

SIMULATION OF A MULTIHOP WIRELESS LAN USING OMNET++

By

AHMAD ROHAIZAD BIN YUSOFF

FINAL PROJECT REPORT

**Submitted to the Electrical & Electronics Engineering Programme
in Partial Fulfillment of the Requirements
for the Degree
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)**

**Universiti Teknologi Petronas
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan**

© Copyright 2005

by

Ahmad Rohaizad Yusoff, 2005

+

TK

5105.78

A 286

2005

1) Wireless LAN

2) EEE -- Thesis

CERTIFICATION OF APPROVAL

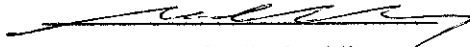
SIMULATION OF A MULTIHOP WIRELESS LAN USING OMNET++

by

Ahmad Rohaizad bin Yusoff

A project dissertation submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Approved:



Ms. Nasreen bt. Badruddin

Project Supervisor

**UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK**

June 2005

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.



(Ahmad Rohaizad bin Yusoff)

ABSTRACT

This project serves as a study of implementing multihop design in wireless local area network (WLAN) technology. The WLAN technology has seen a growth in popularity over the past few years. This project is divided into two parts. The first part of the project is to understand the theory behind the WLAN technology. Most of the materials for this research are taken from IEEEEXPLORE website and several books on wireless LAN and wireless data structure. The fundamental of networking that touches on OSI layer and data services are also studied. The second part of the project concentrated on simulating the multihop design. The simulation method uses a network simulator that can be found freely on the internet. The simulation is important as we want to simulate the system that has been built and to see whether the implementation is suitable or not. A good knowledge on C++ language is essential as most of the building process to achieve a credible simulation uses C++ programming codes. The network simulator offers a great dynamics to simulate the network and results can be plotted easily. The main focus of this study is the wireless standard; IEEE 802.11 that is used for wireless communication. The standard focuses on the MAC and PHY layers for Access Point based networks and ad hoc networks. There are certain objectives to be achieved which to study IEEE 802.11 WLANs standard; to analyze the problems occurred in multihop design in WLANs and to design a simulation platform to implement multihop design. The methodology used in this project is simulation using OMNeT++. The result of the simulation showed in a histogram of the node multihop between the nodes. From this project the possibilities of implementing a multihop design in WLAN could be analyzed and further works can be applied to improve its performance.

ACKNOWLEDGEMENTS

Firstly, I would like to express my utmost gratitude to Allah for giving the strength and blessing to complete this project. Without Him I would not be able to stay fit and constantly pushing my knowledge to meet the objectives of the project and produce a good result.

Secondly, I would like to thank my parents as they always support me and give me advice whenever I really need it.

My heartfelt gratitude also goes to the project supervisor, Ms Nasreen Badruddin for her valuable guidance and support throughout the progress of this project. Being under her supervision has been beneficial; her advice, suggestions and experience have helped me tremendously towards the completion of the project. Besides that, she is always willing to provide essential training sessions and consultation hours to all the students under her supervision if any difficulties or problems are encountered along the way.

I would also like to thank Mr Fawnizu Azmadi Hussin as my co-supervisor that helps me on understanding Linux system. He has helped me much installing and giving advice on the command used in Linux operating system.

Furthermore, I would like to accord my gratitude to Mr. Rasky the lab technician of IT/IS FYP Lab for allowing me to use the lab.

Lastly, I would like to thank all my friends who have helped me, either directly or indirectly by providing important ideas, key points and motivation regarding the project.

Without these people I mentioned above I would not have completed my Final Year Project successfully. Thank you all.

TABLE OF CONTENTS

LIST OF TABLES	ix
LIST OF FIGURES.....	x
LIST OF ABBREVIATIONS	xii
CHAPTER 1 INTRODUCTION	1
1.1 Background of Study.....	1
1.2 Problem Statement	2
1.2.1 Problem Identification	2
1.2.2 Significance of the project	3
1.3 Objectives and Scope of Study.....	3
1.3.1 The Relevancy of the Project.....	3
1.3.2 Feasibility of the Project within the Scope and Timeframe.....	3
CHAPTER 2 LITERATURE REVIEW AND THEORY	4
2.1 Fundamentals of WLAN	4
2.1.1 Wireless LAN Requirement.....	5
2.2 IEEE 802.11 Standard	6
2.2.1 IEEE 802.11 General Architecture	6
2.2.2 System Specification.....	7
2.2.3 IEEE 802.11 Services	7
2.2.4 Association-Related Services	8
2.3 IEEE 802.11 Medium Access Control	9
2.3.1 Distributed Coordination Function	10
2.4 MAC Frame.....	13
CHAPTER 3 METHODOLOGY/PROJECT WORK	14
3.1 Procedure Identification	18
3.2 Tools required.....	19
3.3 Simulation of ad hoc networks in OMNeT++.....	20
CHAPTER 4 RESULTS AND DISCUSSION	21
4.1 Linking and Compiling the Model Files	36
CHAPTER 5 CONCLUSION AND RECOMMENDATION.....	47
5.1 Recommendations	47
REFERENCES.....	49

