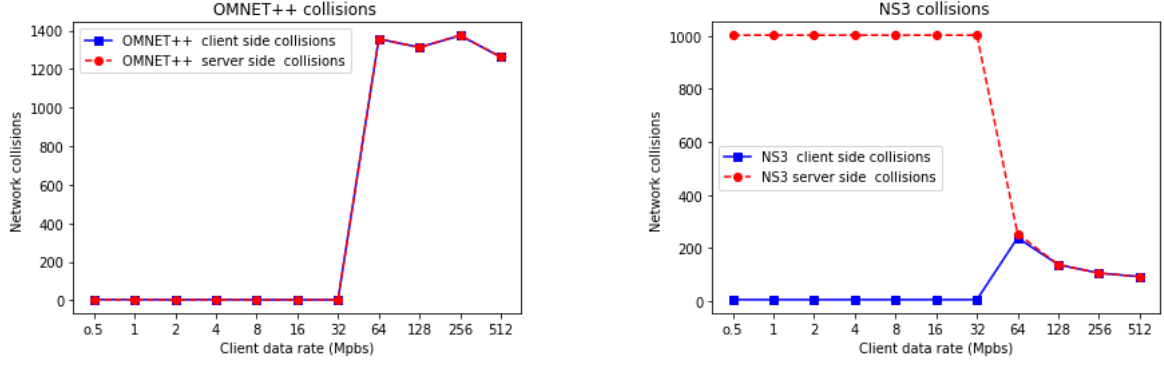


Figure 4.6: Comparison of the network collisions during execution of the simulation in OMNET++ and NS3

Since NS3 does not implement signal jamming, collisions detected by the client machine are not visible to the server machine and vice versa. For this reason, the client and server machines in NS3 display different collisions, unlike OMNET++, where both machines display the same number of collisions.

#### 4.2.5 Network Throughput

In this experiment, client and server throughput is calculated for the application layer and the physical layer for full-duplex and half-duplex modes in OMNET++ and NS3 simulators.

Figure 4.7 below shows the application layer throughput of the client and server in full-duplex mode running PPP protocol and half-duplex mode running CSMA/CD protocol in OMNET++.

Figure 4.8 below shows the application layer throughput of the client and server in full-duplex mode running PPP protocol and half-duplex mode running CSMA/CD protocol in NS3.

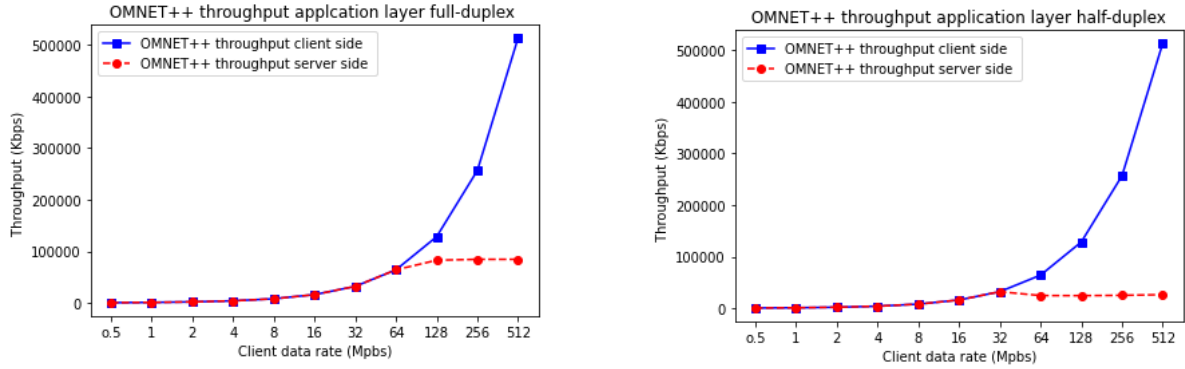


Figure 4.7: The application layer throughput of the client and server in full-duplex mode running PPP protocol and half-duplex mode running CSMA/CD protocol in OMNET++

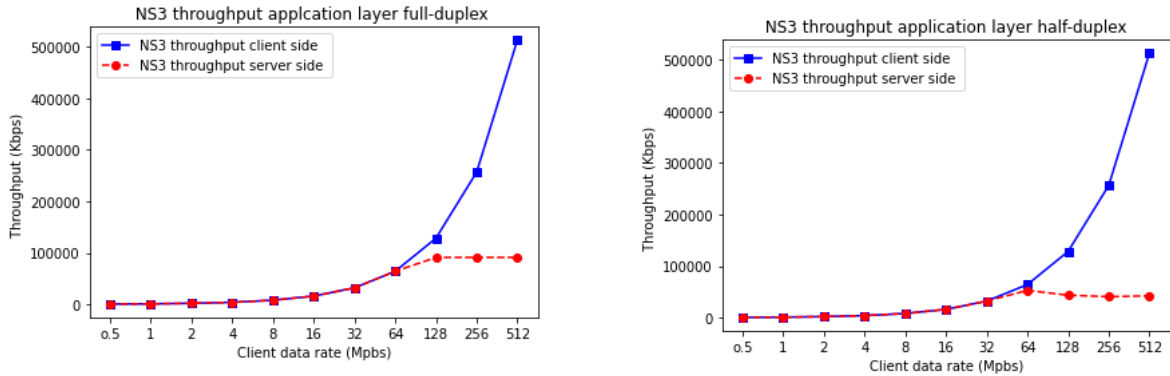


Figure 4.8: The application layer throughput of the client and server in full-duplex mode running PPP protocol and half-duplex mode running CSMA/CD protocol in NS3.

In the two diagrams above , OMNET++ and NS3 throughput application layer client and server in both modes show proportional similarity. The client application layer's throughput directly reflects the data rate increase.

On the other hand, the server application layer throughput increases along with the client application layer throughput. However, it does not increase at the same rate due to the delay between the client and the server, which prevents the server throughput from increasing at the same rate as the client application layer throughput.

The throughput of full-duplex mode for server sides is higher than the half-duplex mode in OMNET++ and NS3.

Figure 4.9 below shows the physical layer throughput of the client and server in full-duplex mode running PPP protocol and half-duplex mode running CSMA/CD protocol in OMNET++.

Figure 4.10 below shows the physical layer throughput of the client and server in full-duplex mode running PPP protocol and half-duplex mode running CSMA/CD protocol in NS3.

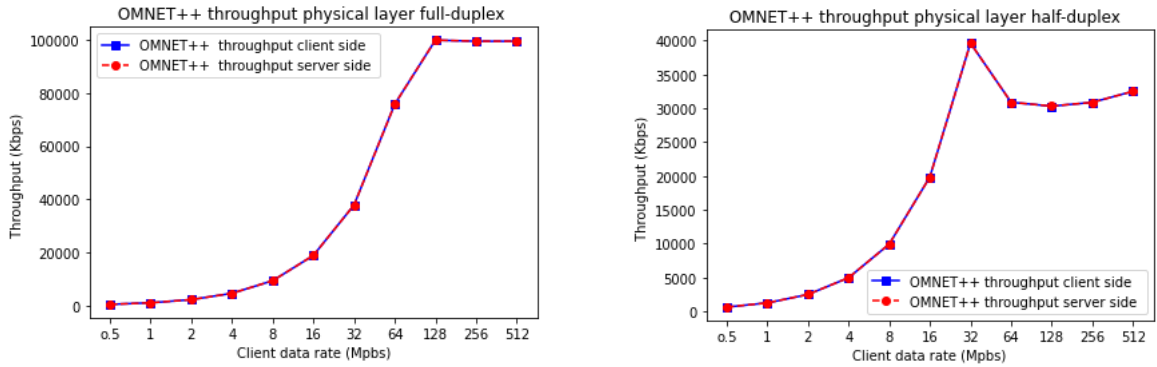


Figure 4.9: The physical layer throughput of the client and server in full-duplex mode running PPP protocol and half-duplex mode running CSMA/CD protocol in OMNET++

