# Chapter 3 Computer Network Simulation

One of the objectives of this thesis is to create a simulated environment that allows testing and experimentation of a NewReno simulation. This chapter will begin in the first section with the introduction of the computer simulation model and study the different types of simulation models.

The second section will discuss the various available simulators. A survey of existing network simulators is performed to show current approaches to network simulation. The section concludes with the advantages and disadvantages of each simulator.

The third section will explain the differences between the procedural approach and the object-oriented approach. A discussion of the advantages of the various programming approaches is conducted. This section will also emphasize the importance of the object-oriented approach in coding structure and reusability, compared to the procedural approach. A brief discussion on an object-oriented programming language, JAVA, is also given. Finally, a discussion on the programming tool that will be used to develop the network simulator is introduced.

The final section of this chapter is a summary of this chapter.

## 3.1 Computer Simulation

"Simulation is defined as the imitation of the behavior of some existing or intended system, or some aspect of that behavior. Examples of where simulation is used include communication network design, where simulation can be used to explore overall behavior, traffic patterns, trunk capacity, etc., " [V.ILLINGWORTH]

Today simulation can be performed on digital computer called computer simulation or digital simulation. Computer simulation can imitate the model of an actual system. Sometimes it only consists of partial characteristics of the vital properties of the actual system.

To implement an actual system is too expensive and difficult as physical network devices themselves are expensive and require a large network working space. From the technical aspect, it is more difficult to configure as well, compared to a computer simulation.

A computer simulation can avoid the same problem faced by the actual system since it only needs a set of Personal Computer (*PC*) and the right simulation software. A computer simulation is a cost-effective solution to imitate an existing actual system.

### 3.1.1 Simulation Model

A model is an abstraction of a system intended to replicate some properties of that system [C.M.OVERSTREET]. With the collection of the properties, model design is based on its objective to replicate some properties of that system. Model design should be workable and solves a problem domain.

Simulation model works on simulation, which imitates the actual system. The simulation should be able to collect the outcome of the simulation process. The simulation-collected result can determine whether the problem domain can be solved. If it cannot, it will provide another alternative to solve the problem. For example, in the case of a sender-side's TCP version of congestion control, if there are more than two consecutive lost segments occurring, Tahoe cannot provide an effective amount of throughput. This is because Tahoe does not consider more than one consecutive lost segments. Reno or NewReno are the other alternatives to get a large amount of throughput because they consider more than one consecutive lost segments.

### 3.1.2 Simulation Approach

According to R. E. Nance, computer simulations may be divided into three categories based on the simulation approach [R.E.NANCE]:

1) *Monte Carlo* simulation, a method by which an inherently non-probabilistic problem is solved by a stochastic process, where explicit representation of time is not required
2) *Continuous*, in which the variables within the simulation are continuous functions (normally involve solving differential equations)
3) *Discrete event*, where the change of the values of program variables happens at finite number of time points in simulation time (not necessarily evenly spaced)

Usually, the actual simulation involves the use of a combination of techniques. For example, a "combined" simulation refers generally to a simulation that has both discrete event and continuous components, whereas a "hybrid" simulation refers to the use of an analytical submodel within a discrete event framework. For network simulations, especially packet-level simulations, the discrete event approach is the most significant. The JaNetSim, as well as all network simulators discussed in this chapter, are based on the discrete event approach.

It is often possible to use a simulation model in conjunction with a less realistic but "cheaper-to-use" analytical model [BRATLEY]. An analytical model describes the system in question with mathematical formulas obtained through analyses (e.g. probability theory, queuing theory, etc.). Once the formulas are derived, the evaluation of the system can be quickly done. An analytical model often involves too many simplifying assumptions and may not represent the actual system correctly. On the other hand, the simulation model more closely resembles the actual system and is generally more accurate, but with a high computation cost. One approach is to first evaluate a system using an analytical model, then use simulation to validate the analytical model.

## 3.2 Current Existing Network Simulators

A network simulator can be either a general-purpose simulator or a special-purpose simulator. A general-purpose simulator consists of a wide range of possible simulations, whereas a special-purpose simulator targets a particular area of research [BRESLAU]. This section reviews a number of major network simulators, describing their features, advantages, and disadvantages.

### 3.2.1 INSANE

The Internet Simulated ATM Networking Environment (INSANE) [INSANE] is a network simulator designed to test various types of 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).

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++. Its scenarios are created using TCL scripting language. INSANE is designed to run in a large simulations environment and its results are processed off-line.

**Advantages**

- Although simulations are all sequential processes, INSANE works quite well on distributed computing clusters.

**Disadvantages**

- INSANE only runs on UNIX-based platforms.
- Not a user-friendly environment.
- The GUI of INSANE does not provide other features related to the creation of the simulation environment.
- Output performance can only be viewed in text.

