UNIVERSITY OF OSLO
Department of Informatics

An Investigation and Comparison on Network Performance Analysis

Mi Lanxiaopu

Network and System Administration
Oslo University College

May 23, 2012
An Investigation and Comparison on Network Performance Analysis

Mi Lanxiaopu

Network and System Administration
Oslo University College

May 23, 2012
Abstract

This thesis is generally about network performance analysis. It contains two parts.

The theory part summarizes what network performance is and inducts the methods of doing network performance analysis. To answer what network performance is, a study into what network services are is done. And based on the background research, there are two important network performance metrics: Network delay and Throughput should be included in network performance analysis.

Among the methods of network analysis, there are two types: real target network analysis and network simulation. In the practical part, the two types of network analysis are both applied under a same target network layout for the sake of comparing them and find out difference. It turns out that the simulated network by ns-3 tends to give a more optimistic analysis results over the two network performance metrics.
## Contents

1 Introduction
   1.1 Motivation ......................................................... 3
   1.2 Status quo .......................................................... 5
   1.3 Overview ............................................................ 6

2 Background Research
   2.1 Computer Networks and Network services .......................... 7
       2.1.1 Computer network .............................................. 7
       2.1.2 Main Network Devices ........................................ 9
       2.1.3 Network Services .............................................. 9
   2.2 Network Performance Classification ................................. 12
       2.2.1 Brief description of network performance ..................... 12
       2.2.2 Network performance pyramid ................................ 13
       2.2.3 Main Performance Metrics .................................... 14
   2.3 Network performance analysis methods ............................ 17
       2.3.1 Analysis based on true networks .............................. 17
       2.3.2 Network Simulation .......................................... 19
       2.3.3 Comments and Comparison .................................... 20
   2.4 Network performance Measuring Tools ............................. 22
       2.4.1 Simulating tool .............................................. 22
       2.4.2 Tools aim on actual network ................................. 23

3 Methodology
   3.1 Network design .................................................... 29
       3.1.1 Network structure ............................................ 29
       3.1.2 Experiment arrangement ..................................... 30
   3.2 Construction of Target Network .................................. 32
   3.3 Simulation of Target Network ..................................... 33
       3.3.1 Network Nodes Simulation ................................... 34
       3.3.2 Network Segmentations Simulation .......................... 34
       3.3.3 Network Topology Simulation ................................ 35
       3.3.4 Network Devices Simulation .................................. 35
       3.3.5 Application Simulation ....................................... 36
   3.4 Experiment Procedures ............................................ 37

4 Data Presentation and Analysis ...................................... 39
   4.1 Throughput from the real network ................................ 39
## CONTENTS

4.2 Network Delay from the real network .................................. 45  
4.3 Throughput from the Simulated Network ............................... 47  
4.4 Network Delay from the Simulated Network ......................... 48  
4.5 Comparing and Analysis .................................................. 48  
4.5.1 Throughput .............................................................. 48  
4.5.2 Network Delay .......................................................... 52  
4.5.3 Comprehensive analysis ............................................... 52  

5 Discussion and Future work ................................................. 54  
5.1 Discussion ................................................................. 54  
5.2 Future work ............................................................... 54  

6 Conclusion ........................................................................... 56  

7 Appendix ............................................................................. 60  
7.1 The ns-3 simulating code .................................................. 60  
7.2 Important Router Configurations ....................................... 64  
7.2.1 Router-4 ................................................................. 64  
7.2.2 Router-2 ................................................................. 66
Chapter 1

Introduction

The earliest history of Computer Networks can be retroactive to the middle of last century, one interesting affair in the early history is "in 1969 the University of California at Los Angeles, the Stanford Research Institute, University of California at Santa Barbara, and the University of Utah were connected as the beginning of the ARPANET network using 50 Kbits/sec circuits[1]". Even viewing from today this speed is not so bad, it just take 10 minutes to download a song.

Several decades has passed, computer networks has permeated into our daily life with an amazing speed and occupied a very important part. There are social commentators said that computer network brought us the forth industry revolution (Information Revolution), and now we are enjoy the benefits it brings to us. As a master student in engineer I do not really know too much about social science or humanities, however, there is no doubt that computer network has changed and improved people’s life significantly.

It is not a exaggeration that network has became a important everyday item, since more and more people choose to meet their requirements in purchasing, entertainment, business and information obtaining counting on computer networks. Hence people need reliable, stable and promised networks. Nobody wants to get stuck on networks while doing important business, which may result in big losses. And this is also what the system administrators are persuading in work: system administration is about the design, running and maintenance of human-computer system[2]. The so called human-computer system is mostly consisted of interactive communication computer networks.

1.1 Motivation

Since computer network has turned to a life tool, the analysis and evaluation of its performance then naturally became a valuable issue.

As we all know that super heroes existed in movies beat the bastards with their
1.1. MOTIVATION

own powerful skills or weapons, which are recognized as their life tools. Just image if their life tool is really bad or has low quality they wouldn’t be able to become a super hero, they may just fallen to normal people like us: go to school, find a job, read newspaper and comment the government.

It is really important for user to understand whether their tool is reliable or not. That when we need to do evaluation and analysis. And our requirements is not so irrational toward the tool: computer networks. Basically, the requirements are all built on the main function which is communication.

Beside the above one that we should know what we are using and how good it is, there are two more reasons for doing network performance analysis.

Although the service goal of computer network is clear, it is a complex entirety itself and has complicated structure and stratification. For the potential system administrators like me, the beginning of the work may be making a layout of a proper network follow the requirements and then construct it in real. During this implementation process, it is a safe and cautious way to simulated the layout somewhere before deployment, as this can avoid the offset of the goal or big missing with doing pre-analysis on simulating platform. One can also choose another way which is constructing the objective network while doing partly performance analysis. No matter how you finally achieve the goal, when you are on the way, the performance analysis helps adjust the working direction and instructs us to some extent.

Not only construction and running network consists our job, maintaining and optimization also take a big part. Obviously the efficient way of optimization is find out the very point that draw back the entire performance and try to improve it. In order to address where the problem is, a complete performance analysis can be quite necessary, since different performance volume represents different work ability of a certain network part.

In a summarization, network performance analysis is an important issue that we can benefit form according to the above three aspects.

- **Evaluate and rank networks.** Although there is no explicit definition for network quality, at least we all know what users mostly care about networks. The corresponding performance analysis may give conclusions or judgments with considerable reference value.

- **Discovery of network problems or potential problems.** Computer networks are complicated, multi functional, changing and composite object, this lead to increasing complexity of network debug when there is something wrong. A complete and systemic performance analysis may help to find where’s the problem, and of cause experience and luck will help also.

- **Construct and improve networks.** Even there’s no problem and the the net-
work works, one may still have the need of improving current networks or find a better solution. Usually there are some specific requirements that tells in which respect and in what level the network should be improved, and one may need to do performance analysis to check if it’s approved.

1.2 Status quo

Since network performance is such a significant topic, inevitably there must be lots of related reports and academic thesis existed. In fact, if you search for "Network Performance Analysis" online, there will be more than one million results appeared. If you go through these search results a little, you will see they can be divided into two classes, one is kind of instruction or description for software that can do network analysis, they are just like advertisements. The other class is consisted of finished paper and thesis related to network performance analysis. For those software and application tools, they are also assorted. Generally speaking, they can be sorted as positive testing tools and negative monitoring tools, more details in chapter 2.

Existed papers and thesis are very valuable to us, as our target is researching and comparing methods of network performance analysis. However, these finished work has a large and wide cover in the field. They are various and different from each other, because they have their own unique emphasis and views. There will be more discussion later, here we only list the inducted situations that caused the mess.

1. Different target network is the most common difference among them. There are large amount of different network types and various internal network structure.

2. Researches in existence focus on different performance volumes. Network performance is complex concept, it contains a lot different sub-concepts. One may not need to analyze all of them in most case. One may only want to follow one aspect or a small part of network performance at a specific time, and do not care about other features that much. This mostly happens in some targeted academic researches and comparative works.

3. There is no authoritative definition for network performance, different group has different explanations to network performance. The missing of delimiting about network performance brings difficulties for defining network performance analysis. Therefore, the existed paper are in a wide variety.

Especially the last point, it resulted in the cases that similar topic name fol-
1.3. OVERVIEW

The whole mater thesis generally follows the below procedure.

Chapter-2 shows the background knowledge that relevant to network performance analysis, and clearly define some ambiguous concepts. Also list the inducted points of relevant works. There are introduction of analysis tools with simple example at the end of chapter-2.

The description of the practical part is in chapter-3, environment construction process is also included. The experiment of this thesis is built on the results of Chapter-2. Generally, it is a applying and comparison of two methods of network performance analysis.

Data collecting and presenting is in the next chapter, also comparison and analysis part. Chapter-6 has the conclusion. The whole process inevitably has parts that imperfect and involves some future work, this will be the last chapter of this thesis.
Chapter 2

Background Research

In this chapter, the relevant background knowledge and theory are presented and summarized. Since lots of definitions and concepts are involved, there will be some inductions also.

2.1 Computer Networks and Network services

The first and most important thing that should be studied is the target object of network performance analysis, which is computer network. There is old wise-saying in China: "He who has a thorough knowledge of the enemy and himself is bound to win in all battles." Now the "enemy" is computer network, and the battle is about the methods of analyzing it.

2.1.1 Computer network

A computer network is the infrastructure that allows two or more computers (called hosts) to communicate with each other. The network achieves this by providing a set of rules for communication, called protocols, which should be observed by all participating hosts.

Network layers

Computer networks are running under multiple protocols with architecture. For people who learning networks or other corresponded subjects, we know there are two networks models which are OSI(Open System Interconnection) Reference model and TCP/IP protocol suite model.

The OSI model was developed in 1977 by the ISO(International Organization for Standardization). Network is defined into seven layers in this model, each layer is responsible for its own work.
However, this model is just a ideal reference model, its promised perfect performance has never come true. What we are using most in real world is TCP/IP model, which is proposed out several years earlier than OSI. But this ideal model indeed complement the network theory system and partly make things clearer.

As we are doing performance analysis in real, we cares more about practical things, like TCP/IP model. In TCP/IP protocol suite model, networks is composed of four layers. The boundaries between layers are not rigidly defined as OSI, and these layers are divided by the operation scopes of different protocols. In 1970s, as John Postel observed, internetwork communication should be viewed as having two components: the hop-by-hop relaying of a message and the end-to-end control of the conversation, afterward the old TCP(Transmission Control Protocol) split into two parts: new TCP and IP(Internet Protocol). In new specification, TCP handles packetization, error control, retransmission and reassembly, while IP is responsible for routing packets[5] [6].
2.1. COMPUTER NETWORKS AND NETWORK SERVICES

As the name suggests, TCP and IP are apparently the two core protocols in the TCP/IP model. TCP works at the upper layer of IP, which is transport layer. This layer has another protocol that is similar to TCP but lighter weight, named UDP. TCP provides a reliable transport layer to loss-sensitive applications through retransmission of lost packets, whereas UDP supports delay-sensitive applications without any retransmission.

TCP/IP became popular and was adopted widely since 1980s because of its simplicity, and also because it’s an open protocol[7]. Nowadays, the most of computer networks around us follow TCP/IP model. Based on this facts, we set our target objective networks as TCP/IP networks, and our performance analysis is therefore referring to TCP/IP networks performances.

Network Topology

Computer networks are composed by amount of network deceives, the arrangement of these network devices is network topology[8]. Network topology may affect network performance, but it’s insignificant. There are eight types of network topology[8][9]:

• Point-to-point • Bus • Star • Ring or circular • Mesh • Tree • Hybrid • Daisy chain

2.1.2 Main Network Devices

Although computer network is a complex entirety, the basic physical devices are not very complicated. No matter how big the network is or how difficult the protocol is running on it, in physic there are only two kinds of net devices, which is the terminal device and forwarding device.

Usually, terminal devices are placed at the terminal nodes of networks, like host machine. And that’s where user initiated new network tasks. Forwarding devices are more flexible and they mostly decide the performance of a certain network. Forwarding devices include hubs, switches, routers and servers, etc.