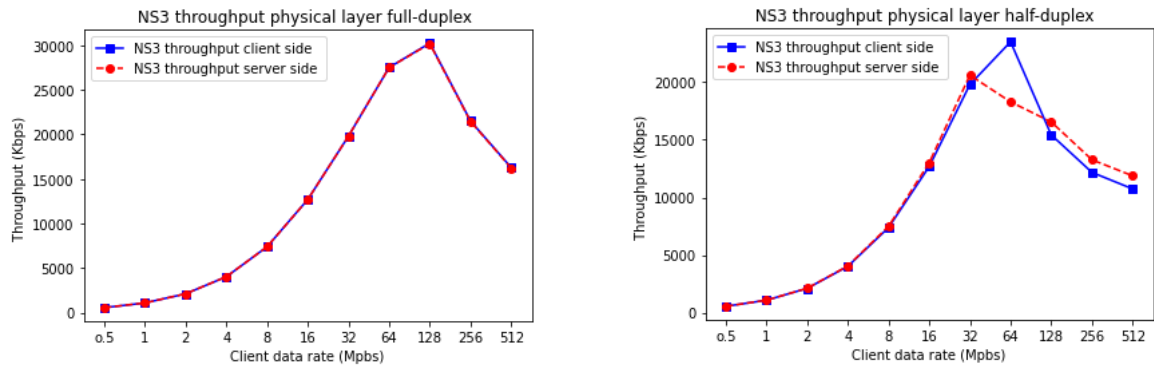


Figure 4.10: The physical layer throughput of the client and server in full-duplex mode running PPP protocol and half-duplex mode running CSMA/CD protocol in NS3.

In comparison, the throughput of the physical layer of OMNET++ is higher than NS3 at both the client and server sides in full-duplex and half-duplex modes. Therefore OMNET++ provides higher efficiency in data transmission and smooth communication at the physical layer compared to NS3.

#### 4.2.6 Number of Simulation Events

In this experiment, the number of events simulated to complete the simulation is computed in OMNET++ and NS3.

Figure 4.11 below shows the simulator events count of peer-to-peer network topology in full-duplex mode running PPP protocol and half-duplex mode running CSMA/CD protocol in OMNET++ and NS3.

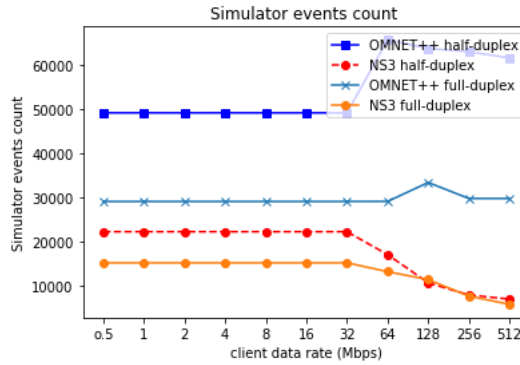


Figure 4.11: Simulator events count

The figure shows interesting compatibility between the two simulators, as the number of simulation events for both is in the same order of magnitude and a narrow range.

The highest number of simulation events is for OMNET++ simulators in half-duplex mode due to the increasing number of collisions which trigger more events such as signal interference and jamming. Next is OMNET++ full duplex, which indicates that OMNET++ divides the simulations into smaller steps, which can lead to more accurate results because each step is treated separately. In comparison, NS3 performs the same scenario using lesser events.

In NS3, the number of events for half and full duplex is almost the same at higher client data rate values, which is mainly related to the decrease in packet delivery, causing lower traffic load and fewer events from the loss of packets.

On the other hand, the number of events increases in OMNET++ when the client data rate value surpasses the throughput of the client's physical layer. However, this is not the case in NS3, where the decrease in the number of collisions, the absence of jamming signals, and the decrease in traffic load at a higher client data rate which all result in less events generated.

### 4.2.7 Simulation Length

In this experiment, the time taken by the connection to close is computed in OMNET++ and NS3.

Figure 4.12 below shows the simulation length of peer-to-peer network topology in full-duplex mode running PPP protocol and half-duplex mode running CSMA/CD protocol in NS3 and OMNET++.

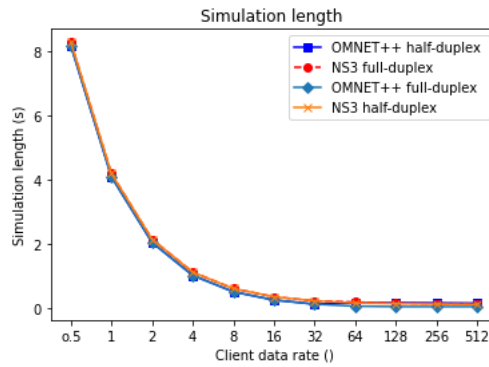


Figure 4.12: OMNeT++ Architecture

The most significant similarity between the two simulators is the simulation time. The simulation time is cut in half for both simulators when the client data rate value doubles. However, except at high data rates from 16 Mbps, the simulation time does not decrease as quickly and reaches a minimum value. This minimum here is mainly caused by the packet propagation delay or the time it takes for the data to travel, which cannot be eliminated by a high client data rate.

### 4.3 Discussion

From the simulations of peer-to-peer network topology in full-duplex mode with PPP protocol and half-duplex mode with CSMA/CD protocol in OMNeT++ and NS3, the increase in data rate affects the packet loss ratio, Throughput, end-to-end delay, jitter, Number of collisions on the channel, and the number of simulated events to complete the simulation. On the other hand, the simulation length is the same in both simulators for the two modes.

When analyzing the packet loss rate, it can be seen that OMNET++ has no packet loss, but NS3 does. From the statistics on the number of bytes in the traffic flow, we can see that at high data rates, not every byte sent by the application layer of the client computer reaches the channel.

Therefore, we can say with certainty that the packet loss occurs within the client machine, so we hypothesized that the packet loss might occur within the protocol stack of NS3, or we can say that the loss occurs in the transport or Internet layer of the OSI model.

The default TCP implementation is used for both simulators. Since the client machine on NS3 recorded packet loss only in transmission mode, we can assume that the loss is because of the memory overflow of the TCP transmit buffer, namely "tx buffer" [45].

A transmit buffer is a memory used to store packets the associated node sends until they are acknowledged [14]. In the TCP stack of NS3, the transmit buffer size can have a specific size that the programmer can specify, and when that buffer is full, packets start to drop [38].

In contrast, the default TCP stack implementation of OMNET++ does not allow the transmit buffer size variation, which means that the buffer size in OMNET++ is virtually unlimited and cannot overflow [37][41]. In comparison, NS3 is more realistic in this regard [36].

However, we increased the TCP transmit buffer size to test our hypothesis, so overflow is impossible. Thus, if packet loss persists after this change, we can confidently say that the packet loss is not caused by the overflow the TCP buffer.

We added the following line to NS3 [38]:

Code snippet 4.1: Network Nodes

```
1 DynamicCast<ns3::TcpSocketBase>(clientApplicationSocket)
2 ->GetTxBuffer()->MaxBufferSize(1000000000));
```

This allowed us to increase the maximum size of the client machine’s TCP buffer, specifically the transmit buffer used by the machine to store packets before sending them down to the physical layer. The value we set is 1 billion bytes, which is about 1 GB, representing the maximum number of bytes that can fit in the buffer. It should be high enough that it is never reached and no overflow occurs. After this change, packet loss in NS3 disappeared, and we got similar results to OMNET++ for packet delivery rate on both machines.