APPENDICES.....50

LIST OF TABLES

Table 1 IEEE 802.11 Original Specifications: Basic Features	7
Table 2 IEEE 802.11 Services	8
Table 3 Sub module parameters.....	27
Table 4 Air gate size	28
Table 5 Sub module parameters.....	29
Table 6 The imported files	34

LIST OF FIGURES

Figure 1 Multihop Design in WLAN.....	2
Figure 2 IEEE 802.11 protocol entities.....	6
Figure 3 IEEE 802.11 Protocol Architecture	10
Figure 4 IEEE 802.11 Medium Access Control Logic	12
Figure 5 IEEE 802.11 MAC Frame Format.....	13
Figure 6 Hidden node problem	15
Figure 7 Intersymbol Interference.....	16
Figure 8 Steps in implementing multihop design in WLANs.....	18
Figure 9 OMNeT++ GNED	19
Figure 10 Logical architecture of the ad-hoc simulator.....	21
Figure 11 Draw menu.....	23
Figure 12 Drawing the first sub module	23
Figure 13 Sub-module appearances window	24
Figure 14 Icon selected	25
Figure 15 Sub module properties	25
Figure 16 Sub module properties (Parameters tab).....	26
Figure 17 Gate size tab.....	27
Figure 18 Finished creating Air module	28
Figure 19 Icon selected	29
Figure 20 Ad Hoc network topology	30
Figure 21 Opening the module contents	31
Figure 22 The submods and conns tabs	31
Figure 23 All model tabs.....	32
Figure 24 Module properties.....	33
Figure 25 Changing the module name to Ad Hoc System.....	33
Figure 26 Adding the imports tab	33
Figure 27 The model menu after adding the import tab.....	34
Figure 28 Import properties menu.....	34
Figure 29 Switch to NED code mode	35
Figure 30 The network tab added.....	35
Figure 31 Network properties menu	36

Figure 32 Saving the model file	36
Figure 33 A new project created	37
Figure 34 New project window.....	37
Figure 35 OMNeT++ AppWizard Page 1	38
Figure 36 OMNeT++ AppWizard Page 2.....	39
Figure 37 OMNeT++ Note	39
Figure 38 Closing a project in Visual C++	40
Figure 39 Ad Hoc System project file menu.....	41
Figure 40 Importing files to the project	42
Figure 41 The files menu after importing the files	42
Figure 42 Setting the project properties.....	43
Figure 43 Setting the C/C++ properties	43
Figure 44 Setting the NED files compiler properties.....	44
Figure 45 Setting the msg compiler properties	45
Figure 46 Compiling the .ned and .msg files	45
Figure 47 Histogram of the network throughput.....	46

LIST OF ABBREVIATIONS

WLANs	Wireless Local Area Network
IEEE	Institute of Electrical and Electronics Engineer
AP	Access Point
MAC	Medium Access Control
OFDM	Orthogonal Frequency Division Multiplexing
ISM	Industrial, Scientific and Medical
NIC	Network Interface Card
PHY	Physical Layer
LLC	Logical Logic Control
PDU	Protocol Data Unit
CSMA/CA	Carrier Sense Multiple Access Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access Collision Detection
DS	Distribution System
BSS	Basic Service Set
MSDU	MAC Service Data Unit
ESS	Extended Service Set
ACK	Acknowledgment
DCF	Distributed Coordination Function
PCF	Point Coordination Function
IFS	Interframe Space
ISI	Intersymbol Interference
FTP	File Transfer Protocol

CHAPTER 1

INTRODUCTION

1.1 Background of Study

The introduction of a committee in 1990 by IEEE has develop a standard for wireless LANs, operating at 1 and 2 Mbps. For some technical reasons the first standard for wireless LANs only emerge in 1997 which is known as IEEE 802.11 system. This first standard allowing it work at a data rate of 1 and 2 Mbps. In the fall of 1999, the standard was extended to break the 10 Mbps barrier. Thus the IEEE 802.11b was born allowing reaching data rates of 5.5 Mbps and 11 Mbps. In the meantime a second group was working on a standard working in the 5- GHz band. This standard, known as IEEE 802.11a, allowed work at velocities of 6, 12, and 24 Mbps defining 9, 18, 36 and 54 Mbps as option.

The trend for today's communication is shifting to wireless communication. It is a common environment to see young executives and businessmen surfing the internet world in first class coffee shops like Starbucks, San Francisco and such in business district in Kuala Lumpur. The need to communicate to the other people around the world and relatives in anywhere we go has seen a rapid increase in the usage of mobile computing. This place is called 'hotspots' as we can access to the internet without wired cables. An access point (AP) located near the customer's tables is connected to a wired backbone that enables customers to surf the net.