Hubs are not intelligent, they just simply copy every packet received and send it to all rest interfaces. It’s a easy way, but not secure and make network full of noise. Also, there are some special servers forward packets, for example a gateway server, they are usually high performance computers.

2.1.3 Network Services

Inside TCP/IP network structure, the lowest layer network access layer is composed of physical equipment and simple logical links. Although this layer is the very place that real data being transported and migrated, compare to other
2.1. COMPUTER NETWORKS AND NETWORK SERVICES

layers this one is more credible, because function and performance of this layer mainly depends on how strong and powerful the hardware is and some simple configuration set previously.

The major of network services we are using in our daily life is defined and provided by the first layer second layer and the third layer. The first layer, application layer, is usually considered as a friendly interface between user applications and network services. Users can begin, configure or end a network service here, but the "real" network service does not happen here. Protocols enabled in this layer include HTTP, SMTP, SSH and so on. And these application protocols can be realized in four categories: remote login category, file transfer category, e-mail category and support services category [10], but they have nothing to do with the true network services.

In fact, most of network performance measurements and analysis that have been done are aim to the second layer, third layer or both. Because the most genuine network services are provided by transport layer and Internet layer. Therefore, understanding the services provided by this two layers is considerably significant for this thesis. And the study begin with core protocols in TCP/IP networks.

TCP

Now we will briefly go through the network service that provided in transport layer and network layer. As it is well known by information technology students, the most convenient and rapidest way to summarize the services is elaborating with the packet head structure of this layer.

![Figure 2.3: The TCP head format.](image)

From the perspective of applications that running in application layer, a trans-
2.1. COMPUTER NETWORKS AND NETWORK SERVICES

The port layer protocol provides logical communication between application processes working on different hosts\[11\]. And so far there are two transport-layer protocols, TCP and UDP.

TCP provides a reliable and connection-oriented service for upper applications. Its function is included in the head structure in the TCP head format figure. The reason for it can provide reliable service is reflected in its using of acknowledge mechanism, this part is presented in the third line and the flags part of its head structure. This smart mechanism endow reliability and security to the TCP end to end communication to some extent.

Meanwhile, the "window size" part is related to flow control, which effectively prevents buffer overflow and dynamically adjust the communication size refer to the receiver.

This "window size" usually start with a small size and then be changed by receiver or from acknowledge mechanism, this is called "slow-start" mechanism, which is responsible for congestion control.

Therefore, TCP specially equipped with acknowledge mechanism and "slow-start" mechanism to ensure the reliability and security of network service while other parts inside this TCP head structure composed the basic setup of communication.

UDP

UDP is a lighter weight version of transport layer protocol\[12\]. Its head structure is very simple, as it only furnish basic transport service. UDP does not include reliability or security mechanism. However, it is faster and easier and quite widely applied in network internal communication and real-time transport.
2.2. NETWORK PERFORMANCE CLASSIFICATION

IP

All the transport-layer packets will be encapsulated in network layer by IP(internet protocol). IP protocol provides unreliable datagram-based service[12][7].

2.2 Network Performance Classification

After the discussion about our target network and it’s crucial services, now we have to summarize and induct what is network performance from them. As we said above, the concept "network performance" dose not have a exact and clear definition, it is a very subjective and open concept. Therefore, we research the network services and some performance volumes in existence. Only based on these information, we can make network performance this vogue concept clearly defined in our thesis work.

2.2.1 Brief description of network performance

As mentioned previously, there are lot of reports and published papers are concerning on network performance, some cares the transport speed, some cares the packet losses[13] and there are even more aspects. A seriously structuring work has be done and their concern points are put in categories. Before listing and explaining these categories, a model must be introduced first: the IO(Input and Output) system model, for this is the cornerstone of our discrimination work.

The IO system, is system with inputs and outputs. Humans are IO system, food is input and excretion is output, which happens after some proper processing that we call it digestion. All communicated networks are belong to IO system, so do computer networks. As a transport system or IO system, basically computer networks has three important components: source, transport processor and receiver. Figure 2.5 shows the general structure of IO system.

![Figure 2.5: Three important parts of IO Systems.](image)

In other words, you can consider the three parts as main network components. The network devices contained have to play one or more than one roles in networks. And based on these three different role, we place network performance volumes into three classes.
2.2. NETWORK PERFORMANCE CLASSIFICATION

The concerning points of different component’s view within a transport system.[14][15]

**From the sauce, or the network service initiative locality:**
- Call set-up time
- Response time
- Transport error rate
- Security

**From the communicating processor:**
- Network availability
- Circuit availability
- Rerouting time
- Throughput
- Transport speed
- Error correction

**From the receiver part:**
- Transport error rate
- Security

2.2.2 Network performance pyramid

Based on the summarized information above, we can easily induct the network performance architecture. There are two network performance layers: basic communication and reliable communication, the former one is the foundation of the latter one. As showed in figure 2.6, their relationship and proportion just like low layer and high layer in a pyramid.

![Figure 2.6: Two main service classes.](image)

For the basic communication, it means the most staple feature patency, which includes reachable and not very slow. And reliable communication is mainly concerning about performance on application layer. In order to make it more clear, two most representative points for each performance layer are listed.
2.2. NETWORK PERFORMANCE CLASSIFICATION

For basic communication:

- Basic connection and communication
- Rapid communication

For reliable communication:

- Accurate communication
- Secure communication

Each of the above can be considered as an interested performance-measuring aspect. However, the reliable communication performance are mostly provided by application layer, which we considered as a friendly user interface rather than where the real network service happens. So the basic communication performance should be the valuable part to focus on.

2.2.3 Main Performance Metrics

After the above discussions, here comes the part about metrics of network performance. Since what should be concerned on is the basic communication, which in more detail means unimpeded and rapid communication. As mentioned, there are variety of names of network performance volumes. But it is easy to find that they are not independent or close concepts. For instance, there are call set-up time and response time, they are containment relationship between them, and even more, they can both be included in a big performance volume named network delay. Also the same with network availability and throughput.

There are two basis, firstly the network is viewed as a IO system. Secondly, similar and mutually inclusive volumes should be inducted to one volume. In line with these two principles, two performance volumes that comparatively compliant are found. They are Time Delay and Throughput. More details below.

Network delay

Network delay is equal to network time delay, it includes all the delay between the initiation and end of one call. Generally speaking, it contains five aspects of delays, which are packetization delay, processing delay, queuing delay, transmission delay and propagation delay[16][17][18]. The following part describe their definitions, also there are calculation methods of some of them.

- Packetization delay
2.2. NETWORK PERFORMANCE CLASSIFICATION

The packetization delay\([16]\) is not only existed in the main network service layers which are transport layer or Internet layer, but also contained in application layer. Because it represents the time consuming by the protocols and algorithms that doing the data preparing job. As mentioned previously, there are protocols of transport layer and Internet layer work on crafting or modifying data packets in order to make them ready to be sent. They take some time to do these work. Also, programs in the application layer may pre-process the data.

However, the packetization delay \(D_p\) is very slight and minor. Usually, it is ignored. In real networks, a packetization delay is usually stable, almost a constant, and equally added to every packet. If one wants to improve network delay, he should not work on this sub-volume.

• Transmission delay

The transmission delay presents the time that used on push all of the packets bits into transport channel. It affected by two factors, the packet size and transmission rate\([11]\). This scene is like people waiting beside a Ferris wheel in an amusement park. Everyone has to wait the one before them be put into a compartment first, then they can enter the next one.

The formula of calculating transmission delay is:

\[
D_t = \frac{N}{R}
\]

\(D_t\) stands for the transmission delay. \(N\) is the number of bits. And \(R\) is the transmission rate, which usually presented the bits per second.

• Propagation delay

The propagation delay is the time ate on the way. Image a train travel from station A to B in uniform speed, the propagation delay is the time interval from when the train head leave A to when the train head reach to B. In networks, the train header converts to the packet head. So, it is obvious that the propagation delay depends on two parameters: distance and the media propagation speed.

The media propagation speed is usually named wave propagation speed in physics. It describes the media’s propagation ability. In wireless communication, propagation speed equal to \(c\), the speed of light. The generally ranges of copper wire’s speeds are from \(.59c\) to \(.77c\)\([19]\) \([20]\). Wires have an approximate propagation delay of 1 ns for every 6 inches (15 cm) of length\([21]\).

The expression of propagation delay \(D_p\) is:

\[
D_p = \frac{d}{s}
\]
2.2. NETWORK PERFORMANCE CLASSIFICATION

The d stands for the distance, and S is the media propagation speed.

- Queuing delay

Queuing delay[18] is the most important time delay within the total network delay, because it can mostly shows the network devices’ working ability. Queuing delay happens when packets need to wait before being served. It depends on how many packets are in front of and how long do they required to be served.

For example, lots of packets from different sources and target to different destinations have arrived at a router. The router needs to process them one by one, like read the recent packet’s head, check errors and send it to the next hop according to the routing table. Each process takes some time, so the rest has to wait in the router’s buffer. The queuing delay $D_q$ does not have a simple formula, it is quite varied. Tracing and measuring $D_q$ are very useful and helpful for network performance analysis. But to simulate the queuing delay is very difficult, the queuing theory must be applied properly. More details about the queuing theory later.

- Processing delay

Processing delay is the time that routers take to process the packet header, it is a key component in network delay[11] [22].

Like in the previous case, finally one of the packets which have been stayed in the router’s buffer for a long time comes to the serving hall. However, still some time consuming here. The router has to read and check a little about the packet, if everything is fine then send it to next hop, which may be another same repeat tribulation.

The processing delay $D_p$ is variable, it affected by different processor, processing strategies and the current packet itself. That’s why it named $D_p$. Usually, for a certain processor it is steady.

The network delay $D$ in this thesis is measured as the sum of all the above. Note that this calculation method is just a rough and simplified one, a true and accurate function of network time delay is much more complex[23].

$$D = D_s + D_t + D_p + D_q + D_v \ [24]$$

Throughput

The two metrics network delay and throughput are not strictly independent, but better to be measured separately. Because they presents the network performance situation in view of time and space.
2.3. NETWORK PERFORMANCE ANALYSIS METHODS

Throughput is the average number of successfully transmitted bits per second\cite{25}. It is closely related to the channel capacity concept\cite{26}. But the latter one is a only theoretic value.

The throughput of real networks can be tested out by using special network analysis tools. However, network simulating tool use queuing theory to simulate this feature.

2.3 Network performance analysis methods

Generally speaking, all analyzing ways are the same, they all follow the same procedure:

1. Identifying problems;
2. Environment construction;
3. Design and experiment;
4. Data collection and processing;
5. Analysis and conclusion.

However, when doing environment construction, there are two different ways. One can either construct the environment in practice, or use simulate tools. This leads to two different methods of network performance analysis.

No matter how complicated the procedure is, if your data comes from a really existed network then you are doing network performance analysis based on true networks. On the other hand, if your analysis is built on the simulated platform, then you are using the network simulation way. The following parts contain specific description of the two methods.

2.3.1 Analysis based on true networks

Undoubtedly, the most intuitive way of doing performance analysis is getting data from the object itself. Just think about If you want to know how good is your axe, why not just go and chop something. So when the objective network is there and available, all we need is just some proper and nice tools.

Quite a lot of related papers in existence are based on practical networks. They either use the ready-made networks, or build it. Then following the general
2.3. NETWORK PERFORMANCE ANALYSIS METHODS

way of analysis. The hinge of this method is the selection of tools, which depends on what kind of performance you are interested in and in which level you want it.

Here is an example procedure of doing performance analysis based on true network. This case is about analyzing and evaluating the networks in a family, and check whether the network bandwidth is in compliance according to what the ISP promised or not.

Identifying problems

In this case, the goal is to find out how good the network is and check if it has reach the standard. So it is clear that the problem is find out the real bandwidth and analyze the result.

Environment Construction

Everything has already been there, the only thing needs be done is get a host computer and plug in the network cable.

Design and experiment

In this step, two tasks need be solved before experiment begin: decide how to test and which tool to use. In the example case, what in need is just a simple tool that can be run in host machine. So the tool, lets say ntop, is chose, and running on the host machine to get the throughput. Moreover, in order to be considerate, the test should be run several times and in different time.

