FACULTY OF
COMPUTER SCIENCE
AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA

# IMPLEMENTATION OF TOPOLOGY CONVERTER INTO UM JANETSIM

ANDREW CHIAM MING JER( WEK010015 )

*Under the Supervision of*
**Mr. Phang Keat Keong**

*Moderator*
**Mr. Ang Tan Fong**
SESSION 2003/2004



This project is submitted to the faculty of Computer Science and Information Technology, University of Malaya, in partial fulfillment requirement of the Bachelor of Computer Science.

# Abstract

This project describes and implement the approach and procedure involve in the development of the network topology converter to be implemented in JaNetSim.

The development of topology converter will made possible the conversion of different topology file format in Ns-2 and JaNetSim to enable wider range of simulation to be carried out more efficiently and effectively. Ns-2 is an established network simulator with the most active research and it can be a great and powerful reference tools to improve and complement the functionality of UM JaNetSim. The topology converter will incorporate several important features. The most important features lies in the object oriented approach which provide the luxurious of manipulating inheritance and polymorphism which provide benefits like reusability, flexibility and extendibility. Another important feature is the friendly, intuitive and simple to use interface. The interface provided by the topology converter will be consistent with the JaNetSim layout. This will greatly cut the amount of time to help user to familiarize and utilize the functionalities provided by the topology converter.

The topology converter will be helpful to translate various topology formats to increase portability and widen the scope for various simulators.

# Acknowledgement

First, I would like to express my utmost gratitude to my project supervisor, Mr. Phang Keat Keong for being such a great and inspiring supervisor for the generous sharing of knowledge and support when it comes to the project discussion. He had always concerned about our work progress and readily took some time off his hectic work schedule to listen to our problems and make things right for us.

Besides that I would like to convey my sincere gratitude to Mr. Ling Teck Chaw and my moderator Mr. Ang Tan Fong for their kind and resourceful help and advice during the discussion and unselfish share of knowledge and technical solutions.

I would also like to express my deepest appreciation to my family members for providing a strong mental and moral support to me during the project development. They had provided me with strong pillar of support in time of despair and this has greatly motivated and inspired me to greater heights in life.

Lastly, I would like to thanks all my others project members especially to Chia Kai Yan, Lim Lee Wen and Tang Geck Hiang for their advice, support and cooperation.

# Table Of Contents

# List of Figures

# List of Tables

# Chapter 1 Introduction

## *1.1 Introduction to Network Topology*

Network topology refers to the specific physical or logical arrangement of the elements in a network (Anon 1998). Two networks have the same topology if the connection configuration is the same, although the networks may differ in physical interconnections, distances between nodes, transmission rates, or signal types. The common types of network topology will be illustrated in Chapter 2 of the thesis under Topology.

The network topology tree plays a critical role in management of the network (Cisco 2003). It consists of four main purposes:

- Identifies key components of the network
- Organizes the settings and convention for the key components
- Defines the physical structure of network topology
- Provides building blocks for the network
- Defines the desired traffic flows across your network

## *1.2 Introduction to Network Simulator*

With the rapid development of high-speed network, network simulator has become a valuable tool to study and investigate the protocol and design issues regarding the performance of the network. It allow user to make correct decision on designing a network without the need to invest into the technology. A network simulator can be used as a tool for network planning or as a tool for protocol performance analysis. It is useful for modeling network behavior under different setting and conditions for the

various network components. Users are able to analyze and predict the performance of the network design based on the generated result of network simulator. Besides that, researchers and network planners are able to analyze networks without the expense of building a real network with the use of a simulator. Huge saving can be made both in terms of investment and the cost in terms of unnecessary restructuring for experimentation. (AU-KBC 2003)

## *1.3 Objective*

The primary objectives of this project are to study and understand the operation of creating and generating a topology in various network environments. This involves researching work on various network simulator topologies to enhance the understanding of the concept behind the topology generation. In the research, comparison of the topology generated in JaNetSim and Ns-2 are examine in details to differentiate the approach used by various simulator to create, simulate and save the topology format.

The goal of the project is to create a converter program that is able to convert topology format between Ns-2 and JaNetSim. The script developed must be able to integrate into the existing JaNetSim system

.

The converter program should have the following capabilities:

- Convert Tcl script from Ns-2 to topology format in UM JaNetSim.
- Convert from topology format in UM JaNetSim to tcl script in Ns-2.
- Allow topology to be saved in two format:

  i) Plain topology file without the parameters

  ii) Topology with logging of all values at saved time.

## *1.4 Scope*

This project mainly involves a lot of research work on the existing network simulator and its functionalities. Thus the scope of the project will be covering two simulator that will be dissect and research intensively, which is the Ns-2 and JaNetSim simulator.

During the research, the overview ,structure and file saving method of both simulator are explore to provide clear and unifying view of the entire system to improve understanding on the work mechanism to ease the process of developing the converter for the project.

## *1.5 Schedule*

| ID | Task Name | Duration | Start | Finish |
|---|---|---|---|---|
| 1 | Project Definition | 12 days | Mon 6/16/03 | Tue 7/1/03 |
| 2 | Literature Review | 20 days | Mon 6/30/03 | Fri 7/25/03 |
| 3 | Research Works | 10 days | Mon 7/21/03 | Fri 8/1/03 |
| 4 | System Analysis | 25 days | Mon 7/28/03 | Fri 8/29/03 |
| 5 | System Design | 20 days | Wed 8/20/03 | Tue 9/16/03 |
| 6 | Implementation | 78 days | Wed 9/17/03 | Fri 1/2/04 |
| 7 | Testing | 31 days | Fri 1/2/04 | Fri 2/13/04 |
| 8 | Documentation | 167 days | Mon 6/30/03 | Tue 2/17/04 |

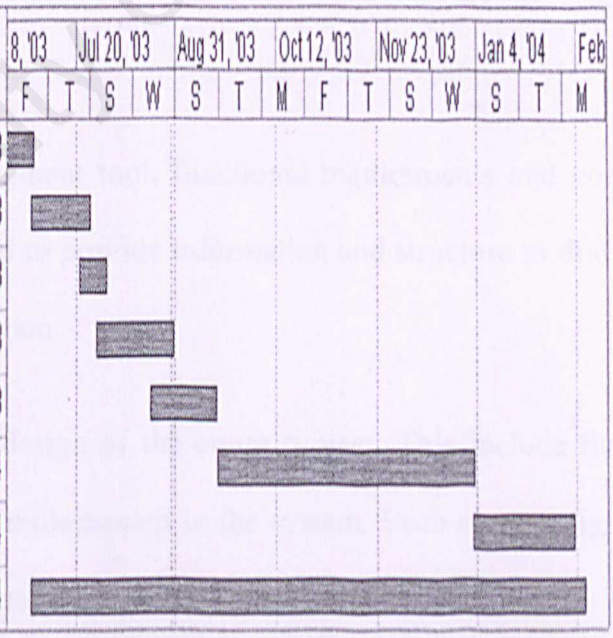



**Figure 1.1 : Project schedule of the task during the research and development**

## *1.6 Report Organization*

The others chapters in the thesis are organized as follow :

***Chapter 2*** covers the research work done during the project to enhance the knowledge and gathering of basic concept in network topology. The chapter mainly covers 3 sections. The first section is a review of network topology concept. The second section covers the evaluation of current existing network simulator and the last section reviews the programming approaches that will used to develop the converter to be integrated into JaNetSim.

***Chapter 3*** will dissect both simulators, Ns-2 and JaNetSim to get more details information on the components and operational process with regard to topology on both system. Besides that, it will be looking on the topology and save file format on both simulator.

***Chapter 4*** will be covering the development tool, functional requirements and non functional requirements that are essential to provide information and structure to draft a design for the topology converter program

***Chapter 5*** will focus on the overview design of the entire project. This include the design proposal of object classes to be implemented in the system. Each class design includes attributes and description of method perform by that class.

***Chapter 6*** covers the implementation of the topology converter. It explains in details of all the operational work flow and programming coding to be implemented in the converter program.

*Chapter 7* will include all the systematically approach taken in order to test the workability of the system to conform the functionalities in the requirements that the system had promised to deliver.

*Chapter 8* will concludes the research and development of the network topology. It summarize the findings of the project, the final product and the constraint during the development and testing stage.

# Chapter 2 Literature Review

## *2.1 Introduction to Topology General Concepts*

A major component of a network consists of links and nodes. The arrangement and interconnection of the links and nodes is known as the network topology. Generally, in network topology, there are few concepts that provide the basic for the relationship:

- Line configuration
- Topology
- Transmission mode

## *2.2 Line Configuration*

Line configuration refers to the way two or more communication devices attach to a link. Link is a physical communication pathway that transfers data from one device to another device (Forouzan 2001). In order for communication to occur, two devices must be connected in some other way to the same link at the same time. There are two possible line configurations: point-to-point and multipoint.

### 2.2.1 Point-to-Point

Point -to-point is a term used to describe a data channel which connects two, and only two, terminal by providing a dedicated link between them. During the transmission, the entire capacity of the channel is reserved for the terminals communication. There are two different types of point-to-point:

- Point-to-point Circuit: A communication circuit, or system connecting two points through a telephone circuit, or line.
- Point-to-point Network: A Point-to-point Network is one in which exactly two stations are connected. It may be dial connection or a leased line.

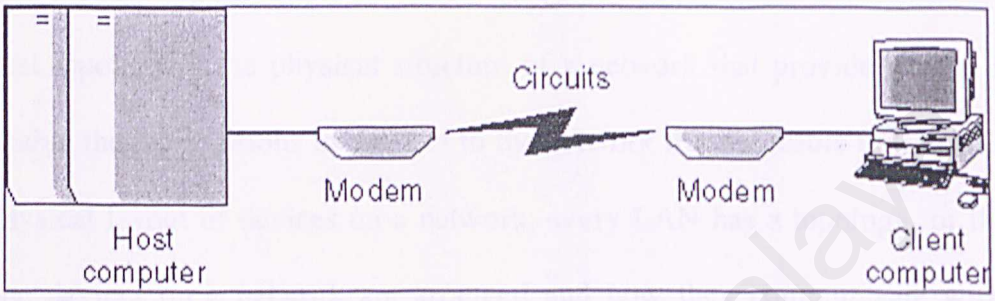



**Figure 2.1 : Point-to-point line configuration**

### 2.2.2 Multipoint

A multipoint line configuration describes a line configuration in which a single transmission facility is shared by several end stations. Line or circuit interconnecting several stations are also called multipoint line .Only one station can send or receive at any time, all others must wait. If several devices can use the link simultaneously, it is a spatially shared line configuration. If users must take turns, it is a time-shared line configuration.

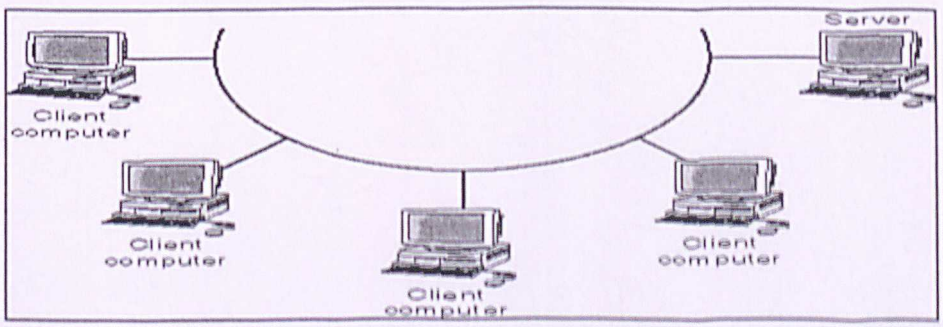



**Figure 2.2 : Multipoint line configuration**

## *2.3 Topology*

### 2.3.1 Introduction to topology

Topology refers to the shape of a network, or the network's layout. The network's topology determined the connectivity of the nodes in the network and the communication method. Topologies are either physical or logical.

Physical topology is the physical structure of a network that provide for the layout that enable the workstations to connect to the network through cable to transmit data. For physical layout of devices on a network, every LAN has a topology, or the way that the devices on a network are arranged and how they communicate with each other.

The logical topology, in contrast, is the way that the signals act on the network media, or the way that the data passes through the network from one device to the next without regard to the physical interconnection of the devices.

### 2.3.2 Reviews of Topology

#### 2.3.2.1 Bus

A bus topology is a network topology in which there is a single line called the bus or backbone to which all nodes are connected.Nodes are connected to the bus by drop lines and taps.A drop line is a connection running between the device and the main cable .A tap is a connector that either splices into the main cable or punctures the sheathing of a cable to create a contact with the metalic core (Anon 2003).

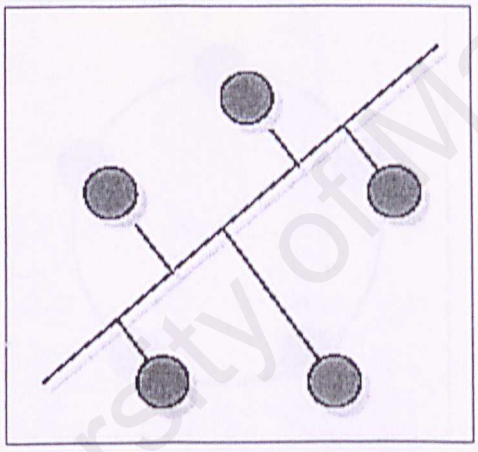


**Figure 2.3 : Bus topology**

Advantages

- Easy installation and uses less cabling than others topologies.
- Connectivity between dedicated nodes is not affected by failure of another node.

Disadvantages

- Difficult reconfiguration and fault isolation.
- A faulty bus cable stops all the transmission even between devices on the same sode of the problem

### 2.3.2.2 Ring

Ring topology is a network topology in which every node has exactly two branches connected to it.All devices are connected to one another in the shape of a closed loop, so that each device is connected directly to two other devices, one on either side of it. A signal is passed along the ring in one direction from device to device until it reaches its destination. Each device in the ring incorporates a repeater. When a device receives a signal intended for another destination, the repeater regenerates the bits and passes them along the networks.

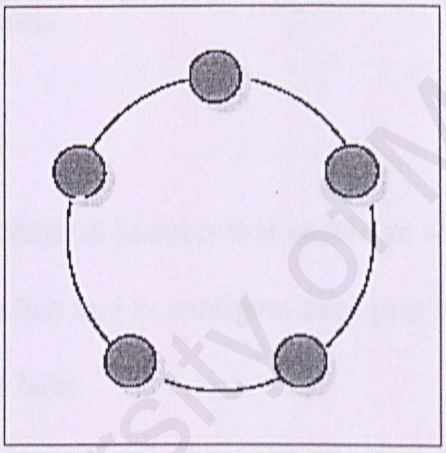


**Figure 2.4 : Ring topology**

Advantages

- Easy to install and reconfigure.
- All stations have equal priority for the medium access.

Disadvantages

- Shared bandwidth for the entire network
- Unidirectional traffic provides no alternatives in case of network failure.

### 2.3.2.3 Star

Star topology is a network topology in which peripheral nodes are connected to a central hub, which rebroadcasts all transmissions received from any peripheral node to all peripheral nodes on the network, including the originating node. All peripheral nodes may thus communicate with all others by transmitting to, and receiving from, the central hub only. There are three different types of hub:

Passive Hub

A passive hub serves simply as a conduit for the data, enabling it to go from one device (or segment) to another.

Intelligent Hub

Intelligent hubs include additional features that enable an administrator to monitor the traffic passing through the hub and to configure each port in the hub. Intelligent hubs are also called manageable hubs.

Switching Hub

A third type of hub, called a switching hub, actually reads the destination address of each packet and then forwards the packet to the correct port.

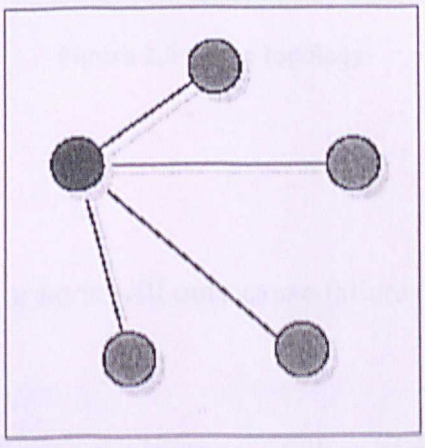


**Figure 2.5 : Star topology**

Advantages

- Simple expansion.
- Straightforward network management.

Disadvantages

- Single point of failure in case of failure of the central hub.

### 2.3.2.4 Tree

Tree topology is a network topology in which the nodes are arranged as a tree. From a topologic viewpoint, this resembles an interconnection of star networks in that individual peripheral nodes are required to transmit to and receive from one other node only and are not required to act as repeaters or regenerators. Unlike the star network, the function of the central node may be distributed.

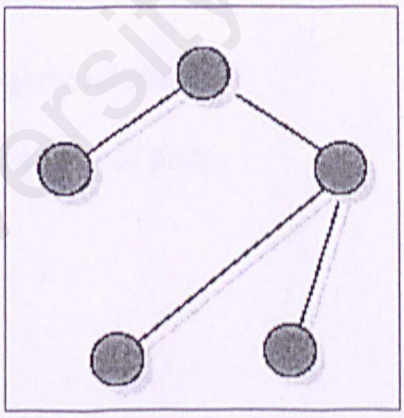


**Figure 2.6 : Tree topology**

Advantages

- Easy to expand
- Failure of the superior node will only cause failure to some subsystems.

Disadvantages

- Branches can be disconnected in case of failure of a superior node

### 2.3.2.5 Mesh

Mesh topology is a network topology in which there are at least two nodes with two or more paths between them. Devices are connected with many redundant interconnections between network nodes. In a true mesh topology every node has a connection to every other node in the network

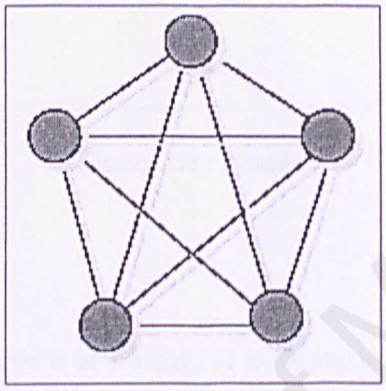


**Figure 2.7 : Mesh topology**

Advantages

- Topology with the highest reliability
- Direct connectivity between all nodes

Disadvantages

- Expensive
- Difficult to expand and reconfigured
- Complex wiring

## *2.4 Transmission Mode*

Transmission mode is used to define the direction of signal flow between two linked devices (Forouzan 2001). There are three types of transmission modes: simplex, half-duplex and full-duplex.

### 2.4.1 Simplex

Data can flow in only one direction. Only one of the two stations on a link can transmit; the other only can receive

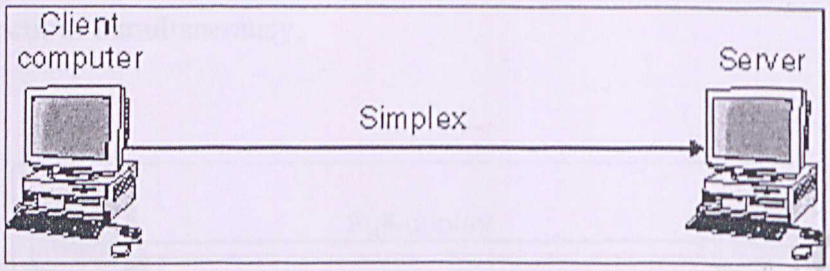



**Figure 2.8 : Simplex**

### 2.4.2 Half-duplex

Data flows in only one direction at a time. It is sometimes called two-way alternate. Each station can both transmit and receive, but not at the same time. When one device is sending, another can only receives and vice versa.

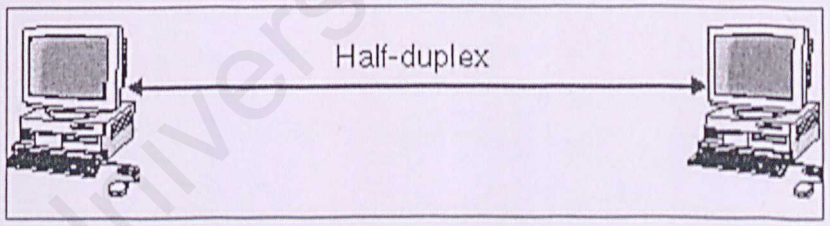



**Figure 2.9 : Half-duplex**

### 2.4.3 Full –duplex

Data flows in both directions at the same time.Most modem connections today transmit full duplex increasing efficiency with data flowing on the same pair of wires in both directions simultaneously.

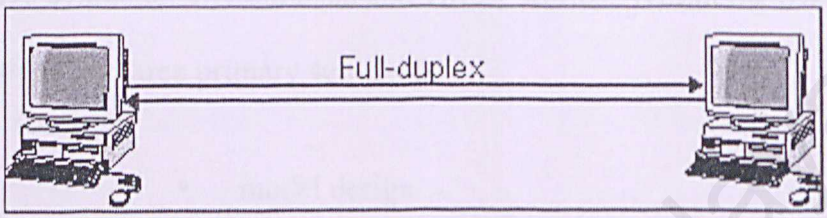



**Figure 2.10 : Full-duplex**

## *2.5 Introduction to Computer Simulation*

Computer simulation is designing of an actual or theoretical physical system, executing the model on a digital computer, and analyzing the execution output. Simulation embodies the principle of "learning by doing" – a model of some sort is build and then operates the model to learn about a system. Computer simulation serves to drive synthetic environments and virtual worlds. Within the overall tasks of simulation, there are three primary sub-fields:

- model design
- model execution
- model analysis

Models are designed to provide answer at a given abstraction level – the more detailed the model, the more detail the output.