The use of this WLAN technology is not only in the coffee shops and shopping complexes. Most of the companies have seen the advantages of this technology and increasing their response to the customers needs. The normal workstation that connects with wired cables is now converting it to WLAN technology. This will enable employees to move freely and communicate with each other easily. It will also

decrease the effects of ergonomics syndrome because of the long period of time sitting in the workstation.

1.2 Problem Statement

1.2.1 Problem Identification

Seeing the increase usage of WLAN technology in Malaysia today, the drawbacks is still exist despite the advantages that this smart technology offers. This project is considering the possibility to implement multihop design in WLAN. The AP located in a certain network that is built has a limitation on its coverage. It only serves the computers that is within a 100 to 300 meters depending on the vendor that produced the AP. IEEE 802.11 standard does not support a multihop concept because of the unfriendliness of medium access control (MAC) to multihop network. This study will present the system that will enable multihop network to be implemented. The layout of the system is shown in Figure 1.

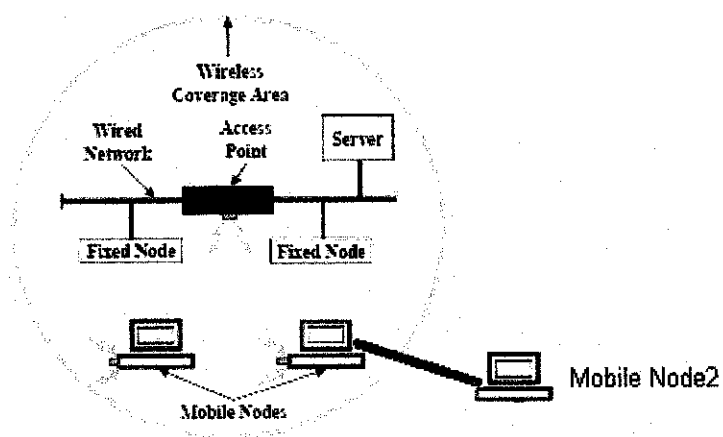


Figure 1 Multihop Design in WLAN

The idea of this multihop network if the mobile node 2 is out of the AP range it still can be connected to it through relaying with the mobile nodes that is inside the AP range. Meaning to say that this concept will extend the coverage of the AP without installing a new AP to support the node that is outside the existing AP. By using this concept the installation cost and the cost for purchasing a new AP could be avoided.

1.2.2 Significance of the project

By implementing the multihop concept in WLAN not only we can decrease the installation cost and AP cost but it also gives a wider coverage to the WLAN.

1.3 Objectives and Scope of Study

1.3.1 The Relevancy of the Project

This project aims at implementing the multihop concept in WLAN technology. But first a detail understanding about the standard and the fundamental concept of WLAN technology including routing, relaying, mobility and others have to be understood clearly. The concept of multihop will not be successful to be implemented if we neglected the idea of how the WLAN sends its message from one node to the other node. The OSI layer that is servicing the sending of the message must be known and the modulation that is used in transmitting from sender to receiver. The objectives of this project are as below:

1. To study IEEE 802.11 WLAN standard.
2. To analyze the problems occurred in multihop design in WLAN.
3. To design the simulation to implement multihop design.

1.3.2 Feasibility of the Project within the Scope and Timeframe

This project is conducted within the timeframe of two semesters which is about one year. During the first semester of the project most of the time was spent to do literature search on wireless technology and the IEEE 802.11 standard. WLAN technology is using the IEEE protocol so it is useful to understand first the theory of the standard, its usage and the advantages and disadvantages of the protocol. The second semester of the project is to simulate the network using a network simulator that is downloaded from internet. The second part of the project is a bit hard in the first two weeks of the semester as it involved the process to install Linux operating system and learn all the coding in OMNeT++. This method will be explained in details in Chapter 4 of the report while in this stage is just worth mentioning.

CHAPTER 2

LITERATURE REVIEW AND THEORY

2.1 Fundamentals of WLAN

The basis for the technology that is used in WLANs today was developed in World War II by the US military as a way for securing the safe, private delivery of voice communications without eavesdropping by the enemy [1]. The WLAN technology is popular because of its advantages in ease of installation, flexibility and mobility. This benefits offer gains in efficiency, accuracy, and lower business costs [2]. There are many technologies that associated with the WLAN system. The technique that is used to transmit voice and data by operating over a range of frequencies is called spread spectrum which is widely used by WLAN equipment today. The spread spectrum is not the only technology that is being used for WLANs. Other technologies such as infrared and UHF have been used for years and newer systems based on orthogonal frequency division multiplexing (OFDM) are being developed. WLANs uses unlicensed spectrum to operate which is called industrial, scientific, and medical (ISM) band. WLANs mostly operate by using both radio technology and infrared techniques in this frequency band of 2.4 GHz. An advantage of radio waves is that they can provide connectivity for a nonline-of-sight (NLOS) situation. A disadvantage of radio waves is that the electromagnetic propagation, which might cause interference with medical equipment and industrial components working at the same frequency. Radio waves propagate through walls, which may also be a security problem [1].