Of course, buffer overflow is not the only possible reason for packet loss. However, in our case with NS3, it was the cause of packet loss. On the other hand, there was no packet loss in OMNET++. We tried to manipulate the buffer size in OMNET++ to cause packet loss, but we found that OMNET++ has no way to change the TCP transmit buffer size in the default TCP implementation.

There is a relation between the number of collisions and the throughput in CSMA/CD protocol: A high throughput increases the chances of concurrent access to the channel, increasing collisions. A high number of collisions results in the execution of the backoff algorithm. The backoff algorithm forces the node to pause for a short period, decreasing the node’s throughput. So a high throughput is associated with a high number of collisions and vice versa.

This only happens if more than one machine is using the channel, but if only one machine has high throughput, it can transmit all the data before any other machine tries to access the channel.

The delay, jitter, and throughput were different for the two simulators. Moreover, this was a projection of the differences between the inner implementation of the algorithms involved. Both simulators react differently when the bandwidth of a channel is exceeded. In OMNET++, the delay of a packet increases when the channel is full compared to NS3 in full-duplex mode and half-duplex mode.





# Chapter 5: Conclusion and Future Work

## 5.1 Conclusion

We saw different results and behavior from each simulator, mainly caused by differences between the inner implementation of the algorithm in each software.

From the perspective of this thesis, NS3 gave a more satisfying research experience. This is because it was more stable, and its API was clearer with detailed documentation and educational resources. Besides, NS3 is entirely open-source software, unlike OMNET++, and these two things explain why there is more learning material for NS3 than for OMNET++. Even though NS3 did not have a detailed implementation for CSMA/CD protocol, the fact that it had a more realistic protocol stack that uses limited size buffers allowed it to give more varied output that better matches the reality. It is more realistic because, in OMNET++, the transmit buffer has a theoretically unlimited size, which is not practical when seeking real results. Software like OMNET++ and NS3 are usually not entirely built from scratch. However, components from other projects are brought and integrated into the project. However, they bring their limitations with them.

Simulation over software can be beneficial, accurate, and educative, but it has manageable limits, especially when combined with a testbed or components from the real world. Also, Open Source network simulators are primarily used in academic research, but businesses need that use more sophisticated software.

## **5.2 Future Work**

In the future, one can take advantage of what we have done by developing our own TCP application, which is not common in the community. Usually, an application from the official collection is used. The TCP client with a configurable echo server does not exist for NS3 in the official collection or the open source community as far as we could reach. Therefore, we think it would be helpful to extend the application in the future to include newer communication technologies such as LTE, 5G, and more complex protocols and different topologies. Routing can be added first, then wireless medium.

# Appendix A: NS3

This chapter is a step-by-step installation process of the NS3 simulation tool on windows operating system. More detailed instructions can be found in the official installation guide. At the end of this chapter, the software will be ready for use.

## A.1 NS3 Installation

NS3 is designed to run on Linux platforms, but it does not prevent being installed in other operating systems such as Mac OS and Windows OS, which can provide users with a Linux environment. In this thesis, the installation is done in the Windows operating system using the virtualization technology WSL "Windows Subsystem for Linux," which allows users to install, manage, and use a total Linux environment without installing VirtualBox or setting up a dual-boot system. This section will cover the step-by-step installation process of NS3 on Kali Linux distribution application windows using WSL technology.

### Step 1: WSL configuration on Windows 10

Windows Subsystem for Linux (WSL) is a compatibility layer for running Linux binary executables natively on Windows 10. It allows installing a Linux distribution as an application from the Windows store in which Basic Linux Commands (i.e., `sudo`, `apt`, `ls`, `cat`, `nano`, `cp`, `mv` ) can be executed and running Bash shell scripts with different program languages such as Languages: C, C++, Python, NodeJS...

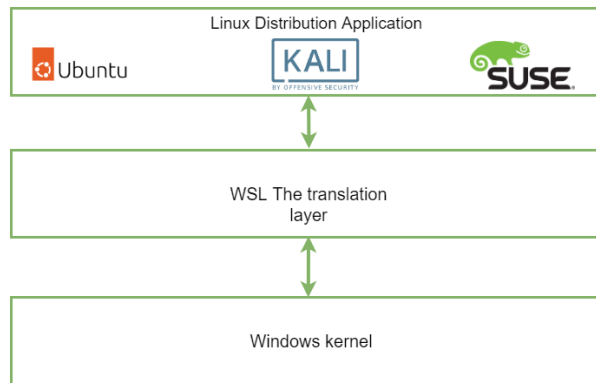


Figure A.1: WSL architecture

To get WSL working, we have to activate "Virtual Machine Platform" and "Windows Subsystem for Linux" feature by typing on the Windows search bar "Turn Windows features on or off".

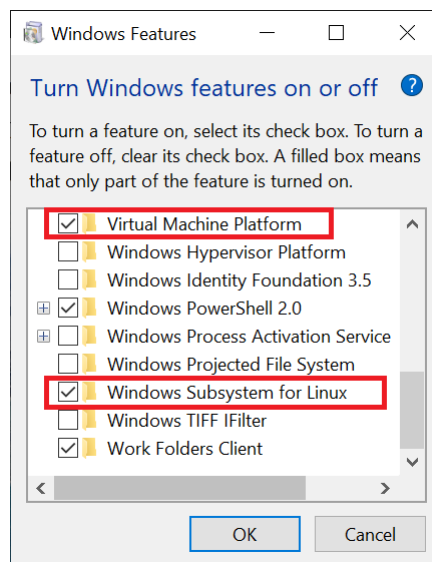


Figure A.2: Enable the "Virtual Machine Platform" and "Windows Subsystem for Linux"

It can also be done in another way through PowerShell. Just open the PowerShell terminal as administrator and type

Code snippet A.1: Enable the "Virtual Machine Platform" and "Windows Subsystem for Linux" via PowerShell terminal

```

1 //Enable Windows Subsystem for Linux
2 dism.exe /online /enable-feature ↵
   /featurename:Microsoft-WindowsSubsystem-Linux /all /norestart
  
```

```
3 //Enable Virtual Machine feature
4 dism.exe /online /enable-feature /featurename:VirtualMachinePlatform ↵
    /all /norestart
```

WSL is now activated. To proceed to the next step, the computer needs to restart

## Step 2: Installation of Linux Distribution

Microsoft Store offers a wide variety of Linux applications like ubuntu, Suse, Kali Linux, and many others. In this thesis, Kali Linux is used.