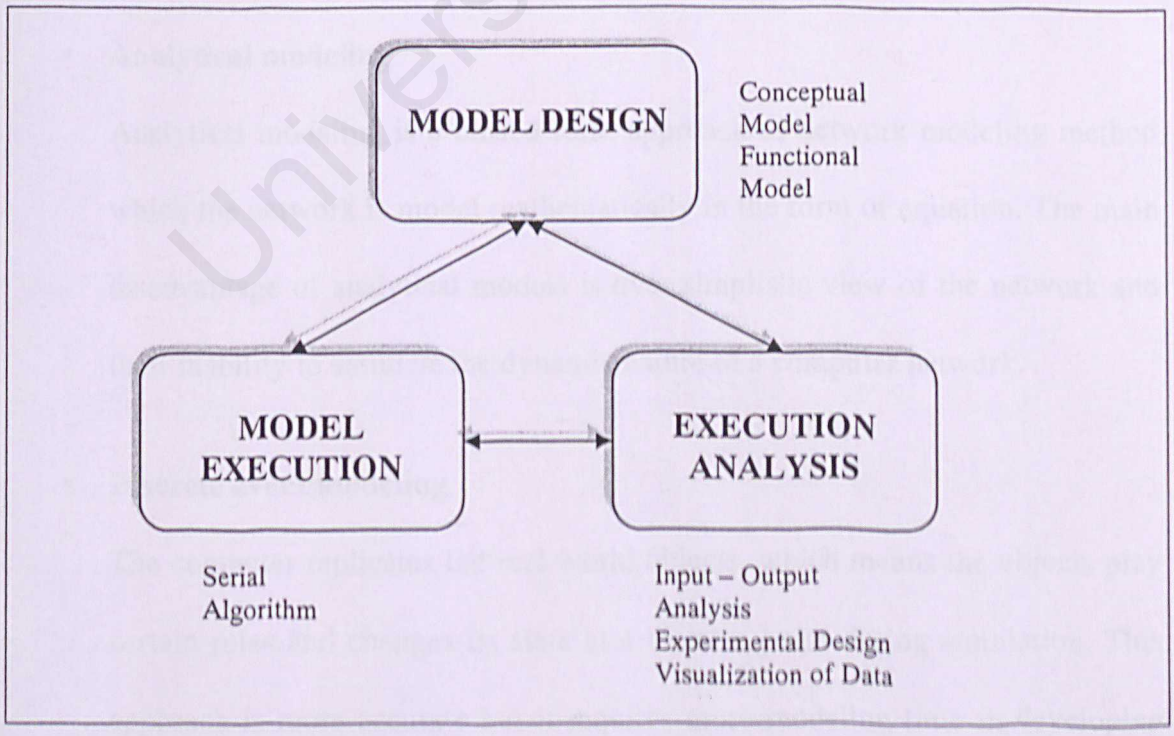



**Figure 2.11 : Three Sub-Fields of Computer Simulation**

Simulation is often essential in the following cases:

- The model is very complex with many variables and interacting components
- The underlying variables relationships are non-linear
- The model contains random variates
- The model output is to be visual as in a 3D computer animation

The power of simulation is that – even for easily solvable linear systems – a uniform model execution technique can be used to solve a large variety of systems. Another important aspects of the simulation technique are to builds a simulation model to replicate the actual system.

### 2.5.1 Network Simulation Approaches

There are two approaches to modeling a network simulator. These two approaches are as below :

- **Analytical modeling**

  Analytical modeling is a closed form approach of network modeling method which the network is model mathematically in the form of equation. The main disadvantage of analytical models is over simplistic view of the network and their inability to simulate the dynamic nature of a computer network.

- **Discrete event modeling**

  The computer replicates the real world objects, which means the objects play certain roles and changes its state at a discrete point during simulation. This approach is more accurate but it requires more modeling time in developing

the system. Besides that, it need more time in processing the real world objects.

### 2.5.2 Study of Various Existing Simulator

A network simulator is used to perform experiments on network without the expanses of building a real network. It help user to perform analysis on the network and obtain accurate information in order to plan and design the network more efficiently.

Generally, ATM network simulators are able to support network performance analysis in varying traffic types and loads, network capacity planning, traffic aggregation studies and ATM network protocol research. The following are the current ATM network simulator evaluated to analysis their strength and weakness:

- INSANE
- NIST ATM/HFC
- YATS
- OMNET++
- NetSim++

#### 2.5.2.1 INSANE

INSANE (Internet Simulated ATM Networking Environment) is designed to test various IPover 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.

the system. Besides that, it need more time in processing the real world objects.

### 2.5.2 Study of Various Existing Simulator

A network simulator is used to perform experiments on network without the expanses of building a real network. It help user to perform analysis on the network and obtain accurate information in order to plan and design the network more efficiently.

Generally, ATM network simulators are able to support network performance analysis in varying traffic types and loads, network capacity planning, traffic aggregation studies and ATM network protocol research. The following are the current ATM network simulator evaluated to analysis their strength and weakness:

- INSANE
- NIST ATM/HFC
- YATS
- OMNET++
- NetSim++

#### 2.5.2.1 INSANE

INSANE (Internet Simulated ATM Networking Environment) is designed to test various IPover 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). Internet protocols supported include large subsets of IP, TCP, and UDP. 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.

INSANE is designed to run large simulations whose results are processed off-line. 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).

Although there is no graphical user interface, a (optional) Tk-based graphical simulation monitor provides an easy way to check the progress of multiple running simulation processes. The bulk of INSANE is written in C++. Customization and simulation configuration is performed with Tcl scripts.

### 2.5.2.2 NIST ATM/HFC

This simulator was developed at the National Institute of Standards and Technology (NIST) and it is a tool to analyze the behavior of ATM and HFC networks without the expense of building a real network. Therefore, this simulator can conceivably be used to plan be used to plan ATM networks as well as analyze ATM and HFC protocols.

It allows the user to interactively model the environment with a graphical user interface. By using the NIST ATM/HFC simulator, the user can create different

network topologies, adjust the parameters of each component's operation, measure network activity, save/load different simulation configuration and log data simulation execution.

#### 2.5.2.3 YATS

YATS (Yet Another Tiny Simulator) is a small cell-level simulation tool for ATM. Its kernel comprises the event scheduler, a symbol manager and a scanner/parser front end An input file describes the - arbitrary - model network configuration, the simulation actions and the way to analyze the results. The input language is a simple script language, which allows for a flexible problem description (loops, macros and basic mathematical capabilities are provided). The discrete-time event scheduler applies a static calendar queue and unusual event memory management, which results in good simulation speed.

The system is written in C++. All network nodes are objects that communicate over standardized messages. Graphical object classes are able to display the time dependent behavior of variables and distributions inside of other model objects (without adding complexity to these network objects).

#### 2.5.2.4 OMNeT++

OMNeT++ (Objective Modular Network Testbed in C++) 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 on Windows platforms (NT recommended). It provides a

simulation library with statistical classes and an environment that supports interactive simulation including the visualization of collected data.The gnu plot-based GUI tool is used for analyzing and plotting simulation results.

#### 2.5.2.5 NetSim++

In a nutshell, NetSim++ is a software package designed to provide a comprehensive work environment for the network modeler. It can be used in areas of communications networks such as performance measurement for existing or future networks under a wide range of conditions.

Besides that, it can perform analysis and simulation of queuing systems. NetSim++ is designed specifically for the development and analysis of communications networks. Models can be hierarchically structured, allowing their re-use in different simulations. Specifications are entered graphically with specialized editors. The editors provide an efficient medium for design capture via a consistent set of modern user-interface elements.

NetSim++ follows for the hierarchy and communication model a subset of SDL-92 semantics. As with SDL, the active parts are processes; a hybrid approach is used to embed C++ language code with a graphically specified Extended Finite State Machine (EFSM).

### 2.5.3 Advantage and Disadvantages of Various Simulator

| Simulator | Advantages | Disadvantages |
|---|---|---|
| INSANE | The Tk-based graphical simulation monitor enable user to check the progress of multiple running simulation process. Besides that, it is able to support the simulation on a large network, which the result is processed off-line. | The simulator can only works on a few hardware and platforms only and this restricted the portability of the simulator. Furthermore, there are a few software requirements to run the simulator and this will be troublesome for the user to use the software |
| NIST ATM/HFC | The user can create different topologies and able to adjust the parameters during the simulation of the network. The user can save and load various simulation configuration.The simulator provides a graphical user interface and enable user to drag and drop the entities in the network. | Users of the simulator might face problems setting up the network topology because they need to input a large number of parameters. The customization of the simulator's component requires user or programmers to have strong foundation in C. Besides that, it is using procedural approach whereby the components have overlapped functions between the components. The simulator only can run on |

| | | |
|---|---|---|
| | | limited platform that is UNIX and LINUX platform. |
| YATS | The simulation of the network has reasonable speed and simple models virtually can run in real time. The simulator has high flexibility of integrated model description, simulation control, and result analysis. The whole simulation experiment can be instrumented via environment variables that in turn allows - together with a shell script - to easily perform complete experiment series over night. Although it is very simple, the online displays are useful to understand what's happening in the model network. This especially holds for ABR, TCP and all this protocol stuff. | The pure slotted operation causes some restriction when simulating different line speeds in the same model. It's only possible to choose speeds for which the cell transfer time is an integer multiple of a basic time used for the whole model. Lower line speeds are emulated by the multiplexer objects classes MuxAF/MuxDF, the ABR multiplexer does not yet support lower speeds. The discrete-time nature excludes some useful source models like the Poisson types. While the language based model description yields a high flexibility, the input may become a bit irritating in case of larger networks |
| OMNET++ | OMNeT++ has a solid and flexible simulation kernel and it provide powerful GUI environment for simulation execution. Users can build | User must use command line to simulate the network and posses knowledge in C or C++ programming languages |

| | | |
|---|---|---|
| | hierarchical and reusable models easily. The interface is human readable and its source code is provided. | to use OMNet++. |
| **NetSim++** | NetSim++ provides an efficient event-driven Simulation Kernel, a Simulation API and a Base Models Library of components. It takes the design specification and automatically generates an executable simulation. A set of analysis tools is provided to interpret and visualize a large volume of simulation results. | The current implementation of NetSim++ is available only for UNIX/X Window System platforms. |

**Table 2.1 : Advantages and Disadvantages of various simulator**

### 2.5.4 Comparison of network simulator

Based on the evaluation of the network simulator, the Table 2.2 below summarize the network simulator on a few feature such as discrete-event simulator, object-oriented, GUI, multithreaded, web enabled, platform independent.

| Simulator | Discrete Event Simulation | Object-Oriented | GUI | Multithread | Web-Enable | Platform Independent |
|---|---|---|---|---|---|---|
| INSANE | √ | √ | √ | √ | X | X |
| NIST ATM/HFC | √ | X | √ | X | X | X |
| YATS | √ | √ | | √ | X | X |
| OMNet++ | √ | √ | √ | | X | X |
| NetSIM++ | √ | X | √ | √ | X | X |

**Table 2.2 : Comparison of various simulator**

## *2.6 Topology Generator*

In order to engineer and design the internet, crucial issue such as the large scale structure of its underlying physical topology, its time evolution and the contribution of its individual components to its overall function need to be well understood. Extensive simulations are usually performed to assess its feasibility, in terms of efficiency and performance.

In general, Internet studies and simulations assume certain topological properties or use synthetically generated topologies. If such studies are to give accurate guidance as to Internet–wide behavior of the protocols and algorithms being studied, the chosen

topologies must exhibit fundamental properties or invariants empirically found in the actual extant structure of the Internet. Otherwise, correct conclusions cannot be drawn.

There are several synthetic topology generators available to the networking research community .Many of them differ significantly with respect to the characteristics of the topologies they generate

### 2.6.1 BRITE

**Design and implementation**

BRITE was designed to be a flexible topology generator, not restricted to any particular way of generating topologies (Medina et al. 2001). As such, it supports multiple generation models. This section will be describing the design goal and approach and behind BRITE implementation. Figure 2.12 depicts a schematic view of the structure of BRITE .The different components in BRITE are labeled from (1) – (4).

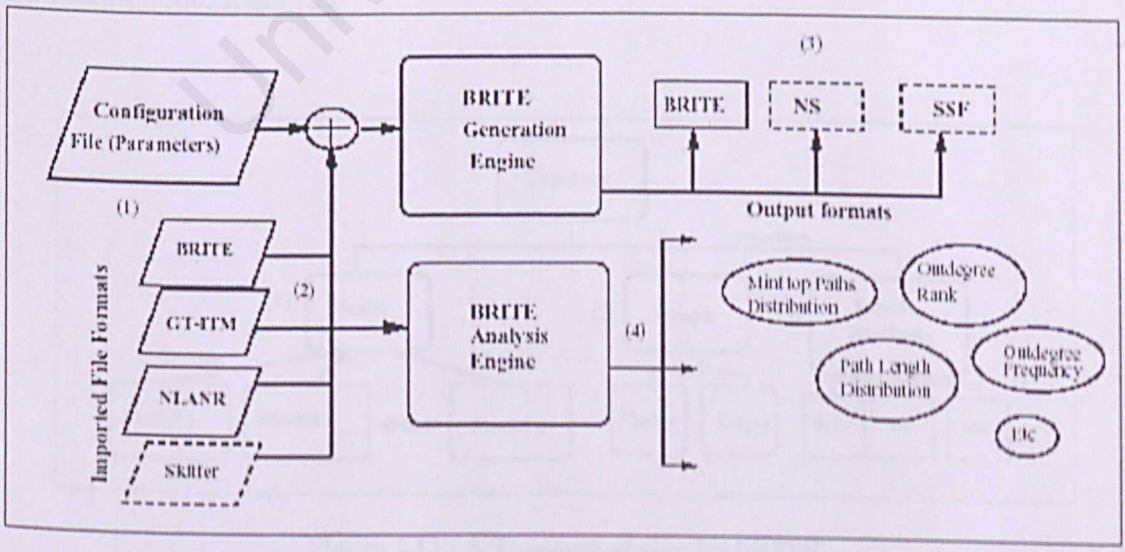



**Figure 2.12: Schematic structure of BRITE**

BRITE reads the generation parameters from a configuration file (1) that can be either hand written by the user or automatically generated by BRITE's GUI .BRITE provides the capability of importing topologies (2) generated by other topology generators (GT-ITM, Inet , Tiers , BRITE 1.0) or topological data gathered directly from the Internet (NLANR, Skitter ). BRITE can be included in the "imported" file formats, because it is possible to generate topologies using BRITE and then reusing them to generate other topologies by combining them with BRITE models or other imported formats. In the current distribution BRITE produces a topology in its own file format (3), and output capabilities for producing topologies that can be used directly by the Network Simulator (NS) and the Scalable Simulation Framework (SSF) simulators are currently being developed.

**BRITE's Architecture**

In BRITE, a topology is represented by a class Topology. This class contains a Model (1) and a Graph (2) as data members, and among others, a set of exporting methods and function members (3).

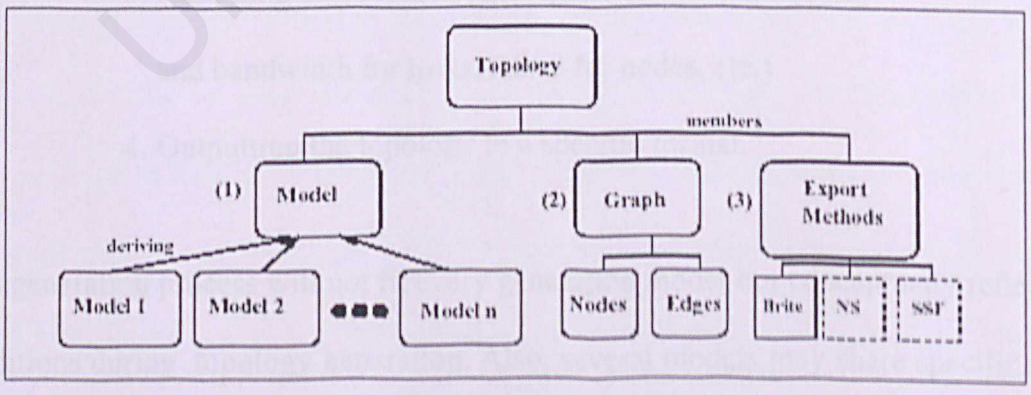



**Figure 2.13 : A Topology as seen by BRITE**

The Model class is an abstract base class from which multiple specific generation models are derived.Each specific topology generated by BRITE can use a single instance of one of the available generation models if the generated topology is flat, or more than one instance if the topology is a combined hierarchical topology. The Graph data member (2) is a Graph class with the minimal functionality required by the generation models. The class may be extended or replaced with minimum effects on the remaining code if the graph component is required.

Finally, the general architecture shows a set of export methods which output BRITE topologies into specific formats.

**Topology generation process**

The specific details regarding how a topology is generated depend on the specific generation model being used. The generation process can divided into a four-step process:

1. Placing the nodes in the plane
2. Interconnecting the nodes
3. Assigning attributes to topological components (delay and bandwidth for links, AS id for nodes, etc.)
4. Outputting the topology to a specific format.

This generation process will not fit every generation model but conceptually reflects conditions during topology generation. Also, several models may share specific steps during the generation process, while other models differ significantly on the individual steps.

### 2.6.2 Other Topology Generator

#### 2.6.2.1 Waxman

Waxman developed one of the first topology generators. This generator produces random graphs based on the Erd¨os-Renyi random graph model, but it includes network–specific characteristics such as placing the nodes on a plane and using a probability function to interconnect two nodes in the Waxman model which is parameterized by the distance that separates them in the plane.

#### 2.6.2.2 GT_ITM

One of the most popular generators available is GT-ITM. The main characteristic of GT-ITM is hat it provides the Transit-Stub (TS) model, which focuses on reproducing the hierarchical structure of the topology of the Internet. In the TS model, a connected random graph is first generated by using the Waxman method or a variant of it. Each node in that graph represents an entire *Transit domain*. Each transit domain node is expanded to form another connected random graph, representing the backbone topology of that transit domain. Next, for each node in each transit domain, a number of random graphs are generated representing *Stub domains* that are attached to that node. Finally, some extra connectivity is added, in the form of "back-door" links between pairs of nodes, where a pair of nodes consists of a node from a transit domain and another from a stub domain, or one node from each of two different stub domains. GT-ITM also includes about five flavors of flat random graphs.

### 2.6.2.3 Tiers

Another generator that implements models trying to imitate the structure of the Internet is Tiers.The generation model of Tiers is based on a three-level hierarchy aimed at reproducing the differentiation between Wide-Area, Metropolitan-Area and Local-Area networks comprising the Internet.

### 2.6.2.4 Inet and PLRG

Inet and PLRG are two generators aimed at reproducing the connectivity properties of Internet topologies. These generators initially assign node degrees from a power-law distribution and then proceed to interconnect them using different rules. Inet first determines whether the resulting topology will be connected, forms a spanning tree using nodes of degree greater than two, attaches nodes with degree one to the spanning tree and then matches the remaining unfulfilled degrees of all nodes with each other. PLRG works similarly to Inet in that it takes as an argument the number of nodes to be generated and value of the exponent .

## *2.7 Programming Language*

There are a few of programming languages that can be used in the development of the network simulator. In addition, the features of the programming language must be able to meet the requirements of the system to be developed. Since most of the simulator is built in Object Oriented Programming approach, the programming language must support the OOP approach. The choices of programming language should be able to support other functionality of the network simulator.

In this section, it reviews on a main OOP programming languages, Java and Tcl, a simple yet flexible and powerful scripting language. These two programming language has become the most popular language used to develop the network simulator.

### 2.7.1 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. (Deitel 2003)

The original intent was to use C++, but as work progressed in this direction, the Java developers realized that they would be better creating their own language rather than extending C++.J ava 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. So, unlike languages like C and C++, on which Java is based, Java is an interpreted language.

Java is best described as a small, simple, safe, object-oriented, interpreted or dynamically optimized, byte-coded, architecture-neutral, garbage-collected, multithreaded programming language with a strongly typed exception-handling mechanism for writing distributed, dynamically extensible programs. Java has several of important features that make it an attractive programming language as below:

- Java is simple

  Java started out as C++ but has had certain features removed, it is certainly a simpler language than C++. Java has simplified C++ programming by both adding features beyond those found in C++ and by removing some of the features that make C++ a complicated and difficult language to master. Java is simple because it consists of only three primitive data types-numbers, Boolean types, and arrays.

- Everything else in Java is a class.

  Java is object-oriented – The design of Java is completely object-oriented. Java provides all the luxuries of object-oriented programming: class hierarchy, inheritance, encapsulation, and polymorphism-in a context that is truly useful and efficient. Java's object-oriented structure enables user to develop more useful, more tailor able, and much simpler software the first time around.

- Java supports the Internet

  Java can be used to build small application modules or applet for use as part of a Web page. Applets make it possible for a Web page user to interact with the page. Java is general purpose Java can be used to create complete applications that may run on a single computer or be distributed among servers and clients in a network.

- Java is robust

  The Java objects can contain no references to data external to themselves

or other known objects. This ensures that an instruction cannot contain the address of data storage in another application or in the operating system itself, either of which would cause the program and perhaps the operating system itself to terminate or "crash". The Java virtual machine makes a number of checks on each object to ensure integrity.

- Java is secure

  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.

- Java is platform-independent

  The programs created are portability 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. The Java virtual machine interprets the byte code into code that runs on the real computer hardware. This means that individual computer platform differences such as instruction lengths can be recognized and accommodated locally just as the program is being executed. Platform-specific versions of the program are no longer needed.

- Java supports 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. Such an application

may be communicating with a remote server in one thread while interacting with a user in a different thread.

## 2.7.2 TCL

Tcl is the leading scripting language for a wide variety of integration application needs, whether to build a powerful GUI, embed Tcl in application, create a multi-threaded application, or develop a cross-platform (Anon 2001). Tcl provides a dramatically easier way to build integration applications ranging from simple graphical user interfaces to complex financial, Web, and management applications.