WLANs consists of wireless network interface cards (NICs), often known as stations (STAs) and wireless bridges referred to access points (APs). The STAs and APs interface the wireless network with the wired network. There are two types of WLANs connection which are AP-based network and ad-hoc network. AP-based

network is also known as infrastructure network which uses AP as a bridge connecting wireless and wired LANs. When AP is present stations do not communicate on a peer to peer basis. All communications between stations and a wired network client go through AP. The other type of WLANs is the ad hoc network which does not have AP in it but purely communicating between several nodes. Stations communicate with each other on a peer-to-peer level sharing a given cell coverage area. This type of network is often formed on a temporary basis.

Wireless is an option for these scenarios; mobility, short-term usage needs, speed of deployment, and the need to overcome difficult wire installation situations. Mobility means allowing users to move while using their equipment (for example, a laptop PC). Users in an office can move to meetings or roam around the building with their laptops while still being connected to the network. WLANs technology also allows short-term user as-needed basis without concern for cost of justification for wired solutions. WLANs permit quick connectivity to the network. Forming and disbanding work groups can be done easily with WLANs. Many situations do not permit easy installation of wires. There are situations where wires cannot be laid, for example, across busy streets. Building-to-building connections are also difficult to facilitate because no underground cabling is present. Using wireless bridges to connect physically separated LANs or Internet connections can be very effective.

2.1.1 Wireless LAN Requirement

There are a number of requirements that a wireless LAN need to have to build wireless environment. A wireless LAN must meet the same sort of requirements typical of any LAN, including high capacity, ability to cover short distances, full connectivity among attached stations, and broadcast capability [3]. The following are among the important requirements for wireless LANs:

1. Throughput
2. Number of nodes
3. Connection to backbone LAN
4. Service area
5. Battery power consumption

6. Transmission robustness and security
7. Collocated network operation
8. License-free operation
9. Handoff/roaming
10. Dynamic configuration

2.2 IEEE 802.11 Standard

2.2.1 IEEE 802.11 General Architecture

The IEEE 802.11 is a standard constituted by a PHY layer and a MAC layer [4]. A topology provides a means of explaining necessary physical components of a network, but the logical architecture defines the network's operation. The logical architecture of the 802.11 standard that applies to each station consists of a single MAC and one multiple PHY. The architecture is shown in Figure 2.

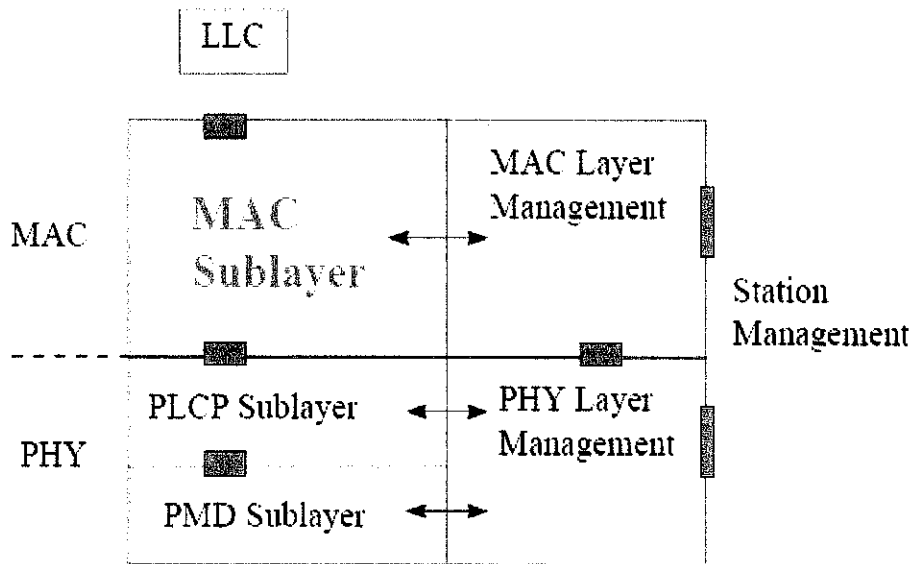


Figure 2 IEEE 802.11 protocol entities

The goal of the MAC layer is to provide access control functions (such as addressing, access coordination, frame check sequence generation and checking, and LLC PDU delimiting) for shared-medium PHYs in support of the LLC layer. The MAC layer performs the addressing and recognition of frames in support of the LLC. The 802.11

standard uses CSMA/CA, whereas standard Ethernet uses CSMA/CD. It is very impractical to transmit and receive on the same radio channel at the same time due to the extreme ratio between the transmitted and received power. Therefore, an 802.11 WLAN only takes measures to avoid collisions, not detect them.

2.2.2 System Specification

Some basic features of the original IEEE 802.11 specification are sketched in Table 1.