### 3.2.2 OMNET++

Objective Modular Network Test bed in C++ (OMNET++) [OMNET] is a discrete event simulation tool. This simulator is designed to simulate computer networks, distributed systems and multi-processors. This simulation tool is developed on Linux. It works well on most UNIX systems and Windows platforms.

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

**Advantages**

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

**Disadvantages**

- Users require knowledge in C or C++ programming languages to use OMNET++.
- In order to simulate the network, users need to use the command line.

### 3.2.3 OPNET

OPtimised Network Engineering Tool (OPNET) [OPNET] is a discrete event network simulator. The OPNET Modeler was introduced in 1987 and developed at MIT. This simulator supports signaling; call setup and teardown, segmentation and reassembly of cells, cell transfer, traffic management and buffer management.

Its actual network structure and network components are built from graphical editors using object oriented modeling approach. Each component's behaviors are specified with a state transition diagram.

**Advantages**

- OPNET is able to model complex network topologies with unlimited sub-network testing within its hierarchical network models.
- 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 supports modeling of mobile and satellite networks.
- OPNET includes a numbers of comprehensive library of detailed networking protocols and application models.

**Disadvantages**

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

### 3.2.4 PARSEC

Parallel Simulation Environment for Complex Systems (PARSEC) [PARSEC] is a C-based discrete event network simulator. PARSEC represents a set of objects in the physical system as logical processes.

**Advantages**

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

**Disadvantages**

- The entire simulation process involves the use of the command line without a GUI.
- PARSEC is less portable among different platform.

### 3.2.5 REAL Network Simulator

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

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

**Advantages**

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

**Disadvantages**

- Users must have strong knowledge in C programming language.
- No graphical user interface (GUI) representation capabilities.
- Not a cross-platform simulator.

### 3.2.6 NIST ATM/HFC

The NIST ATM/HFC network simulator [NIST] was developed at the National Institute of Standards and Technology (NIST). This simulator provides a flexible test bed for studying and evaluating the ATM performance and HFC network without the cost of building a real network.

This simulator is written in C Structural Programming Language. NIST ATM/HFC gives users an interactive modeling environment with GUI that provides the user with a mean to display network topology from defined parameters and connectivity of the network, log data from simulation run, and to save and load the network configuration.

**Advantages**

- This simulator has a well-defined message passing mechanism based on the sending of events among simulation components, which is handled by an event manager.
- This simulator allows users to create different network topologies.
- Users are allowed 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 (GUI).
- Provides various instantaneous performance measurements displayed in graphical or text form on the screen while the simulation is running.

**Disadvantages**

- Users might face the network topology set up problems due to the requirement to consider a large number of parameters.
- This simulator lacks portability between different platforms because the simulator relies on the X Window System for its GUI. It can only run on UNIX or Linux platforms.
- For users and programmers, a strong foundation in C programming language is needed to customize and understand the simulator's components.

### 3.2.7 NS-2

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

**Advantages**

- NS-2 allows simulation with multiple levels of abstraction, where higher abstraction levels trade off accuracy for performance.
- NS-2 measurements do not impact the network by adding extra traffic.
- NS-2 includes a network emulation interface that permits network traffic to pass between real-world network nodes and the simulator.

**Disadvantages**

- Although NS-2 has a network animation tool that provides network visualization features, but it does not have a GUI for general simulation manipulation and scenario set up.

### 3.2.8 DELSI

DELSI (Delphi Simulation) [DELSI88] is a discrete-event simulation tool. It's purpose is to simulate the queuing of systems with complicated logic. Based on Piecewise Linear Aggregate Simulation Architecture, DELSI is implemented as a set of components for Borland Delphi [DELPHI99] 3.0 and 4.0. DELSI's internal simulation logic has been optimized for high loading that is for use with a large number of transactions in the model. The model's logic is implemented using Delphi event handling. The development of DELSI within the Delphi RAD allows users to combine DELSI with other components, which include GUI design, report building and databases.

**Advantages**

- DELSI can be combined with other components within the Delphi Rapid Application Development platform.
- DELSI is also very capable of handling a large number of transactions in the model.
- DELSI includes GUI for designing and the building of reports and databases.

**Disadvantages**

- Due to DELSI's origins as commercial simulation software, its actual building components are not clear to users or programmers.
- DELSI's use is limited only to the Borland Delphi tool – a tool not widely used in the industry and not portable to other platforms.

### 3.2.9 UMJaNetSim

The UMJaNetSim [UMJANETSIM] network simulator is a flexible test bed for studying and evaluating the TCP with IP network performance without the expense of building an actual network. This simulator is written in JAVA Language whereby it is developed in an object-oriented programming approach. This 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 runs, and save and load the network configuration.