Since Tcl can be used for such a wide range of purposes user can now standardize on just one scripting language for all their needs. This is a great benefit by reducing the cost for organization since user only need to learn, maintain, and support one scripting solution as well as significantly improving the ability for all applications to integrate smoothly.

### 2.7.2.1 Tcl Capabilities

In contrast, Tcl provides a superb platform for creating integration applications. Tcl's capability comes from two basic features.

First, Tcl makes it easy to connect to any of the object that user need to integrate. If user need to connect any X to any Y, it is easy to create one Tcl extension that connects to X, another that connects to Y, and use Tcl as the intermediary between them. Dozens of free extensions are already available for database access, network management, and many other purposes.

Second, with Tcl it is easy to write scripts that manage the connections in powerful ways. In contrast to system programming languages, Tcl is interpreted and typeless. The interpreted nature of Tcl makes it easy to modify and extend applications on the fly and evolve them rapidly. By being typeless and string-oriented, Tcl hides the differences between components and makes it easy to move information between them.

The combination of these two features allows integration applications to be developed 5-10 times more efficiently with Tcl than with system programming languages such as C++ or Java, measured either in development time or in lines of code. Furthermore, the applications created with Tcl are more powerful and flexible.

There are many different reasons why people use Tcl but most of them fall into just a few categories. Here are the ten benefits of why people use Tcl today:

- Rapid development

  The most important reason why people use Tcl is that it gets their job done faster. The applications can be implemented five to ten times faster with Tcl than with other languages, especially if the application involves GUIs, string-handling, or integration. Once an application is built in Tcl, it can also be evolved rapidly to meet changing needs.

- Graphical user interfaces

  With its Tk toolkit, Tcl provides facilities for creating GUIs that are incredibly simple yet remarkably powerful. For example, the Tk canvas widget makes it easy to create displays with graphics, yet it also provides powerful facilities such as bindings and tags. The text widget provides sophisticated hypertext capabilities and more. No other toolkit has the same combination of simplicity

and power. Tcl attracted much of its early following because it was the only sane way to create user interfaces under Unix; now it provides these same benefits on Windows and Macintosh platforms too.

- Cross-platform applications

  Tcl runs on Windows (95 and NT), Macintosh, and nearly every imaginable Unix platform. This makes it an outstanding tool for creating cross-platform applications. For example, the same Tcl script can run on Unix, Windows, and Macintosh and display a graphical user interface; the GUI will have a different look and feel on each platform, to match the user's expectations for that platform. Because it runs on all major platforms Tcl provides an excellent management and integration tool for mixed environments, such as those with Windows desktops and Unix servers.

- Extensible applications

  If user want to create a powerful application that can be scripted and extended by its other users and modified in the field, user will need to include an interpreted scripting language in the application. Tcl is unmatched for this purpose. The Tcl interpreter was designed from the start to be embedded in a variety of applications. It is easy to incorporate Tcl into an application, and the Tcl Interpreter melds naturally with the application, almost as if the Tcl language were designed exclusively for that particular application.

- Flexible integration

  With Tcl it is easy to coordinate existing components and applications so that they work together effectively. For example, it is easy to use Tcl as a control language for special-purpose hardware and protocols, add a GUI or network interface to a legacy application, or integrate new Java applications with

legacy code in C or C++. This makes Tcl a powerful tool in areas such as network management and factory automation.Ready for the enterpriseWith the Tcl 8.1 release, Tcl became the first (and only) scripting language suitable for large server applications and other mission-critical enterprise uses. The benefits of scripting, such as rapid development, flexible evolution, and easy integration, have been known for years, but until Tcl 8.1 no scripting language provided all the facilities needed for enterprise applications, which include internationalization, thread safety, cross-platform portability, great GUI capabilities, embeddability, Internet support, and database access. Tcl 8.1 added internationalization and thread safety, making Tcl the first scripting language to meet all these requirements and bring the benefits of scripting to the enterprise.

- Testing

Tcl is an ideal language to use for automated hardware and software testing, and it may well be the dominant language used for this purpose. Tcl can easily connect to testing hardware or internal APIs of an application, invoke test functions, check the results, and report errors. Tcl's interpreted implementation allows tests to be created rapidly, and the tests can be saved as Tcl script files to reuse for regression testing. If you are testing a software application, Tcl allows you to connect directly to lower-level APIs within the application, which provides much more precise and complete testing.

- Easy to learn

  Tcl is a very simple language. Experienced programmers can learn Tcl and produce their first interesting application in just a few hours or days. Casual programmers can also learn Tcl quickly. Tcl is often used in situations where experienced programmers create a base set of facilities, and more casual programmers write Tcl scripts to customize those facilities, create business rules,etc.

- Network-aware applications

  Tcl provide easier acess to networking facilities. Servers and clients can be created in a few minutes with just a few lines of code. Tcl provides a great way to add network interfaces to legacy applications.

- The Tcl community

  Another attractive reason for using Tcl is the large and helpful community of Tcl users and developers. The Tcl community is a constant source of ideas, free extensions, applications, and technical support.

### 2.7.2.2 Other scripting language

Tcl is a member of the class of languages known as scripting languages. There are many other scripting languages besides Tcl, including JavaScript, Visual Basic, Perl, and others. As a group, all of the scripting languages tend to be used for integration applications, and all offer significant benefits over system programming languages.

Each scripting language has particular strengths. For example, JavaScript is known for its smooth integration with Web browsers, Visual Basic for its easy-to-learn development environment, and Perl for its string-handling capabilities. Tcl's greatest

Strength is its versatility: it can be embedded in applications or used standalone, it has outstanding GUI capabilities, and it can easily be connected to nearly any other application or protocol. Tcl was designed from the start to be used for many different purposes in many different situations, and the tremendous diversity of Tcl applications demonstrates that it has met this design goal.

In contrast, most other scripting languages were designed for a narrower set of tasks. They perform well for those specific tasks but they aren't used for as many different things as Tcl. For example, JavaScript is the obvious choice to use for simple scripting in a browser, but it is rarely used for anything outside the browser. Visual Basic provides excellent facilities for creating Windows GUIs, but it isn't suitable for integrating Windows desktops with Unix servers. Perl's string handling makes it an excellent choice for system administration tasks, report generation, and Web scripting, but it doesn't have native GUI capabilities and it isn't as easily embeddable as Tcl.

The scripting language comparison chart below gives an overview of the features available in each of the most popular scripting languages today.

| Features | | Tcl | Perl | Java Script | Visual Basic |
|---|---|---|---|---|---|
| Speed of use | Rapid development | ✓ | ✓ | ✓ | ✓ |
| | Flexible, rapid evolution | ✓ | ✓ | ✓ | ✓ |
| | Great regular expressions | ✓ | ✓ | | |
| Breadth of functionality | Easily extensible | ✓ | | | ✓ |
| | Embeddable | ✓ | | | |
| | Easy GUIs | ✓ | | | ✓ |
| | Internet and Web-enabled | ✓ | ✓ | ✓ | ✓ |
| Enterprise usage | Cross platform | ✓ | ✓ | ✓ | |
| | Internationalization support | ✓ | | ✓ | ✓ |
| | Thread safe | ✓ | | | ✓ |
| | Database access | ✓ | ✓ | ✓ | ✓ |

**Table 2.3: Comparison of today popular scripting languages**

## *2.8 Summary*

This chapter has covered the primary research background of this project and relevant knowledge needed to develop the network simulator. A more detailed explanation of the simulator will be presented in the following chapter.

# Chapter 3 Ns-2 and JaNetSim

## *3.1 Ns-2 Concept Overview*

Ns-2 is an Object-Oriented, discrete event network Simulator developed at UC Berkely. It is written in C++ and OTcl(Object-Oriented Tcl) and primarily uses OTcl as Command and Configuration Language. Ns is mailnly used for simulating local and wide area networks. It simulates a wide variety of IP networks.(Fall et al. 2003)

It implements network protocols such as TCP and UDP, traffic source behaviour such as FTP, Telnet, Web, CBR & VBR. router queue management mechanisms such as Drop Tail, RED and CBQ, routing algorithms such as Dijkstra and more. Ns also implements multicasting and some of the MAC layer protocols for LAN protocols for LAN simulations. The Ns project is now part of the VINT project that develops tools for Simulation results display, analysis & converters that convert n/w topologies generated by well-known generators to Ns formats.

### 3.1.1 OTcl Linkage

Originally, Ns is written in C++, with OTcl interpreter as a user front end. In C++, it supports a class hierarchy called Compiled hierarchy and in OTcl interpreter, the similar version are called interpret hierarchy. There is a one-one correspondence between classes of these two hierarchies. The root of the hierarchy is Class TclObject. User instantised objects are mirrored through methods defined in Class TclObject.

Users create new simulator objects through interpreter that are instantiated within the interpreter. The interpreted hierarchy is automatically established through methods defined in the TclClass.

| | |
|---|---|
| **TclObject** | Root of ns-2 object hierarchy |
| | bind(): link variable values between C++ and OTcl |
| | command(): link OTcl methods to C++ implementations |
| **TclClass** | Create and initialize TclObject's |
| **Tcl** | C++ methods to access Tcl interpreter |
| **TclCommand** | Standalone global commands |
| **EmbeddedTcl** | ns script initialization |

**Figure 3.1 : C++/OTcl Linkage**

### 3.1.2 Duality Need for Different Language

Ns developer can be considered working on Tcl, running simulations in Tcl using the simulator objects in OTcl library. The event scheduler & most of the components are implemented in C++ and available to OTcl through an OTcl linkage

C++

For detailed simulations of protocols, programming language like C++ can efficiently handles bytes, packet header and implements algorithm efficiently,

Tcl

In order to vary the parameter or configuration of changing scenarios, iteration time is more important than run-time of the part of task.Thus this can be accomplished by a scripting language like Tcl.

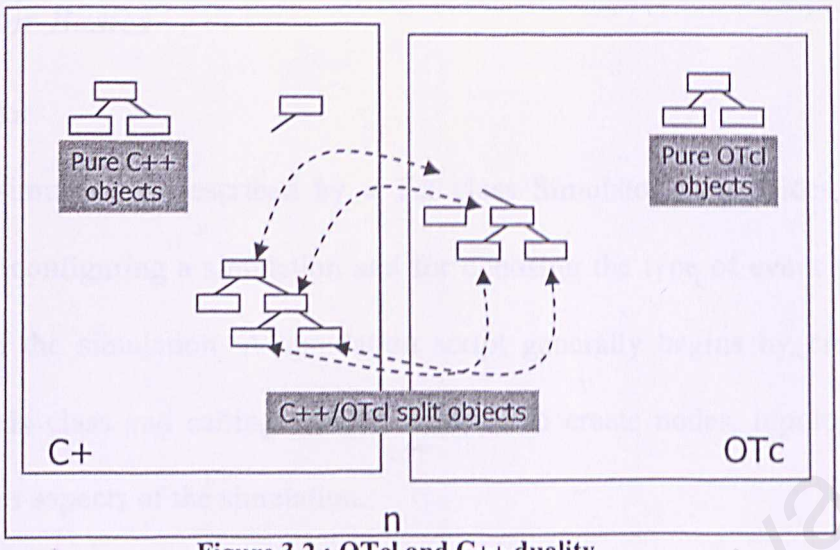



**Figure 3.2 : OTcl and C++ duality**

Ns is Object-Oriented TCl (OTcl) script interpreter that has a simulation event scheduler and network component object libraries and network setup modules called plumbing modules. The program that runs Ns is in OTcl script Language. The basic script sets up & runs a simulation of network. This initiates an event scheduler, sets up network topology using network objects and plumbing functions in the library and tells traffic source when to start & stop. When a user wants to make a new object, they can either write a new object from the scratch or make a compound object from the object library & plumb data through it.

Simulation results are usually contains in files called 'Trace files'. When the simulation is over Ns produces one or more text based output files that contain simulation data as specified in the input script. It can also be viewed using a nice graphical tool called 'Network Animator' or NAM in short.

## *3.2 Simulator Basics*

Class Simulator

The overall simulator is described by a Tcl class Simulator. It provides a set of interfaces for configuring a simulation and for choosing the type of event scheduler used to drive the simulation. A simulation script generally begins by creating an instance of this class and calling various methods to create nodes, topologies, and configure other aspects of the simulation.

### 3.2.1 Simulator Initialization

When a new simulation object is created in tcl, the initialization procedure performs the following operations:

- initialize the packet format (calls create_packetformat)
- create a scheduler (defaults to a calendar scheduler)
- create a "null agent" (a discard sink used in various places)

The packet format initialization sets up field offsets within packets used by the entire simulation.The scheduler runs the simulation in an event-driven manner and may be replaced by alternative schedulers which provide somewhat different The null agent is created with the following call:

```
set nullAgent_ [new Agent/Null]
```

This agent is generally useful as a sink for dropped packets or as a destination for packets that are not counted or recorded.

### 3.2.2 Schedulers and Events

The simulator is driven by event based activities. Firstly, the scheduler runs by selecting the following earliest event, executing it to completion and proceed with execution of the next event. The events are measured in seconds. Currently, the simulator is able to support only a single process of event execution at any given time. Presently, there are four types of schedulers defined in the simulator:

- Simple Linked List Scheduler

  The list scheduler (class Scheduler/List) implements the scheduler using a simple linked-list structure. The list is kept in time-order (earliest to latest), so event insertion and deletion require scanning the list to find the appropriate entry.Choosing the next event for execution requires trimming the first entry off the head of the list. This implementation preserves event execution in a FIFO manner for simultaneous events.

- Heap Scheduler

  The heap scheduler (class Scheduler/Heap) implements the scheduler using a heap structure. This structure is superior to the list structure for a large number of events. This implementation in *ns* v2 is borrowed from the MaRS-2.0 simulator; it is believed that MaRS itself borrowed the code fromNetSim, although this lineage has not been completely verified.

- Calendar Queue Scheduler

  The calendar queue scheduler (class Scheduler/Calendar) uses a data structure analogous to a one-year desk calendar, in which events on the same month/day of multiple years can be recorded in one day.

- Real Time Scheduler

  The real-time scheduler (class Scheduler/RealTime) attempts to synchronize the execution of events with real-time. It is currently implemented as a subclass of the list scheduler. The real-time capability in ns is still under development, but is used to introduce an *ns* simulated network into a real-world topology to experiment with easily-configured network topologies, cross-traffic, etc. This only works for relatively slow network traffic data rates, as the simulator must be able to keep pace with the real-world packet arrival rate, and this synchronization is not presently enforced.

### 3.2.3 Nodes and Packet Forwarding

Each simulation requires a single instance of the Simulator to control and operate that simulation. The class provides instance procedures to create and manage the topology, and internally stores references to each element of the topology. The basic primitive for creating a node is

```
set ns [new Simulator]        $ns node
```

The instance procedure node constructs a node out of more simple classifier. The Node itself is a standalone class in OTcl. However, most of the components of the node are themselves TclObjects. This simple structure consists of two TclObjects: an

address classifer (classifer_) and a port classifier (dmux_). The function of these classifiers is to distribute incoming packets to the correct agent or outgoing link. All nodes contain at least the following components:

An address or id_, monotonically increasing by 1, A list of neighbors (neighbor_), A list of agents (agent_), A node type identifier (nodetype_) and a routing module Nodes can be configured by the users themselves using one of the control functions, Address and port number management, routing functions, Agent Management and neighbor tracking functions. The function of a node when it receives a packet is to examine the packet's fields, usually its destination address, and on occasion, its source address. It should then map the values to an outgoing interface object that is the next downstream recipient of this packet. In this task is performed by a simple classifier object. Multiple classifier objects, each looking at a specific portion of the packet forward the packet through the node. A node uses many different types of classifiers for different purposes. A classifier provides a way to match a packet against some logical criteria and retrieve a reference to another simulation object based on the match results. Each classifier contains a table of simulation objects indexed by slot number. The job of a classifier is to determine the slot number associated with a received packet and forward that packet to the object referenced by that particular slot.

A node is essentially a collection of classifiers. The simplest node (unicast) contains only one address classifier and one port classifier. When one extends the functionality of the node, more classifiers are added into the base node, for instance, the multicast node. As more function blocks is added, and each of these blocks requires its own

classifier(s),it becomes important for the node to provide a uniform interface to organize these classifiers and to bridge these classifiers to the route computation blocks.

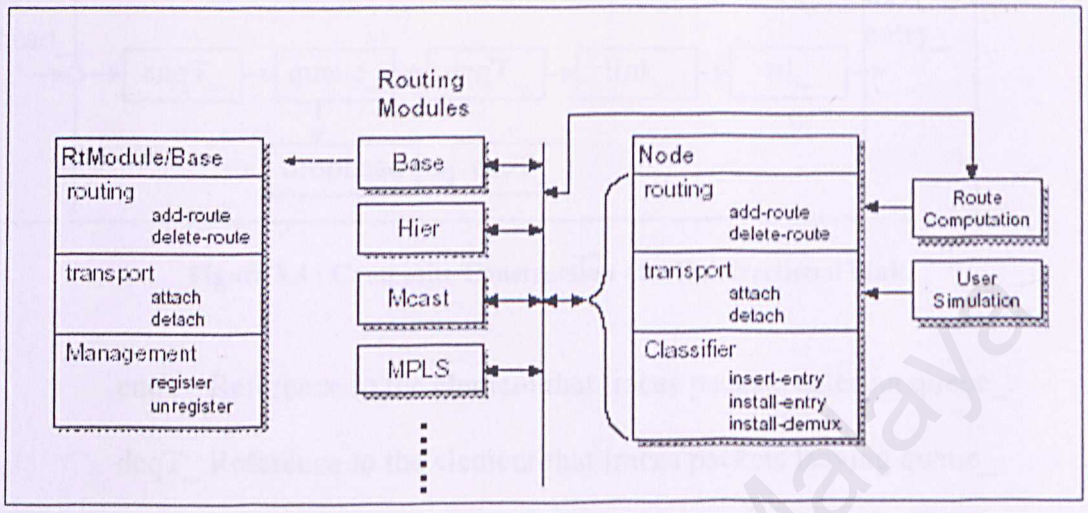



**Figure 3.3 : Interaction among node, routing module, and routing**

### 3.2.4 Links

This is the second aspect of defining the topology. Links are used to connect the nodes and complete the topology. Apart from simple point to point links, it supports a variety of other media, including an emulation of a multi-access LAN using a mesh of simple links, and other true simulation of wireless and broadcast media

set ns [new Simulator] $ns simplex-link {node0} {node1} {bandwidth} {delay} {queue_type}

The command creates a link from node0 to node1, with specified bandwidth and delay characteristics. The link uses a queue of type queue_type. The procedure also adds a TTL checker to the link. Five instance variables define the link- namely head, queue, link, ttl,drophead.

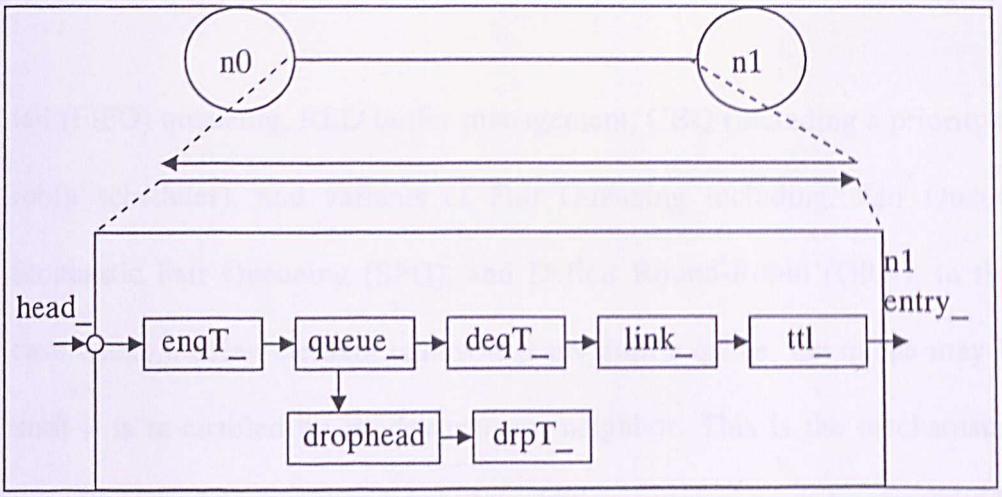



**Figure 3.4 : Composite Construction of a Unidirectional Link**

enqT_ Reference to the element that traces packets entering queue_.

deqT_ Reference to the element that traces packets leaving queue_.

drpT_ Reference to the element that traces packets dropped from queue_.

rcvT_ Reference to the element that traces packets received by the next node.

The instance variables enqT, deqT, drpT, rcvT track the trace elements. Delays represent the time required for a packet to traverse a link. A special form of this object (``dynamic link'') also captures the possibility of a link failure. The amount of time required for a packet to traverse a link is defined to be speed of the link in bits/sec, and is the link delay in seconds. The implementation of link delays is closely associated with the blocking procedures.

### 3.2.5 Queue Management and Packet Scheduling

Queues represent locations where packets may be held (or dropped). Packet scheduling refers to the decision process used to choose which packets should be serviced or dropped. Buffer management refers to any particular discipline used to regulate the occupancy of a particular queue. At present, support is included for drop-

tail (FIFO) queueing, RED buffer management, CBQ (including a priority and round-robin scheduler), and variants of Fair Queueing including, Fair Queueing (FQ), Stochastic Fair Queueing (SFQ), and Deficit Round-Robin (DRR). In the common case where a delay element is downstream from a queue, the queue may be blocked until it is re-enabled by its downstream neighbor. This is the mechanism by which transmission delay is simulated. In addition, queues may be forcibly blocked or unblocked at arbitrary times by their neighbors (which is used to implement multi-queue aggregate queues with inter-queue flow control). Packet drops are implemented in such a way that queues contain a ``drop destination''; that is, an object that receives all packets dropped by a queue. This can be useful to (for example) keep statistics on dropped packets. The Queue class is derived from a Connector base class. It provides a base class used by particular types of (derived) queue classes, as well as a call-back function to implement blocking.