Spectrum	2.4 GHz
Maximum physical rate	2 Mbps
Maximum data rate, layer 3	1.5 Mbps
MAC	CSMA/CA
Fixed network support	IEEE 802 wired LANs and others

Table 1 IEEE 802.11 Original Specifications: Basic Features

2.2.3 IEEE 802.11 Services

In this protocol there are nine services that need to be provided by the WLAN to provide functionality equivalent to that which is inherent to wired LANs. Table 2 lists the services and indicates two ways of categorizing them.

1. The service provider can be either the station or the DS. In every 802.11 station, station services are implemented including AP stations. Distribution services are provided between BSSs; these services may be implemented in an AP or in another special-purpose device attached to the distribution system.
2. There are three services that are used to control IEEE 802.11 access and confidentiality. To support delivery of MAC service data units (MSDUs) between stations six services are used. The MSDU is block of data passed down from the MAC user to the MAC layer. The MAC frame may need to be fragmented and transmitted in a series of MAC frames if the MSDU is too

large to be transmitted in a single MAC frame.

Service	Provider	Used to support
Association	Distribution system	MSDU delivery
Authentication	Station	LAN access and security
Deauthentication	Station	LAN access and security
Dissassociation	Distribution system	MSDU delivery
Distribution	Distribution system	MSDU delivery
Integration	Distribution system	MSDU delivery
MSDU delivery	Station	MSDU delivery
Privacy	Station	LAN access and security
Reassociation	Distribution system	MSDU delivery

Table 2 IEEE 802.11 Services

2.2.4 Association-Related Services

The transfer of MSDUs between MAC entities are transferred by the MAC layer. This purpose is fulfilled by the distribution service. Information about stations within the ESS that is provided by the association-related services is needed for that service to function. That station must be *associated* before the distribution service can deliver data to or accept data from a station. The distribution service needs to know where the destination station is located to deliver a message within a DS. Specifically, the DS needs to know the identity of the AP to which the message should be delivered in order for that message to reach the destination station. A station must maintain an association with the AP within its current BSS to meet this requirement. Three services relate to this requirement:

1. Association: This service function is to establish an initial association between a station and an AP. Its identity and address must be known before a station can transmit or receive frames on a wireless LAN. For this purpose, a station must establish an association with an AP within a particular BSS. The AP can then communicate this information to other APs within the ESS to facilitate

routing delivery of addressed frames.

2. **Reassociation:** This service enables an established association to be transferred from one AP to another, allowing a mobile station to move from one BSS to another.
3. **Disassociation:** A notification from a station or an AP that signals an existing association is terminated. Before leaving an ESS or shutting down a station will give a notification to indicate the station is leaving. However, the MAC management facility protects itself against stations that disappear without notification.

2.3 IEEE 802.11 Medium Access Control

Medium Access Control (MAC) protocol functions as a means of controlling access to the transmission medium. It is needed for an orderly and efficient use of that capacity. This is because all LANs and MANs consist of collections of devices that must share the network's transmission capacity. That is why MAC protocol is needed to make the transmission of message is in order and efficient. For WLANs technology the IEEE 802 protocol layers is different compared to OSI model.

For the IEEE 802.11 MAC protocol it allows interoperability between compatible PHYs through the use of the CSMA/CA protocol with acknowledgment (ACK) and of a random back-off time following a busy medium condition. The 802.11 CSMA/CA reduce the probability of collision between multiple stations accessing the medium at the point in time where collisions would most likely occur. Usually the collision would most likely occur just after the medium becomes free or just after busy medium conditions. These conditions always happened when multiple stations are waiting for the medium to become available again. This is where a random back-off arrangement is used to resolve and minimize conflicts in medium contention. The protocol also describes the way beacon frames are sent by the AP at regular intervals (like 100 ms) to enable stations to monitor the presence of the AP. The MAC also gives a set of management frames that allow station to actively scan for other APs on any available channel. Best-suited AP can be decided by the stations using this scan information.

There are two types of proposals that has been considered by the 802.11 working group which are: distributed access protocols, the decision to transmit over all the node is distribute using a carrier-sense mechanism; and centralized access protocols, which is the transmission is regulated by a centralized decision maker. For an ad hoc network of peer workstations it is obvious for a distributed access protocol and may also be attractive in other wireless LAN configurations that consist primarily of bursty traffic. A number of wireless stations that are interconnected with each other and some sort of base station that attaches to a backbone wired LAN is natural for a centralized access protocol configuration.

