Chapter 3 Computer Network Simulation

One of the objectives of this thesis is to create a simulated environment that allows the testing as well as experimentation of the above technologies. This chapter will first begin with the introduction of the computer simulation model. A study on different types of simulation models is performed.

The following sections will describe the various available simulators. A survey of existing network simulators is performed to show current approaches to network simulation. A comparison is done among the simulators on their strengths as well as their limitations.

The third section will explain the differences between procedural approach and object-oriented approach. A discussion of the strengths and drawbacks of the various programming approaches is done.

The final section will conclude a review on the network simulation. The choice of the simulation model to be used in this project is finalized. Besides that, the network simulator is chosen as well as the type of programming language approach that is needed for this project is finalized.

3.1 Computer Simulation

Computer simulation is the discipline of designing a model of an actual or theoretical physical system. After designing, the model is executed on a digital computer. Finally, the execution output is analyzed to get the final conclusion or result [31]. Normally, a simulation is the execution of a model, represented by a computer program that gives information about the system being investigated. The simulation approach of analyzing a model is opposed to the analytical approach, where the method of analyzing the system is purely theoretical.

Simulations generate dynamic environments with which users can interact “as if they were really there”. A simulation pictures reality and the complexity of a real network environment.
There are currently three primary sub-fields, which are model design, model execution as well as model analysis that must be implemented during simulation [32].

### 3.1.1 Simulation Model

A model is an abstraction of a system intended to replicate some properties of that system [33]. Thus, the model design of the simulation plays an important role in the entire simulation. The model design must adhere strictly to the problem and must place accuracy as its primary priority. The simulation model will become useless if it is workable but not entirely correct in term of its semantics as applied to a problem domain. This will cause misinterpretation that lead to the incorrect decision.

The execution model handles the semantic of the simulation base on the problem domain for the simulation. The execution can run as serial and parallel. Each of the execution models must be relevant to the problem domain. In order to ensure that the execution is correct, the execution analysis has to be performed.

### 3.1.2 Simulation Approach

In the context of a network simulator, there are two approaches to modeling. These two approaches are analytical modeling and discrete event modeling. Analytical modeling simulation is a closed form approach to model simulation in which the network is modeled mathematically in the form of function and equations. However, this approach is not suitable to simulate the dynamic nature of a computer network [34].

Discrete event modeling is an approach where the computer model replicates the real-world objects. Each object changes its state only at discrete points during the simulation time. The primary concern of this model is the accuracy, thus, it requires greater computation time and this approach is more desirable [34].
3.1.3 Advantages and Disadvantages of Simulation

A network simulator provides a means for researchers and network planners to analyze the behavior of the network without the expense of building a real network. Normally, to build an actual system for every experiment is usually too expensive, impractical and sometimes simply impossible. The simulator usually contains tools that give the user an interactive modeling environment with a graphical user interface.

A network planner can use the simulator as a planning tool. Basically, the network analyzer will setup the various network configuration and traffic loads to obtain statistics result such as utilization of network links as well as throughput rates of virtual circuits. The statistics result are normally saved or logged in a data file for further processing and analysis. Besides that, a researcher could use the simulator to design various type of new topology. By using the simulation, the researcher is able to verify the correctness of designs.

Another advantage of using simulation is that it can be used even in cases where the problem is an easily solvable linear system. This is because uniform execution techniques can be employed for a large variety of systems without the need to choose special purpose and sometimes-arcane solution methods to avoid simulation.

Despite the advantages, the simulation also has its disadvantages. Sometimes, the simulations implementation and result could be differing considerably from a real environment.

3.2 Introduction to Various Simulator

A network simulator can be either a general-purpose simulator that enables a wide range of possible simulations, or a special-purpose simulator targeting a particular area of research. There are currently a wide variety of network simulation tools available on the market. The following sub-sections describe various simulators that have been implemented, including INSANE, OMNET++, OPNET, PARSEC, REAL Network Simulator, NIST ATM/HFC, Ns-2 and UMJaNetSim. This section will describe their features, advantages and weakness.
3.2.1 INSANE

The Internet Simulated ATM Networking Environment (INSANE) is a network simulator designed to test various IP-over-ATM algorithms with realistic traffic loads derived from empirical traffic measurements. INSANE's ATM protocol stack provides real-time guarantees to ATM virtual circuits by using Rate Controlled Static Priority (RCSP) queuing. ATM signaling is performed using a protocol similar to the Real-Time Channel Administration Protocol (RCAP) [35].