### 3.2.6 Agents

Agents represent endpoints where network-layer packets are constructed or consumed, and are used in the implementation of protocols at various layers. The Agent has an implementation partly in OTcl and partly in C++. The C++ internal Agent includes enough internal state to assign various fields to a simulated packets before its sent. The state includes the following addr-the node address, dst-where pkts are sent to, size, type-the type of the packet, fid-the flow identifier, prio-the IP priority field, flags-packet flags, defttl-default ip TTL. Agent supports packet generation & reception. The common agent methods are meant to allocate packets, receiving the packets, specifying timeout methods.

## *3.3 NAM*

Nam is a Tcl/TK based animation tool for viewing network simulation traces and real world packet tracedata. It is is used to visualize the ns simulations and real world packet trace data. The design theory behind nam was to create an animator that is able to read large animation data sets and be extensible enough so that it could be used indifferent network visualization situations. Under this constraint nam was designed to read simple animation event commands from a large trace file. In order to handle large animtion data sets a minimum amount of information is kept .The first step to use nam is to produce the trace file. The trace file contains topology information, e.g., nodes, links, as well as packet traces.

Usually, the trace file is generated by ns. During an ns simulation, user can produce topology configurations, layout information, and packet traces using tracing events in Ns.

However any application can generate a nam trace file. When the trace file is generated, it is ready to be animated by nam. Upon startup, nam will read the tracefile, create topology, pop up a window, do layout if necessary, and then pause at time 0. Through its user interface, nam provides control over many

aspects of animation.

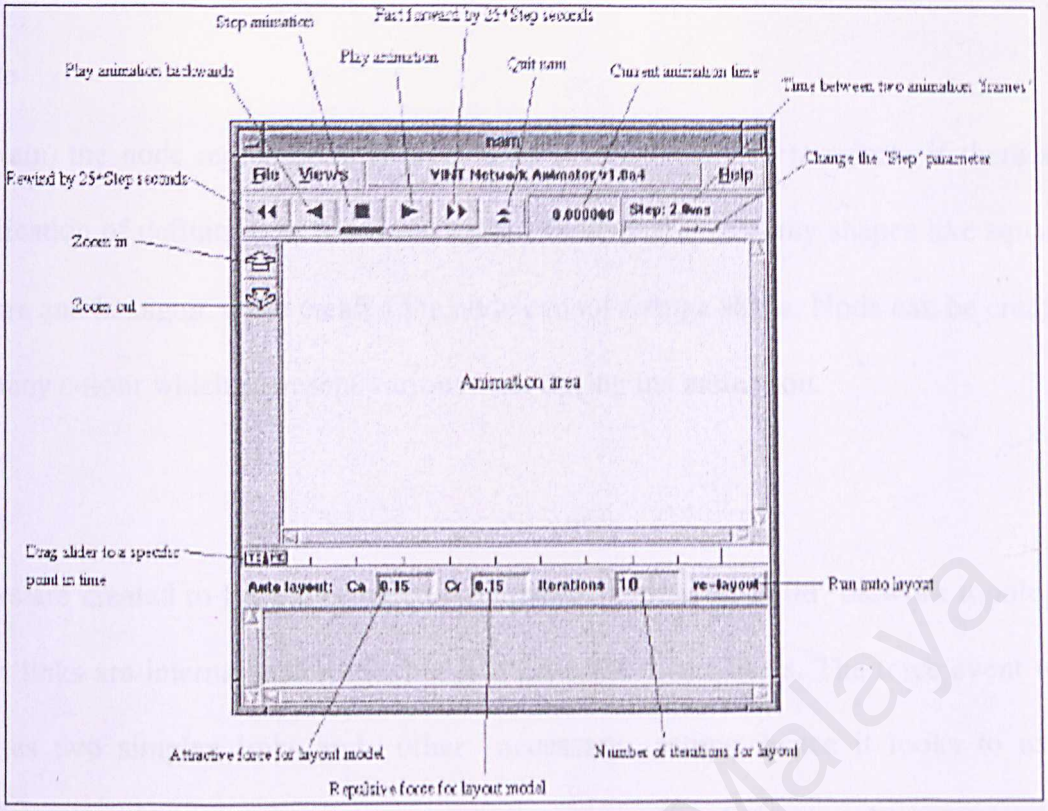



**Figure 3.5 : Screenshot of Nam interface**

### 3.3.1 Objects In Nam

In Nam the animation is supported by five different components. The building blocks in Nam are :

- Node
- Link
- Queue
- Packet
- Agent

Node

In Nam, the node represents a source/ host/ router. Nam will terminate if there are duplication of definition for the same node. Node may have many shapes like square, square and hexagon. Once created the node cannot change shape. Node can be created in many colour which represent various state during the animation.

Link

Links are created to form a connection between nodes to build network topology. Nam links are internally simplex, but it is invisible to the users. The trace event will creates two simplex links and other necessary setups, hence it looks to users identical to a duplex link. Link may be represented in different colors, to simulate flow of traffic during animation.

Queue

Queue needs to be constructed in Nam between two nodes. Unlike link, Nam queue is associated to a simplex link. Queues are visualized as stacks of packet that need to be transmitted between nodes. In trace events, parameters like angle between the line and horizontal line can be logged to trace events.

Packet

In Nam, packet is visualized as a block with an arrow. The direction of the arrow shows the flow direction of the packet. Queued packets are shown as little squares. A packet may be dropped from a queue or a link. Dropped packets are shown as rotating squares, and disappear at the end of the screen. Dropped packets are not visible during backward animation.

Agent

Agents are used to separate protocol states from nodes. They are always associated with nodes. An agent has a name, which is a unique identifier of the agent. It is shown as a square with its name inside, and a line link the square to its associated node.

## *3.4 Creating Topology*

In Ns-2, Tcl script is developed to simple toplogy. Tcl script defines the simulation scenario by including the topology and events.The script is able to create some output on stdout, write a trace file or start nam to visualize the animation .Below show the generic script structure in Tcl format.( Greis, Marc 2001 )

```
set ns [new Simulator]
# [Turn on tracing]
# Create topology
# Setup packet loss, link dynamics
# Create routing agents
# Create:
#  - multicast groups
#  - protocol agents
#  - application and/or setup traffic sources
# Post-processing procs
# Start simulation
```

First of all, create a simulator object. This is done with the command

```
set ns [new Simulator]
```

**Figure 3.6 : Basic topology script in Tcl (Part 1 of 5)**

Now open a file for writing that is going to be used for the nam trace data.

```
set nf [open out.nam w]
$ns namtrace-all $nf
```

**Figure 3.7 : Basic topology script in Tcl (Part 2 of 5)**

The first line opens the file 'out.nam' for writing and gives it the file handle 'nf'. The second line tell the simulator object created above to write all simulation data that is going to be relevant for nam into this file.

The next step is to add a 'finish' procedure that closes the trace file and starts nam.

```
proc finish {} {
    global ns nf
    $ns flush-trace
    close $nf
    exec nam out.nam &
    exit 0
}
```

**Figure 3.8 : Basic topology script in Tcl (Part 3 of 5)**

The next line tells the simulator object to execute the 'finish' procedure after 5.0 seconds of simulation time.

```
$ns at 5.0 "finish"
```

**Figure 3.9 : Basic topology script in Tcl (Part 4 of 5)**

The last line finally starts the simulation.

```
$ns run
```

**Figure 3.10 : Basic topology script in Tcl (Part 5 of 5)**

The segment of the above code will be use as a starting point to write a Tcl script for more complex example. If the coding above is execute, the system will prompt an error message 'nam : empty trace file out.nam 'because objects and events have not been defined yet

**Two nodes and one link**

The following two lines define the two nodes.

```
set n0 [$ns node]
set n1 [$ns node]
```

**Figure 3.11 : Simple script in Tcl (Part 1 of 3)**

A new node object is created with the command '$ns node'. The above code creates two nodes and assigns them to the handles 'n0' and 'n1'.

The next line connects the two nodes.

```
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
```

**Figure 3.12 : Simple script in Tcl (Part 2 of 3)**

This line tells the simulator object to connect the nodes n0 and n1 with a duplex link with the bandwidth 1Megabit, a delay of 10ms and a DropTail queue. Now the file can be save and the script can be started with command line 'ns example1.tcl'.Nam will be started automatically resembling the pictures below

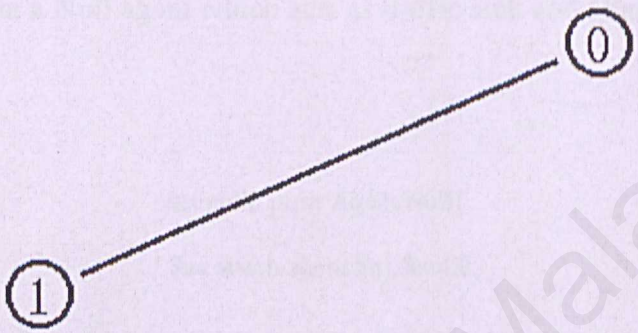



**Figure 3.13 : Simple script in Tcl (Part 3 of 3)**

**Sending data**

In Ns-2, data is always being sent from one 'agent' to another. So the next step is to create an agent object that sends data from node n0, and another agent object that receives the data on node n1.

```
#Create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0


# Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
```

**Figure 3.14 : Create data source between nodes (Part 1 of 5)**

These lines create a UDP agent and attach it to the node n0, then attach a CBR traffic generator to the UDP agent. CBR stands for 'constant bit rate'. The packetSize is being set to 500 bytes and a packet will be sent every 0.005 seconds (i.e. 200 packets per second).

The next lines create a Null agent which acts as traffic sink and attach it to node n1.

```
set null0 [new Agent/Null]
$ns attach-agent $n1 $null0
```

**Figure 3.15 : Create data source between nodes (Part 2 of 5)**

Now the two agents have to be connected with each other.

```
$ns connect $udp0 $null0
```

**Figure 3.16 : Create data source between nodes (Part 3 of 5)**

And now inform the CBR agent when to send data and when to stop sending. It's probably best to put the following lines just before the line '$ns at 5.0 "finish"'.

```
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
```

**Figure 3.17 : Create data source between nodes (Part 4 of 5)**

The file can be saved and restart the simulation again. When click on the 'play' button in the nam window, after 0.5 simulation seconds, node 0 starts sending data packets to node 1.

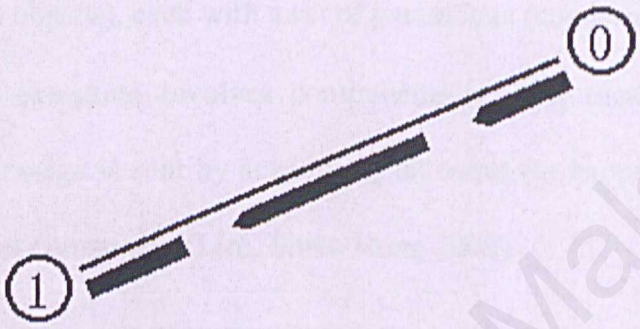



**Figure 3.18 : Create data source between nodes (Part 5 of 5)**

## *3.5 Java Network Simulator ( JaNetSim )*

The basic underlying concepts used of JaNetSim are:

- . discrete-event model
- . central simulation engine with a centralized event manager.
- simulation scenario consists of a finite number of interconnected components (simulation objects), each with a set of parameters (component properties).
- Simulation execution involves components sending messages among each other. A message is sent by scheduling an event (to happen some time later) for the target component (Lim, Shiau Hong 2002).

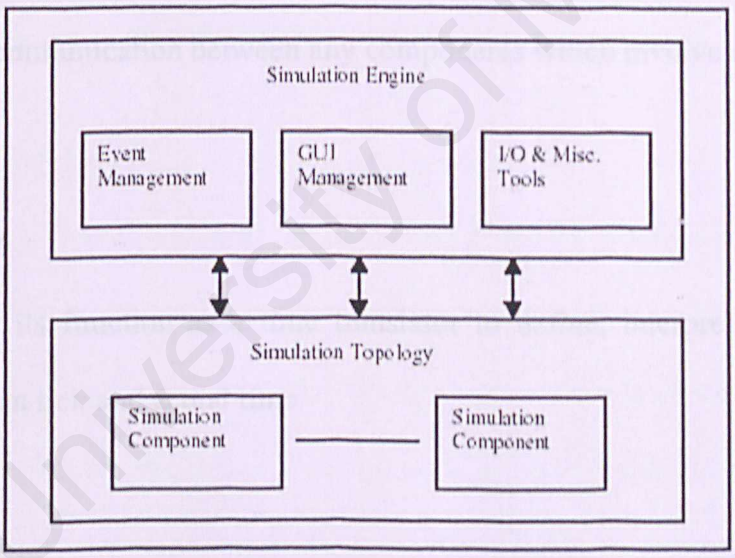



**Figure 3.19 : JaNetSim Overall Architecture**

The architecture of the simulator enables a wide simulation of various network components virtually possible by emulating transmission of data over the network These concepts are modeled from the NIST ATM/HFC Network Simulator.

### 3.5.1 JavaSim

JavaSim is the main object that contains all the functionalities in the simulator. It provides all GUI functions (together with SimPanel) and main JFrame for the application. Besides that, it provides the event manager to handle event-passing among all components. There will be only one instances of the JavaSim object throughout the simulation.

There are few services provided by the JavaSim object as below:

- Provide the current simulation time in tick
- Provide a list of all existing SimComponent
- Provide communication between any components which involve creation of a SimEvent

### 3.5.2 SimClock

This class serves its function as a time translator to define, interpret and provide conversion between tick and actual time

### 3.5.3 SimEvent

SimEvent caters for communications between components by using enquiring method. In order for communication to happen between source and destination ,the source component will create SimEvent and name the destination component as receiver and enqueue the event. The SimEvent will define time as one of its parameter to enable the source component to react to the event that had been targeted at that component

There are two types of events:

- Public Event
  - can be enqueued for itself or for another SimComponent
  - defined in the SimProvider object
  - development of new SimComponent and event types require the recompilation of this object
- Private Event
  - can only be enqueued for itself
  - private events are defined within the particular SimComponent source itself
  - private events must be greater than a constant (SimProvider.EV_PRIVATE) defined in SimProvider

### 3.5.4 SimComponent

This is the most important class to understand in the simulator in order to development new components. Every network component in the simulation inherits SimComponent. The SimComponent class itself should not be instantiated because it only provides the skeleton for an actual component. A new component should extends SimComponent and override its various methods in order to provide meaningful operations for the component.

Below are the methods available in the SimComponent:

| METHOD | DESCRIPTION |
|---|---|
| start() | perform any operation needed when the simulation starts |
| reset() | perform a reset operation in order to bring the status of the component back to the same status as if it is just newly created |
| action(SimEvent e) | this is the event handler of this component, and will be called by the simulator as the destination fires. Besides that, all private events will be handled in this method |
| isConnectable(SimComponent comp) | this is called by the simulation engine when a component is about to be connected to this component. The *comp* is a reference to the new component. |
| addNeighbor(SimComponent comp) | this is called by the simulation engine when a new neighbor is connected to this component. |
| removeNeighbor(SimComponent comp) | this is called by the simulation engine when a neighbor is disconnected from this component. |
| copy(SimComponent comp) | this method is used to copy parameter values of another SimComponent of the same type. |
| getImage() | this method is used to load an image file to represent the component in the simulator |

**Table 3.1 : Methods define in SimComponent**

Besides that, every SimComponent must have a component class and a component type, as defined in the SimProvider class. The *getCompClass()* method can be used to obtain the component class whereas *getCompType()* method can be used to get the component type.

### 3.5.5 SimParameter

Every SimComponent can have internal parameters or external parameters, which can be shown or accessible by users. All external parameters must inherit SimParameter. By extending SimParameter, one obtains parameter logging and meter display features automatically.

Any parameter that inherits SimParameter will provide a constructor that includes at least 4 parameters, which are name of the parameter, name of the component own the parameter, time when the parameter is created and whether the parameter can be logged in the log file.

The current simulation engine provides 3 general purpose classes that all inherit from SimParameter: *SimParamInt, SimParamDouble,* and *SimParamBool.* Obviously, these 3 objects provide support for integer, double and Boolean parameters. Extending SimParameter accordingly can create other types of parameters.

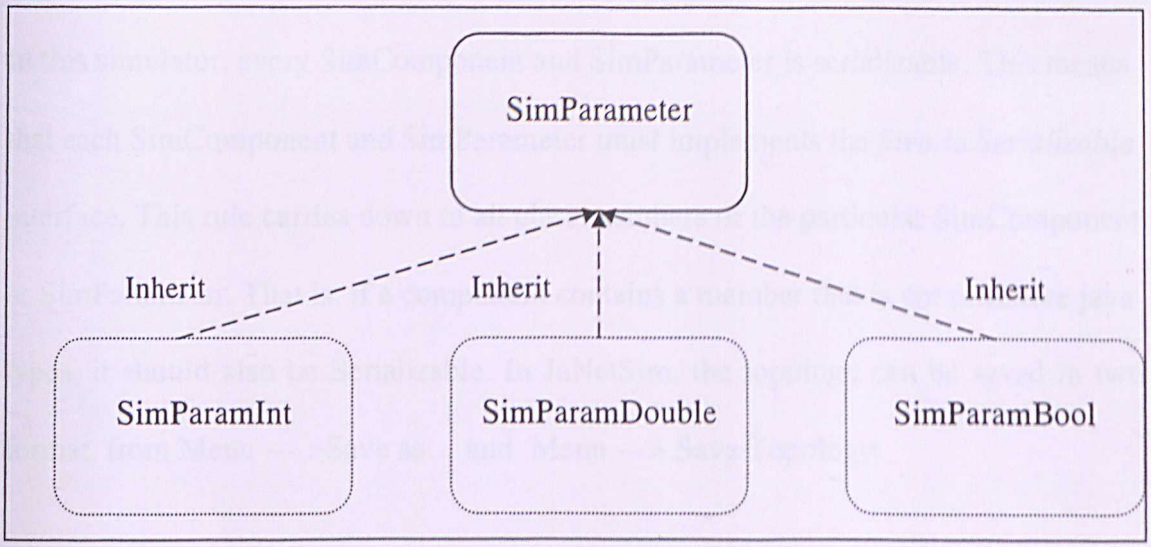



**Figure 3.20 : Inheritance from SimParameter class**

There is one important requirement to all parameters that may be added/ removed after the creation of a component. Any addition or removal of SimParameter from the component's parameter list *(java.util.List params)* should be followed by a notification call to the main JavaSim object, by this statement: *theSim.notifyParametersChange(this),* which ensure that any opened dialogs containing those parameters get updated/closed.

When a SimParameter is created with *isLoggable* true, it's value will not get logged when for example, a *setValue()* call is done. This is to avoid excessive or unnecessary logging of data. Each component is responsible in controlling the rate of logging. In order to make sure a new value of a SimParameter is logged, one must call the *update(long tick)* method.

## *3.6 Object Serialization and Load/Save Function*

The simulator uses object serialization as a form of light-weight persistence. This allows accurate saving and restoring of the simulation states without much effort from the components developers.

In this simulator, every SimComponent and SimParameter is serializable. This means that each SimComponent and SimParameter must implements the *java.io.Serializable* interface. This rule carries down to all class members of the particular SimComponent or SimParameter. That is, if a component contains a member that is not primitive java types, it should also be Serializable. In JaNetSim, the topology can be saved in two format, from Menu --- >Save as... and Menu ---> Save Topology.

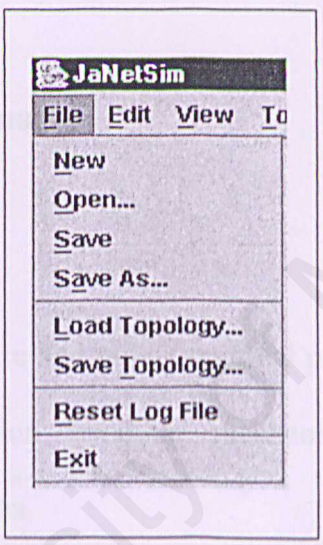



**Figure 3.21 : Screenshot of save format in JaNetSim**

From the first format, the layout of the topology will be saved with all the exact values in parameter at the time of saving. When the topology is reloaded again it will display the layout of the topology with the values logged by different devices before saving. If the user selects to save topology from the second method it will only save the layout of the topology without all the parameter. All the parameter from the previous simulation will be wiped off.

# Chapter 4 System Analysis

## *4.1 Development Tools*

The most suitable and appropriate tools for developing the system have been identified and selected. The tools have been selected include the development language as well as the entire platform on which the development of the project is developed.

### 4.1.1 Programming Language

#### 4.1.1.1 Java Programming

Java is a small, simple, dynamic and object oriented programming language coupled with strongly typed execution handling mechanism for writing distributed, dynamically extensible programs.

Java is object oriented programming language especially designed for use in internet environment. Object Oriented Programming technique use method that models the characteristics of abstract or real objects using classes and objects. Software objects have state and behavior as they are modeled after-real world objects. An object is a software bundle of variables and related methods. A software object maintains its state in one or more variables. A variable is an item of data named by an identifier. A software object implements its behavior with methods. A method is a function (subroutine) associated with an object.