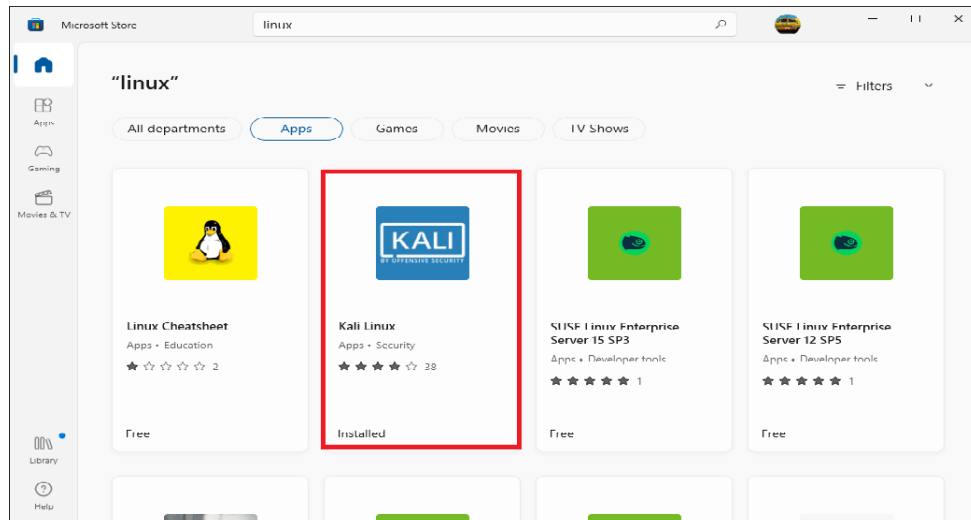


Figure A.3: Kali Linux

After the installation is finished, go back to the windows search bar and type kali Linux to set up a user account and password then we have to make sure that the distribution is working under WSL by typing the next command in PowerShell terminal.

```
1 wsl -l -v
```

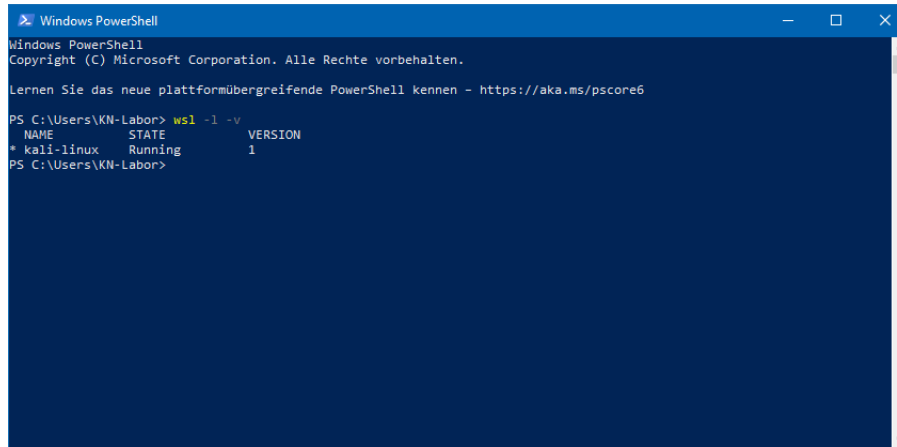


Figure A.4: Checking Kali Linux is running and the WSL

### Step 3: NS3 Installation

As Kali Linux is working under the WSL technology, we need to install some packages before installing NS3. It is preferred to update and upgrade the system first. For that, we have to get root privileges.

```

1 //root user in Linux
2 sudo su
3 //Update and Upgrade the System
4 sudo apt-get update && apt-get upgrade
5 //Dependencies installation
6 sudo apt-get install gcc g++ git mercurial cmake libc6-dev ↵
    libc6-dev-i386 g++-multilib gdb valgrind gsl-bin libgsl0-dev ↵
    flex bison libfl-dev tcpdump wireshark sqlite3 libsqlite3-dev ↵
    libxml2 libxml2-dev libgtk2.0-0 libgtk2.0-dev vtun lxc ↵
    uncrustify doxygen graphviz imagemagick texlive ↵
    texlive-extra-utils texlive-latex-extra texlive-font-utils ↵
    dvipng dia libboost-filesystem-dev openmpi-bin openmpi-common ↵
    openmpi-doc libopenmpi-dev gnuplot
7 //exit from root
8 exit

```

At this point the system is ready for downloading and installing the NS3 Simulation tool. Before downloading and installing NS3 from the official website (<https://www.nsnam.org/releases/NS3-34/>), a new folder named NS3 is created under the Home directory.

```

1 //navigate to home directory
2 cd ~
3 //create folder where NS3 simulator will be installed
4 mkdir NS3
5 //navigate to NS3 folder
6 cd NS3

```

the last stable version of NS3 is NS3.34 which is available for download at NS3 Releases. To download NS3 simulator we have to type this two command in Kali Linux terminal under the folder NS3 that was created.

```

1 //downloading NS3 installation Pack from the internet
2 wget https://www.nsnam.org/release/ns-allinone-3.34.tar.bz2
3 //extracting the files
4 tar xvjf ns-allinone-3.34.tar.bz2

```

```

kali@UET-G518-02: ~/ns3
(kali@ UET-G518-02)-[~/ns3]
$ wget https://www.nsnam.org/release/ns-allinone-3.34.tar.bz2
--2022-06-01 16:57:48-- https://www.nsnam.org/release/ns-allinone-3.34.tar.bz2
Resolving www.nsnam.org (www.nsnam.org)... 143.215.76.161
Connecting to www.nsnam.org (www.nsnam.org)[143.215.76.161]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 28938226 (28M) [application/x-bzip2]
Saving to: 'ns-allinone-3.34.tar.bz2'

ns-allinone-3.34.tar.bz2      100%[=====] 27.60M  5.55MB/s  in 14s
2022-06-01 16:58:03 (1.96 MB/s) - 'ns-allinone-3.34.tar.bz2' saved [28938226/28938226]

(kali@ UET-G518-02)-[~/ns3]
$ tar xvjf ns-allinone-3.34.tar.bz2
ns-allinone-3.34/ns-3.34/utlis.py
ns-allinone-3.34/ns-3.34/wscript
ns-allinone-3.34/ns-3.34/README.md
ns-allinone-3.34/ns-3.34/CONTRIBUTING.md
ns-allinone-3.34/ns-3.34/LICENSE
ns-allinone-3.34/ns-3.34/waf.bat
ns-allinone-3.34/ns-3.34/AUTHORS
ns-allinone-3.34/ns-3.34/CHANGES.html
ns-allinone-3.34/ns-3.34/waf
ns-allinone-3.34/ns-3.34/VERSION
ns-allinone-3.34/ns-3.34/Makefile
ns-allinone-3.34/ns-3.34/wutlis.py
ns-allinone-3.34/ns-3.34/test.py

```

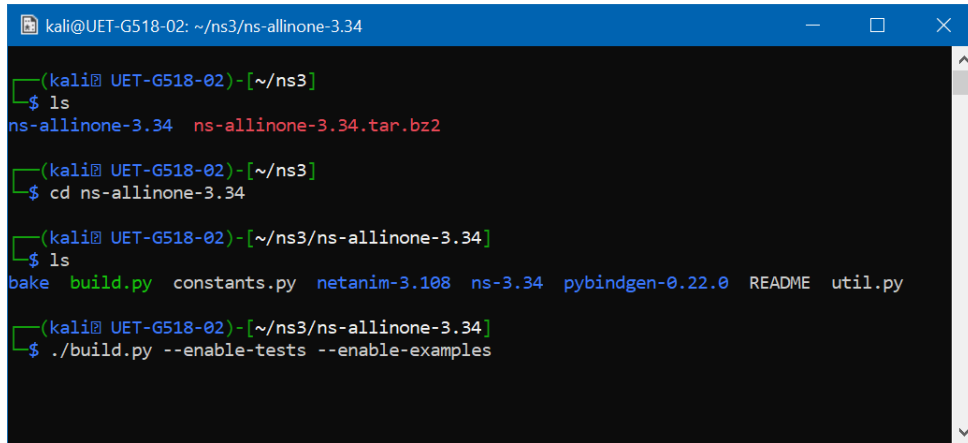
Figure A.5: Download and Extract NS3

To complete the installation we have to navigate to ns-allinone-3.34 folder by typing those commands the output should be like in figure A.6 presented.

```

1 //navigate to ns-allinone-3.34
2 cd ns-allinone-3.34
3 // Build NS3 with build command
4 ./build.py --enable-tests --enable-examples