Data collection and processing

After all tests are finished, the data should be collected, and the average should be calculated from them. The reason for using the average number is it’s more convincing and objective. One may also choose some other mathematical way to process data.

Analysis and conclusion

Based on the data results from last step, more consideration and explanation may be needed. After this is the time for making conclusion.
2.3. NETWORK PERFORMANCE ANALYSIS METHODS

2.3.2 Network Simulation

In reality, there may be many limitations that bring the impossibility of testing and analyzing the true networks. One has to simulate, or partly simulate the target network. The whole analysis procedure would not change, this method is just construct the testing environment in an alternative way.

There are network simulating tools, and generally, all types of network simulating tools are based on queuing theory. Because the queue feature included in computer networks is one of the most important decisive factor of network performance. The following part expounds what is queuing theory and how it works.

Queue Theory

The queue theory is built on queue system, it’s a mathematical theory system that can be applied in waiting lines(queues), and allow researchers calculate and obtain the important variables for further analysis. Queues, of some sort, are central in the majority of models of computer and other communication system[27]. Therefore, when doing network simulation, we need to utilize queue model from queue theory.

Queuing model is consist of three key components, which are arrival process, buffer(waiting room) and service process as showed in figure 2.7. The arrival process describe the features of incoming tasks, such as how many of them, how often they come and so on. After the tasks get inside the system, they may need to wait a little before being served, since the processor may be busy. Also, in this waiting room, there are some strategies decide which task should be the next be processed. Finally the service process, it depicts the features like how long each task need and then let them out after served.

Also, queues are various, this leads to classification of queue systems. There are three important conditions can decide which class a queue system is belong to, the characteristic of arrival process, the service time distribution and the numbers of the processor(server).

Therefore, according to Kendall’s Notation the basic form of different queues is wrote as A/S/m where A stands for nature of inter-arrival times, S denotes the service time distribution and m represents the number of servers.[28]
2.3. NETWORK PERFORMANCE ANALYSIS METHODS

People may self-define the "A" and "S" these two parts with special regulations or use existed rules. For instance, when AS=M, it means the arrival process is a random process or the service time obey the random distribution such as Poisson distribution; and if AS=G, this is a General distribution; if AS=D, it stands for deterministic, which means the inter-arrival times or the service times is constant. The inter-arrival times and service times may follow different distribution rules or regulations.

Not only the queues in queue system are different, the internal processor’ behaviors are divergent too. What we mean processor’s behavior here is decisions that concerned on which "customer" in the queue is going to be served next. There are mainly four types as elaborated below[29][25].

• FIFO -First in first out.
• LIFO -Last in first out.
• PS -Processor sharing.
• IS -Infinite server.
• RED -Random early detection.

All the above parts are general explanation for queuing theory, this theory system allows us to find a specialized queuing model that is suitable and may be properly applied to simulate a certain computer network, and this is not easy at all[30]. The motivation behind the application is one can obtain several significant values by using proved algorithms and formulas within queue model, and what’s more, we have the Little’s Rule that developed by different authors, can help us in the same way. These values, to some extent, represent the performance level of the original system.

Little’s Rule
Although each of the specific queue models are diverse, the Little’s Rule is conditionally universal. It is shown that, if the three means(mean of customers number in queue ‘L’, mean of arrival rate “λ” and mean of times that each customer spend in queue system) are finite and the corresponding stochastic processes strictly stationary, and, if the arrival process is metrically transitive with nonzero mean, then:

\[ L = \lambda W. \] \[28][31]\]

2.3.3 Comments and Comparison
Problem solving may begin with modeling, and in our case queue models that based on queue theory is really a powerful and helpful theory system. It describes the performance level of the original system.
2.3. NETWORK PERFORMANCE ANALYSIS METHODS

However, it is not difficult for us to realize that queue theory is mainly concerning on system resource assignment caused altering of system variables.

As we discussed before, computer networks is a data IO system, and data IO system has three crucial parts: source, processor and receiver. When applying queue theory, the source part, or in another word the arrival process is considered as obeying some certain distributions which means it's pre-fixed, and so does the service times. It is a processors’ performance evaluation based on idea modeling and it focus on evaluating the entire system resource(processor). Here comes the limitation of queue theory:

• Although the mean numbers we obtain from queue theory can be considered as a evaluation, but as they are mean numbers of whole system, it might be less significant to a single user.

• The entire queue theory is based on idea case, things are different in real world. Especially on the input distribution and time consuming distribution two parts. Therefore, to find a suitable or a serial of suitable models is complicated and involved more mathematics.

• This theory system is built on a single aspect that is resource sharing, and it is obviously that only one single view from one respect is not enough for a complete and systemic performance analysis.

There are indeed lots of existed network performance analysis work are done by following other idea. We can see many packet tracing tools and network monitoring tools are popular and widely applied. They can give us real values of network traffic’s volumes. Although those real numbers might have much haphazard, coincidence and affected by random noise, but since they are real numbers without any assumptions, they are quite helpful. Moreover, the random factors can be ignored by increasing the sample amount.

Beside of the side effects caused by factors such as noise, there is another malpractice of real data analysis on network performance. It must be set up on a network that in the actual existence, which means it can not do anything about prediction analysis before construction a network constructed. And this becomes like a belated action, it may be correct but just too late. It also brings problem like high costs of hardware resources and time.

The two methods of network performance analysis both have their own advantages and shorts. One may need to trade-off between them and make a smart choice. However, this leads in more problems such as which method is more accurate? And will they both lead to the same conclusion? An experiment based on these questions is presented in following chapters.
2.4 Network performance Measuring Tools

In this section, some popular tools aim on network performance analysis are listed. Based on the two methods we discussed above, we introduce tools for them one by one. There are also simple examples for some of them.

2.4.1 Simulating tool

In fact, there are not many blameless or at least good enough network simulator in existence. Fortunately, we did find one works, which named ns(Network Simulator). Ns is a open-source project targets on simulating networks, it is a object-oriented simulator and written in C++. The development team( which is an open team) behind has been working on it for a long time. Ns3, the newest version of ns is utilized in the practical part of this thesis.

One may need to know about C++ language, since the simulating scripts must be written in C++. What the user need to do is writing codes and invoking the header files that ns3 provide. It is not easy to use at all, but there is a way. We are using it in our experiment, so there will be details and compiled codes available late. Here we only present some major working principles and its structures.

Since ns-3 is a object oriented project, the most important thing is to find out what kind of network object that ns-3 is providing to us and how it provides.

Node

Generally, the basic computing device is called node in ns-3. This concept of node is abstracted by the class Node in C++. The Node class provides methods for managing the representations of computing devices in simulations.[32]

To be more specifically, "Node" is a abstract device, which is going to be equiped and become a certain device such as host computer and router. Switches are different, it does not need to be installed on a node.

Application

As the name suggested, the application class is used to define what kind of application a user want from a node. For example, it can be specializations class named UdpEchoClientApplication and UdpEchoServerApplication. They are obvious for managing UDP communication.
2.4. NETWORK PERFORMANCE MEASURING TOOLS

Channel

The Channel class represents the links between nodes, or in another word network devices. This class is used for describe the features of channels. Specifically, there are PointToPointChannel subclass, CsmaChannel subclass and so on.

Net Device

The network device abstraction is represented by the class NetDevice in C++. This class provides methods for managing connections to Node and other objects. It can be specialized by developers[32], or in another word users.

Topology Helpers

The topology helpers are used for creating networks contain complicated topology structures. This allows user to choose a proper topology model instead of trying to connect multiple devices or individual sub-networks.

The figure 2.8 and 2.9 show the architecture of ns-3 and how it works with the classes it provides. The only difference is the network in 2.8 using Channel class since it’s a simple network while the other one using Topology Helper as it’s comparatively complicated.

![Figure 2.8: The architecture of ns-3](image)

2.4.2 Tools aim on actual network

A network analyzer, also called a "packet analyzer," "traffic analyzer" and "protocol analyzer."[33]

As discussed before, we can also collect data through tools. If we are on the way of doing real performance analysis, which means we collect real data from networks in practical, we have another kind of tools such as tcpdump, ntop,
2.4. NETWORK PERFORMANCE MEASURING TOOLS

There are quite a lot of tools, and they are and they are all kind of good. This field is more developed compare to network simulating, and much more easier.

Although there’s lots of options, they are similar in principles. All these tools can be recognized into two types. One is passive surveillance, another one is proactively testing. The difference between them is the former one dose not generate big amount of test packets, while the latter one obtaining consequential data by creating lots of testing packets. That is to say, they can also be categorized as monitoring tools and benchmarking tools[7]. It’s the same classification but different names. Certainly, there are other classification to these network performance analysis tools, like Software tools and Hardware tools[34] that are based on the different implementations.

The figure 2.10 shows the types of network performance analyzing tools.

Among these tools, which one to choose is very flexible and depends on each specific case. In the case of this thesis, a combination of several tools is chose. The following parts will give a briefly description to each tool.

nmap

Strictly speaking, nmap is not a performance analysis tool. The most famous feature of it is showing the network structure. However, it’s scanning function do give very useful and helpful information for performance analysis. Its features included host discovery, port scanning, OS detection, and in addition to these, Nmap can provide further information on targets, such as reverse DNS names, device types, and MAC addresses[35].

There is a simple example about a small home network, the target is to see the
2.4. NETWORK PERFORMANCE MEASURING TOOLS

Figure 2.10: The types of network performance analyzing tools.

internal structure and situation. The nmap is running on machine 10.0.0.17, the command is:

```
- nmap 10.0.0.*
```

<table>
<thead>
<tr>
<th>PORT</th>
<th>STATE</th>
<th>SERVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>80/tcp</td>
<td>open</td>
<td>http</td>
</tr>
<tr>
<td>139/tcp</td>
<td>open</td>
<td>netbios-ssn</td>
</tr>
<tr>
<td>445/tcp</td>
<td>open</td>
<td>microsoft-ds</td>
</tr>
<tr>
<td>9100/tcp</td>
<td>open</td>
<td>jetdirect</td>
</tr>
<tr>
<td>53/tcp</td>
<td>open</td>
<td>domain</td>
</tr>
<tr>
<td>80/tcp</td>
<td>open</td>
<td>http</td>
</tr>
<tr>
<td>111/tcp</td>
<td>filtered</td>
<td>rpcbind</td>
</tr>
</tbody>
</table>

The output of nmap.
2.4. NETWORK PERFORMANCE MEASURING TOOLS

The output lists all the network devices that has an IP-address and their opened ports. And from the services, one may find out what the device’s role in the network. For more information, one can run some more commands target to the very address.

ntop

For those Linux user, ntop just likes the command "top". It shows network usage information and variety of network performance parameters. It is a powerful tool, all the data and statistical results are presented in web. One can just open a web browser and check all the information.

The figure 2.11 is the output excerpts of ntop. Still in the small home network, and ntop is running on machine 10.0.0.17. Figure 2.11 shows the throughput of the last 10 minutes, also includes some prediction lines such as anomaly, upper, lower and trend based on real data. These prediction lines are using some algorithms, but has nothing to do with network performance.

However, the throughput result from ntop is the actual consumed channel capacity but not the maximum throughput of the channel. And ntop is lack of information about network time delay.

iperf

The above two are both belong to passive surveillance tools, and here comes iperf, which is a proactively testing tool. It can create large amount of packets and generate result data based on those packets.

The iperf now has a graphical user interface front end version created with java, named jperf. However, the internal working core is still the iperf. The main feature of iperf is measuring the throughputs of networks. The below part shows a example
2.4. NETWORK PERFORMANCE MEASURING TOOLS

with iperf.

The example test environment is still the small home network that being used before, and we use jperf. It is different from the above two, jperf needs two host machines, one acts like a server and another one is the client. The client machine send packets to server, and the throughput is actually the maximum throughput of the link between these two machines.