In particular, the simulated TCP implementation performs connection management, slow start, flow and congestion control, retransmission, and fast retransmits. Various application simulators mimic the behavior of standard Internet applications to provide a realistic workload, including: telnet, ftp, WWW, real-time audio, and real-time video. The simulation core and primitive objects are implemented in C++ while the simulation scenarios are created using the Tcl scripting language. INSANE is designed to run large simulations whose results are processed off-line [35].

Advantages

- It works quite well on distributed computing clusters (although simulations are all sequential processes, a large number of them can easily be run in parallel).

Weaknesses

- The GUI of INSANE does not provide other features concerning the creation of the simulation environment.
- Not a user-friendly environment.
- Output performance can only be viewed in text based.
- It only runs on Unix-based platforms.
3.2.2 OMNET++

Objective Modular Network Test bed in C++ (OMNET++) is a discrete event simulation tool. It is primarily designed to simulate computer networks, multi-processors and other distributed systems, but it may be useful for modeling other systems tool. OMNET++ has been developed on Linux, but it works just as well on most Unix systems and Windows platforms (NT recommended) [36].

OMNET++ has an execution environment that supports interactive simulation including the visualization of collected data. Besides that, it has a gnu plot-based GUI tool that is used for analyzing and plotting simulation results.

Advantages

- OMNET++ has a solid and flexible simulation kernel and it provides a powerful GUI environment for simulation execution.
- Users can build hierarchical and reusable models easily. The interface is human readable and its source code is provided.

Weaknesses

- User must posses knowledge in C or C++ programming languages to use OMNET++.
- In order to simulate the network, the user needs to enter in command line.

3.2.3 OPNET

Optimised Network Engineering Tool (OPNET) is a discrete event simulation package. Modeler is originally developed at MIT and introduced in 1987 as the first commercial network simulator. It provides support for signaling; call setup and teardown, segmentation and reassembly of cells, cell transfer, traffic management and buffer management [37].
OPNET used object oriented modeling approach and graphical editors showing the structure of actual network and network components. The behaviors of each component are specified by a state transition diagram.

Advantages

- OPNET is able to model complex network topologies with unlimited sub-network testing within its hierarchical network models.
- OPNET support modeling of mobile and satellite networks.
- OPNET model can be compiled into executed code, which means that the simulation can be debugged or executed to produce output data.
- OPNET can be used to simulate the dynamic nature of networks, protocols and their interaction, to develop new or optimize existing protocols, to analyze the performance of network systems and to explore new technologies and their impacts on networks.
- OPNET include a numbers of comprehensive library of detailed networking protocol and application models.

Weaknesses

- The OPNET Modeler is not fully platform independent since it supports only the Solaris, Windows NT/2000, and the HP-UX operating system.
- The use of the OPNET Modeler for research is costly from a financial point of view.

3.2.4 PARSEC

Parallel Simulation Environment for Complex Systems (PARSEC) is a C based discrete event simulation tool. PARSEC represents objects (physical processes) or sets of objects in the physical system as logical processes [38].

Advantages

- PARSEC is able to execute the simulation model either sequentially or in parallel by using several different asynchronous parallel simulation techniques.
• It provides for powerful message receiving constructs that result in shorter and more
natural simulation programs.
• It also includes debugging facilities and a front-end for visual specification of the
simulation model as well as the runtime output.

Weaknesses

• The entire simulation process involves the use of the language itself without a GUI.
• It is less portability among different platforms.

3.2.5 REAL Network Simulator

The REAL network simulator is a network simulator designed for testing congestion and flow
control mechanisms. REAL is a simulator for studying the dynamic behavior of flow and
congestion control schemes in packet switch data networks. It provides users with a way of
specifying such networks and to observe their behavior [39].

The simulator takes as input a scenario, which is a description of network topology, protocols,
workload and control parameters. It produces as output statistics such as the number of
packets sent by each source of data, the queuing delay at each queuing point, the number of
dropped and retransmitted packets and other similar information.

Advantages

• User can modify the simulator software to accommodate network components
• REAL provides a flexible test-bed to study the dynamic behavior of flow control and
  congestion control schemes in packet switch data networks.