```



```

kali@UET-G518-02: ~/ns3/ns-allinone-3.34
(kali@ UET-G518-02)~[~/ns3]
$ ls
ns-allinone-3.34  ns-allinone-3.34.tar.bz2

(kali@ UET-G518-02)~[~/ns3]
$ cd ns-allinone-3.34

(kali@ UET-G518-02)~[~/ns3/ns-allinone-3.34]
$ ls
bake  build.py  constants.py  netanim-3.108  ns-3.34  pybindgen-0.22.0  README  util.py

(kali@ UET-G518-02)~[~/ns3/ns-allinone-3.34]
$ ./build.py --enable-tests --enable-examples

```

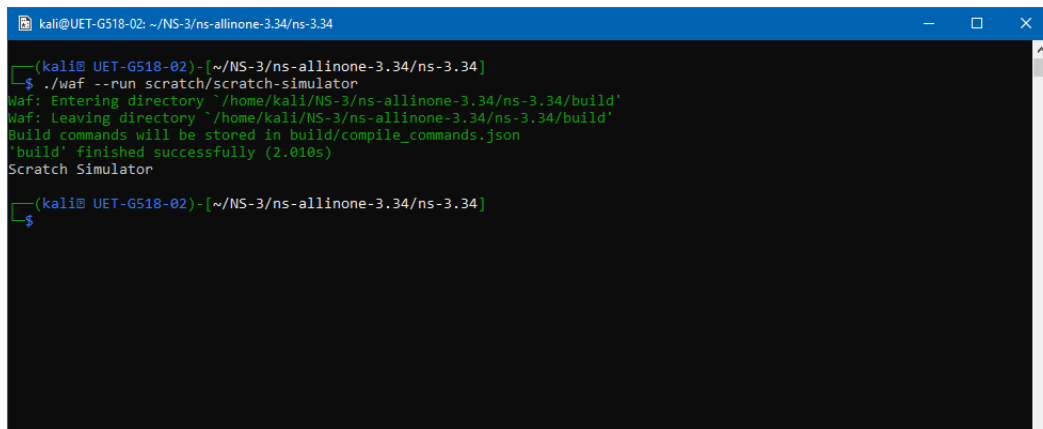
Figure A.6: compile and build NS3 simulator

The build of the whole NS3 simulators takes a several minutes (around 30 minutes). After that, the software set-up is complete. The last step is to check if the simulator is working fine by typing

```

1 //testing
2 ./waf --run hello-simulator

```



```

kali@UET-G518-02: ~/NS-3/ns-allinone-3.34/ns-3.34
(kali@ UET-G518-02)~[~/NS-3/ns-allinone-3.34/ns-3.34]
$ ./waf --run scratch/scratch-simulator
Waf: Entering directory `/home/kali/NS-3/ns-allinone-3.34/ns-3.34/build'
Waf: Leaving directory `/home/kali/NS-3/ns-allinone-3.34/ns-3.34/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (2.010s)
Scratch Simulator

(kali@ UET-G518-02)~[~/NS-3/ns-allinone-3.34/ns-3.34]
$

```

Figure A.7: Installation check

As before mentioned the script has to be created under the "scratch" folder of NS3 otherwise, it will be no possible to execute any simulation .Therefore we have to navigate from the home directory to "/NS3/ns-allinone-3.34/NS3.34/scratch" by typing this command in the Kali Linux terminal after that, we have to create a file with .cc extension.



```

1 //navigate to scratch folder
2 cd NS3/ns-allinone-3.34/NS3.34/scratch
3 //crate a file with .cc extention (the name chosen in this work is ←
  network.cc)
4 touch network.cc
5 //open the Kali Linux internal script editor
6 nano network.cc

```

```

GNU nano 6.3 scratch/network.cc
#include "ns3/bridge-module.h"
#include "ns3/rip-helper.h"
#include "ns3/ipv4-list-routing-helper.h"
#include "ns3/flow-monitor-module.h"
#include "ns3/olsr-helper.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE("Master Thesis");

int main (int argc, char *argv[])
{
    uint32_t SentPackets = 0;
    uint32_t ReceivedPackets = 0;
    uint32_t LostPackets = 0;

    CommandLine cmd;
    cmd.Parse (argc, argv);

    /* # Creating and declaring Nodes */
    NS_LOG_INFO ("Create nodes.");

```

Figure A.8: Kali Linux internal script editor

Although we are able to write scripts directly from the Kali Linux terminal, it is not practical to do it because it is not flexible as external code editors such as Visual Studio code, PyCharm, and Sublime Text editor and many others. For executing a script in NS3, the command "waf -run scratch/filename" is used to run the simulation. Here the filename is network.cc, in which the proposed network topology is modeled.

```

1 ./waf -v --run "scratch/network.cc"

```



# Appendix B: OMNET++

This chapter is a step-by-step installation process of the OMNET++ simulation tool on windows operating system. More detailed instructions can be found in the official installation guide. At the end of this chapter, the software will be ready for use.

## B.1 OMNET++ Installation

Unlike NS3, OMNET++, the official website of OMNET++ (<https://omnetpp.org/>) provides the user with different installation platform packages, such as Linux, Mac, and windows operating system. In our work, the Installation is done on the windows operating system.

The first step is to create a new folder called, for example named OMNET++ on the Desktop of the computer. Then download the source code from OMNET++ official web page (<https://omnetpp.org/download/old>) and extract it under the OMNET++ folder. The latest stable version of OMNET++ is OMNET++ 5.6.2 . After selecting the correct version, which is OMNET++ 5.6.2 source code, and the right operating system windows, we can now click the download button.

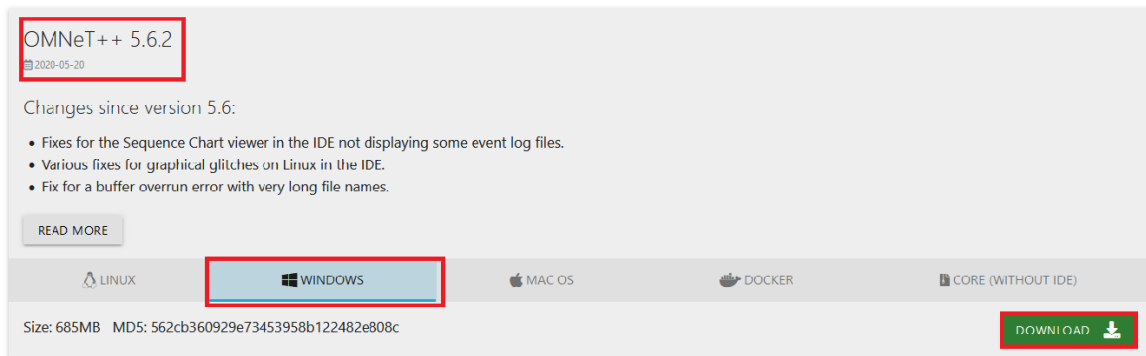


Figure B.1: downloading the OMNET++ source code

When the download is complete open the folder that was created OMNET++ on the desktop and extract it on the same folder. Now we have look for the the terminal

window named "mingwenv" from the extracted folder . Open it By double click and then press enter.