**Advantages**

- Better graphical user interface (GUI), which provides a user-friendly environment.
- Output performance can be viewed in text and graphical representation on the screen while the simulation is running.
- Users are allowed to create different network topologies.
- Users are allowed to adjust the parameters of each component's operation, measure network activity, save/load different simulation configurations and log data during simulation execution.
- UMJaNetSim has a high portability among the various platforms and is readily web-enabled by using an applet version of the simulator.
- Users can add in new components without affecting the whole simulation as it is written in an 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.

**Disadvantages**

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

### 3.2.10 Summary of Existing Simulator

To summarize, all the simulators that we have discussed have their own advantages and weaknesses in terms of platform independence, network research focusing area, simulation techniques, the programming approaches and the availability of graphical user interface.

All the simulators that have been studied here are discrete event simulators. There is no web-enabled simulator yet today. Table 3.1 gives a comparison among these simulators in terms of object-oriented, graphical user interface (GUI), multithread and platform independence.

Table 3.1 shows that all of these simulators are not platform independent except UMJaNetSim. Most of the network simulators are written in object-oriented programming language. Only UMJaNetSim and OMNET++ have a good graphical user interface (GUI).

| Simulator | Object Oriented | GUI | Multithreaded | Platform Independence |
|---|---|---|---|---|
| INSANE | Yes | Poor | Yes | No |
| OMNET++ | Yes | Good | No | No |
| OPNET | Yes | Normal | No | No |
| PARSEC | No | Poor | Yes | No |
| REAL NS | No | Poor | No | No |
| NIST ATM/HFC | No | Normal | No | No |
| NS-2 | Yes | Normal | Yes | No |
| DELSI | Yes | Normal | No | No |
| UMJaNetSim | Yes | Good | Yes | Yes |

*Table 3.1 Comparison of Simulators*

## 3.3 Programming Technique

This thesis will use the UMJaNetSim network simulator to develop and to test NewReno. In order to develop a new component for a network simulator, the appropriate programming language, programming tools and programming approach need to be selected. This section will discuss in terms of each of the programming approaches and reasons behind the programming language chosen to develop the new network simulator component.

### 3.3.1 Approach

There are several programming approaches for developing a network simulator. These include procedural approach, structural approach and object-oriented approach. They are widely used in developing a network simulator (refer Table 3.1).

#### 3.3.1.1 Procedural Approach

The procedural approach makes use of procedural languages. Each program code is place into blocks. Each block is referred to as a procedure or a function. A function or procedure is coded in such a way that it will perform a certain function. In some occasions it may return a value. Examples of procedural programming languages are C, FORTRANS and Pascal.

Normally, a programmer writes a procedural-based program using the "from start to end" technique. The programmer must be clear of what he wants at the beginning of the coding. After completing a program, if the programmer intends to make or add a component he has to restructure the program so this approach is not suitable for developing a network simulator because although it has the same function name, the function definition are different, between components.

### 3.3.1.2 Structure Programming Approach

In the structure programming approach, before writing a single line of coding, the programmer has to design the program completely. In this case, a large amount of schematics, flow charts and other tools to document each interactive function and each piece of data flow are needed to develop a program.

The main idea of the structured-based programming is the division of tasks. A program might be a complex task but this task can be broken down into a set of smaller tasks. This division can continue until the tasks are sufficiently small and self-contained enough to be understood.

This approach is suitable for complex problems. It needs an experienced programmer to design the program structure carefully before the commencement of coding.

### 3.3.1.3 Object-Oriented Programming

Object Oriented Programming [JAVA] is a dominant programming paradigm. It has replaced structured procedure based programming techniques since the early 1970s. Object-oriented programming utilizes performs the concept of encapsulation, inheritance and polymorphism.

Encapsulation is used to combine data and behavior in one package while hiding the data implementation 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 called instance variable or fields. The methods are referred to as the object operation.

Inheritance is a concept of the properties' inheritance and the definitions of parent class to child class. When a programmer creates an object class he does not need to recreate certain classes again as he can inherit its properties from its parent class. For example, when creating a Proton Wira car object, the programmer can inherit car object (parent class) properties to the Wira car object. This action increases the speed of development. It also ensures an inherent validity to the defined subclass object.

Inheritance is the inheriting of the functionality of the parent classes to the smaller components. This concept is very useful in building a network simulator.

Polymorphism is the ability of objects of different classes related by inheritance to respond differently to the same function call. This is useful in creating a network simulator framework because the core simulator engine only interacts with various well-defined network interfaces; each component behaving in its own in the same function call.

In traditional structured programming, the algorithm would come first followed by the data structure. This means that this programming method requires designing a set of functions first before identifying the appropriate ways to store the data. This limits the way programmers work as they have to structure the code first. Object-oriented programming is the reverse. It puts data structure first before looking at the algorithm that operates the data.