Host 10.0.0.17 is the server, and another laptop connected with wireless is the client. The laptop has the IP address 10.0.0.6. The corresponding iperf command is:

`iperf -c 10.0.0.17 -P 1 -i 1 -p 5001 -f k -t 10`

The command run on the laptop, "-c" means the current machine is client, the following address is the server address, "-P" specifies the number of parallel streams, here is only 1.

-`-i 1`: the report interval, here is report each second.
-`-p 5001`: the server’s opened port for testing.
-`-f k`: the output format, "k" stands for KBits.
-`-t 10`: the testing time, here the test lasts for 10 seconds.

The figure 2.12 shows the output of iperf in graphic. It is not so stable, and this is caused by using wireless. To make the result more convincing, more repeat tests and mathematical opeations should be applied.

![Figure 2.12: The graphical output of iperf.](image)

**hp_ing**

These two are packets generators. Usually, ping is the default tool for checking if a link is smooth or not, it has options to specify what kind of packets you want to send, however it can’t choose the protocol. And hping is kind of an advance version of ping[36]. Network time delay is included in their outputs.

Here’s a example command abut generates some tcp packets with hping3. The host 10.0.0.17 sends a packets under TCP protocol(default) to 10.0.0.6:9234, and the data content size is 1024 bytes.
2.4. NETWORK PERFORMANCE MEASURING TOOLS

**hping3**
```
hping3 10.0.0.6 -p 9234 -d 1024
```

**tcpdump**

Usually, tcpdump is used to read the traffic through a host, it is a packet analyzer. Since there are packets generated from hping, tcpdump can be used to read them. The tcpdump command below is listen to the network interface en1, "-n" indicates show the numbers instead of names.

```
tcpdump -i en1 -n
```

The output is partly showed below with first volume, which shows the time bypassed.

```
1  ... IP 10.0.0.17.1618 > 10.0.0.6.9234: Flags [ ], seq 861730578:861731602, win 512, length 1024
2  ... IP 10.0.0.6.9234 > 10.0.0.17.1618: Flags [R.], seq 0, ack 861731602, win 0, length 0
```
Chapter 3

Methodology

Based on the previous research and discussion, there are two general way of doing network performance analysis: network simulation and true network analyzing. For network simulation, the analyzing data is from a simulated platform. And for another method, the analyzing data comes from real networks. Except the different data source, other procedures are the same.

In general, researchers only choose the proper one according to the available conditions. However, are these two ways equal effective is not clear. There should be difference between them, and the reference values of their results.

The practical part of this thesis are extracted from the previous content, and will explore the above issues. The practical experiment contains two sections, one is analysis built on simulated platform, another one is built on real networks. Both of them have the same network layout and system parameters.

3.1 Network design

In this part, the network layout and the experiment ideas will be presented.

3.1.1 Network structure

There are different network topology structures and different network devices. To compare two network performance analysis ways, the layout of the network for testing must contain various network devices and blend of multiple structures.

The main idea of the network design is making a comparatively comprehensive network under the limited resource, then place a pairs of host machines at different positions each time. The data of each time comes from different links, this renders the diversity of the data source. The designed network is closed, it does not have any exit to the Internet. And it doesn’t have very complicated network topology such as grid structure, even it do contain different topologies. Complicated network structure brings difficulties to simulation work[37][38].

The network layout is showed in figure 3.1. There are five host computers included, but they are not necessary to be there at the same time, the figure just showing the
3.1. NETWORK DESIGN

Figure 3.1: The test network layout.

possible spots for placing a host. In the experiment, a pair of host machines is chose every time.

3.1.2 Experiment arrangement

Based on the network layout, the concrete plan of the experiment is running couples of tests among groups. There are four groups, each group has two hosts. As showed in figure 3.2. For each group, the two network performance metrics will be measured are network delay and throughput.

Moreover, in order to make the network more natural, there mast be some segmentations. Since there are routers, the network are easily segmented with routers. The distribution of sub-networks are showed in figure 3.3, and their network addresses and related setups are showed in the table below.

<table>
<thead>
<tr>
<th>Sub-networks</th>
<th>Address</th>
<th>Gateway Address</th>
<th>Gateway Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network 1</td>
<td>10.0.1.0/24</td>
<td>10.0.1.3</td>
<td>Router 3</td>
</tr>
<tr>
<td>Network 2</td>
<td>10.0.2.0/24</td>
<td>10.0.2.2</td>
<td>Router 2</td>
</tr>
<tr>
<td>Network 3</td>
<td>10.0.3.0/24</td>
<td>10.0.3.4</td>
<td>Router 4</td>
</tr>
<tr>
<td>Network 4</td>
<td>10.0.4.0/24</td>
<td>10.0.4.1</td>
<td>Router 1</td>
</tr>
<tr>
<td>Network 5</td>
<td>10.0.5.0/24</td>
<td>10.0.5.1</td>
<td>Router 1</td>
</tr>
<tr>
<td>Network 6</td>
<td>10.0.6.0/28</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Network 7</td>
<td>10.0.7.0/28</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Network 8</td>
<td>10.0.8.0/28</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Note that in this network, all the switches do not have their own IP address, they are actually act like a network bridge. And for all the router, static routing is the routing
3.1. NETWORK DESIGN

Figure 3.2: The arrangement of test groups.

Figure 3.3: The network segmentations.

strategy that applied.

As the experiment part of this thesis contains two sections, which are built on different platforms, the following parts elaborate them separately. Besides, the schedule about tools for measuring the two network performance metrics is that iperf and hping will be used on the true network, and for the simulated network the related data will be generated by the internal mechanism of ns3.
3.2 Construction of Target Network

In order to get definite information and parameters of network system for simulating, the true network must be constructed first. There is no mandatory requirements about what type of network device to choose. Using Cisco 2800 serial or 2900 serial is same in this case. The point is to record the information of the specific type network device, and configure the simulating platform under the information.

The types of network devices are listed below:

<table>
<thead>
<tr>
<th>Device name</th>
<th>Device type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Router 1-4</td>
<td>Cisco 2800 Serial</td>
</tr>
<tr>
<td>Switch 1-2</td>
<td>D-Link DES-1008D</td>
</tr>
<tr>
<td>Switch 3</td>
<td>Catalyst C2980 G-A</td>
</tr>
</tbody>
</table>

And the information of cables:

<table>
<thead>
<tr>
<th>Cable Name</th>
<th>Cable Type</th>
<th>Propagation Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Straight-through</td>
<td>CAT 5E[40]</td>
<td>4.80-5.30 nsm</td>
</tr>
<tr>
<td>Serial Cable[41]</td>
<td>CL 2</td>
<td>4.92-5.18 nsm</td>
</tr>
</tbody>
</table>

Propagating delay is a important part of network delay, and it depends on the inherent delay rate and the length of cable. All the serial cables in using are 5 meters, but straight-through cables have various length. It is showed in figure 3.4. Also, the serial cable has DCE and DTE two parts, and the router connected to the DCE cable controls the clock rate. These information is included in figure 3.4 as well.

Figure 3.4: The cable information of network.
3.3. SIMULATION OF TARGET NETWORK

According to the above information, the time consuming on propagation is easy to calculate. As an instance, for the 5 meters long straight-through cable, its propagation delay is known as 5 nsm in rate form, thus its total propagation delay is:

\[ D_p = \text{Propagation delay rate} \times \text{Cable length} \]
\[ = 5 \text{ nsm} \times 5 \text{ meters} \]
\[ = 25 \text{ ns} \]

The construction of real network is not difficult but a cumbersome work. And a lot of information and values must be recorded carefully. The final entity of the network is showed in figure 3.5.

![Figure 3.5: The final entity of the network.](image)

3.3 Simulation of Target Network

As discussed before, ns-3 is a comparatively mature and complete simulating tool, therefore it is chosen for simulating in this part. Since ns-3 is an open source project wrote in C++, user has to have some basic knowledge of C++ and be able to write C++ code.
3.3. SIMULATION OF TARGET NETWORK

There are three kinds of network devices involved, they are router, switch and host machine. In total, there are 12 network nodes and 13 links. All these have to be simulated step by step.

3.3.1 Network Nodes Simulation

Firstly, 13 blank nodes should be created for equipping them with certain network devices later.

Name rules

Normally, the names of the nodes will be created with the first character of the device type they are going to be and the number they have according to figure 3.1. For instance, router 1 is named r1 in the simulating code. And for those nodes for host machine, their name start with n and followed by the device they are directly connected.

However, not or of them follow the rules. I made some mutations to allow the names be more easier to be recognized by me, like switch-1 is named s2, switch-2 is s22 and switch-3 is s222. The code below shows how the nodes are created. The "Ptr<Node>" returns a smart pointer to a node object.[...]

```cpp
Ptr<Node> r3 = CreateObject<Node>(); // router R3
Ptr<Node> r2 = CreateObject<Node>(); // router R2
Ptr<Node> nR3 = CreateObject<Node>(); // node connected to R3
Ptr<Node> nR2 = CreateObject<Node>(); // node connected to R2
Ptr<Node> s2 = CreateObject<Node>(); // Switch S2
Ptr<Node> nS2 = CreateObject<Node>(); // node connected to S2
Ptr<Node> r1 = CreateObject<Node>(); // Router R1
Ptr<Node> nR4 = CreateObject<Node>(); // node connected to R4
Ptr<Node> s22 = CreateObject<Node>(); // Switch S22
Ptr<Node> nS22 = CreateObject<Node>(); // node connected to S22
```

3.3.2 Network Segmentations Simulation

After the 13 nodes have been created, now the network segmentations should be arranged. The object NodeContainer uses the word "container" to define the concept of network area, it has the index of nodes that belong to a certain network area. In this section of coding work, a node-container is first crated with only one node, then a function named "Add" of this object is invoked and more nodes get contained.

```cpp
// Nodes grouped by what network they are on, excluding switches
NodeContainer nodes_10_0_1_0 = NodeContainer(r3); // NodeContainer(r3);
NodeContainer nodes_10_0_2_0 = NodeContainer(r2); // NodeContainer(r2);
NodeContainer nodes_10_0_3_0 = NodeContainer(r4); // NodeContainer(r4);
NodeContainer nodes_10_0_4_0 = NodeContainer(r3); // NodeContainer(r3);
NodeContainer nodes_10_0_5_0 = NodeContainer(r1); // NodeContainer(r1);
NodeContainer nodes_10_0_5_0 = NodeContainer(r1); // NodeContainer(r1);
NodeContainer nodes_10_0_5_0 = NodeContainer(r1); // NodeContainer(r1);
NodeContainer nodes_10_0_5_0 = NodeContainer(r1); // NodeContainer(r1);
NodeContainer nodes_10_0_5_0 = NodeContainer(r1); // NodeContainer(r1);
NodeContainer nodes_10_0_5_0 = NodeContainer(r1); // NodeContainer(r1);
...`
```
3.3. SIMULATION OF TARGET NETWORK

The reason why switches are excluded is they don’t have their own IP address in this case, they are acting like network bridges and learn the current local network segmentation automatically.

Besides, \texttt{Ipv4AddressHelper} is needed. This is how the network address is assigned. As its name suggested, this object helps us process IPv4 address information. Especially in this case, there are networks with different length of sub-net mask. The below code is about the network information of a serial network.

```c
1 Ipv4AddressHelper net_10_0_7_0;
2 net_10_0_7_0.SetBase("10.0.7.0", "255.255.255.240");
3 Ipv4InterfaceContainer i_10_0_7_0 = net_10_0_7_0.Assign(devices_7);
```

3.3.3 Network Topology Simulation

The next thing needs be done is creating and configuring the connection between nodes, then put the created connection and connected nodes together in a \texttt{NetDevice-Container} object allow a easier management. One should be very careful when deciding which topology helper to use. Take the sub-network 10.0.1.0 for instance, it has two nodes R3(router 3) and nR3(laptop 2), the code is showed below:

```c
1 ....
2 PointToPointHelper p2p_1;
3 p2p_1.SetDeviceAttribute("DataRate", StringValue("100Mbps"));
4 p2p_1.SetChannelAttribute("Delay", StringValue("50ns")); // 10m cables
5 NetDeviceContainer devices_1;
6 devices_1 = p2p_1.Install(nodes_10_0_1_0);
7 ....
```

The topology simulation around switch is not so direct, since they don’t have IP address and they are connected to many other devices. Hence the object \texttt{CsmaHelper} is leaded in, it implements a carrier sense multiple access communication and gives Ethernet-like functionality.[....] The code below shows us a example on creating a csma connection with the cable length equal to 2 meters.

```c
1 ....
2 CsmaHelper csma_2m; // cables used for S2 is 2 m
3 csma_2m.SetChannelAttribute("DataRate", StringValue("100Mbps"));
4 csma_2m.SetChannelAttribute("Delay", StringValue("10ns"));
5 ....
```

3.3.4 Network Devices Simulation

Normal host machine node dose not need to be specifically simulated, just need to assign it an IP address and install the necessary net device and proper protocols. And router is almost same as host machine, only one thing is different: router knows more routing rules. However, the simulation of switch is quite complicated.