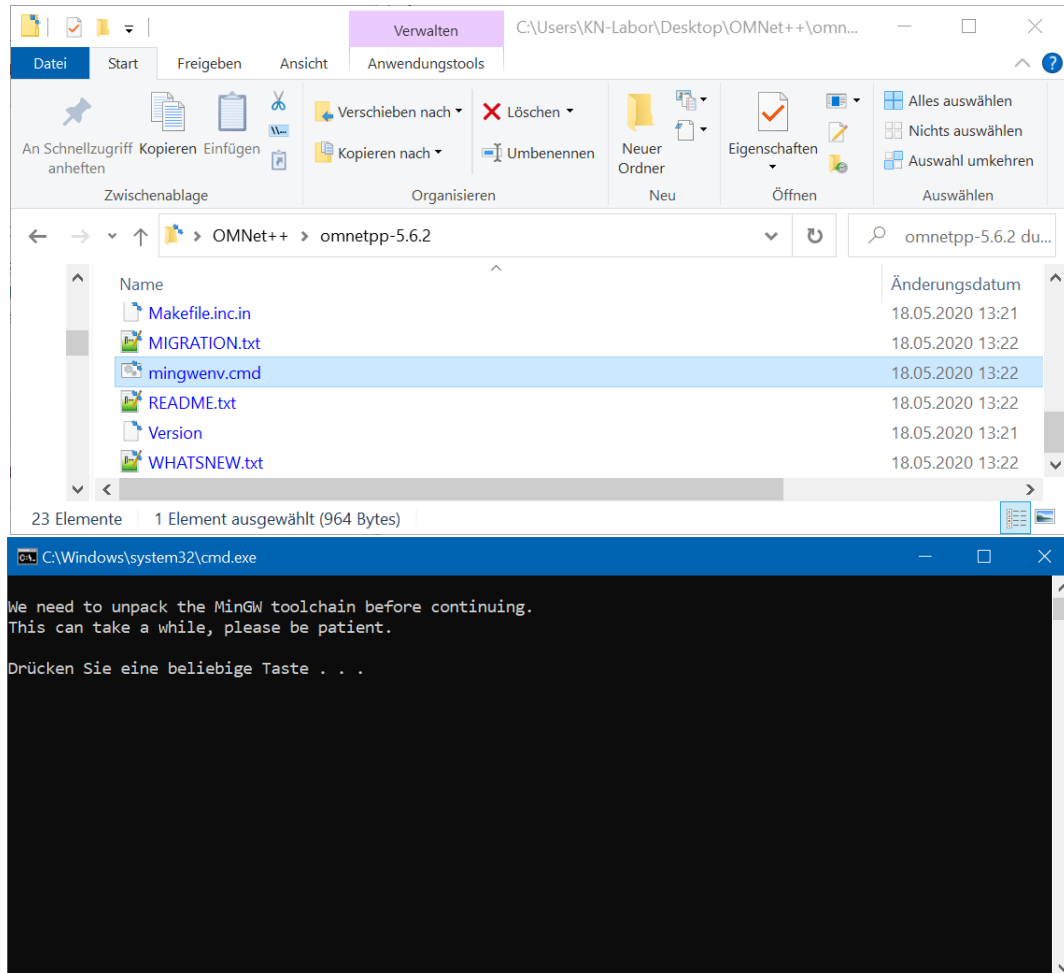
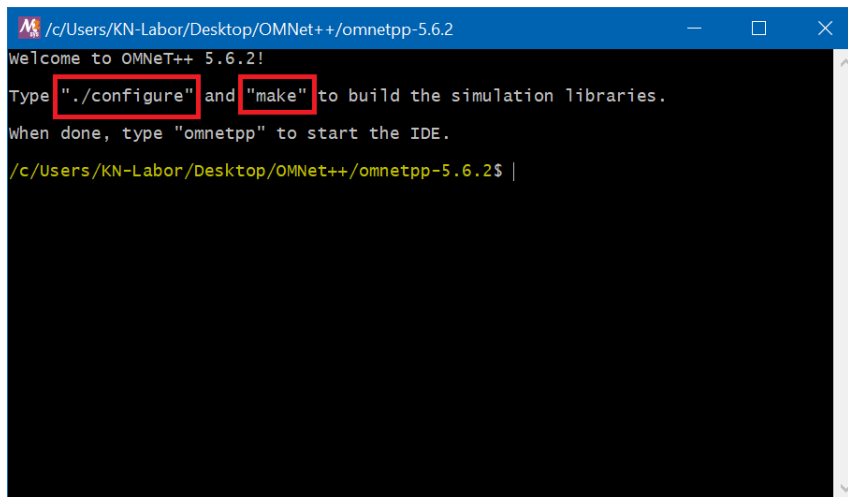


Figure B.2: mingwenv window terminal

When the the compilation is complete a new terminal window will appear and there type the following commands to compile and install OMNET++.

```
1 ./configure
2 make
```

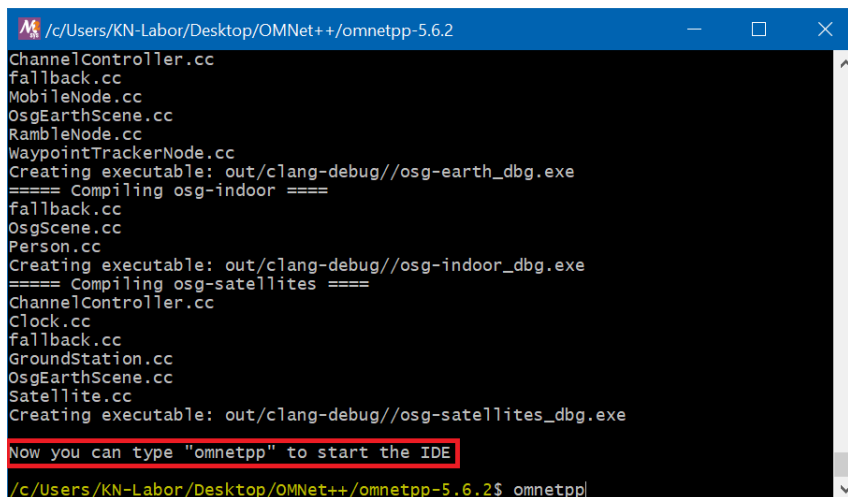


```

/c/Users/KN-Labor/Desktop/OMNet++/omnetpp-5.6.2
Welcome to OMNet++ 5.6.2!
Type "./configure" and "make" to build the simulation libraries.
When done, type "omnetpp" to start the IDE.
/c/Users/KN-Labor/Desktop/OMNet++/omnetpp-5.6.2$

```

Figure B.3: compile and install OMNET++



```

/c/Users/KN-Labor/Desktop/OMNet++/omnetpp-5.6.2
ChannelController.cc
fallback.cc
MobileNode.cc
OsgEarthScene.cc
RambleNode.cc
WaypointTrackerNode.cc
Creating executable: out/clang-debug//osg-earth_dbg.exe
==== Compiling osg-indoor ====
fallback.cc
OsgScene.cc
Person.cc
Creating executable: out/clang-debug//osg-indoor_dbg.exe
==== Compiling osg-satellites ====
ChannelController.cc
Clock.cc
fallback.cc
GroundStation.cc
OsgEarthScene.cc
Satellite.cc
Creating executable: out/clang-debug//osg-satellites_dbg.exe
Now you can type "omnetpp" to start the IDE
/c/Users/KN-Labor/Desktop/OMNet++/omnetpp-5.6.2$ omnetpp

```

Figure B.4: compile and install OMNET++ is finish

Now the installation is completed to start the IDE of OMNET++ type in the mingwenv terminal and the OMNET++ simulator screen will immediately appear

```
1 | omnetpp
```



Figure B.5: screen of the OMNET++ simulator

The INET framework will automatically be installed to provide the user with primary library for the simulation of communication networks.