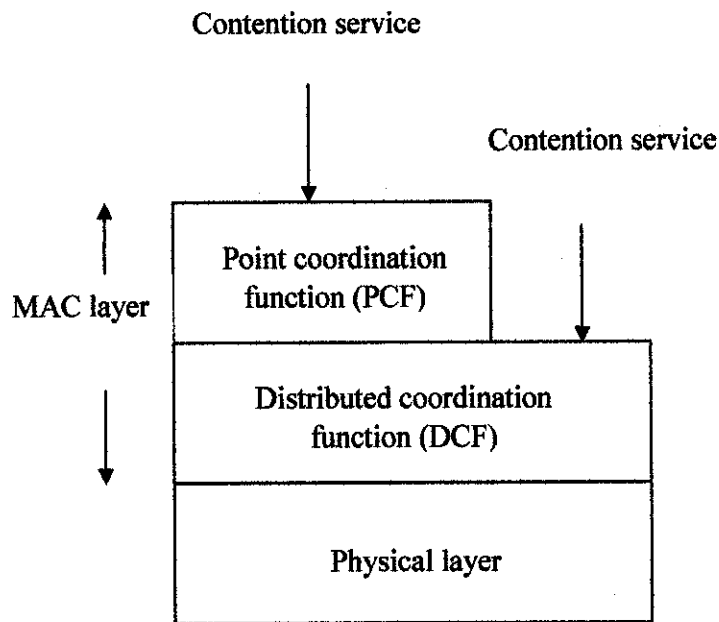


Figure 3 IEEE 802.11 Protocol Architecture

Figure 3 illustrates the architecture of the IEEE 802.11 protocol. The distributed coordination function (DCF) located at the lower sublayer of the MAC layer. DCF uses a contention algorithm to provide access to all traffic. It is considered as a fundamental access method, which support asynchronous data transfer on a best effort basis. It operates solely in an ad hoc network, and in an infrastructure network, the DCF either coexists with the PCF or operates solely.

2.3.1 Distributed Coordination Function

The DCF sublayer is built by a simple CSMA algorithm. A station listens to the

medium if it has a MAC frame to transmit. The station may transmit if the medium is idle otherwise the station must wait until the current transmission is complete then it can transmit. There is no collision detection function in the DCF MAC sublayer (i.e., CSMA/CD) because collision detection is not practical on wireless network. Because of the dynamic range of the signals on the medium is so large that a transmitting station cannot effectively distinguish incoming weak signals from noise and the effects of its own transmission [3]. The carrier sensing is performed, in IEEE 802.11, both the air interface (referred to as physical carrier sensing) and at the MAC sublayer (referred to as virtual carrier sensing). The other IEEE 802.11 WLAN users are detected by physical carrier sensing. It analyzes all detected packets, and also detects the activity in the channel via relative signal strength from other sources.

DCF includes a set of delays that amounts to a priority scheme to ensure the smooth and fair functioning of this algorithm. Using an IFS, the rules for CSMA access are as follows (Figure 4):

1. A station with a frame to transmit senses the medium. If the medium is idle, it waits to see if the medium remains idle for a time equal to IFS. The station may transmit immediately if it is equal.
2. If the medium is busy (either because the station initially finds the medium busy or because the medium becomes busy during the IFS idle time), the station defers transmission and continues to monitor the medium until the current transmission is over.
3. The station delays another IFS once the current transmission is over. If the medium remains idle for this period, then the station backs off a random amount of time and again senses the medium. The station may transmit if the medium is still idle.