Weaknesses

• No graphical user interface (GUI) representation capabilities
• User must have strong knowledge in C programming language.
• Not a cross-platform simulator.
3.2.6 NIST ATM/HFC

This NIST ATM/HFC Network Simulator was developed at the National Institute of Standards and Technology (NIST) to provide a flexible test bed for studying and evaluating the performance of ATM and HFC network without the expense of building a real network [40].

This simulator is written in C Language whereby it is written in structural programming approach. Typically, the simulator is a tool that give user an interactive modeling environment with a graphical user interface which provides the user with a means to display the topology of the network, define the parameters and connectivity of the network, log data from simulation run, and to save and load the network configuration.

Advantages

- The simulator has a well-defined message passing mechanism based on the sending of events among simulation components, handled by an event manager.
- Allows user to create different network topologies.
- Allows user to adjust the parameters of each component's operation, measure network activity, save or load different simulation configuration and log data during simulation execution.
- Provides graphical user interface.
- Provide various instantaneous performance measurements displayed in graphical or text form on the screen while the simulation is running.

Weaknesses

- Users might face problems setting up the network topology because of the requirement to consider a large number of parameters.
- It lacks portability between different platforms because the simulator relies on the X Window System for its GUI. The simulator can only run on UNIX or LINUX platforms.
- User or programmers needs to have strong foundation in C programming language to customize and understand the simulator's components.
3.2.7 NS-2

Ns-2 is a discrete event simulator targeted at networking research. Ns-2 provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. Ns is derived from the REAL network simulator and now supported by DARPA through the VINT project. The simulator is written in C++ (for its core), and simulation scenarios are designed using the Tcl scripting language (or OTcl for Ns-2) [41].

Advantages

- Ns includes a network emulation interface that permits network traffic to pass between real-world network nodes and the simulator.
- Ns allows simulation with multiple levels of abstraction, where higher abstraction levels trade off accuracy for performance
- The simulator measurements do not impact the network by adding extra traffic

Weaknesses

- Although, Ns has a network animation tool that provides network visualization features, but it does not have a GUI for general simulation manipulation and scenario setup.

3.2.8 UMJaNetSim

This UMJaNetSim simulator is a flexible test bed for studying and evaluating the performance of MPLS network without the expense of building a real network. This simulator is written in JAVA Language whereby it is developed in object-Oriented programming approach. Typically, the simulator is a tool that gives the user an interactive modeling environment with a graphical user interface which provides the user with a means to display the topology of the network, define the parameters and connectivity of the network, log data from simulation run, and save and load the network configuration [42].
Advantages

- Better Graphical user Interface that provide a very user-friendly environment.
- Output performance can be viewed in text based and graphical representation on the screen while the simulation is running.
- Allows user to create different network topologies.
- Allows user to adjust the parameters of each component's operation, measure network activity, save/load different simulation configuration and log data during simulation execution.
- UMJaNetSim has high portability among various platforms. Furthermore, the simulator is readily web-enabled by using an applet version of the simulator.
- User can add in new components without affecting the whole simulator because it is written in object-Oriented programming approach.
- The UMJaNetSim API simplifies component development and shifts the development effort to the actual research focus rather than general simulation management.

Weaknesses

- It is not web-enabled for the application version of UMJaNetSim.
- The simulator requires a lot of memory processing during the simulation.

3.2.9 Summary of Existing Simulators

To summarize, the simulators that have been discussed all have their advantages and weaknesses in terms of simulation techniques, the programming approach, availability of a graphical user interface, platform dependence or independence and focus of network research areas.

All simulators that have been studied are discrete event simulators. However, not even one simulator is web-enabled. Table 3.1 gives a comparison among the studied simulators in terms of object-oriented, graphical user interface (GUI), multithread and platform independence.
Table 3.1 Comparisons of Simulators

<table>
<thead>
<tr>
<th>Simulator</th>
<th>Object Oriented</th>
<th>GUI</th>
<th>Multithreaded</th>
<th>Platform Independence</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSANE</td>
<td>Yes</td>
<td>Poor</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>OMNET++</td>
<td>Yes</td>
<td>Good</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>OPNET</td>
<td>Yes</td>
<td>Normal</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PARSEC</td>
<td>No</td>
<td>Poor</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>REAL NS</td>
<td>No</td>
<td>Poor</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>NIST ATM/HFC</td>
<td>No</td>
<td>Normal</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>NS-2</td>
<td>Yes</td>
<td>Normal</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>UMJaNetSim</td>
<td>Yes</td>
<td>Good</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

3.3 Programming

In order to develop a network simulator, the appropriate programming approach, the programming language as well as the programming tools that are needed to develop a network simulator, need to be selected carefully. This section will discuss in term of each of the programming approach, as well as the reasoning behind the programming language choice to create the network simulator components.