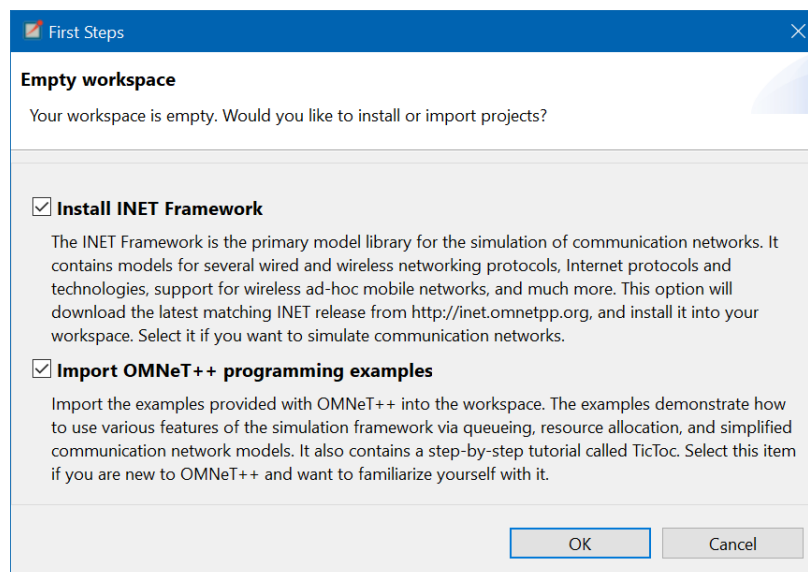


Figure B.6: INET framework installation

# Abbreviations

|               |   |
|---------------|---|
| ACK .....     | Acknowledgement                                   |
| CSMA .....    | CSMA Carrier Sense Multiple Access                |
| HTTP .....    | HyperText Transfer Protocol                       |
| IEEE .....    | Institute of Electrical and Electronics Engineers |
| IP .....      | Internet Protocol                                 |
| IS-IS .....   | Intermediate System to Intermediate System        |
| ISO .....     | International Standard Organization               |
| LAN .....     | Local Area Network                                |
| MAC .....     | Media Access Control Protocol                     |
| MAC .....     | Media Access Control                              |
| MAN .....     | Metropolitan Area Network                         |
| Mbps .....    | Megabits per Seconds                              |
| NCI .....     | Network Interface Card                            |
| NS-3 .....    | network simulator version 3                       |
| OMNET++ ..... | Objective Modular Network Test-bed in C++         |
| OMNeT++ ..... | Objective Modular Network Testbed in C++          |
| OSI .....     | Open System Interconnexion                        |
| PPP .....     | Point to Point Protocol                           |
| TCP .....     | Transmission Control Protocol                     |
| TCP .....     | Transport Control Protocol                        |
| TCP/IP .....  | Transport Control Protocol/Internet Protocol      |
| UDP .....     | User Datagram Protocol                            |
| WAN .....     | Wide Area Network                                 |
| WiFi .....    | Wireless Fidelity                                 |

Wsl ..... windows subs system for linux



# List of Figures

|      |   |    |
|------|---|----|
| 1.1  | Simulation cycle [26]   | 2  |
| 2.1  | Point to point topology [27]  | 7  |
| 2.2  | Line topology [27]  | 8  |
| 2.3  | Bus topology [27]   | 8  |
| 2.4  | Star topology [27]  | 9  |
| 2.5  | Ring topology [27]  | 9  |
| 2.6  | Fully mesh topology [27]  | 10 |
| 2.7  | partially meshed topology [27]  | 10 |
| 2.8  | Tree topology [27]  | 11 |
| 2.9  | Relationship between a service and a protocol [35]  | 11 |
| 2.10 | OSI and TCP/IP reference model [35]   | 12 |
| 2.11 | flow diagram for the CSMA/CD protocol [5]   | 14 |
| 2.12 | TCP Connection Establishment [28]   | 16 |
| 2.13 | TCP Connection Termination [28]   | 17 |
| 2.14 | Example of data flow for different network components based on<br>simplified OSI model [18]   | 19 |
| 2.15 | Transmission delay  | 20 |
| 2.16 | Propagation delay   | 21 |
| 3.1  | Software organization of NS3 [9]  | 26 |
| 3.2  | NS3 Node architecture [9]   | 27 |
| 3.3  | OMNET++ Architecture [42]   | 28 |
| 3.4  | Scripting languages shared and unique for OMNET++ and NS3                                     | 31 |
| 3.5  | NS3 and OMNET++ platforms   | 31 |
| 4.1  | Comparison of the packet loss ratio during execution of the simulation<br>in OMNET++ and NS3  | 38 |
| 4.2  | Number of bytes transmitted into the channel  | 39 |
| 4.3  | Number of bytes sent and received client side from application layer<br>to the physical layer | 40 |

|      |   |    |
|------|---|----|
| 4.4  | Comparison of the end-to-end delay during execution of the simulation in OMNET++ and NS3 . . . . .  | 41 |
| 4.5  | Comparison of the jitter during execution of the simulation in OMNET++ and NS3 . . . . .  | 42 |
| 4.6  | Comparison of the network collisions during execution of the simulation in OMNET++ and NS3 . . . . .  | 43 |
| 4.7  | The application layer throughput of the client and server in full-duplex mode running PPP protocol and half-duplex mode running CSMA/CD protocol in OMNET++ . . . . . | 44 |
| 4.8  | The application layer throughput of the client and server in full-duplex mode running PPP protocol and half-duplex mode running CSMA/CD protocol in NS3. . . . .      | 44 |
| 4.9  | The physical layer throughput of the client and server in full-duplex mode running PPP protocol and half-duplex mode running CSMA/CD protocol in OMNET++ . . . . .    | 45 |
| 4.10 | The physical layer throughput of the client and server in full-duplex mode running PPP protocol and half-duplex mode running CSMA/CD protocol in NS3. . . . .         | 45 |
| 4.11 | Simulator events count . . . . .  | 46 |
| 4.12 | OMNeT++ Architecture . . . . .  | 47 |
| A.1  | WSL architecture . . . . .  | 54 |
| A.2  | Enable the "Virtual Machine Platform" and "Windows Subsystem for Linux" . . . . .   | 54 |
| A.3  | Kali Linux . . . . .  | 55 |
| A.4  | Checking Kali Linux is running and the WSL . . . . .  | 56 |
| A.5  | Download and Extract NS3 . . . . .  | 57 |
| A.6  | compile and build NS3 simulator . . . . .   | 58 |
| A.7  | Installation check . . . . .  | 58 |
| A.8  | Kali Linux internal script editor . . . . .   | 59 |
| B.1  | downloading the OMNET++ source code . . . . .   | 61 |
| B.2  | mingwenv window terminal . . . . .  | 62 |
| B.3  | compile and install OMNET++ . . . . .   | 63 |
| B.4  | compile and install OMNET++ is finish . . . . .   | 63 |
| B.5  | screen of the OMNET++ simulator . . . . .   | 64 |
| B.6  | INET framework installation . . . . .   | 64 |

# List of Tables

|     |  |    |
|-----|--|----|
| 4.1 | Simulation Parameters used for the simulation experiment . . . . . | 37 |
|-----|--|----|



# Listings

|     |  |    |
|-----|--|----|
| 4.1 | Network Nodes . . . . .  | 49 |
| A.1 | Enable the "Virtual Machine Platform" and "Windows Subsystem for<br>Linux" via PowerShell terminal . . . . . | 54 |



# Bibliography

- [1] Release ns-3-dev. *CSMA NetDevice*. URL: <https://www.nsnam.org/docs/release/3.10/manual/html/csma.html>.
- [2] Release ns-3-dev. *NS3 Library*. URL: <https://www.nsnam.org/docs/models/ns-3-model-library.pdf>.
- [3] Release ns-3-dev. *NS3 Manual*. URL: <https://www.nsnam.org/docs/manual/ns-3-manual.pdf>.
- [4] Release ns-3-dev. *PointToPoint NetDevice*. URL: <https://www.nsnam.org/docs/release/3.10/manual/html/point-to-point.html>.
- [5] Ibrahim Sayed Ahmad, Ali Kalakech, and Seifedine Kadry. “Minimizing mobiles communication time using modified binary exponential backoff algorithm”. In: *International Journal of Computer Networks & Communications* 5.6 (2013), p. 85.
- [6] Mukarram AM Almuahaya et al. “A survey on Lorawan technology: Recent trends, opportunities, simulation tools and future directions”. In: *Electronics* 11.1 (2022), p. 164.
- [7] Gayatry Borboruah and Gypsy Nandi. “A study on large scale network simulators”. In: *International Journal of Computer Science and Information Technologies* 5.6 (2014), pp. 7318–7322.
- [8] Gustavo Carneiro, Pedro Fortuna, and Manuel Ricardo. “Flowmonitor: a network monitoring framework for the network simulator 3 (ns-3)”. In: *Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools*. 2009, pp. 1–10.
- [9] Rachna Chaudhary et al. “A study of comparison of Network Simulator-3 and Network Simulator-2”. In: *International Journal of Computer Science and Information Technologies* 3.1 (2012), pp. 3085–3092.
- [10] Kennedy Clark and Kevin Hamilton. *Cisco LAN Switching (CCIE Professional Development series)*. Cisco Press, 1999.