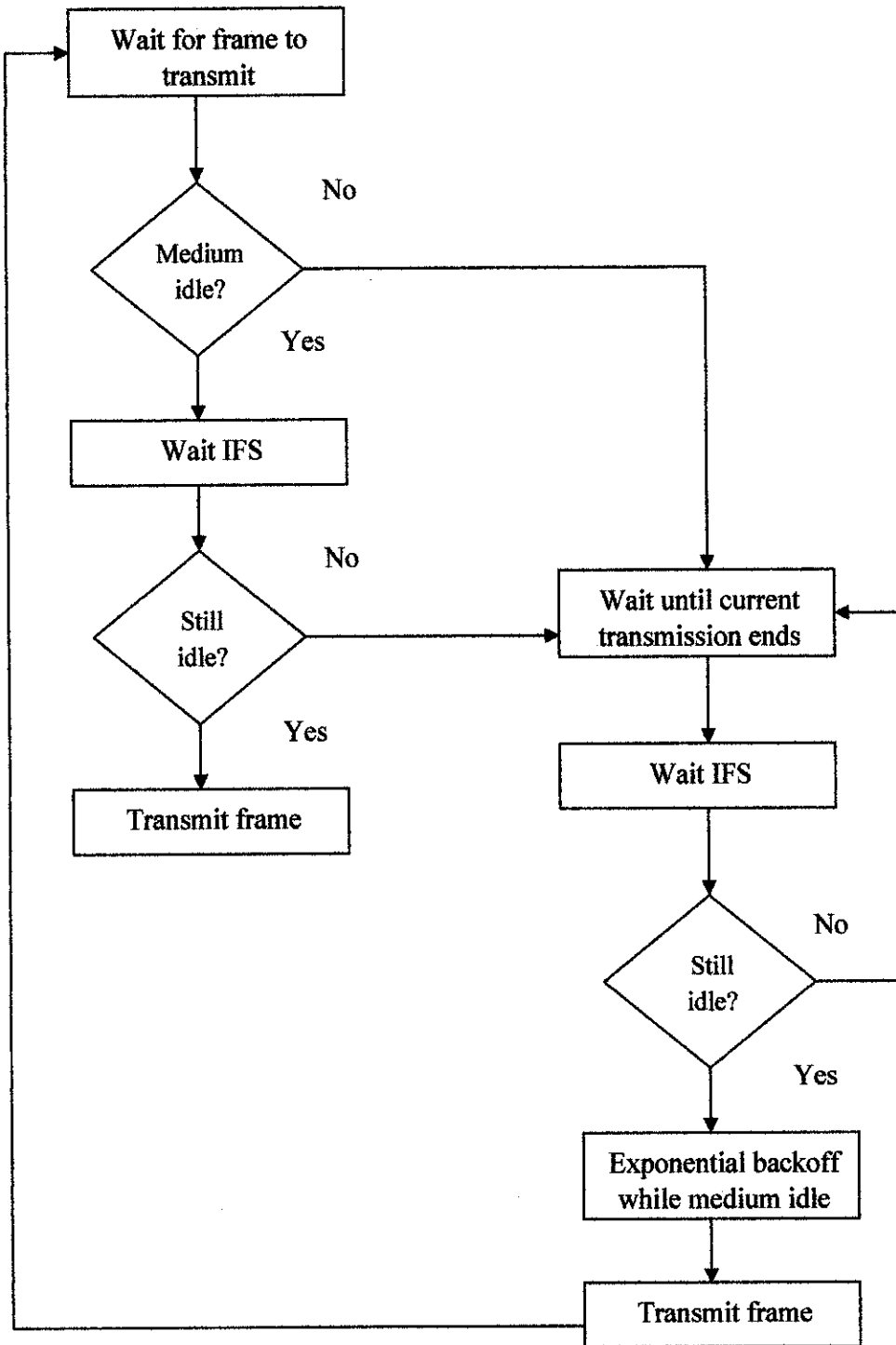


Figure 4 IEEE 802.11 Medium Access Control Logic

2.4 MAC Frame

Figure 5 shows the 802.11 frame format. The general format is used for all data and control frames, but not all fields are used in all contexts. The fields are as follows:

1. **Frame Control:** Indicates the type of frame (control, management, or data) and provides control information. Control information includes whether the frame is to or from a DS, fragmentation information, and privacy information.
2. **Duration/Connection ID:** If used as a duration field, indicates the time (in microseconds) the channel will be allocated for successful transmission of a MAC frame.
3. **Addresses:** The number and meaning of the address fields depend on context. Address types include source, destination, transmitting station, and receiving station.

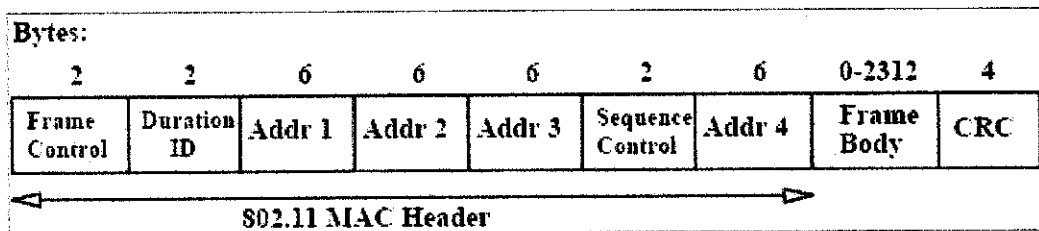


Figure 5 IEEE 802.11 MAC Frame Format

4. **Sequence Control:** Contains a 4-bit fragment number subfield, used for fragmentation and reassembly, and a 12-bit sequence number used to number frames sent between a given transmitter and receiver.
5. **Frame Body:** Contains an MSDU or a fragment of an MSDU. The MSDU is a LLC protocol data unit or MAC control information.
6. **Frame Check Sequence:** A 32-bit cyclic redundancy check

CHAPTER 3

METHODOLOGY/PROJECT WORK

Basically the aspects that have been conducted during this project are:

- **Simulation**

During this project a few simulations have been conducted to analyze the performance of wireless LAN. OMNeT++ is a network simulation tools that is publicly released in the internet for the use of network simulation. The name itself stands for Objective Modular Network Testbed in C++ [5]. Familiarization with the OMNeT++ tools is done by trying the examples that comes with the program like FIFO model, Token Ring LAN, and client-server application. The simulator also can be used for:

- traffic modeling of telecommunication networks.
- protocol modeling
- modeling queuing networks.
- modeling multiprocessors and other distributed hardware systems.
- validating hardware architectures
- evaluating performance aspects of complex software system.