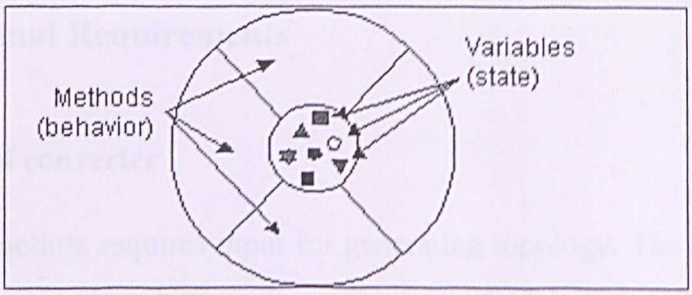



**Figure 4.1 : Common visual representation of a software object.**

#### 4.1.1.2 Tcl

One of the most powerful and versatile scripting language today for creating integration applications is Tcl. Tcl is a very simple scripting language made up of commands separated by new lines or semicolons.

Integration applications have characteristics quite different from traditional programming tasks. and often incorporate business rules and processes. Therefore, it tend to ill-structurized and evolve easily. Besides that, Tcl provide syntax that can be understand easily thus it is often popular with not sophisticated programmer.

## *4.2 System Requirements*

### 4.2.1 Functional Requirements

#### 4.2.1.1 Input of converter

The converter module requires input for generating topology. The input should be in topology format with extensions of *.tcl (Ns-2) and *top (JaNetSim).The topology should contain all the information of the network configuration to be simulated.

Basic and minimum information to be included into the topology file :

- Information of the network components like nodes and link
- Interconnection with neighboring components
- Values of parameters of each components

#### 4.2.1.2 Converter

The converter module is the core of the thesis. It is developed to enable conversion of topology format from Ns-2 to JaNetSim to simulate a network in different environment.

The converter should be able to:

- Read topology file format in *.tcl in Ns-2 and convert the topology to file format to *.top in JaNetSim and vice versa.
- Write to topology file format in *.tcl for Ns-2 and *.top for JaNetSim.

- Recreate the converted topology with minimum conversion including the parameters and other needed values as in the original topology file.
- Save the layout of topology in choice of file format
- Log all the values and parameters failed for conversion.

#### 4.2.1.3 Output

There should be two outputs by the end of conversion:

- The converted version of the topology file format for Ns-2 or JaNetSim in text file format
- A log file for informing the users of containing all the values of the data and parameters failed to be converted during the process

### 4.2.2 Non-Functional Requirements

Non-functional requirements are requirements which are not directly concerned with the specific function delivered by the system. Rather it may relate to the system properties or alternatively define the constraints of the system. Below are the functional requirements for the proposed system:

#### 4.2.2.1 Physical Environment

The converter will be developed in Windows environment by using Microsoft Window 98 with Intell Pentium III 800 Mhz processor and memory size of 128 MB.For testing purposes, *.tcl topology format have to be execute in Ns-2 which run on Linux Red Hat 9.0 under the same hardware configuration.

#### 4.2.2.2 Users and human factor

The user of the converter program should have at least understood the basic requirements in creating a topology to simulate a network .Users too should aware of all the parameters of the component used in the simulator. Users should understand about the relation of different network components and their interconnection to form a topology

#### 4.2.2.3 .Flexibility

The converter program should be able to change easily to support for changes and redesign purposes. Besides that it should be able to integrate without seamless into the existing simulator. The components and objects should be independent so that modification of the simulation will be fuss free and time saving.

#### 4.2.2.4 Usability

The program should be user friendly and user must be able to ues the system in shortest time. GUI interface of the system should accustom the user with sense of presence and familiarity with the Windows interface. The system functionalities must be self explaining and consistent with the design of the existing system.

#### 4.2.2.5 Maintainability

The program should be designed in a way that required less effort to maintain and robust. The task of locating and fixing an error must be simple and less troublesome.

#### 4.2.2.6 Constraint

As the conversion involves two topology formats for two different systems with slight different in components and connection method, not all the components can be mapped directly to each another. During the conversion some of the values and parameters may have to be alter to suit the best possible condition. All the values lost will be save into a log file for reference.

# Chapter 5 System Design

## *5.1 Introduction*

Design of the topology converter is based on the information collected during the system analysis and requirements stage. The topology converter is designed for conversion of topology file format in Ns-2 and JaNetSim system. The converter should be able to integrate with the existing system and comes with user friendly window based graphical user interface to reduce user time to learn the usage of topology converter..

## *5.2 Technique used*

There are two techniques involves in the design for the components. The first technique is modular decomposition and the second technique will be event oriented decomposition. Description of the both technique will be explained in details in the following segment.

### 5.2.1 Modular decomposition

For this technique, the system is constructed based on assigning functions to the components. Design of the system begins with a high-level description of the function to be implemented. From here, the details and relation of component organization will be produced.

### 5.2.2 Event-oriented decomposition

This approach allows the system design to be base on events that are handled by the system. Events are actions that prompt the system to perform some processing activities and several events will be discuss in the next part.

## *5.3 System Design*

The system design can be divided into three components, the input design, functionality design and output design

### 5.3.1 Input design

The converter program will take Ns-2 Tcl script and JaNetSim topology file as an input to the system. Basically both of the file will contain information of the topology including parameters and values of input and output of each component. The file also must contain the list of neighboring components and lists of routes in the class.

### 5.3.2.1 File reading

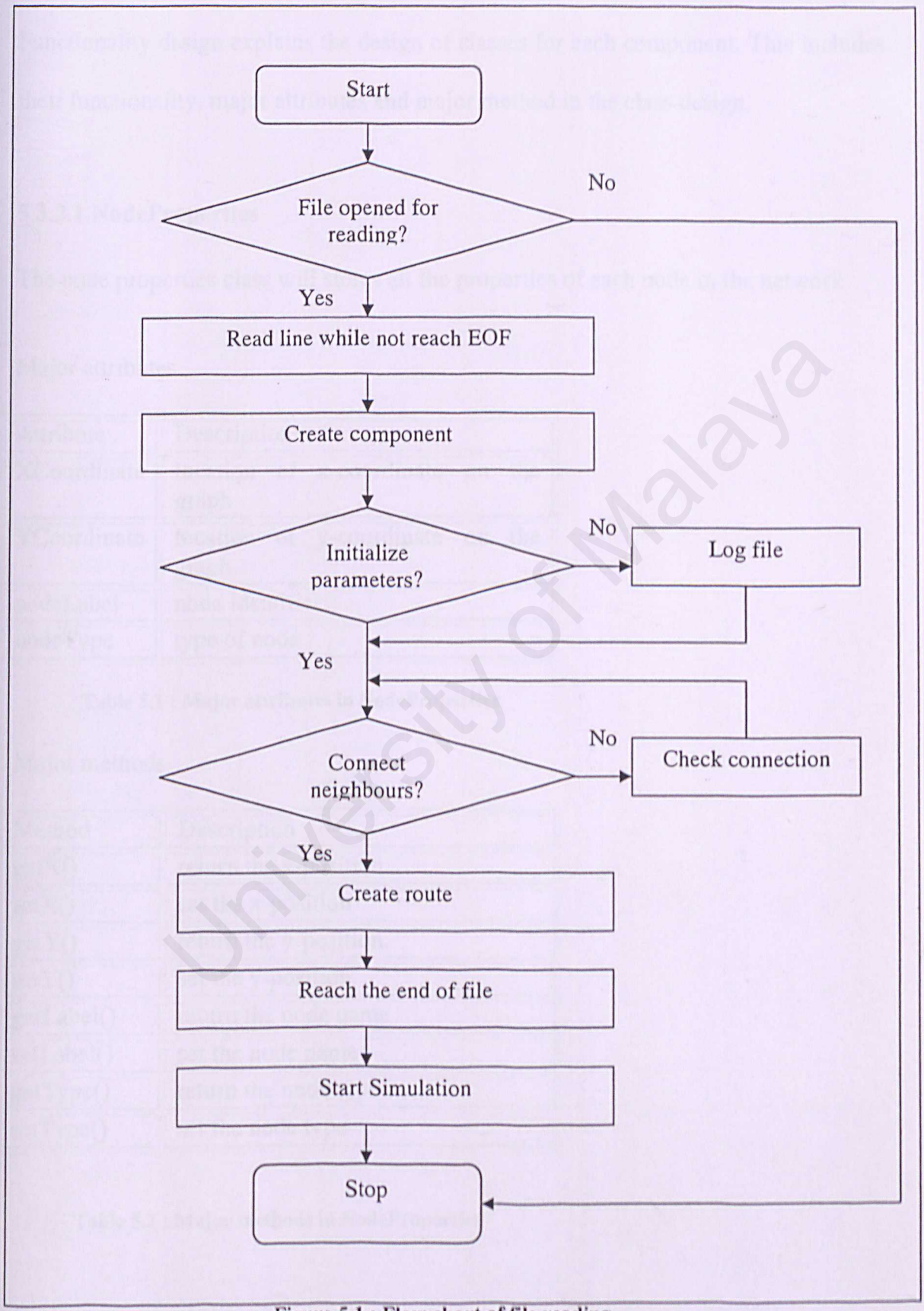



**Figure 5.1 : Flow chart of file reading**

## 5.3.2 Functionality Design

Functionality design explains the design of classes for each component. This includes their functionality, major attributes and major method in the class design.

### 5.3.2.1 NodeProperties

The node properties class will stores all the properties of each node in the network

Major attributes

| Attribute | Description |
|---|---|
| XCoordinate | location of x-coordinate on the graph |
| YCoordinate | location of y-coordinate on the graph |
| nodeLabel | node identifier |
| nodeType | type of node |

**Table 5.1 : Major attributes in NodeProperties**

Major methods

| Method | Description |
|---|---|
| getX() | return the x-position |
| setX() | set the x-position |
| getY() | return the y-position |
| setY() | set the y-position |
| getLabel() | return the node name |
| setLabel() | set the node name |
| getType() | return the node type |
| setType() | set the node type |

**Table 5.2 : Major methods in NodeProperties**

### 5.3.2.2 LinkProperties

Link properties will store all the properties of each node in the network

Major attributes

| Attribute | Description |
|---|---|
| distance | the distance between two nodes |
| bandwidth | the bandwidth of the link |
| delay | the delay between the link |
| label | link identifier |
| unidirectional | type of link, eiher simlpex or duplex |

**Table 5.3 : Major attributes in LinkProperties**

Major methods

| Method | Description |
|---|---|
| setBandwidth() | set the link bandwidth |
| getBandwidth() | return the value of bandwidth |
| setLabel() | set the link identifier |
| getLabel() | return the link identifier |
| getDirection() | get the line type |
| setDirection() | return 0 for simplex,1 for duplex |
| setDistance() | set the distance between node |
| getDistance() | return distance |
| setDelay | set the delay between nodes |
| getDelay | return the delay |

**Table 5.4 : Major methods in LinkProperties**

### 5.3.2.3 Node

The node class represents individual node on the network.The properties of the node can be invoke by calling node.getProperties().

Major attribute

| Attribute | Description |
|---|---|
| key | unique key identifier for individual node |

**Table 5.5 : Major attributes in Node**

Major method

| Method | Description |
|---|---|
| getKey() | return the key |
| setKey() | assign unique key value to node |

**Table 5.6 : Major methods in NodeProperties**

### 5.3.2.4 Link

Link class represents a link between nodes.The link of the properties cn be invoke by calling link.getProperties()

Major attributes

| Attribute | Description |
|---|---|
| fromNode | source node |
| toNode | destination node |

**Table 5.7 : Major attributes in Link**

Major methods

| Method | Description |
|---|---|
| fromNode() | return source node |
| toNode() | return destination node |
| turnaround() | connect the source and destination node |
| toString() | display the source and destination node |

**Table 5.8 : Major methods in Link**

#### 5.3.2.5 Converter

This class contains methods to perform the conversion method. The converter class will also be handling a set of nodes and links

| Method | Description |
|---|---|
| readNS2() | read a file in Tcl script |
| readSim() | read JanetSim topology format |
| fromNS2() | convert the Tcl script to JaNetSim topology file format |
| toNS2() | convert JanetSim topology file fomat to tcl script |

**Table 5.9 : Major methods in Converter**

#### 5.3.2.6 MainClass

This class will create a command line interface to the topology conversion functions for the converter.The main topology conversion functions are included in Converter class.

#### 5.3.2.7 ConverterException

This class will be handling exception for error that occur during the conversion stage

### 5.3.3 Output design

The output design of the system will be the converted topologies and the log file. The converter will record all the values of the parameter that fail to be converted for alerting the user in the log file.

### 5.3.3.1 File Writing

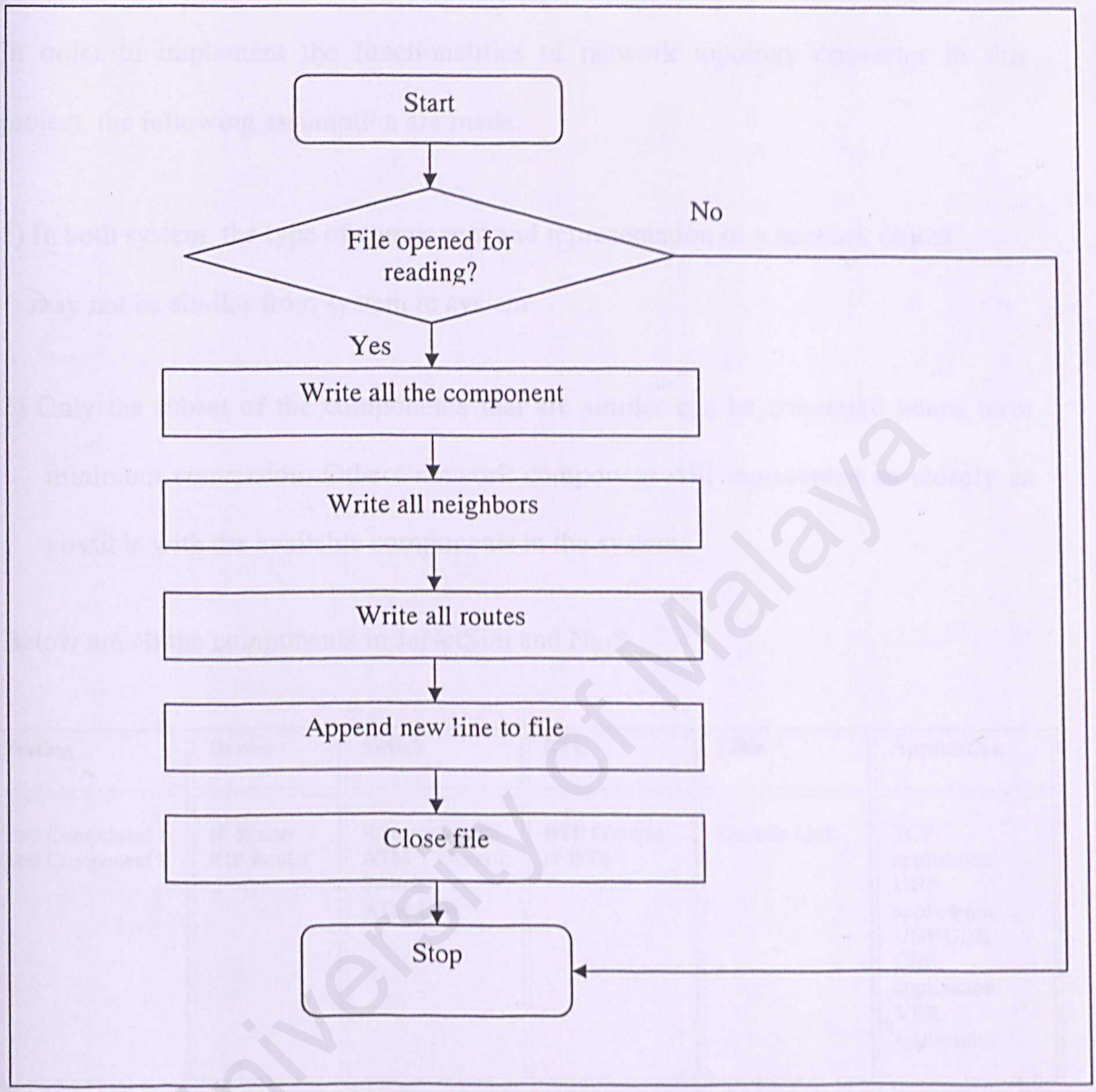



**Figure 5.2 : Flow chart of file writing**

## *5.4 Design Constraint*

In order to implement the functionalities of network topology converter in this project, the following assumption are made:

1) In both system, the type of component and representation of a network object may not be similar from system to system

2) Only the subset of the components that are similar can be converted hence term minimum conversion. Others network component will represented as closely as possible with the available components in the system.

Below are all the components in JaNetSim and Ns-2

| Testing | Router | Switch | BTE | Link | Application |
|---|---|---|---|---|---|
| Test Component 1<br>Test Component 2 | IP Router<br>RIP Router | EthernetSwitch<br>ATM Generic Switch<br>ATM LSR | BTE Generic<br>IP BTE | Generic Link | TCP application<br>UDP application<br>UDP CBR<br>CBR application<br>VBR application |

**Table 5.10 : Listing of all major component in JaNetSim**

| Agent | Traffic Source | Loss Model | Link | Queue Type |
|---|---|---|---|---|
| TCP<br>TCP/RENO<br>TCP/Vegas<br>TCP/Fack<br>UDP<br>---<br>Null<br>TCP Sink<br>TCP Sink / Del Ack<br>TCP Sink / Sack 1 | CBR<br>Exponential<br>FTP<br>Parento<br>Telnet | Periodic<br>Uniform | Generic Link | Drop Tail<br>Fair Queue<br>Stocastic Fair Queue<br>Deficit Round Robin<br>RED |

**Table 5.11 : Listing of all major component in Ns-2**

# Chapter 6 System Implementation

System implementation plays an important role of converting previously analysis, design and requirements into a real world system. The designs done in the earlier stages were meant to provide ease in combining the sub modules into a fully functioning system during implementation. Object oriented methodology is used at the implementation phase of the development process through the Java programming language.

## 6.1 JaNetSim to NS-2 conversion

The JaNetSim to NS-2 conversion will be implemented as the call function in the SimGui class. The action will be perform when the JaNetSim to NS-2 button is click by the user.The algorithm of the conversion function is shown below:

```
else if(cmd.equals("JaNetSIM--->Ns-2"))
{
        //open a save file dialog
        //read file
        //create component
        //create link
        //create agent
        //write to file
        //close file
}
```

Firstly, the converter will prompt a save dialog message to allow user to chose the file save location.

```
fileChooser=new JFileChooser(System.getProperty("user.dir"));
```

Then it will create components, links and agents for conversion by keeping all the essential parameters and the location coordinate.

```
SimComponent thiscomp=(SimComponent)components.get(i);
```

Next, all the component will be assign an index number and this number contain the component linking information. The linking information consists of the total neighbors and the index number of every component attached to its neighboring components.

```
for(int i=0;i<components.size();i++)
{
  SimComponent thiscomp=(SimComponent)components.get(i);
  SimComponent [] neighbors=thiscomp.getNeighbors();
}
```

Finally after writing all the topology information to the selected file, the file will be save to the desired location and close.

```
outfile.close();
```

A dialog box will be prompt to user to show the summary of the conversion including the number of nodes, links and TCP application successfully converted to NS topology file.

## 6.2 NS-2 to JaNetSim conversion

The NS-2 to JaNetSim conversion will be implemented as the call function in the SimGui class. The user will perform the action when the NS-2 to JaNetSim button is click. The algorithm of the conversion function is shown below:

```
else if(cmd.equals("Ns-2--->JaNetSIM"))
{
        //open a open file dialog
        //read file
        //set flag
        //create component
        //create agent
        // close open file dialog
        //assign link
        //open a save file dialog
        //write to file
        //close file
}
```

First, the converter will prompt a open dialog message to allow user to chose the file location for conversion.

```
fileChooser=new JFileChooser(System.getProperty("user.dir"));
```

As they is a slight variation in the NS topology file created in by NS-2 and the topology converter, the flag function will check the selected topology file and set the flag to value of 1 if the file is created by the topology converter. The flag is set to distinguish certain parts of the program to perform specific function according to the type of NS topology file.

```
if(aline.startsWith("## Generated"))
{
  flag=1;
}
```

Next, the file reader will store all the nodes and the agents information in an array to be retrieve in the later part. A new vector is declare to keep the information of the link in non-redundant form.

```
Vector all=new Vector();

 for(int i=0;i<node1.length;i++)
{
 if(!all.contains(node2[i]+","+node1[i]))
  all.add(node1[i]+","+node2[i]);
 }
```

After having all the important value store in an array, the file open for reading will be close.

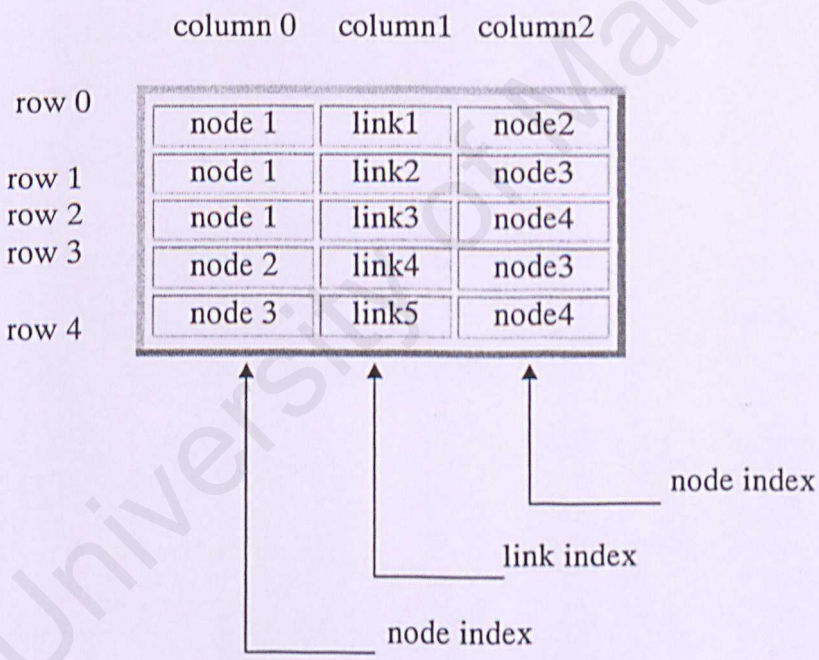
```
infile.close();
```

Then the program will be checking for extreme condition where the amount of node to be converted is less than zero and more than maximum of twenty. If the amount of node fail to comply with the allow number of nodes for conversion, an error message will be prompt to the user and the program will exit back to mainframe. Else if the conversion is successful, the following part will proceed.