Switch

The way of simulating a switch is following the procedure: put all the csma connections that belong to one switch node, together with a switch controller. Treat them as parameters and use the object \texttt{OpenFlowSwitchHelper} install all of them on the corresponding node.
3.3. SIMULATION OF TARGET NETWORK

```c++
// BEGIN SWITCH S2 (network 10.0.4.0) //
connect R3 to S2
NetDeviceContainer devices_4_1;
devices_4_1 = cmc_2m.Install(NodeContainer(r3, s2));
connect R1 to S2
NetDeviceContainer devices_4_2;
devices_4_2 = cmc_2m.Install(NodeContainer(r1, s2));
connect R2 to S2
NetDeviceContainer devices_4_3;
devices_4_3 = cmc_10m.Install(NodeContainer(R2, s2));
OpenFlowSwitchHelper switchS2Helper;
Ptr<ns3::ofi::LearningController> controllerS2 = CreateObject<ns3::ofi::LearningController>();
NetDeviceContainer switch_S2_devices; // the ports of S2
switch_S2_devices.Add(devices_4_1.Get(1));
switch_S2_devices.Add(devices_4_2.Get(1));
switch_S2_devices.Add(devices_4_3.Get(1));
switchS2Helper.Install(s2, switch_S2_devices, controllerS2);
// END SWITCH S2 //
```

The above code shows how a switch is created.

**Router**

The object for simulating a static routing router is named `Ipv4StaticRoutingHelper`, it actually installs a routing table inside a normal node, and router just have more routing rules than normal host machine. Normal host machine need this object as well, for it helps to configure and save the gateway information.

The reason for choosing `Ipv4StaticRoutingHelper` is within the true network all the routers are configured using static routing rules, here must be the same too.

The code below shows the instance of how the routing table of router-2 is created.

```c++
Ipv4StaticRoutingHelper routingTable;
// Routing for R2, 10.0.2.0 is directly connected
routingeTable.GetStaticRouting(r2, GetObject<Ipv4>()); //->AddRouteTo(Ipv4Address("10.0.1.0"),
Ipv4Mask("255.255.255.0"), 2); // R2 -> 10.0.1.0 = R3
routingeTable.GetStaticRouting(r2, GetObject<Ipv4>()); //->AddRouteTo(Ipv4Address("10.0.3.0"),
Ipv4Mask("255.255.255.0"), 3); // R2 -> 10.0.3.0 = R1
routingeTable.GetStaticRouting(r2, GetObject<Ipv4>()); //->AddRouteTo(Ipv4Address("10.0.4.0"),
Ipv4Mask("255.255.255.0"), 2); // R2 -> 10.0.4.0 = R3
routingeTable.GetStaticRouting(r2, GetObject<Ipv4>()); //->AddRouteTo(Ipv4Address("10.0.5.0"),
Ipv4Mask("255.255.255.0"), 3); // R2 -> 10.0.5.0 = R1
```

### 3.3.5 Application Simulation

The application simulation part is very important, because the network performance measuring is based on this. If no communication within the network, there will be no data, no performance analysis and no comparison. So there must be some communications.

In the real network section, iperf and hping are used for measuring the two network performance metrics. But here in this simulated platform, the values of the target metrics come from processed traffic data. So the main idea of simulating applications is
3.4. EXPERIMENT PROCEDURES

generating packets and put the traffic information inside log files.

For creating traffic, the object PacketSinkHelper and OnOffHelper are used. The former helps the host machine collecting packets like a server, while the latter helps another host generate and send packet like a client. The is one more thing, the OnOffHelper does not send packets uniformly, but in some high and low alternative pattern. This is why it called OnOffHelper. The code below shows how to simulate communication between host nR4 and nS222, the whole communicating process take 400 seconds.

```cpp
uint16_t port = 50000; // set the port that going to use
Address sinkLocalAddress(InetSocketAddress(Ipv4Address::GetAny(), port));
PacketSinkHelper sinkHelper("ns3::TcpSocketFactory", sinkLocalAddress); // packet protocol is TCP
ApplicationContainer sinkApp = sinkHelper.Install(nR4);
sinkApp.Start(Seconds(1.0)); // server duration
sinkApp.Stop(Seconds(401.0));

OnOffHelper clientHelper("ns3::TcpSocketFactory", InetSocketAddress("10.0.3.2", port));
clientHelper.SetAttribute("OnTime", RandomVariableValue(ConstantVariable(2)));
clientHelper.SetAttribute("OffTime", RandomVariableValue(ConstantVariable(1)));
clientHelper.SetAttribute("DataRate", DataRateValue(DataRate(100*1024*1024*8)))); // send 1MB per second
ApplicationContainer clientApps = clientHelper.Install(nS222);
clientApps.Start(Seconds(2.0)); // client duration
clientApps.Stop(Seconds(400.0));
```

And certainly, some record and statistical work is need about the traffic. Because it is not possible to install any network performance analysis tool on ns-3 simulated host, the traffic information must be processed with C++ code. The target performance metrics are network delay and throughput.

```cpp
uint32_t totalRxBytesCounter = 0;
for (uint32_t i = 0; i < sinkApp.GetN(); i++)
|
  Ptr<Application> app = sinkApp.Get(i);
  Ptr<PacketSink> pktSink = DynamicCast<PacketSink>(app);
  totalRxBytesCounter += pktSink->GetTotalRx();

NS_LOG_UNCOND("TcpVariant : unk
Goodput Bytes/sec : 
Simulator : Now ().GetSeconds ()");
```

3.4 Experiment Procedures

In general, the target network are tested in four groups. According to the name rules in simulation, the four groups are listed below:

1. nS222 ↔ nR4;
2. nS22 ↔ nR4;
3. nR2 ↔ nR4;
4. nR3 ↔ nR4.

For the real network, four tests are applied to each group with iperf. As a iperf server, the node nR4 will not change. The four tests last for different times and have different time intervals:

- Lasts for 30 seconds with interval equaling to 1 second.
- Lasts for 90 seconds with interval equaling to 1 second.
3.4. EXPERIMENT PROCEDURES

- Lasts for 60 seconds with interval equaling to 1 second.
- Lasts for 30 seconds with interval equaling to 2 second.

As there are always random noise and uncontrollable factors in real, increasing the tests’ amount and changes may reduce the random affection. However, this only give us the throughput values. For network time delay, hping is utilized and the time measured is rtt(round-trip-time)[43]. Similarly, 30 numbers are gained from each group and the four averages are the final one counted when analyzing.

For the simulation section, the data of throughput and network time delay are obtained with C++ script through the internal mechanism of ns-3.
Chapter 4

Data Presentation and Analysis

4.1 Throughput from the real network

There are four test groups, and the throughputs of them are measured with iperf. For each group, several tests were ran. The test commands are showed below, they have different total running time or different time interval. This arrangement is for getting a comparatively accurate value.

Test 1: `iperf -c 10.0.3.19 -P 1 -i 1 -p 5001 -f k -t 30`
Test 2: `iperf -c 10.0.3.19 -P 1 -i 1 -p 5001 -f k -t 90`
Test 3: `iperf -c 10.0.3.19 -P 1 -i 1 -p 5001 -f k -t 60`
Test 4: `iperf -c 10.0.3.19 -P 1 -i 2 -p 5001 -f k -t 60`

• For group-1, it actually measures the link between nS222 and nR4. The four tests results is showed from figure 4.1 to figure 4.4. The averages of throughput are 5844, 5684, 4797 and 5810 Kbits/sec. As iperf measures throughput second by second, the throughput line is quite unstable.

![Figure 4.1: The test-1 on group-1. (Total time: 30 sec; Time interval: 1 sec.)](image)

• For group-2, it actually measures the link between nS22 and nR4. The four tests results is showed from figure 4.5 to figure 4.8. The averages of throughput are 88.0, 65.2, 55.9 and 70.9 kbits/sec. The performance is much worse, as within this link there
4.1. THROUGHPUT FROM THE REAL NETWORK

Figure 4.2: The test-2 on group-1. (Total time: 90 sec; Time interval: 1 sec.)

Figure 4.3: The test-3 on group-1. (Total time: 60 sec; Time interval: 1 sec.)

Figure 4.4: The test-4 on group-1. (Total time: 60 sec; Time interval: 2 sec.)
4.1. THROUGHPUT FROM THE REAL NETWORK

is a serial connection, which is usually worse than fast Ethernet connection. Also this link is longer compare to group-1, and more devices are involved.

Figure 4.5: The test-1 on group-2. (Total time: 30 sec; Time interval: 1 sec.)

Figure 4.6: The test-2 on group-2. (Total time: 90 sec; Time interval: 1 sec.)

• For group-3, it actually measures the link between nR2 and nR4. The four tests results is showed from figure 4.9 to figure 4.12. The averages from four tests are 87.1, 64.9, 64.4 and 64.5 Kbits/sec. The throughput of this group is still low, and it tends to be worse than the previous group. This is because the group-3’s link is even longer and passed more devices.

• For group-4, it actually measures the link between nR3 and nR4. The four tests results is showed from figure 4.13 to figure 4.16. The averages of throughput are 88.4, 59.1, 66.4 and 71 Kbits/sec. The link of this group is the longest, and involved most devices, but its throughput is roughly equal to the previous two.

According to all the results among four groups, at least one thing is easy to tell: the first group’s performance is much better than the other three, and the performance in throughput of the other three groups are quite similar. So what’s the reason that causes a large reducing of the throughput? There are only two difference between
4.1. THROUGHPUT FROM THE REAL NETWORK

Figure 4.7: The test-3 on group-2. (Total time: 60 sec; Time interval: 1 sec.)

Figure 4.8: The test-4 on group-2. (Total time: 60 sec; Time interval: 2 sec.)

Figure 4.9: The test-1 on group-3. (Total time: 30 sec; Time interval: 1 sec.)
4.1. THROUGHPUT FROM THE REAL NETWORK

Figure 4.10: The test-2 on group-3. (Total time: 90 sec; Time interval: 1 sec.)

Figure 4.11: The test-3 on group-3. (Total time: 60 sec; Time interval: 1 sec.)

Figure 4.12: The test-4 on group-3. (Total time: 60 sec; Time interval: 2 sec.)
4.1. THROUGHPUT FROM THE REAL NETWORK

Figure 4.13: The test-1 on group-4. (Total time: 30 sec; Time interval: 1 sec.)

Figure 4.14: The test-2 on group-4. (Total time: 90 sec; Time interval: 1 sec.)

Figure 4.15: The test-3 on group-4. (Total time: 60 sec; Time interval: 1 sec.)
4.2 Network Delay from the real network

In this part, hping was applied on all four groups with all the same enactments. For each group, 30 values are obtained, as showed from the figure 4.17 to 4.20, together with each group’s calculated average of total delay.

```
sudo hping3 10.0.3.19 -p 9234 -c 30
```

The averages of network time delay among four groups are 7.4867, 7.4933, 28.983 and 28.99 ms. As mentioned before, there are mainly four types of network delay, compare the first two groups and the last two, the sub-volumes that constructed most to the differences should be propagation delay and queuing delay.
4.2. NETWORK DELAY FROM THE REAL NETWORK

Figure 4.18: The network time delay of group-2.

Figure 4.19: The network time delay of group-3.