3.3.1 Approach

There are several programming approaches to develop a network simulator. These include procedural approach, structured approach and object-oriented approach that are widely used in developing a network simulator.

3.3.1.1 Procedural Approach

The procedural approach makes use of procedural language, in which the program codes are placed into blocks that are referred to as procedures or functions. A function or procedure is a
relatively simple program that is called by other programs and returns a value to the program that called it. With the use of procedural approach, the task is broken down into separate blocks, in which separate blocks would perform separate tasks. Computer languages like Pascal, C and FORTRAN are procedural programming languages.

Basically, in a procedural-based programming language, a programmer writes out instructions that are followed by a computer from start to finish. Besides that, the programmer needs to plan carefully in the beginning. This is due to the big changes that need to be done if there is a little change in the components. It is not a suitable approach to develop a network simulator using the procedural approach. This is due to overlapping functions between components.

3.3.1.2 Structured Programming Approach

In the structure programming approach, the program was to be completely designed before even a single line of code was to be written. Therefore, the development of programs would involve huge amounts of schematics, flow charts and other such tools to document every interaction of each function and how each piece of data flows through the program.

The principle idea behind structured programming is as simple as the idea of divide and conquer. A computer program can be thought of as consisting a set of tasks. Any task that is too complex to be described simply would be broken down into a set of smaller component tasks, until the tasks were sufficiently small and self-contained enough to be understood.

Structured programming remains an enormously successful approach for dealing with complex problems. By the late 1980s, however, some of the deficiencies of structured programming has become all too clear.

3.3.1.3 Object-Oriented Programming

Object Oriented is the dominant programming paradigm these days, having replaced the structured procedure based programming techniques that were developed in the early 70s. The primary concepts of the object-oriented paradigm are encapsulation, inheritance and polymorphism [43].
Encapsulation is nothing more than combining data and behavior in one package and hiding the implementation of the data from the user of the object. Every object is associated with a set of properties and a set of methods. The data or properties are usually called its instance variables or fields. Meanwhile, the methods normally refer to the operation of the object.

Inheritance is the concept that when a class of objects is defined any subclass that is defined can inherit the definitions of one or more general classes. This means for the programmer that an object in a subclass need not carry its own definition of data and methods that are generic to the class of which it is a part. This not only speeds up program development, it also ensures an inherent validity to the defined subclass object. This is very useful in building a network simulator because the small component can inherit the functionality of their parent classes.

Polymorphism is the characteristic of being able to assign a different meaning to a particular symbol or "operator" in different contexts. This allows objects to behave differently within different situation that adding flexibility to the program design. This is very useful in the creation of a network simulation framework because the core simulation engine only interacts with various network components through well-defined interfaces, while each component behaves according to their own functionality.

Traditional structured programming consists of designing a set of functions to solve problem. The next step is to find appropriate ways to store the data. This means the algorithm will come first and the data structure, second. This minimized the way programmers worked at that time. Object-oriented programming reversed the order and puts data structures first, then looks at the algorithm that operate on the data.

The key to being most productive in Object-oriented programming is to make each object responsible for carrying out a set of related tasks. If an object relies on a task that isn’t its responsibility, it needs to have access to an object whose responsibilities include that task. The first object then asks the second object to carry out the task, just like a network component ask another network components to perform some task. This is done with a more generalized version of the function call that use in procedural approach. Besides that, an
object should never directly manipulate the internal data of another object. All communications should be via message, that is, method calls.

Object-oriented programming has the many advantages over the conventional approaches. Following are the key benefits of Object-oriented programming:

- Simplicity – OOP are simple and intuitive.
- Maintainability and Reusability – It is easy to maintain and modify existing code, as new objects can be created with small differences to existing ones.
- Modifiable – OOP provides a good framework for code libraries, where supplied software components can be easily adapted and modified by the programmer.
- Modularity - This makes it good for defining abstract datatypes, where implementation details are hidden and the unit has a clearly defined interface.
- Extensibility – OOP lets programmer extends the functionality of each simulator by adding more classed without affecting the core of the system.