A three dimension array will be create to store the information of the node number and assigning link index number to the respective node in an sequence where it read from.

```
for(int i=0;i<m/2;i++)
    {
        conn_matric[i][0]=array1[i]; //node1
        conn_matric[i][1]=continuenodenum; //linkindex
        conn_matric[i][2]=array2[i];//node2
        continuenodenum++;
    }
```

A typical size of the three dimension array is conn_matric[i][3] where the size of the row will be determine by the number of link and the column size is three to store the information of node1 index number, the link index number and node 2 index number. The array function will be illustrated by the sample below.




After the three dimension array is created the link information like the link index number and the coordinate will be saved into another separate array to be retrieve later. The node information will be sorted out in order in ascending order to facilitate easy retrieval.

```
Object[] linknode=node.toArray();
Arrays.sort(linknode);
```

Then a hash map function will sort all the value in the three-dimension array to get the following values

i) Total number of connection for the node

ii) Index number of the neighboring node

The value will be saved into another array to be printed out to the file later. After reading and storing all the necessary information in the buffer reader and reach end of file, another save dialog box will be prompted to user to choose the file save location.

**File theFile2=fileChooser.getSelectedFile();**

After that all the information will be write to an outfile and a dialog box will be prompt to user to show the summary of the conversion including the number of nodes, links and TCP application successfully converted to JaNetSim topology file.

# Chapter 7 System Testing

Testing is one of the critical phases in project development as it determines the final outcome of the system. Testing was conducted before and after the system implementation to detect pending errors. It represents the complete and extensive review and challenge on the design and coding specification. Testing also provides method to uncover logic errors and to test the system reliability. In this project, four error detection concepts were used to test the system:

- Error detection to help to identify errors by inspection, walkthrough or other type of errors .
- Error removal to debug and remove identified errors.
- Error tracking to find the cause of errors and fix the flaw
- Regression testing where the testing is conducted to find out whether the fixed error is working properly and the rework codes actually fixes the error or fixes it in one part and fails another part of the code.

## 7.1 Unit Testing

### 7.1.1 JaNetSim to Ns2 Unit Testing

#### 7.1.1.1 Normal Conversion

For conversion of JaNetSim to NS2, unit testing is done on the number of nodes and the TCP applications to determine the converted script contains the correct number of nodes/applications with the right connectivity in respective coordinates. To test the conversion , the JaNetSim topology file with 11 nodes is tested to determine the correctness of the functionality.

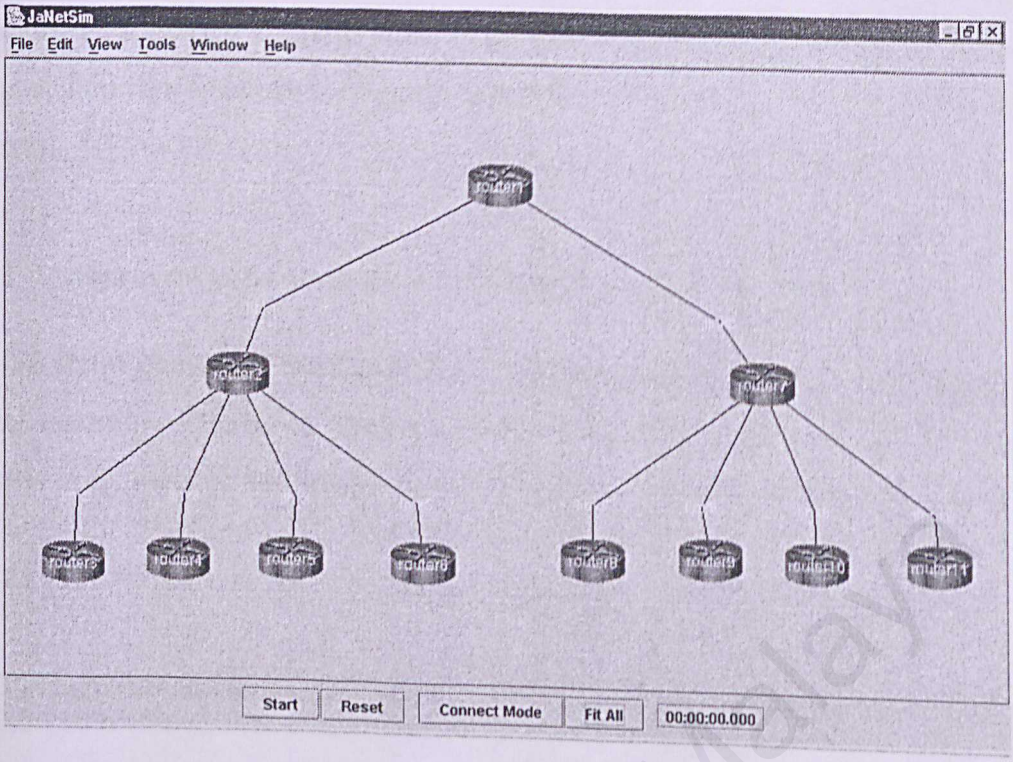



**Figure 7.1 : 11 nodes in JaNetSim**

The topology files with 11 nodes is converted to the NS2 topology file format. The converted file is open with the NAM editor for displaying the node to check for the number of converted nodes and the connection of between the nodes to ensure that the converter is working properly.

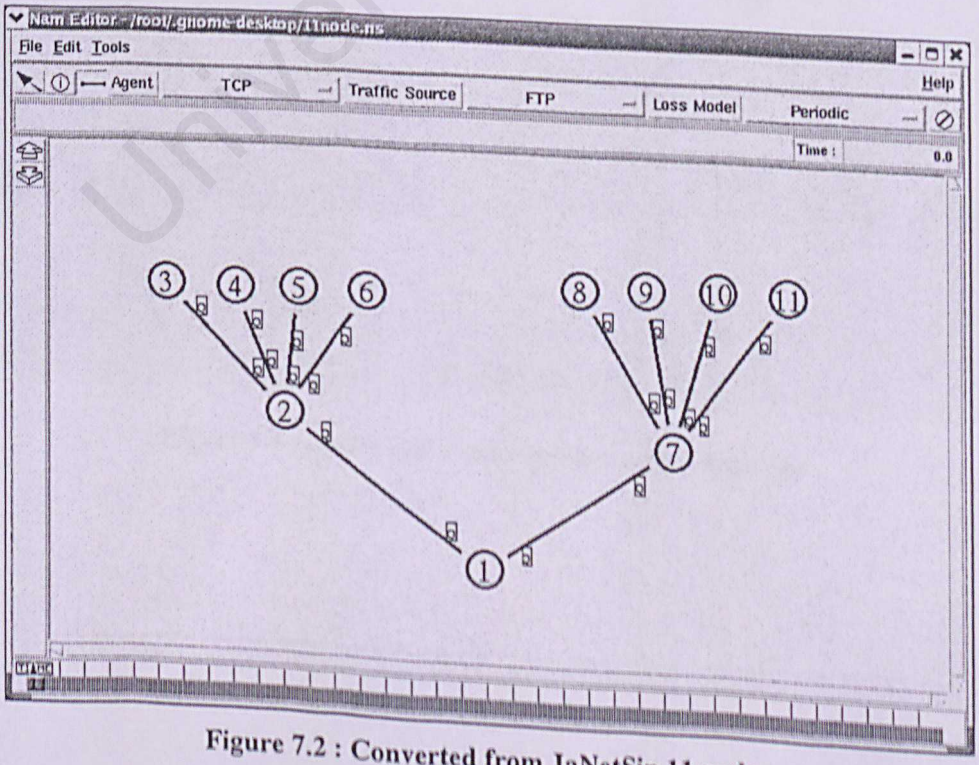



**Figure 7.2 : Converted from JaNetSin 11 nodes**

The converted topology contains exactly the same information of nodes number and the connection link between the nodes. Therefore, this proved that the conversion is successful.

### 7.1.1.2 Backward Conversion from NS2 topology to JaNetSim topology

The NS2 script converted from JaNetSim topology also can be converted back to its original topology. This is to ensure that the NS2 script generated by the topology converter is compatible and works equally as good as the original NS2 script.

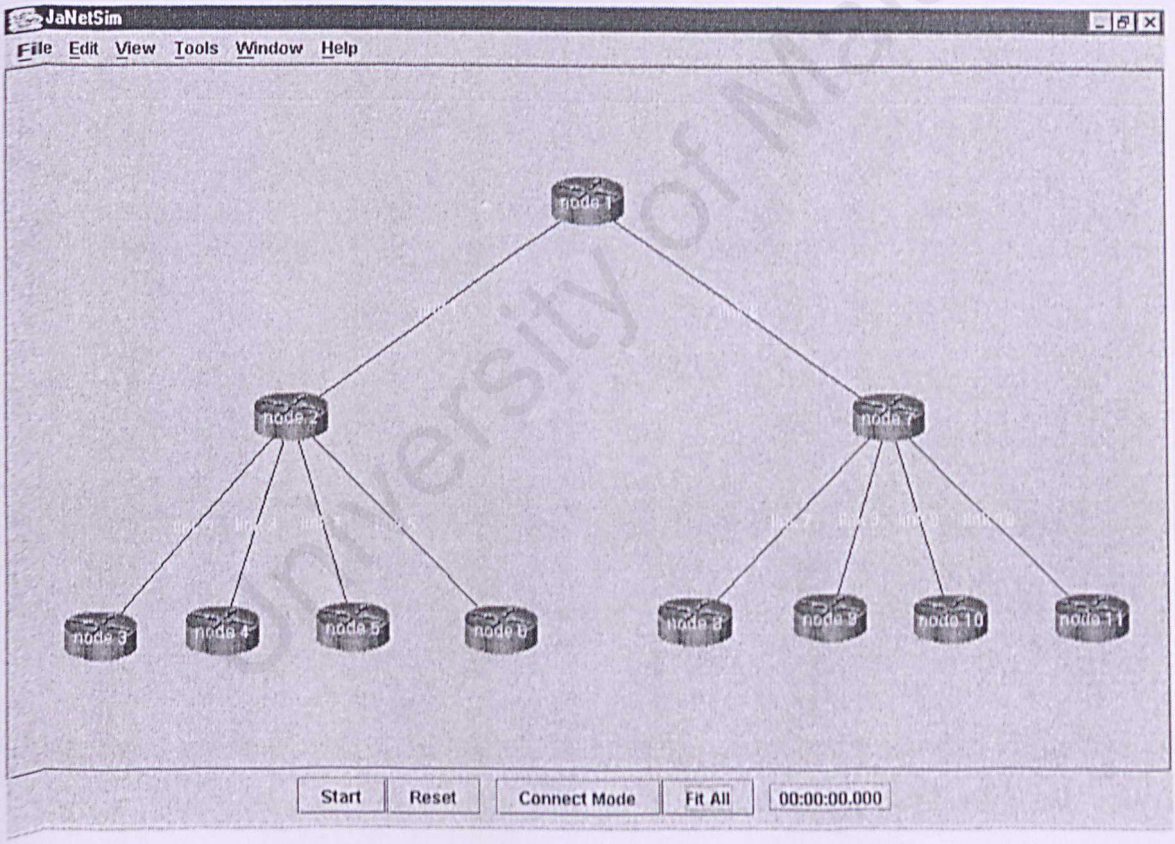



Figure 7.3 : Backward conversion from NS-2 topology

### 7.1.1.3 Conversion with TCP Application

The conversion of JaNetSim to NS2 is further tested by attaching TCP application to the node to be converted. The test case for this conversion contains four TCP application attached to 11 nodes to determine the correctness of the functionality.

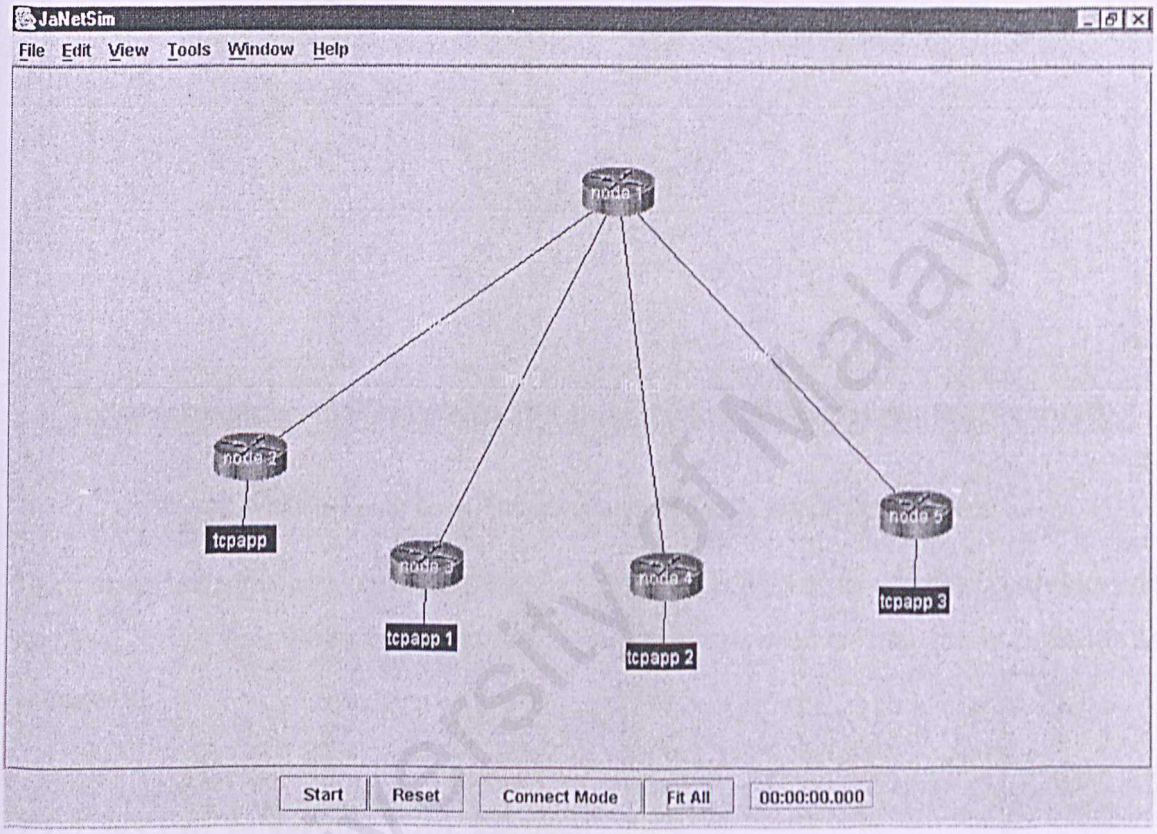



**Figure 7.4 : 5 nodes with 4 TCP application in JaNetSim**

The topology files with 4 TCPs and 11 nodes is converted to the NS2 topology file format. The converted file is open with the NAM editor for displaying the node to check for the number of converted nodes and the connection of between the nodes to ensure that the converter is working properly.

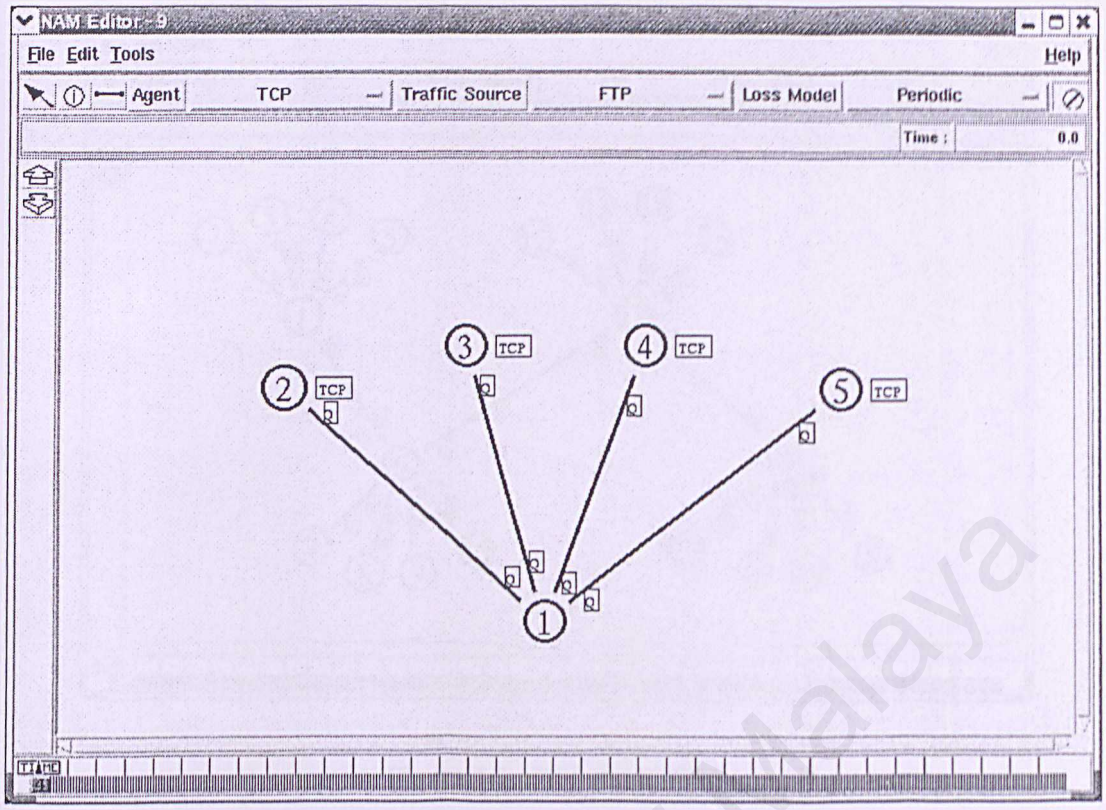



**Figure 7.6 : Converted from JaNetSim with 5 nodes with 4 TCP application**

The converted topology contains exactly the same information of nodes number and the connection link between the nodes. Therefore, this proved that the conversion is successful.

## 7.1.2 NS2 to JaNetSim Unit Testing

### 7.1.2.1 Normal Conversion

For conversion of NS2 to JaNetSim, unit testing is done on the number of nodes and the TCP applications to determine the converted script contains the correct number of nodes/applications with the right connectivity in respective coordinates. To test the conversion , the NS2 topology file with 20 nodes is tested to determine the correctness of the functionality.

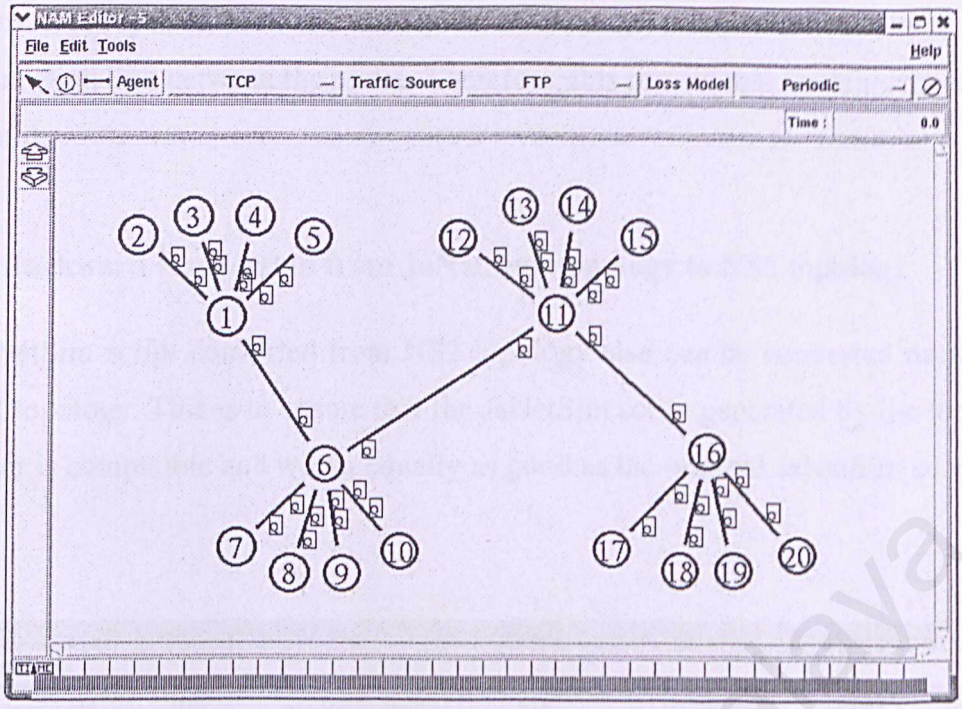



**Figure 7.7 : 20 nodes in NS-2**

The topology files with 20 nodes is converted to the JanetSim topology file format. The converted file is open with the JanetSim for displaying the node to check for the number of converted nodes and the connection of between the nodes to ensure that the converter is working properly.