Figure 4.20: The network time delay of group-4.
4.3. THROUGHPUT FROM THE SIMULATED NETWORK

4.3 Throughput from the Simulated Network

The throughput value of the network simulated by ns-3 is always the same, no matter how many times you have run the simulating script. Since every parameter is fixed, and no system noise and random factors, there is no reason for running multiple times. All the results are the same.

For group-1, the output is listed below:

The throughput of group-1 with ns-3.

```
1 milan@Mabel:~/ns3/ns-allinone-3.13/ns-3.13$ ./waf --run milan
3 [ 626/1788] cxx: scratch/milan.cc -> build/scratch/milan.cc.3.o
4 [1692/1788] cxxprogram: build/scratch/milan.cc.3.o -> build/scratch/milan
5 Waf: Leaving directory '/home/milan/ns3/ns-allinone-3.13/ns-3.13/build'
6 'build' finished successfully (2.299s)
7 ------------------------------
8 Bytes/sec: 839270
9 ------------------------------
```

As in the real network section, the iperf use "Kbits/sec" as the unit of network throughput, here the unit is different.

Conversion: 839270 Bytes/sec = 6714.16 Kbits/sec.

For group-2, the output is listed below:

The throughput of group-2 with ns-3.

```
1 milan@Mabel:~/ns3/ns-allinone-3.13/ns-3.13$ ./waf --run milan
3 [ 626/1788] cxx: scratch/milan.cc -> build/scratch/milan.cc.3.o
4 [1714/1788] cxxprogram: build/scratch/milan.cc.3.o -> build/scratch/milan
5 Waf: Leaving directory '/home/milan/ns3/ns-allinone-3.13/ns-3.13/build'
6 'build' finished successfully (2.593s)
7 ------------------------------
8 Bytes/sec: 14458.6
9 ------------------------------
```

Conversion: 14458.6 Bytes/sec = 115.6688 Kbits/sec.

For group-3, the output is listed below:

The throughput of group-3 with ns-3.

```
1 milan@Mabel:~/ns3/ns-allinone-3.13/ns-3.13$ ./waf --run milan
3 [1452/1788] cxx: scratch/milan.cc -> build/scratch/milan.cc.3.o
4 [1723/1788] cxxprogram: build/scratch/milan.cc.3.o -> build/scratch/milan
5 Waf: Leaving directory '/home/milan/ns3/ns-allinone-3.13/ns-3.13/build'
6 'build' finished successfully (2.746s)
7 ------------------------------
8 Bytes/sec: 14096.8
9 ------------------------------
```

Conversion: 14096.8 Bytes/sec = 112.7744 Kbits/sec.

For group-4, the output is listed below:
4.4. NETWORK DELAY FROM THE SIMULATED NETWORK

The throughput of group-4 with ns-3.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>[1452/1788]</code></td>
<td>cxx: scratch/milan.cc -&gt; build/scratch/milan.cc.3.o</td>
</tr>
<tr>
<td><code>[1724/1788]</code></td>
<td>cxxprogram: build/scratch/milan.cc.3.o -&gt; build/scratch/milan</td>
</tr>
<tr>
<td><code>Waf: Leaving directory</code></td>
<td><code>./waf --run milan</code></td>
</tr>
<tr>
<td><code>Bytes/sec: 14431.8</code></td>
<td></td>
</tr>
</tbody>
</table>

Conversion: 14431.8 Bytes/sec = 115.4544 Kbits/sec.

4.4 Network Delay from the Simulated Network

In this part, a simulated ping function is applied in ns-3, several pings were sent and the round-trip delay time is recorded. However, within each group the rtt values are constant, the figure 4.21 shows the time delay of groups.

![Network Delay of Groups](image)

Figure 4.21: The network delay of groups in ns-3.

As showed, the network delay of group-1 is "0 ms", but this does not mean there is no network delay in group-1. It may be the value is too slight for the unit "ms".

4.5 Comparing and Analysis

4.5.1 Throughput

As for each group, there are four average values of throughput from four tests, and one value from simulated platform, these values are put together inside one figure for a better view of comparison.

In group-1, the tested network link is between nS222 and nR4. The throughput values gained in the real network section and the one from simulation section are showed in figure 4.22. As you can see, the result gained from ns-3 is higher than the others. It is
4.5. COMPARING AND ANALYSIS

about 21% higher than the data from real network.

Figure 4.22: The five throughput values of group-1.

Also for group-2, the results are showed in figure 4.23. Just like the previous group, the result provided by ns-3 is higher, and much more this time. Take a comparison with the average of the other four values, it is about 65% higher.

Figure 4.23: The five throughput values of group-2.

For group-3, the value from ns-3 is 60% higher than real case, as you can see in figure 4.24.

And finally, for group-4 the value from ns-3 is 62% higher than real case. The results are showed in figure 4.25.

From group-1 to group-4, the tested link become longer and more network devices are involved. In group-1, the two host machine has only one switch and one router between, the structure is quite straight, and it turns out the result is the most closest.
4.5. COMPARING AND ANALYSIS

Figure 4.24: The five throughput values of group-3.

Figure 4.25: The five throughput values of group-4.
4.5. COMPARING AND ANALYSIS

Analysis with R

In this part, a professional mathematical tool R is used to evaluate how different the real network result and simulated network result are. The output below is following the group order.

(Group-1) P-value analysis with R.

```r
> x<-c(5864, 5684, 4797, 5810)
> y<-c(6714.16, 6714.16, 6714.16, 6714.16)
> t.test(x,y)

Welch Two Sample t-test

data: x and y

t = -4.6996, df = 3, p-value = 0.01823
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  -1971.3688  -379.4512
sample estimates:
mean of x  mean of y
  5538.75  6714.16
```

Group-1 is the least difference group among all, however, there are still much difference. According to the p-value above, you can see that the possibility of the true difference in means of the two serials of data equal to 0 is about 1.8%. It’s very low.

The below parts show the corresponding outputs of group-2, group-3 and group-4. Their p-value are even lower.

(Group-2) P-value analysis with R.

```r
> x<-c(88, 65.2, 55.9, 70.9)
> y<-c(115.6688, 115.6688, 115.6688, 115.6688)
> t.test(x,y)

Welch Two Sample t-test

data: x and y

t = -6.7663, df = 3, p-value = 0.006596
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  -67.14854  -24.18906
sample estimates:
mean of x  mean of y
  70.0000  115.6688
```

(Group-3) P-value analysis with R.

```r
> x<-c(87.1, 64.9, 64.4, 64.5)
> y<-c(112.7744, 112.7744, 112.7744, 112.7744)
> t.test(x,y)

Welch Two Sample t-test

data: x and y

t = -7.5629, df = 3, p-value = 0.004794
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  -60.45396  -24.64484
sample estimates:
mean of x  mean of y
  70.2250  112.7744
```

51
4.5. COMPARING AND ANALYSIS

(150x329 to 433x488)

> x<-c(88.4,59.1,66.4,71)
> y<-c(115.4544,115.4544,115.4544,115.4544)
> t.test(x,y)

Welch Two Sample t-test

data:  x and y
t = -7.1027, df = 3, p-value = 0.005742
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
-64.04694 -24.41186
sample estimates:
mean of x  mean of y
71.2250    115.4544

4.5.2 Network Delay

Generally speaking, the network delays obtained from ns-3 are always lower. As showed in figure 4.26, the lower rate is around 30% if exclude the group-1.

![Network time delays of 4 groups.](image)

Compare to throughput, the difference between network delays is not so big. As you can see, there are some pattern in both outputs, and they are roughly matching. However, ns-3 gives a more faster estimation.

4.5.3 Comprehensive analysis

Generally speaking, ns-3 gives more optimistic results in both throughput and network delay. Especially in throughput, ns-3 gives about 0.5 times higher results in average.

Notice that the simulated network by ns-3 is a purely quiet environment, there's no realistic interference, while random noise is existed in real. The simulated platform is very loyal, all the results follows the internal simulation mechanism. Hence the difference between reality and simulation is underestimation of the real system random
4.5. COMPARING AND ANALYSIS

factors.

Also, all the configuration information has been recorded, but not all of them are used when simulating. There are more options and parameters in the real, and the simulating mechanism simplified the process. Quite amount of things have to stay under unknown default values. And this may cause significant differences between results. As a instance, there are two types of switch are in using, and they naturally have difference in performance. However, there are no corresponding options in ns-3, switch is a switch, no difference.

Moreover, the process of simulating the network is really tough. It takes almost 10 times of the time consumed on real network, compare to the results it provides, one should really think twice before start.
Chapter 5

Discussion and Future work

5.1 Discussion

In general, this thesis contains two partitions, one theory research part and one practical part. For understanding deeply about network performance analysis, several problems must be solved first.

In the theory research part, many difficulties in finding own way. As there are too much related work has been done, like a large ocean, one may lost his way easily. Fortunately, exit found, problems got solved and questions got answers. Issues like what is network performance, how to do a network performance analysis and what's different between methods of analyzing were explored deeply.

In the practical part, a comparative experiment has been done. When constructing the real network, much efforts has been made for obtaining all the system parameters. As two experiment platforms should be similar to each other as much as possible. However, there are still some points slip through. And ns-3 is really not a easy tool, lots of pre-study work must be done before start to use it. Even after preparing, there may still be some confusions. The simulating code is not perfect, but it matches the real in majority.

The general analysis way is applied in both sections of practical part. As expected, the results from different platforms are different, and the difference presents consistency. The ns-3 simulated network tends to give optimistic results in both throughput and network time delay two aspects. Therefore, their work effects are obviously unequal.

5.2 Future work

For the difference between results from real network and simulated network, there may be several causes. Firstly, uncontrollable interference factors like random noise pulls down the performance of real network, but this does not happen in ns-3 simulated platform.

Also, there are some real system parameters are not be able to be configured in the ns-3 simulation. Like the protocols of the internal communication among routers and switches. Because the internal traffic can also affect the whole network performance.
5.2. FUTURE WORK

The above problems are mentioned in this thesis, however, without a more deeper exploration or completely analysis. It is a very interesting topic to find out the true reason that causes the difference. In order to find the true reasons, one may need to construct more groups for testing and configure more options, and the most important thing: proficient in the simulation tool ns-3. This will be a very valuable research.
Chapter 6

Conclusion

Network performance analysis is quite important and helpful to system administration. Large amount of work in this field has been done over years. However, there are still some unclear concepts and plausible definitions.

Based on the services that networks provide and users’ requirements about networks, the general concept of network performance can be divided into two layers: one built on basic communication, the other one built on reliable communication. For reliable communication, user interface and software performance on application layer are involved, and this thesis only targets on basic network performance.

To analysis basic network performance, there are two metrics should be cover: network throughput and network time delay. These two metrics represents the network performance in both time view and space view.

The general procedure of analysis never change, but when doing network performance analysis, there are two ways of collecting data: data from real network and data from network simulation. In order to compare these two methods, a experiment contains two sections is constructed. With one network layout, a real network and a simulated one are implemented. The results of the practical part shows that these two ways failed in achieving the same endpoint, but indeed has some relevance between the results. The ns-3 simulated network gives more optimistic data in both metrics. The results give some constructive information to related stuff.

The difference may caused by some uncontrollable noise in the real network or the compromise on some system parameters that are not able to configure in ns-3 scripts, or both. It would be a very interesting and valuable research to find out the true reason behind the difference.
Bibliography


[17] Brian Boyko. The five different types of network delay whiteboard series, JANUARY 9, 2009. ASK THE SERVICE ASSURANCE EXPERT.


[41] Ltd. CTC Union Technologies Co. E1/t1 to fiber solution. Fiber Optical E1/T1, Datacom and Ethernet Multiplexer.

[42] 3M Worldwide United States Electronics Manufacturing. 3m1 round, jacketed, flat cable, 3759/09, 100 ft. sf. 3M Id : 80-6108-0256-5.