The main benefits of Object-oriented programming (OOP) are as below:

- Simplicity – OOP is simple and intuitive
- Maintainability and Reusability – It is easy to maintain and modify the existing codes. New components can be created with slight differences from the existing one.
- Modifiable - OOP provides a good framework for code libraries, where supplied software components can be easily adapted and modified by the programmer
- Modularity – OOP is good for defining abstract data types where implementation details are hidden and the unit has a clearly defined interface.
- Extensibility - OOP lets the programmer to extend the functionality of each simulator by adding more classes without affecting the core of the system.

### 3.3.2 Java

Java [JAVA] is a platform-independent language and operating system. Java was developed by Sun Microsystems. Java is written in Java language that is both a programming language and an environment for executing programs. Traditional compilers convert source codes into machine-level instructions but the Java compiler translates Java source codes into instructions that are interpreted by the runtime Java Virtual Machine.

Java programming language has the advantages below:

- Simplicity – The fundamental concepts of Java, based on the object-oriented paradigm are simple and intuitive. It is often thought of as a C++ minus the "complexity and ambiguity" plus "security and portability".
- Object-oriented – Java is an Object-oriented language. Most of the properties in Java is object-oriented.
- Robust – Java is a reliable program emphasizing on early checking for possible problems, dynamic checking and eliminating errors. Pointer models of Java eliminates the possibility of overwriting memory and corrupting data.
- Platform Independent – Java programs are able to run on any platform, such as UNIX, Windows and other Operating System environment, in a network. The programs are compiled into Java byte codes that can be run in a network on a Java virtual machine, which may be a server or a client.
- Multithreaded – In Java language and runtime environments, Java supports multiple and synchronized threads.
- Architecture neutral – In Java runtime system, Java compiled codes are executable on different processors and different operating system.
- Security - Java does not use pointers to directly reference memory locations, like C++ and C. Java controls existing codes within the Java environment. Java's robustness focuses on security.

### 3.3.3 Tool

To make use of the Java programming language, the use of language tools is needed. These tools consist of necessary functional, aids for Java programming, such as debugger, and libraries. The next sub section will discuss the three most popular tools in developing Java application. These 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. This Java tool is used for creating e-business applications that targets the IBM WebSphere software platform. VisualAge for Java allows transforming existing applications for the Web.

This Java tool has such advantages as improving interoperability with other tools; easy to use in which it has a consistent framework and a unit test environment that provides a fast way to develop, test and deploy end-to-end e-business applications; scalable data solutions, and Java is an OOP that has the inheritance feature. This feature allows data and method of other class as to be used.

### 3.3.3.2 Visual J++

Microsoft Visual J++ is an integrated Windows-hosted development tool for Java programming. This tool is used to create, compile, modify, debug and run a Java program. Visual J++ is built around the Developer Studio, Microsoft's common development environment.

This Java tool has the advantage of providing GUI in the development environment, e.g. Java's Abstract Window Toolkit (AWT). This AWT package is used to create visual components on the user's screen. Examples of visual components are Frame, Button, List, Text Area and Text Field.

#### 3.3.3.3 Borland's JBuilder

Borland JBuilder Enterprise version 4.0 is one of the tools for those who want to develop the TCP NewReno network simulator component to enable cross-platform development and to enable web-based deployment. JBuilder 4.0 uniquely delivers the key features required for productive Java development including:

- JBuilder 4.0 can be used in Window-based platform as well as UNIX-based platforms
- Component Wizards and designers for creating reusable JavaBeans library and Enterprise JavaBeans
- Support for the Java 2 platform to deliver the most reliable, scalable and preferred Java solutions
- Visual tools and reusable components for rapidly creating platform independent Java applications, servlets and applets.

## 3.4 Chapter Summary

This chapter covered Computer Simulations, Simulation Models and Simulation Approaches. Three categories of simulation approaches were discussed being Monte Carlo, Continuous and Discrete Event categories

The features, advantages and disadvantages of the current existing network simulators was discussed.

Three Approaches to Programming Techniques was briefly discussed. They are the Procedural Approach, the Structure Programming Approach and the Object-Oriented Programming Approach. After comparisons were made, the Object-Oriented Programming approach was deemed the most suitable approach to use in developing the network simulator.

Java tools such as Visual Age, Visual J++ and Borland's JBuilder were also discussed.

The TCP NewReno components will therefore be developed using the object-oriented approach. These components will be incorporated into the existing UMJaNetSim v 0.5. The JBuilder 4.0 will be used as the tool to develop the TCP NewReno simulator. The next chapter will discuss the UMJaNetSim architecture as well as the TCP NewReno architecture.