3.3.2 Java

Java was developed at Sun Microsystems. Work on Java originally began with the goal of creating a platform-independent language and operating system for consumer electronics. Java is both a programming language and an environment for executing programs written in the Java language. Unlike traditional compilers, which convert source code into machine-level instructions, the Java compiler translates Java source code into instructions that are interpreted by the runtime Java Virtual Machine [43].

Java has several important features that make it attractive programming language compared to other languages. The following are some of the advantages of JAVA programming language:

- Simplicity - Java is designed as closely as C++ to make it more comprehensive. In addition, Java omits many rarely used, poorly understood, confusing feature of C++.
- Object-oriented - Java is fully object-oriented even more so than C++. Everything in Java, except for a few basic types like numbers is an object.
- Robust - Java is intended for writing programs that must be reliable in a variety of ways. Java puts a lot of emphasis on early checking for possible problems, later dynamic
checking and eliminating situations that are error-prone. Java has a pointer model that eliminates the possibility of overwriting memory and corrupting data.

- Architecture neutral - The compiler generates an architecture neutral object file format in which the compiled code is executable on many processors given the presence of Java run time system.
- Security - Closely related to Java's robustness is its focus on security. Because Java does not use pointers to directly reference memory locations, as is prevalent in C and C++, Java has a great deal of control over the code that exists within the Java environment.
- Platform Independent - The programs created in Java are portable in a network. The program is compiled into Java byte code that can be run anywhere in a network on a server or client that has a Java virtual machine.
- Multithreaded - Java support for multiple, synchronized threads that are built directly into the Java language and runtime environment. Synchronized threads are extremely useful in creating distributed, network-aware applications.

3.3.3 Tool

In order to make use of the JAVA programming language, the use of language tools is needed. These tools will include the essential functionality such as language compiler to additional functionality to aid programming in JAVA, such as debugger, GUI and language libraries. The following sections will discuss three of the most popular tools in developing JAVA application. Three of these tools are Visual Age, Visual J++ and Borland's JBuilder.

3.3.3.1 Visual Age

The VisualAge for Java product is IBM’s integrated development environment (IDE) for Java developers. It is a Java tool for creating e-business applications that target the IBM WebSphere software platform for e-business. VisualAge for Java allows transforming existing applications for the Web.

Advantages
Improved interoperability with other tools - Investments in tools and skills are maintained through integration with VisualAge for Java

Easy to use - With a persistence framework and unit test environment for WebSphere, VisualAge for Java, provides a fast way to develop, test, and deploy end-to-end e-business applications.

Scalable data solutions - Leverage the complete range of WebSphere servers and supporting platforms.

3.3.3.2 Visual J++

Microsoft Visual J++ is an integrated Windows-hosted development tool for Java programming. It is used to create a Java program, make modifications, compilation, running and debugging. Visual J++ is built around Developer Studio, Microsoft's common development environment. However, there are substantial differences between Java and Microsoft J++ when interfacing with the external code.

Advantages

- Provides GUI under the development environment
- Java's Abstract Window Toolkit (AWT) available.

3.3.3.3 Borland's JBuilder

Borland JBuilder Enterprise version 4.0 is the choice to develop the MPLS network simulator to enable cross-platform development and enabling web-based deployment. JBuilder 4.0 uniquely delivers the key features required for productive Java development including:

Advantages

- Unrivaled support for the Java 2 platform to deliver the most reliable, scalable, and preferment Java solutions.
- Visual tools and reusable components for rapidly creating platform independent Java applications, servlets, and applets.
Components Wizards and designers for creating reusable JavaBeans and Enterprise JavaBeans

JBuilder 4.0 can be used in Windows platform as well as UNIX platform.

3.4 Chapter Summary

This chapter covers the detail of the network simulation. It discussed the reason why simulation is used to simulate network topology as well as the approach used to develop a simulator. It also covers in depth details of the current network simulator. The final section covers the major analysis on programming language and tools available to develop a simulator.

MPLS VPN components will be developed using the object-oriented approach. These components will be incorporated into the existing UMJaNetSim. The JBuilder 4.0 will be used as the tools to develop the MPLS VPN simulator. The next chapter will discuss the analysis of the UMJaNetSim architecture as well as the MPLS VPN architecture.