Chapter 7
Appendix

7.1 The ns-3 simulating code

```c
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/cmac-module.h"
#include "ns3/ point-topoint-module.h"
#include "ns3/ipv4-routing-module.h"
#include "ns3/applications-module.h"
#include "ns3/ipv4-static-routing.h"
#include "ns3/bridge-helper.h"
#include "ns3/flow-monitor-helper.h"
#include "ns3/netanim-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("milan") ;

int main (int argc , char *argv [])
{
    LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO) ;
    LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO) ;

    // 0.1
    Ptr<Node> r3 = CreateObject<Node> () ; // router R3
    Ptr<Node> r2 = CreateObject<Node> () ; // router R2
    Ptr<Node> nR3 = CreateObject<Node> () ; // node connected to R3
    Ptr<Node> nR2 = CreateObject<Node> () ; // node connected to R2
    Ptr<Node> s2 = CreateObject<Node> () ; // Switch S2
    Ptr<Node> s22 = CreateObject<Node> () ; // Switch S22
    Ptr<Node> s222 = CreateObject<Node> () ; // Switch S222

    // Nodes grouped by what network they are on, excluding switches
    NodeContainer nodes_10_0_1_0 = NodeContainer (r3) ;
    NodeContainer nodes_10_0_2_0 = NodeContainer (r2) ;
    NodeContainer nodes_10_0_4_0 = NodeContainer (r3) ;
    nodes_10_0_1_0 . Add (nR3) ;
    nodes_10_0_2_0 . Add (nR2) ;
    NodeContainer nodes_10_0_4_0 = NodeContainer (r3) ;
    nodes_10_0_4_0 . Add (nR3) ;
    nodes_10_0_4_0 . Add (nR2) ;
    NodeContainer nodes_10_0_5_0 = NodeContainer (r1) ;
    nodes_10_0_4_0 . Add (nR1) ;
    nodes_10_0_5_0 . Add (nR2) ;
    nodes_10_0_5_0 . Add (nR3) ;
    nodes_10_0_5_0 . Add (nR4) ;
    NodeContainer nodes_10_0_6_0 = NodeContainer (r3) ;
    NodeContainer nodes_10_0_6_0 = NodeContainer (r3) ;
    nodes_10_0_6_0 . Add (nR3) ;
    nodes_10_0_6_0 . Add (nR2) ;
    nodes_10_0_6_0 . Add (nR4) ;
    nodes_10_0_6_0 . Add (nR5) ;
    nodes_10_0_6_0 . Add (nR6) ;
    NodeContainer serial_nodes_10_0_6_0 = NodeContainer (r3) ;
```

// Routers grouped by their serial cable networks
NodeContainer serial_nodes_10_0_6_0 = NodeContainer (r3) ;
```
7.1. THE NS-3 SIMULATING CODE

```c
serial_nodes_10_0_6_0 .Add( r2 );
NodeContainer serial_nodes_10_0_7_0 = NodeContainer ( r2 );
serial_nodes_10_0_7_0 .Add( r1 );
NodeContainer serial_nodes_10_0_8_0 = NodeContainer ( r1 );
serial_nodes_10_0_8_0 .Add( r4 );

NodeContainer allNodes ;
allNodes .Add( r1 ) ;
allNodes .Add( r2 ) ;
allNodes .Add( r3 ) ;
allNodes .Add( r4 ) ;
allNodes .Add( nR2 ) ;
allNodes .Add( nR3 ) ;
allNodes .Add( nR4 ) ;
allNodes .Add( nS2 ) ;
allNodes .Add( nS22 ) ;
allNodes .Add( nS222 ) ;

////////////////////////////////////
// Topology Construction //

PointToPointHelper p2p_1 ;
p2p_1 . SetDeviceAttribute ( " DataRate " , StringValue ( " 100Mbps" ) ) ;
p2p_1 . SetChannelAttribute ( " Delay " , StringValue ( " 50 ns " ) ) ;
NetDeviceContainer devices_1 ;
devices_1 = p2p_1 . Install ( nodes_10_0_1_0 ) ;

PointToPointHelper p2p_2 ;
p2p_2 . SetDeviceAttribute ( " DataRate " , StringValue ( " 100Mbps" ) ) ;
p2p_2 . SetChannelAttribute ( " Delay " , StringValue ( " 50 ns " ) ) ;
NetDeviceContainer devices_2 ;
devices_2 = p2p_2 . Install ( nodes_10_0_2_0 ) ;

PointToPointHelper p2p_3 ;
p2p_3 . SetDeviceAttribute ( " DataRate " , StringValue ( " 100Mbps" ) ) ;
p2p_3 . SetChannelAttribute ( " Delay " , StringValue ( " 50 ns " ) ) ;
NetDeviceContainer devices_3 ;
devices_3 = p2p_3 . Install ( nodes_10_0_3_0 ) ;

CsmaHelper csma_2m ;
// cables used for S2 is 2 m
csma_2m . SetChannelAttribute ( " DataRate " , StringValue ( " 100Mbps" ) ) ;
csma_2m . SetChannelAttribute ( " Delay " , StringValue ( " 10ns" ) ) ;

CsmaHelper csma_5m ;
// cables used from r1 to s22 and r4 to s222 is 5 m
csma_5m . SetChannelAttribute ( " DataRate " , StringValue ( " 100Mbps" ) ) ;
csma_5m . SetChannelAttribute ( " Delay " , StringValue ( " 25ns" ) ) ;

CsmaHelper csma_10m ;
// 10m cable, used between node<->switch and switch<->switch
csma_10m . SetChannelAttribute ( " DataRate " , StringValue ( " 10Mbps" ) ) ;
csma_10m . SetChannelAttribute ( " Delay " , StringValue ( " 50ns" ) ) ;

// BEGIN SWITCH S2 ( network 10.0.4.0 ) //
// connect R3 to S2
NetDeviceContainer devices_4_1 ;
devices_4_1 = csma_2m . Install ( NodeContainer ( r3 , s2 ) ) ;

// connect R1 to S2
NetDeviceContainer devices_4_2 ;
devices_4_2 = csma_2m . Install ( NodeContainer ( r1 , s2 ) ) ;

// connect nS2 to S2
NetDeviceContainer devices_4_3 ;
devices_4_3 = csma_10m . Install ( NodeContainer ( nS2 , s2 ) ) ;

// BEGIN SWITCH S2 ( network 10.0.5.0 ) //
// connect R1 to S2
NetDeviceContainer devices_5_1 ;
devices_5_1 = csma_5m . Install ( NodeContainer ( r1 , s2 ) ) ;
```
7.1. THE NS-3 SIMULATING CODE

```
150 // connect S22 to S22
151 NetDeviceContainer devices_5_2;
152 devices_5_2 = csma_10m.Install(NodeContainer(s22, s22));
153
154 // connect nS22 to S22
155 NetDeviceContainer devices_5_3;
156 devices_5_3 = csma_5m.Install(NodeContainer(nS22, s22));
157
158 OpenFlowSwitchHelper switchS22Helper;
159 Ptr<ns3::ofi::LearningController> controllerS22 = CreateObject<ns3::ofi::LearningController>();
160 NetDeviceContainer switch_S22_devices;
161 // the ports of S22
162 switch_S22_devices.Add(devices_5_1.Get(1));
163 switch_S22_devices.Add(devices_5_2.Get(1));
164 switch_S22_devices.Add(devices_5_3.Get(1));
165
166 NetDeviceContainer devices_5;
167 // devices of all the nodes connected to S22
168 devices_5.Add(devices_5_1.Get(0));
169 devices_5.Add(devices_5_3.Get(0));
170 switchS22Helper.Install(s22, switch_S22_devices, controllerS22);
171 // END SWITCH S22 //
172
173 // BEGIN SWITCH S222 (network 10.0.5.0, same as S222!)
174 // connect R4 to S222
175 NetDeviceContainer devices_5_4;
176 devices_5_4 = csma_5m.Install(NodeContainer(r4, s222));
177
178 // S22 is already connected to S222 ...
179
180 // connect nS222 to S222
181 NetDeviceContainer devices_5_5;
182 devices_5_5 = csma_10m.Install(NodeContainer(nS222, s222));
183
184 OpenFlowSwitchHelper switchS222Helper;
185 Ptr<ns3::ofi::LearningController> controllerS222 = CreateObject<ns3::ofi::LearningController>();
186 NetDeviceContainer switch_S222_devices;
187 // the ports of S222
188 switch_S222_devices.Add(devices_5_2.Get(1));
189 switch_S222_devices.Add(devices_5_3.Get(1));
190 switch_S222_devices.Add(devices_5_5.Get(1));
191 // devices of all the nodes connected to S222
192 devices_5.Add(devices_5_4.Get(0));
193 devices_5.Add(devices_5_5.Get(0));
194 switchS222Helper.Install(s222, switch_S222_devices, controllerS222);
195 // END SWITCH S222 //
196
197 // serial cables . . 5m
198 PointToPointHelper p2p_6;
199 p2p_6.SetDeviceAttribute("DataRate", StringValue("128Kbps"));
200 p2p_6.SetChannelAttribute("Delay", StringValue("25.9ns"));
201 NetDeviceContainer devices_6;
202 devices_6 = p2p_6.Install(serial_nodes_10_0_6_0);
203
204 PointToPointHelper p2p_7;
205 p2p_7.SetDeviceAttribute("DataRate", StringValue("128Kbps"));
206 p2p_7.SetChannelAttribute("Delay", StringValue("25.9ns"));
207 NetDeviceContainer devices_7;
208 devices_7 = p2p_7.Install(serial_nodes_10_0_7_0);
209
210 PointToPointHelper p2p_8;
211 p2p_8.SetDeviceAttribute("DataRate", StringValue("128Kbps"));
212 p2p_8.SetChannelAttribute("Delay", StringValue("25.9ns"));
213 NetDeviceContainer devices_8;
214 devices_8 = p2p_8.Install(serial_nodes_10_0_8_0);
215
216 //////////////////////////////////////////////////////////////////////////////////
217 // Internet stack and ipv4 addresses //
218 InternetStackHelper stack;
219 stack.Install(allNodes);
220 // lan //
221 Ipv4AddressHelper net_10_0_1_0;
222 net_10_0_1_0.SetBase("10.0.1.0", "255.255.255.0");
223 Ipv4InterfaceContainer i_10_0_1_0 = net_10_0_1_0.Assign(devices_1);
224 Ipv4AddressHelper net_10_0_2_0;
225 net_10_0_2_0.SetBase("10.0.2.0", "255.255.255.0");
226 Ipv4InterfaceContainer i_10_0_2_0 = net_10_0_2_0.Assign(devices_2);
227```
7.1. THE NS-3 SIMULATING CODE

```
// switch nodes

Ipv4AddressHelper net_10_0_3_0;
net_10_0_3_0.SetBase("10.0.3.0", "255.255.255.0");
Ipv4InterfaceContainer i_10_0_3_0 = net_10_0_3_0.Assign(devices_3);

Ipv4AddressHelper net_10_0_4_0;
net_10_0_4_0.SetBase("10.0.4.0", "255.255.255.0");
Ipv4InterfaceContainer i_10_0_4_0 = net_10_0_4_0.Assign(devices_4);

Ipv4AddressHelper net_10_0_5_0;
net_10_0_5_0.SetBase("10.0.5.0", "255.255.255.0");
Ipv4InterfaceContainer i_10_0_5_0 = net_10_0_5_0.Assign(devices_5);

// serial

Ipv4AddressHelper net_10_0_6_0;
net_10_0_6_0.SetBase("10.0.6.0", "255.255.255.240");
Ipv4InterfaceContainer i_10_0_6_0 = net_10_0_6_0.Assign(devices_6);

Ipv4AddressHelper net_10_0_7_0;
net_10_0_7_0.SetBase("10.0.7.0", "255.255.255.240");
Ipv4InterfaceContainer i_10_0_7_0 = net_10_0_7_0.Assign(devices_7);

Ipv4AddressHelper net_10_0_8_0;
net_10_0_8_0.SetBase("10.0.8.0", "255.255.255.240");
Ipv4InterfaceContainer i_10_0_8_0 = net_10_0_8_0.Assign(devices_8);

// Routing Tables

Ipv4StaticRoutingHelper routingTable;

// Set default gateways for all the nodes
routingTable.GetStaticRouting(nR3->GetObject<Ipv4>())->SetDefaultRoute(i_10_0_1_0.GetAddress(0), 1); // R3 -> R3

routingTable.GetStaticRouting(nR2->GetObject<Ipv4>())->SetDefaultRoute(i_10_0_2_0.GetAddress(0), 1); // R2 -> R1

routingTable.GetStaticRouting(nR4->GetObject<Ipv4>())->SetDefaultRoute(i_10_0_3_0.GetAddress(1), 1); // R4 -> R1

routingTable.GetStaticRouting(nS2->GetObject<Ipv4>())->SetDefaultRoute(i_10_0_5_0.GetAddress(0), 1); // R1 -> R2

routingTable.GetStaticRouting(nS22->GetObject<Ipv4>())->SetDefaultRoute(i_10_0_7_0.GetAddress(2), 1); // R2 -> R3

// Routing for R3, 10.0.1.0 and 10.0.4.0 are directly connected

routingTable.GetStaticRouting(r3->GetObject<Ipv4>())->AddNetworkRouteTo(Ipv4Address("10.0.2.0"), Ipv4Mask("255.255.255.0"), 0); // R3 -> R2
routingTable.GetStaticRouting(r3->GetObject<Ipv4>())->AddNetworkRouteTo(Ipv4Address("10.0.3.0"), Ipv4Mask("255.255.255.0"), 1); // R3 -> 10.0.2.0

// Routing for R2, 10.0.2.0 is directly connected

routingTable.GetStaticRouting(r2->GetObject<Ipv4>())->SetDefaultRoute(i_10_0_6_0.GetAddress(0), 2); // R2 -> R1

routingTable.GetStaticRouting(r2->GetObject<Ipv4>())->AddNetworkRouteTo(Ipv4Address("10.0.1.0"), Ipv4Mask("255.255.255.0"), 2); // R2 -> 10.0.1.0

routingTable.GetStaticRouting(r2->GetObject<Ipv4>())->AddNetworkRouteTo(Ipv4Address("10.0.3.0"), Ipv4Mask("255.255.255.0"), 3); // R2 -> 10.0.3.0

// Routing for R1, 10.0.0.0 and 10.0.5.0 are directly connected

routingTable.GetStaticRouting(r1->GetObject<Ipv4>())->SetDefaultRoute(i_10_0_7_0.GetAddress(0), 1); // R1 -> R2
routingTable.GetStaticRouting(r1->GetObject<Ipv4>())->AddNetworkRouteTo(Ipv4Address("10.0.1.0"), Ipv4Mask("255.255.255.0"), 1); // R1 -> R4

// Routing for R4, 10.0.3.0 and 10.0.5.0 are directly connected

routingTable.GetStaticRouting(r4->GetObject<Ipv4>())->SetDefaultRoute(i_10_0_5_0.GetAddress(0), 1); // R4 -> R1
```
7.2. IMPORTANT ROUTER CONFIGURATIONS