- [11] *Documentation: Discrete Event Simulator*. URL: <https://omnetpp.org/documentation/>.
- [12] Annabel Z Dodd. *The Essential Guide to Telecommunications*, 2000.
- [13] Simson Garfinkel and Gene Spafford. “Practical UNIX & Internet Security”. In: (1999).
- [14] Imtiaz A Halepoto et al. “Effect of TCP Buffer Size on the Internet Applications”. In: *International Journal of Advanced Computer Science and Applications* 9.6 (2018).
- [15] Fred Halsall. *Data communications, computer networks and open systems*. Addison Wesley Longman Publishing Co., Inc., 1995.
- [16] Andreas Hanemann et al. “A study on network performance metrics and their composition”. In: *Campus-Wide Information Systems* (2006).
- [17] Mohammed Humayun Kabir et al. “Detail comparison of network simulators”. In: *International Journal of Scientific & Engineering Research* 5.10 (2014), pp. 203–218.
- [18] James F Kurose and Keith W Ross. *Computernetzwerke: Der Top-Down-Ansatz*. Pearson Deutschland GmbH, 2008.
- [19] And OpenSim Ltd. *OMNET++ UserGuide*. URL: <https://doc.omnetpp.org/omnetpp/UserGuide.pdf>.
- [20] Levente Mészáros, Andras Varga, and Michael Kirsche. “Inet framework”. In: *Recent Advances in Network Simulation*. Springer, 2019, pp. 55–106.
- [21] *Model Catalog*. URL: <https://inet.omnetpp.org/Protocols.html>.
- [22] AA Obiniyi, MB Soroyewun, and MM Abur. “New innovations in performance analysis of computer Networks: A review”. In: *International journal of Applied information systems* 6.2249-0868 (2014).
- [23] *OMNeT++ Simulation Library*. URL: <https://doc.omnetpp.org/omnetpp/api/modules.html>.
- [24] Jianli Pan and Raj Jain. “A survey of network simulation tools: Current status and future developments”. In: *Email: jp10@cse.wustl.edu* 2.4 (2008), p. 45.
- [25] Dr L RAJA. “Study of various network simulators”. In: *International Research Journal of Engineering and Technology (IRJET)* 5.12 (2018).



- 
- [26] Sebastian Rampfl. “Network simulation and its limitations”. In: *Proceeding zum seminar future internet (FI), Innovative Internet Technologien und Mobilkommunikation (IITM) und autonomous communication networks (ACN)*. Vol. 57. Citeseer. 2013.
- [27] Santanu Santra and Pinaki Pratim Acharjya. “A study and analysis on computer network topology for data communication”. In: *International Journal of Emerging Technology and Advanced Engineering* 3.1 (2013), pp. 522–525.
- [28] Ravi Kiran Sattineni. *TRANSMISSION CONTROL PROTOCOL*. URL: <https://webuser.hs-furtwangen.de/~heindl/ebte-08ss-TCP-Sattineni-Ravi.pdf>.
- [29] Salman M Al-Shehri et al. “Common metrics for analyzing, developing and managing telecommunication networks”. In: *arXiv preprint arXiv:1707.03290* (2017).
- [30] JB Shukla et al. “Modeling and analysis of the effects of antivirus software on an infected computer network”. In: *Applied Mathematics and Computation* 227 (2014), pp. 11–18.
- [31] A Discrete-Event Network Simulator. *NS3 Documentation: A Discrete-Event Network Simulator*. URL: <https://www.nsnam.org/documentation/>.
- [32] William Stallings. “Local networks”. In: *ACM Computing Surveys (CSUR)* 16.1 (1984), pp. 3–41.
- [33] W Richard Stevens. *TCP/IP illustrated vol. I: the protocols*. Pearson Education India, 1993.
- [34] Winarno Sugeng et al. “The impact of QoS changes towards network performance”. In: *Int. J. Comput. Networks Commun. Secur* 3.2 (2015), pp. 48–53.
- [35] Andrew S Tanenbaum and David J Wetherall. “Computer networks fifth edition”. In: *Pearson Education, Inc.* Prentice Hall, 2011.
- [36] *TCP models in ns-3*. URL: <https://www.nsnam.org/docs/release/3.10/manual/html/tcp.html#ns-3-implementation>.
- [37] *TCP protocol implementation*. URL: <https://doc.omnetpp.org/inet/api-current/neddoc/inet.transportlayer.tcp.Tcp.html>.
- [38] *TcpTxBuffer Class Reference*. URL: [https://www.nsnam.org/doxygen/classns3\\_1\\_1\\_tcp\\_tx\\_buffer.html](https://www.nsnam.org/doxygen/classns3_1_1_tcp_tx_buffer.html).
- [39] *The Ethernet Model*. URL: <https://inet.omnetpp.org/docs/users-guide/ch-ethernet.html>.

- [40] L. Felipe Perrone Tom Henderson. *NS-3 Advanced Tutorial: Visualization and Data Collection*. URL: <https://webuser.hs-furtwangen.de/~heindl/ebte-08ss-TCP-Sattineni-Ravi.pdf>.
- [41] *Transport Protocols*. URL: <https://inet.omnetpp.org/docs/users-guide/ch-transport.html>.
- [42] András Varga. “Using the OMNeT++ discrete event simulation system in education”. In: *IEEE Transactions on Education* 42.4 (1999), 11–pp.
- [43] Andras Varga. “OMNeT++”. In: *Modeling and tools for network simulation*. Springer, 2010, pp. 35–59.
- [44] Sergio Verdú. “Wireless bandwidth in the making”. In: *IEEE communications Magazine* 38.7 (2000), pp. 53–58.
- [45] Ivan Vujović and Maroje Delibašić. “Influence of Buffer Size on TCP Performance in Heterogeneous Wired/Wireless Networks”. In: *Internet of Things, Smart Spaces, and Next Generation Networking*. Springer, 2013, pp. 224–235.
- [46] Klaus Wehrle, Mesut Günes, and James Gross. *Modeling and tools for network simulation*. Springer Science & Business Media, 2010.
- [47] Felix Wortmann and Kristina Flüchter. “Internet of things”. In: *Business & Information Systems Engineering* 57.3 (2015), pp. 221–224.

# Statement of Academic Integrity

I hereby certify that I have written this Master Thesis independently and only with the use of using only the sources and aids indicated. The work has not been submitted in the same or similar form to any other examination authority.

Berlin, den 2022-10-04

Majd Rekik

A handwritten signature in black ink, appearing to be 'MR' or similar, enclosed within a rough, hand-drawn rectangular border.