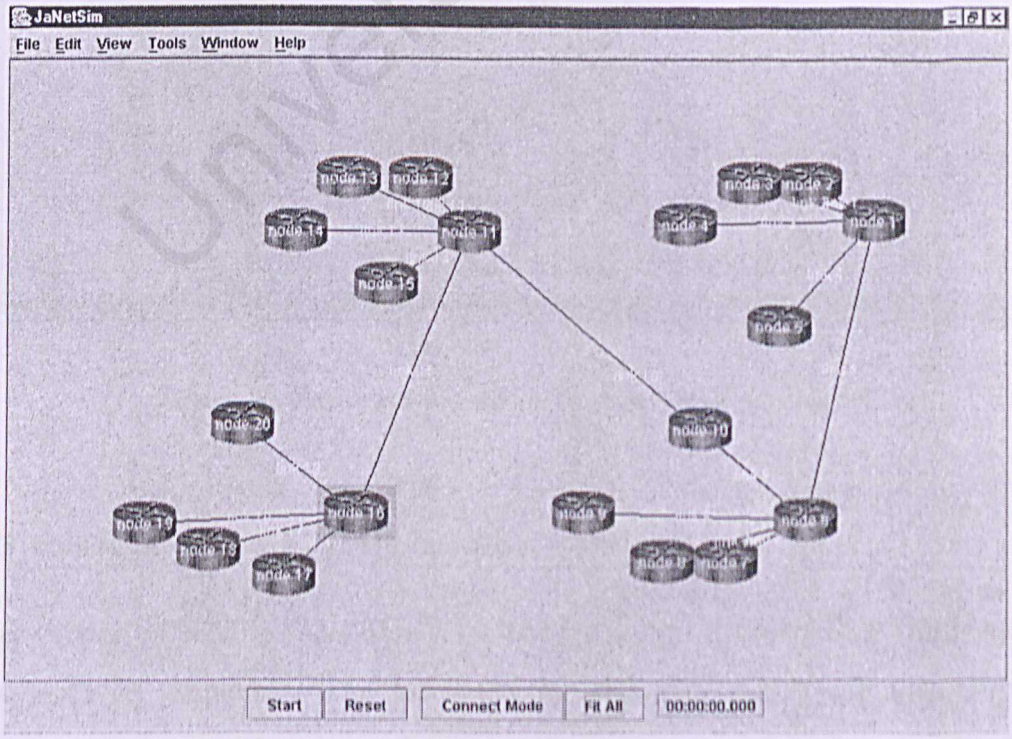



**Figure 7.8 : Converted from NS-2 with 20 nodes**

The converted topology contains exactly the same information of nodes number and the connection link between the nodes. Therefore, this proved that the conversion is successful.

### 7.1.2.2 Backward Conversion from JaNetSim topology to NS2 topology

The JaNetSim script converted from NS2 topology also can be converted back to its original topology. This is to ensure that the JaNetSim script generated by the topology converter is compatible and works equally as good as the original JaNetSim script.

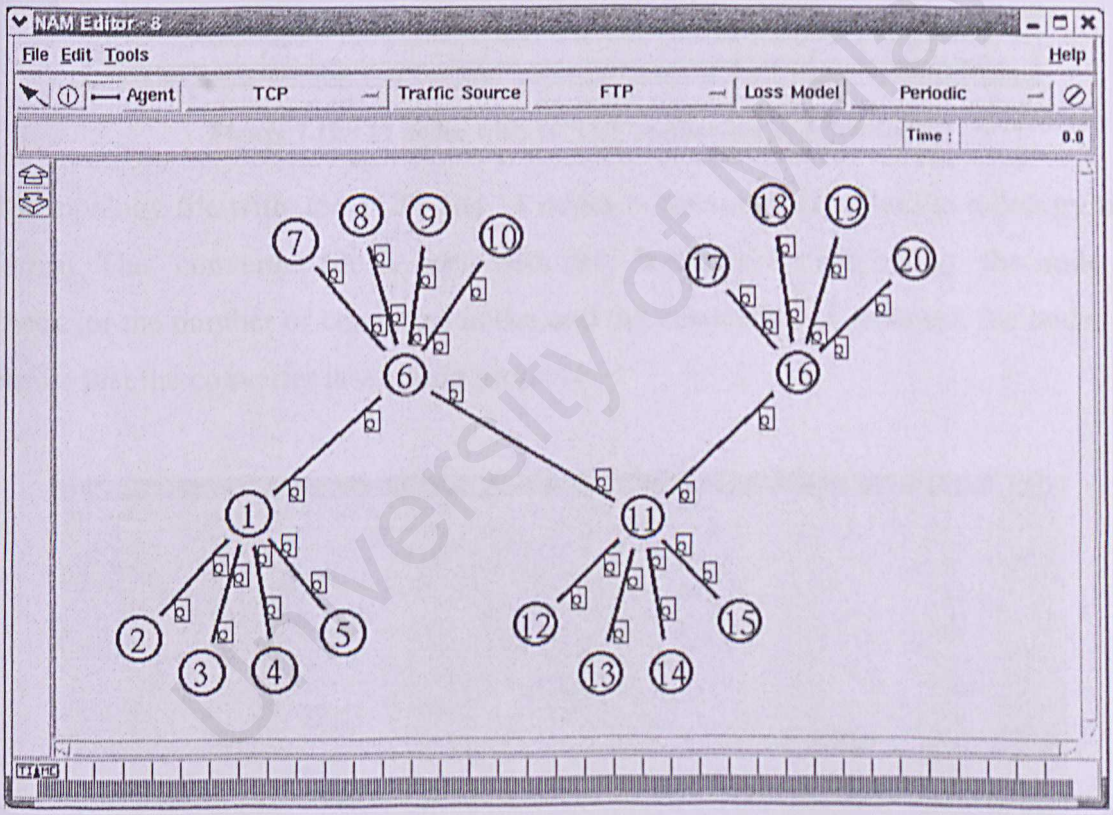



**Figure 7.9: Backward conversion from JaNetSim topology**

### 7.1.2.3 Conversion with TCP Application

The conversion of NS2 to JaNetSim is further tested by attaching TCP application to the node to be converted. The test case for this conversion contains ten TCP application attached to 11 nodes to determine the correctness of the functionality.

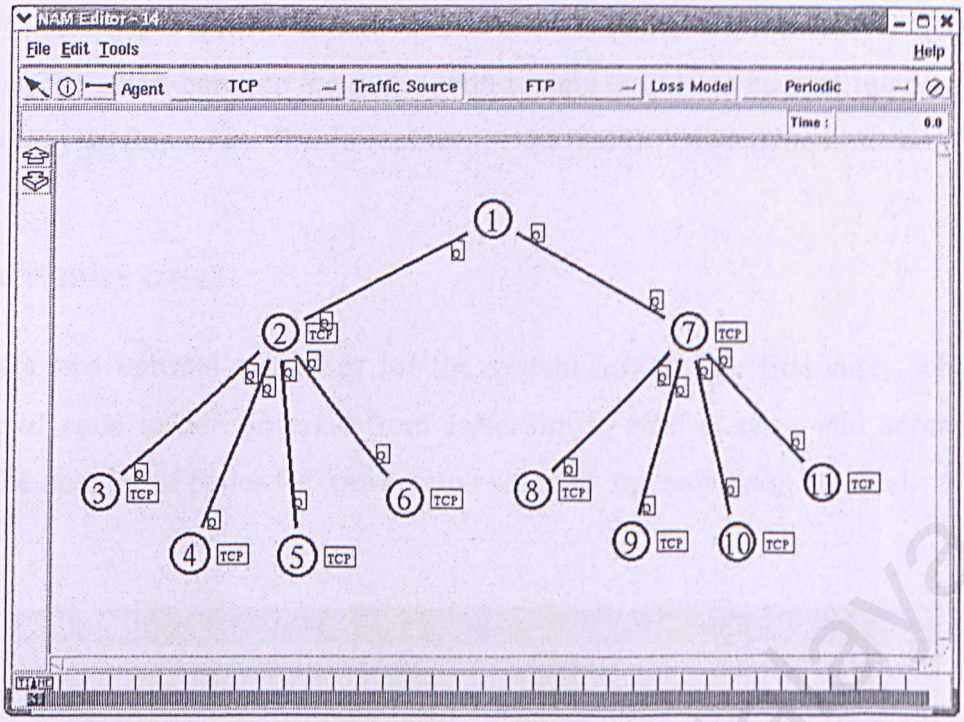



**Figure 7.10 : 11 nodes with 10 TCP application in JaNetSim**

The topology file with ten TCPs and 11 nodes is converted to JaNetSim topology file format. The converted file is open with the JanetSim for displaying the node to check for the number of converted nodes and the connection of between the nodes to ensure that the converter is working properly.

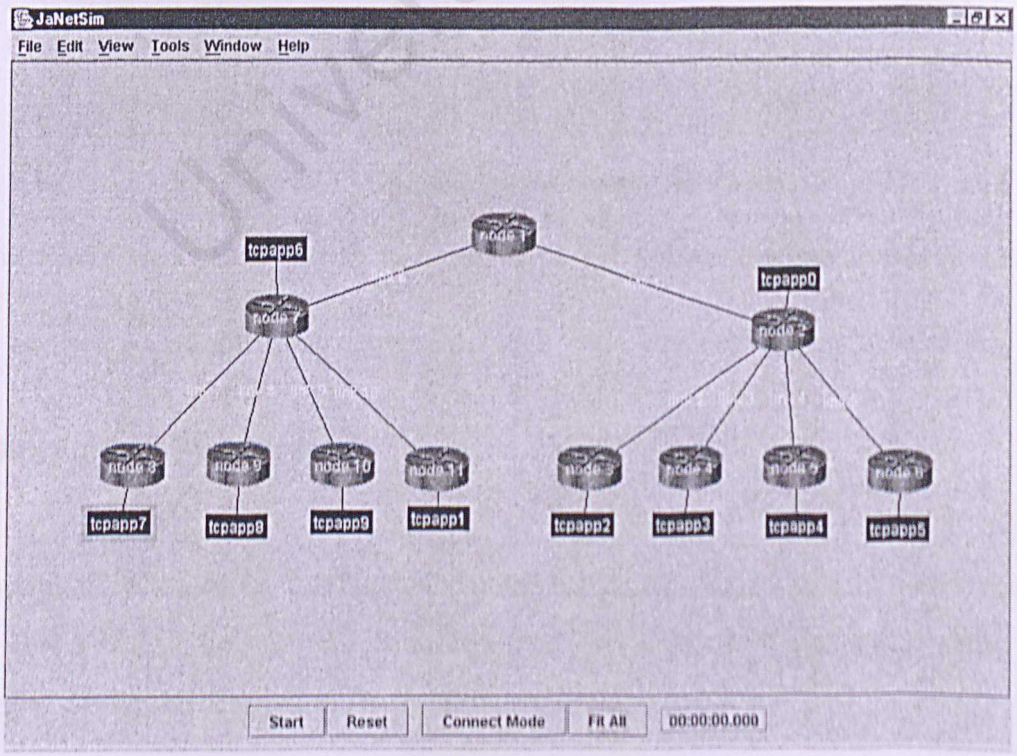



**Figure 7.11 : Converted from JaNetSim with 5 nodes with 4 TCP application**

The converted topology contains exactly the same information of nodes number and the connection link between the nodes with certain degree of manual intervention in the topology arrangement. Therefore, this proved that the conversion is successful.

## 7.2 Extremity cases

There are two unusual test cases for the system testing, the first case where the number of node to be converted from JaNetSim to NS2 is zero and another case where the number of nodes for conversion exceeded the maximum allowed.

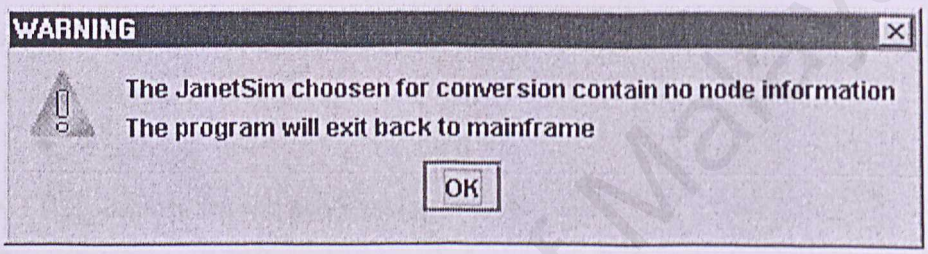



Figure 7.12 : Number of node is zero

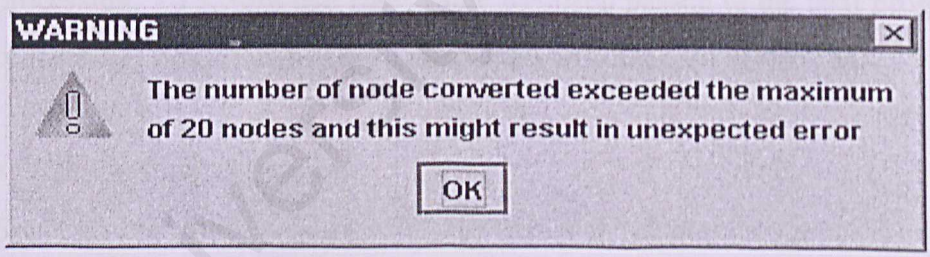



Figure 7.13 : Number of node exceed maximum

The successfulness of the testing to detect the error proved that the simulation model could simulate in a proper manner.

## 7.3 Debugging Strategies

Debugging is actually of finding and fixing the errors. There are various types of errors that exist in the system; compile error, run time error and logic error. The debugging strategies applied in the system are listed as below:

- Built-in Error Detection

Java also has built-in error detection. If an error found during application execution, an error message together with the lines number where the error occurred will be debugged. With this features, the debugging work becomes much easier and faster.

- Reviewing the Algorithm Used

If a program is running well, but the information is not what as intended, then may be a logic error or database error have occurred. Reviewing algorithm and computations for their correctness and efficiency is needed for this purpose. Sometimes, by using different algorithms, the efficiency of the program will be increasing.

- Display on Screen the Passing Value

One of the possibilities of wrong information being retrieved is that the wrong value is being passed from one page of another page that will do the processing. To ensure that right value has been passed to the next program for processing, the passing value is displayed on screen for reviewing.

- Check Success Status

Some processes are dependent where failure in the previous process will affect many other processes. In order to avoid chain reaction from this kind of process, a success status is purposely set to return a true or false value. The success status is checked to determine whether to continue process or to exit from the program and display error message.

## 7.4 Chapter Summary

Testing is one of the important steps in developing a system precision and accuracy of output data is considered during this process. Unit testing has been carried out for the developed system. The objective of a system will only achieve after all the thorough testing done by different user with different aspects.

At the end of the testing phase, the system should be able to perform the tasks required. The system should be ready to use by the users. However, some critical problems and errors will occurred only after some time of the using system. Therefore, testing should not just end up in this phase but have to keep on consistently to make sure the system is functioning well.

The following chapter presents the system evaluation. The evaluation reveals the problem encountered and solutions, system strengths and system constraints and future enhancements.

# Chapter 8 System Evaluation

## 8.1 Introduction

System evaluation is a process of that occurs continuously, drawing on a variety of sources and information. Generally, many technical and non-technical problems were encountered during the development stage. However, most of the problems were detected and resolved eventually but some are not.

The role of the evaluation phase was to determine :
The extent to which the expected outcomes have been realized
The prescription value of the process where extraneous factors were taken into consideration

Besides, the system strengths and system constraints have been list out as detail as possible in the chapter. So that any weakness of the system can be improve in the future enhancements.

## 8.2 Problems Encountered and Solutions

**Lack of experience in programming Language**

The development of the Topology Converter involves mainly on Java programming language. There are times when I am facing difficulties in translating my idea and certain algorithm to Java language. In order to cope with this problems, I would refer to various sources like the internet and e-books and having a group discussion to share and to solve the problems.

**Difficulties in determining the system scope**

When first starting the implementation of the project, the scope or boundaries of the system are still unclear and hard to determine. Many problems were face in designing an algorithm and coding because the system is not working very well when the number of nodes are too large plus the types of application to be converted. Eventually, this problem was solved through discussion with the lecturer and a agreement on the system scope was agreed.

### Problems in Development Environment

Developing the topology converter to be compatible with NS2 system on Linux environment unfold a new challenge as the unfamiliarity with the new environment had somehow causing the initial progress of the development to be stagnant as a lot of testing and effort are needed to familiarize with the system in order to start the development of the project. The problem was solves by intensive testing on the system and through discussion with experienced seniors.

## 8.3 System Strength

### Platform Independent

The topology converter is developed by using Java programming language and it is cross platform. Thus, the system is able to works well in both Windows and Linux environment.

### System Transparency

System transparency refers to the condition where the users do not need to know the file structuring of the system and the mechanism of conversion. The user only need to convert the file to the required format and load the topology with in the right simulator.

## Consistency

The screen design is consistent throughout the whole system. The menus are always displayed at the same position although the user switched from one module to another. Users can easily seek for a particular option that they require in the system.

### Flexibility in conversion method selection

The topology converter enable user to choose either converting the topology script from JanetSim to NS2 or vice versa to conduct research on the both simulator concurrently.

### Ease of use

The converter is built in such way that the user do not feel the complexity of the system. They only need to choose an appropriate script for conversion and the save destination and with a click of button within seconds, the converted file will be save to desire location.

## 8.4 System Constraints

### Rare inconsistency

The constraint of the system is the need of some manual intervention to rearrange the topology when the number of nodes for conversion increase. Besides that, there will be some rare cases of inaccuracy in the conversion where tcp component are involved.

## 8.5 Future Enhancements

### Support for node naming

The topology converter only support labeling of the nodes in numeric format. The future version should be able to support labeling of nodes by alphanumeric format.

### Script viewer

A new window is needed to display the selected script to ensure that the correct sciprt is chosen for conversion.

## 8.6 Knowledge and Experience Gained

By developing the system, personally I feel that I have learned a lot of things, which I have never, knew or realized before this. Some of the knowledge and experience gained are listed below.

### Self Expression

Developing the topology converter has really given me a great chance to express my own opinions and ideas in designing and coding of the system. Involvement and experiences gained during system development has greatly improved my self-confidence and self-esteem.

### Project Planning Skills

System development steps, stages and planning are just a theory before I developed the system. But during the development of this system, I actually have the chance to put into practice all the knowledge and theory about system development and planning.

### Development Tools Knowledge

Developing the system has given me the opportunity to explore the advance features of Java programming language, running JaNetSim and NS2 network simulator and using Linux system. By developing this system, I have discovered more practical knowledge and firsthand experience on those system rather than reading about the theory from book.s.

## 8.6 Reviews on Goal

At the final stage of the project, there were certain expectations on what would be achieved. The following is the expectations that were achieved:

### Expectation Achieved

In overall, the system had fulfilled the expectations stated by the project. The basic foundation of the system was designed and implemented. It was eligible for future growth and implementation.

### Objectives Achieved

The project had successfully created a converter that supported conversion of topology script between two different systems. It could be deduced that the objectives to establish the application had been achieved.

## 8.7 Summary

This project had managed to achieve the overall objectives and requirements determined during the system analysis. The testing phase has proved that the project is implemented successfully. Huge efforts, analytical thinking and endurance to time pressure are what it takes to bring the completion of the project. Overall experiences gained are memorable and meaningful.

However, there are still many rooms for improvements for the system. I hope that the system will provide a good foundation and open up more opportunities for research and improvement on the topology converter in the near future.