7.2.1 Router-4

```cpp
// IMPORTANT CONFIGURATION OF ROUTER-4 //

FastEthernet0/0 is up, line protocol is up
Hardware is GbE6, address is 0023.33ec.c19e (bia 0023.33ec.c19e)
MTU 1500 bytes, BW 100000 Kbit, DLY 100 usec,
reliability 255/255, txload 1/255, rxload 1/255
Encapsulation ARPA, loopback not set
Keepalive set (10 sec)
Full-duplex, 100Mb/s, 100BaseTX/FX
```
ARP type: ARPA, ARP Timeout 04:00:00
Last input 00:00:00, output 00:00:00, output hang never
Last clearing of "show interface" counters never
Input queue: 0/75/130 (size/max/drops/flushes); Total output drops: 0
Queueing strategy: fifo
Output queue: 0/40 (size/max)
5 minute input rate 0 bits/sec, 0 packets/sec
5 minute output rate 0 bits/sec, 0 packets/sec
148810 packets input, 12066378 bytes
Received 65377 broadcasts, 0 runts, 0 giants, 0 throttles
0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
0 watchdog
0 input packets with dribble condition detected
201934 packets output, 19020717 bytes, 0 underruns
0 output errors, 0 collisions, 3 interface resets
0 babbles, 0 late collision, 0 deferred
0 lost carrier, 0 no carrier
0 output buffer failures, 0 output buffers swapped out
FastEthernet0/1 is up, line protocol is up
Hardware is Gt96k FE, address is 0023.33ec.c19f (bia 0023.33ec.c19f)
Internet address is 10.0.3.4/24
MTU 1500 bytes, BW 10000 Kbit, DL Y 100 usec,
reliability 255/255, txload 1/255, rxload 1/255
Encapsulation ARPA, loopback not set
Keepalive set (10 sec)
Full-duplex, 100Mbit/s, 100BaseTX/FX
ARP type: ARPA, ARP Timeout 04:00:00
Last input 00:07:26, output 00:00:04, output hang never
Last clearing of "show interface" counters never
Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
Queueing strategy: fifo
Output queue: 0/40 (size/max)
5 minute input rate 0 bits/sec, 0 packets/sec
5 minute output rate 0 bits/sec, 0 packets/sec
39703 packets input, 2628513 bytes
Received 17 broadcasts, 0 runts, 0 giants, 0 throttles
0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
0 watchdog
0 input packets with dribble condition detected
237814 packets output, 129620694 bytes, 0 underruns
0 output errors, 0 collisions, 8 interface resets
0 babbles, 0 late collision, 0 deferred
0 lost carrier, 0 no carrier
0 output buffer failures, 0 output buffers swapped out
Serial0/2/0 is up, line protocol is up
....
Internet address is 10.0.8.4/28
MTU 1500 bytes, BW 1024 Kbit, DL Y 20000 usec,
reliability 255/255, txload 1/255, rxload 1/255
Encapsulation HDLC, loopback not set
Keepalive set (10 sec)
Last input 00:00:07, output 00:00:00, output hang never
Last clearing of "show interface" counters never
Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 487
Queueing strategy: weighted fair
Output queue: 0/1000/64/487 (size/max total/threshold/drops)
Conversations 0/3/32 (active/max active/max total)
Reserved Conversations 0/0 (allocated/max allocated)
Available Bandwidth 768 kilobits/sec
5 minute input rate 0 bits/sec, 0 packets/sec
5 minute output rate 0 bits/sec, 0 packets/sec
163896 packets input, 13773720 bytes, 0 no buffer
Received 160367 broadcasts, 0 runts, 0 giants, 0 throttles
0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
169683 packets output, 12929076 bytes, 0 underruns
# IMPORTANT ROUTER CONFIGURATIONS

## 7.2. Router-2

### Important Configuration of Router-2

<table>
<thead>
<tr>
<th>Serial0/1/0 is up, line protocol is up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware is GT96K Serial</td>
</tr>
<tr>
<td>Internet address is 10.0.6.2/28</td>
</tr>
<tr>
<td>MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, reliability 255/255, txload 1/255, rxload 1/255</td>
</tr>
<tr>
<td>Encapsulation HDLC, loopback not set</td>
</tr>
<tr>
<td>Keepalive set (10 sec)</td>
</tr>
<tr>
<td>CRC checking enabled</td>
</tr>
<tr>
<td>Last input 00:00:09, output 00:00:03, output hang never</td>
</tr>
<tr>
<td>Last clearing of &quot;show interface&quot; counters never</td>
</tr>
<tr>
<td>Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 7</td>
</tr>
<tr>
<td>Queueing strategy: weighted fair</td>
</tr>
<tr>
<td>Output queue: 0/1000/64/7 (size/max total/threshold/drops)</td>
</tr>
<tr>
<td>Conversations 0/2/256 (active/max active/max total)</td>
</tr>
<tr>
<td>Reserved Conversations 0/0 (allocated/max allocated)</td>
</tr>
<tr>
<td>Available Bandwidth 1158 kilobits/sec</td>
</tr>
<tr>
<td>5 minute input rate 0 bits/sec, 0 packets/sec</td>
</tr>
<tr>
<td>5 minute output rate 0 bits/sec, 0 packets/sec</td>
</tr>
<tr>
<td>152744 packets input, 10385280 bytes, 0 no buffer</td>
</tr>
<tr>
<td>Received 152189 broadcasts, 0 runts, 0 giants, 0 throttles</td>
</tr>
<tr>
<td>0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort</td>
</tr>
<tr>
<td>153814 packets output, 10705095 bytes, 0 underruns</td>
</tr>
<tr>
<td>0 output errors, 0 collisions, 5 interface resets</td>
</tr>
<tr>
<td>0 output buffer failures, 0 output buffers swapped out</td>
</tr>
<tr>
<td>0 carrier transitions</td>
</tr>
<tr>
<td>DCD=up DSR=up DTR=up RTS=up CTS=up</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Serial0/1/1 is up, line protocol is up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware is GT96K Serial</td>
</tr>
<tr>
<td>Internet address is 10.0.7.2/28</td>
</tr>
<tr>
<td>MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, reliability 255/255, txload 1/255, rxload 1/255</td>
</tr>
<tr>
<td>Encapsulation HDLC, loopback not set</td>
</tr>
<tr>
<td>Keepalive set (10 sec)</td>
</tr>
<tr>
<td>CRC checking enabled</td>
</tr>
<tr>
<td>Last input 00:00:09, output 00:00:02, output hang never</td>
</tr>
<tr>
<td>Last clearing of &quot;show interface&quot; counters never</td>
</tr>
<tr>
<td>Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 68</td>
</tr>
<tr>
<td>Queueing strategy: weighted fair</td>
</tr>
<tr>
<td>Output queue: 0/1000/64/8 (size/max total/threshold/drops)</td>
</tr>
<tr>
<td>Conversations 0/3/256 (active/max active/max total)</td>
</tr>
<tr>
<td>Reserved Conversations 0/0 (allocated/max allocated)</td>
</tr>
<tr>
<td>Available Bandwidth 1158 kilobits/sec</td>
</tr>
<tr>
<td>5 minute input rate 0 bits/sec, 0 packets/sec</td>
</tr>
<tr>
<td>5 minute output rate 0 bits/sec, 0 packets/sec</td>
</tr>
<tr>
<td>175420 packets input, 13202010 bytes, 0 no buffer</td>
</tr>
<tr>
<td>Received 152193 broadcasts, 0 runts, 0 giants, 0 throttles</td>
</tr>
<tr>
<td>0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort</td>
</tr>
<tr>
<td>174473 packets output, 14713519 bytes, 0 underruns</td>
</tr>
<tr>
<td>0 output errors, 0 collisions, 4 interface resets</td>
</tr>
<tr>
<td>0 output buffer failures, 0 output buffers swapped out</td>
</tr>
<tr>
<td>0 carrier transitions</td>
</tr>
<tr>
<td>DCD=up DSR=up DTR=up RTS=up CTS=up</td>
</tr>
</tbody>
</table>

FastEthernet0/0 is up, line protocol is down
7.2. IMPORTANT ROUTER CONFIGURATIONS

Hardware is Gt96k FE, address is 0026.cb73.a868 (bia 0026.cb73.a868)
Internet address is 10.0.2.2/24
MTU 1500 bytes, BW 100000 Kbit, DLY 100 usec,
reliability 255/255, txload 1/255, rxload 1/255
Encapsulation ARPA, loopback not set
Keepalive set (10 sec)
Auto-duplex, Auto Speed, 100BaseTX/FX
ARP type: ARPA, ARP Timeout 04:00:00
Last input 2w1d, output 2w1d, output hang never
Last clearing of "show interface" counters never
Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
Queueing strategy: fifo
Output queue: 0/40 (size/max)
5 minute input rate 0 bits/sec, 0 packets/sec
5 minute output rate 0 bits/sec, 0 packets/sec
3495 packets input, 3151205 bytes
Received 192 broadcasts, 0 runts, 0 giants, 0 throttles
0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
0 watchdog
0 input packets with dribble condition detected
135875 packets output, 9042018 bytes, 0 underruns
0 output errors, 0 collisions, 5 interface resets
0 babbles, 0 late collision, 0 deferred
0 lost carrier, 0 no carrier
0 output buffer failures, 0 output buffers swapped out