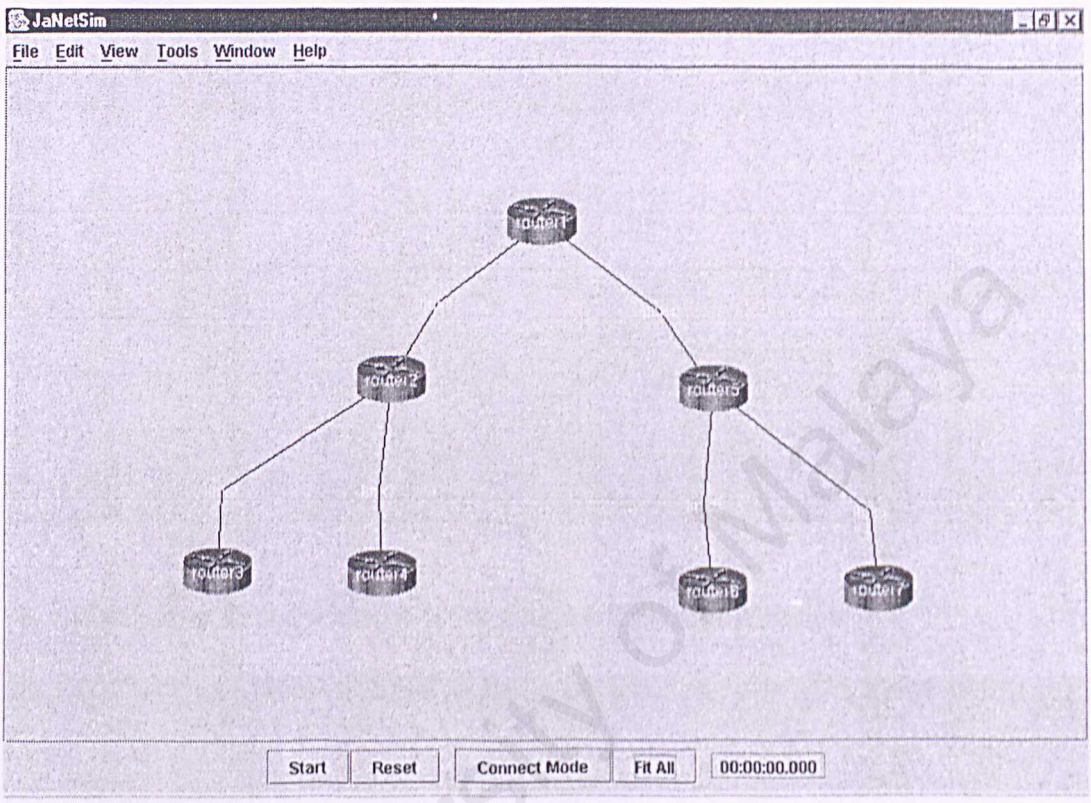
# APPENDIX

# User Manual on Topology Converter

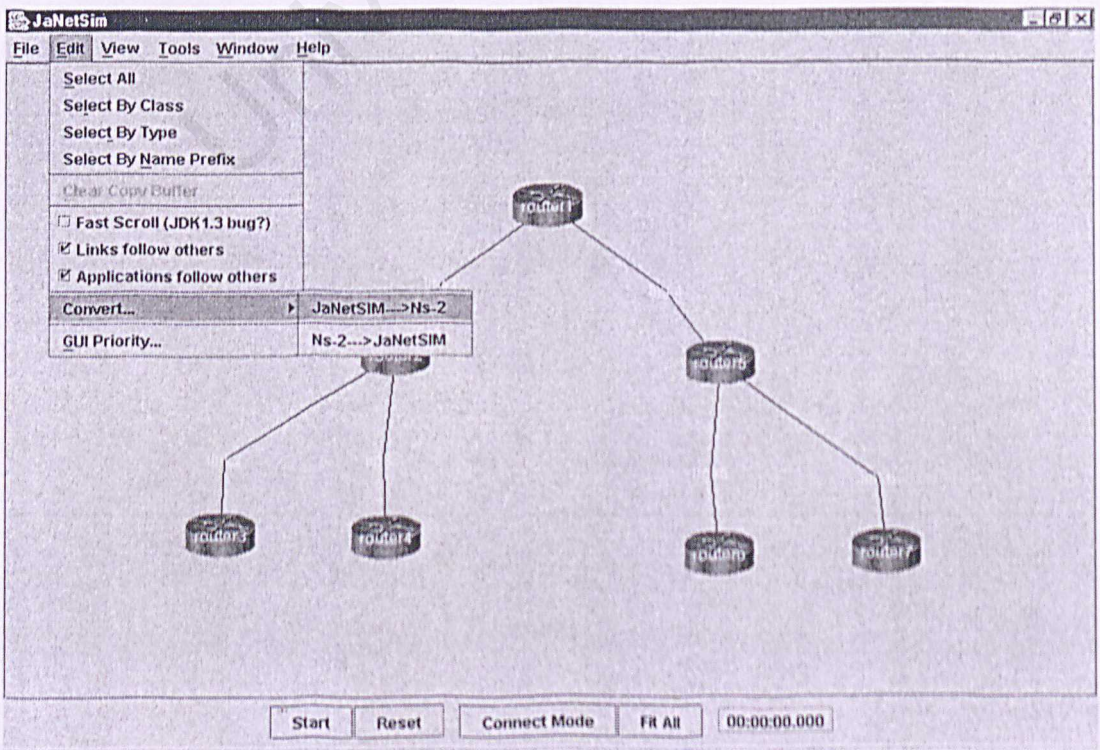
# Creating a Topology File

## Create NS-2 topology file

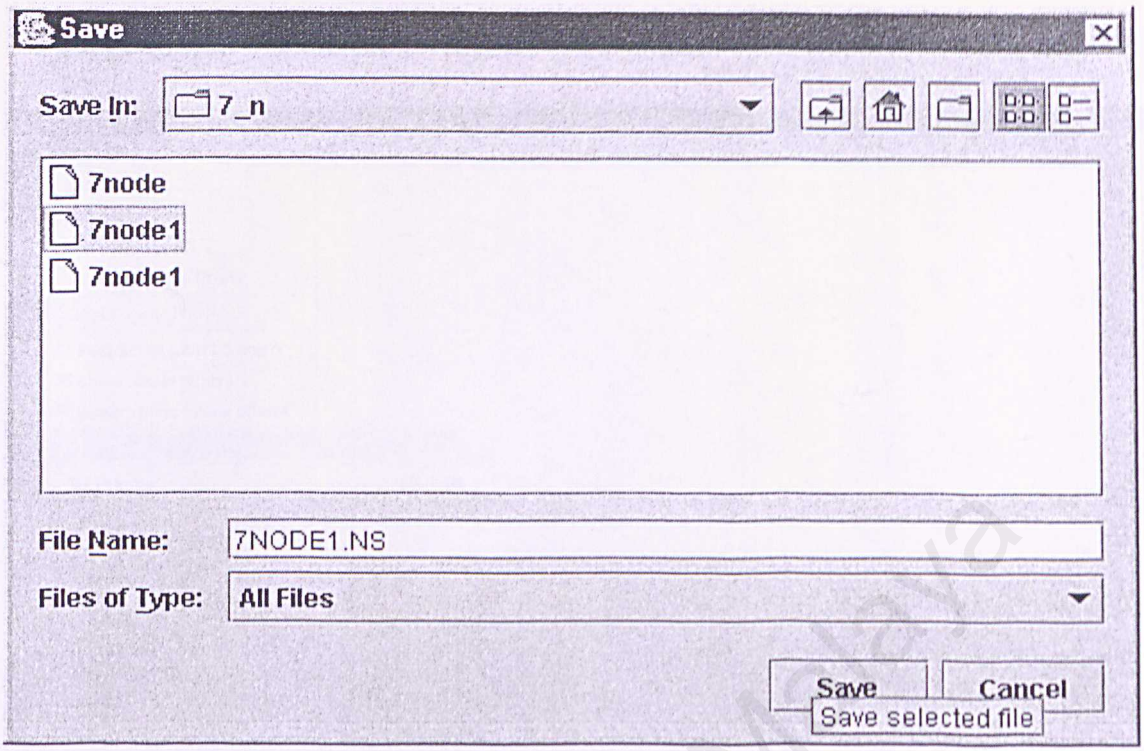
1. Load /create a new topology on the JaNetSim



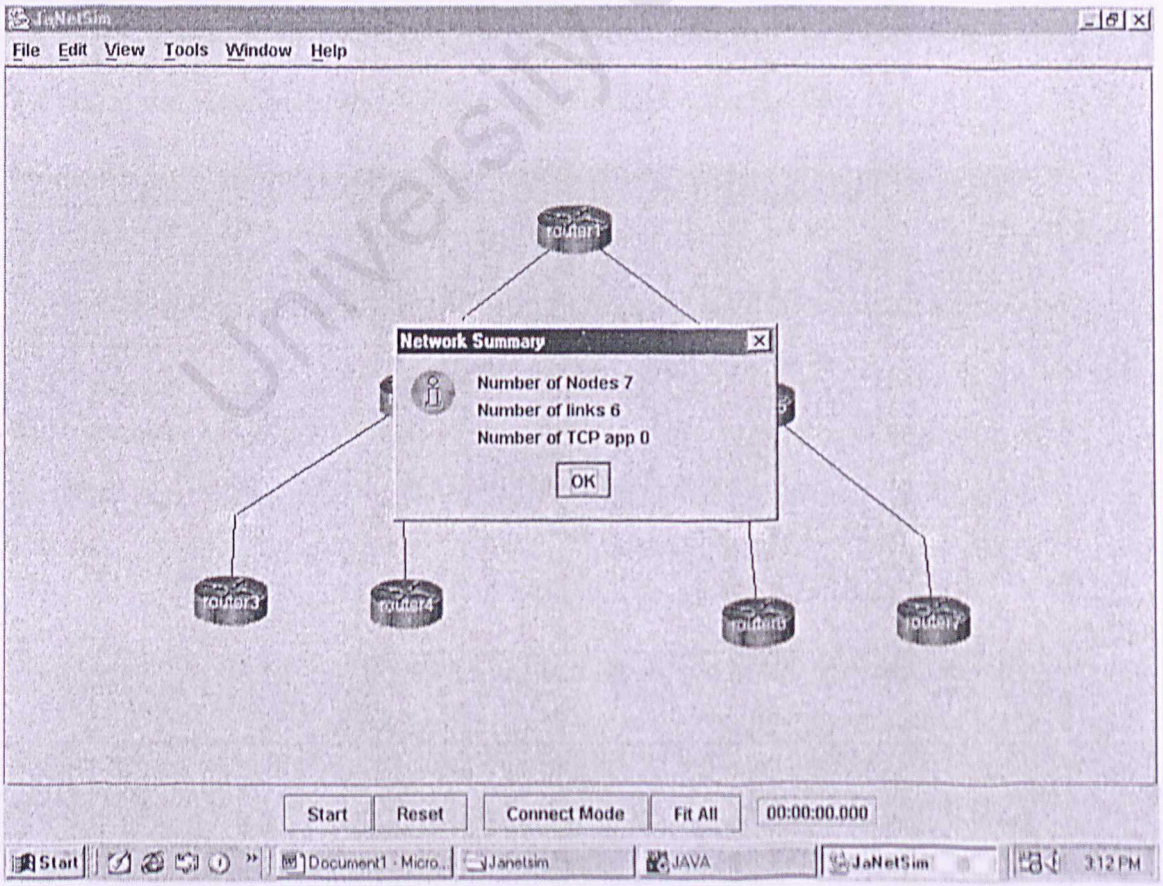

2. Next, go to Edit→ Convert ... → JaNetSim---->Ns-2



3. Choose the location to save the topology file and save the file with *.ns extension

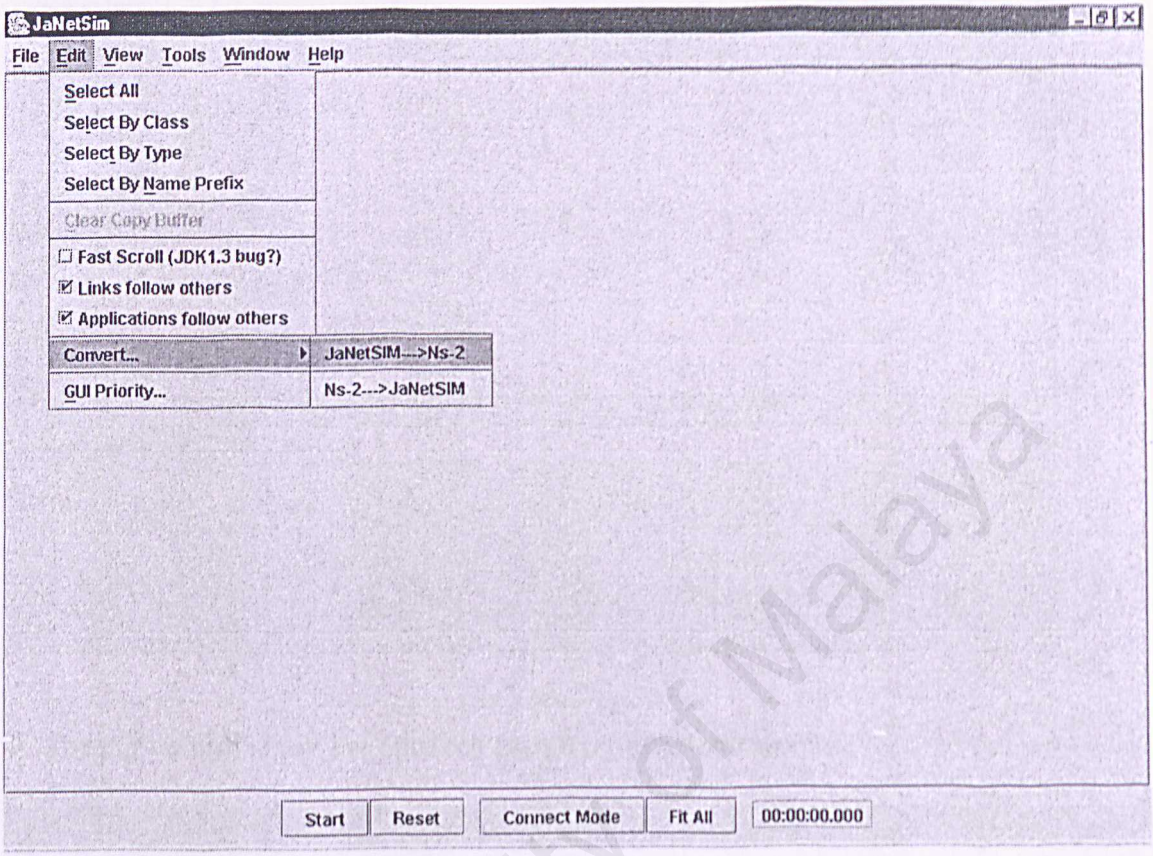



4. The popup will show the network summary after conversion

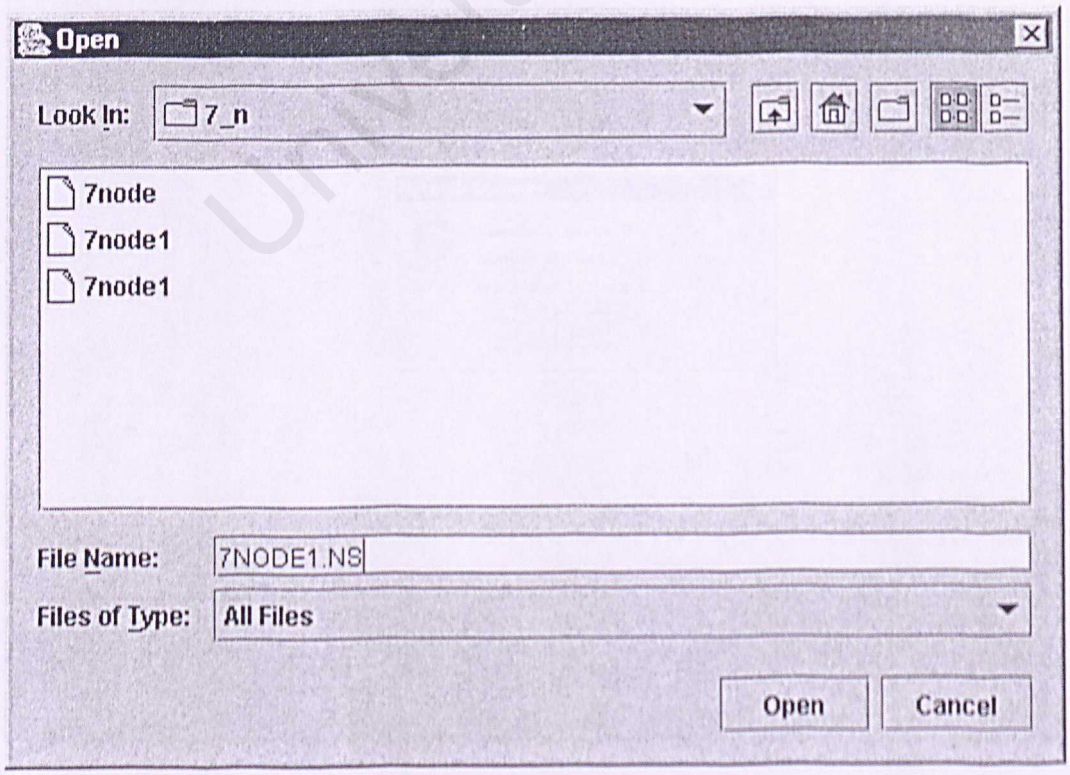


# Create JaNetSim topology file

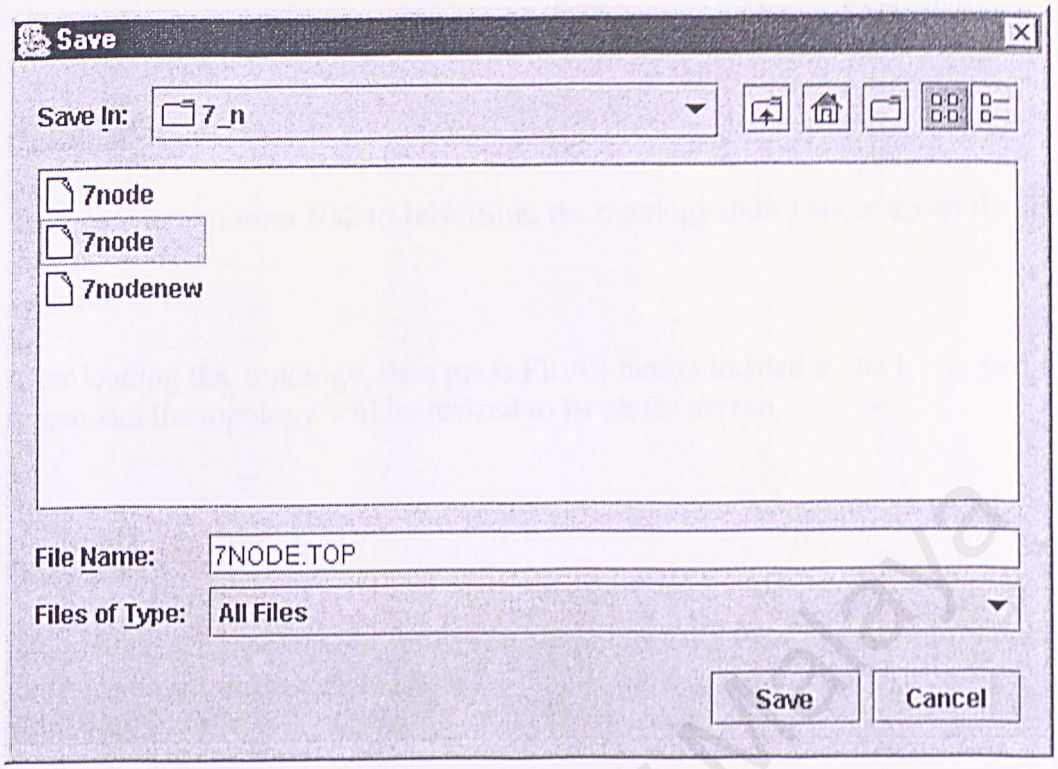
1. First go to Edit→ Convert... → Ns-2---->JaNetSim



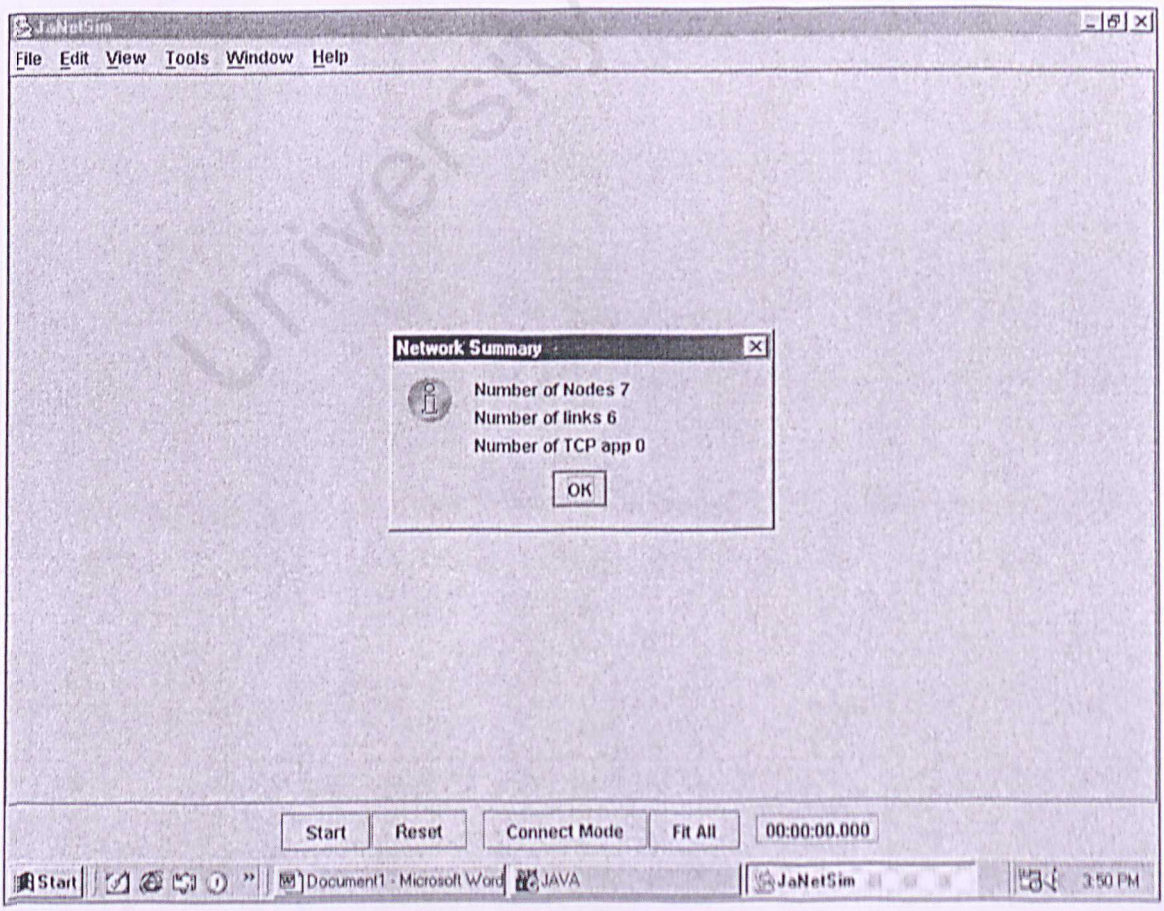

2.Choose the location of the NS topology file and open the file



3. A new window will prompt for save location.Save the file with *.top extension




4. The popup will show the network summary after conversion.



# Troubleshoot

**Question :**

After conversion from Ns2 to JaNetSim, the topology didn't show up on the JaNetSim screen.

**Solution :**

After loading the topology, then press Fit All button located at the lower part of the screen and the topology will be resized to fit on the screen.

**Question :**

After creating a topology on JaNetSim, when pressing the Convert button it didn't yield any result or save dialog box.

**Solution :**

Make sure that after connecting all the topology, press the End Connect button located at the lower part of the screen before the conversion process.