One of the simulations that have been done is on Ethernet network. The simulation material was presented by Dr Jürgen Jaspeneite of University of Applied Science. This step by step tutorial really comes in handy when conducting the simulation for this project. It shows how to use the functions in OMNeT++ simulator which at first is confusing and hard to understand. It also shows us how to link and compile the model's files to produce executable simulation files using Microsoft Visual C++. We could also know where to change the parameters in OMNeT++ to test various environments. It is also applicable to any network topology. The objective of the tutorial is to build network topology not to build models of the element. The models are prepared

earlier and the appropriate codes were added and simulation is done in the network tools. The models are selected from the OMNeT++ library.

- Identify problems.

One of the problems associated with the deployment of multihop design is the hidden (non-sited) node problem. Before we could design the multihop concept the hidden node problem lies in the ad hoc network creating a drawback in the wireless communication. In this case some of the nodes are within the range of the intended receiver but not of the transmitter [6]. Figure 6 shows two nodes (A and C) that are within communication range of a third node B. Node A and node C are not in range which means they cannot communicate with each other because of the limitations that it is not in the communicating range. So node A and node B did not know when each other will transmit data and this could result in collision in node B if the two send signals simultaneously.

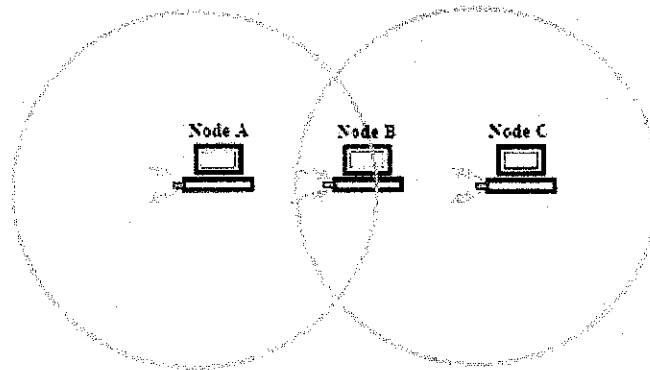


Figure 6 Hidden node problem

Another problem in wireless transmission is multipath fading. This scenario occurs when a signal takes multiple paths to reach the receiver. This could result the destruction of the signal and interference at the receiving antenna. Besides multipath propagation a more serious impairment caused by it is intersymbol interference (ISI) [6]. ISI occurs when duration of data pulse (symbol) becomes comparable to the delay spread, which corresponds to the difference in arrival times of the various signal components dispersed by multipath propagation. The received data pulses overlap in time and this result in digital errors (Figure 7).

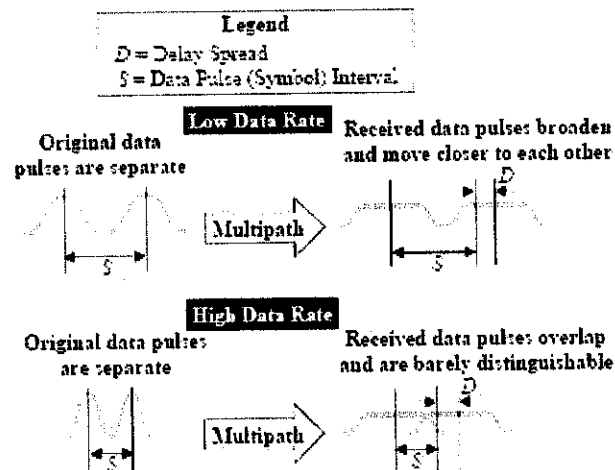


Figure 7 Intersymbol Interference

- Compare performance of multihop design.

Researches around the world have constantly studied the concept of multihop WLAN. One of the papers that are taken from IEEE XPLOR is on a study proposed by [7] to support communication between access points. Its backbone is composed of the Multi-hop Wireless LAN (M-WLAN), where mobile ad hoc network technologies are used to support communication between access points (AP). They suggest a multihop concept of wired backbone in the WLAN architecture. This study is motivated by the need to provide wider service area so the backbone should be easily extended. This wireless backbone networks can be extended easily in low cost. Compared the method that we have right now to cover large service area by using wired backbone it is expensive [7]. It is interesting to know that we could improve the performance of IEEE 802.11 with multihop concept. This method is proposed by [8] that used a novel MAC layer relay-enabled point coordination function (PCF) protocol, called rPCF. This method is proposed to exploit the capability of IEEE 802.11 physical layer that has multi-rate capability.

- Design solution of multihop wireless LAN.

To find solution for the problems that occurred in the wireless communication a study based on what has been proposed by other researchers were referred.

Kenichi and Yasunori [7] from Niigata University implemented a concept that is kind of similar to what this project's objective. The difference is that they build a wireless backbone while this project maintains with the wired backbone. That can be used a strong basis to design a solution for this paper. Some modifications need to be done to suit with this project's objectives.

Zhu and Cao [8] from Pennsylvania State University proposed a technology that will improve the performance of IEEE 802.11 using multihop concept. They proposed a novel MAC layer relay-enabled point coordination function (PCF) protocol. The presentation is good and they stated the improvement that is expected when using multihop concept.

3.1 Procedure Identification

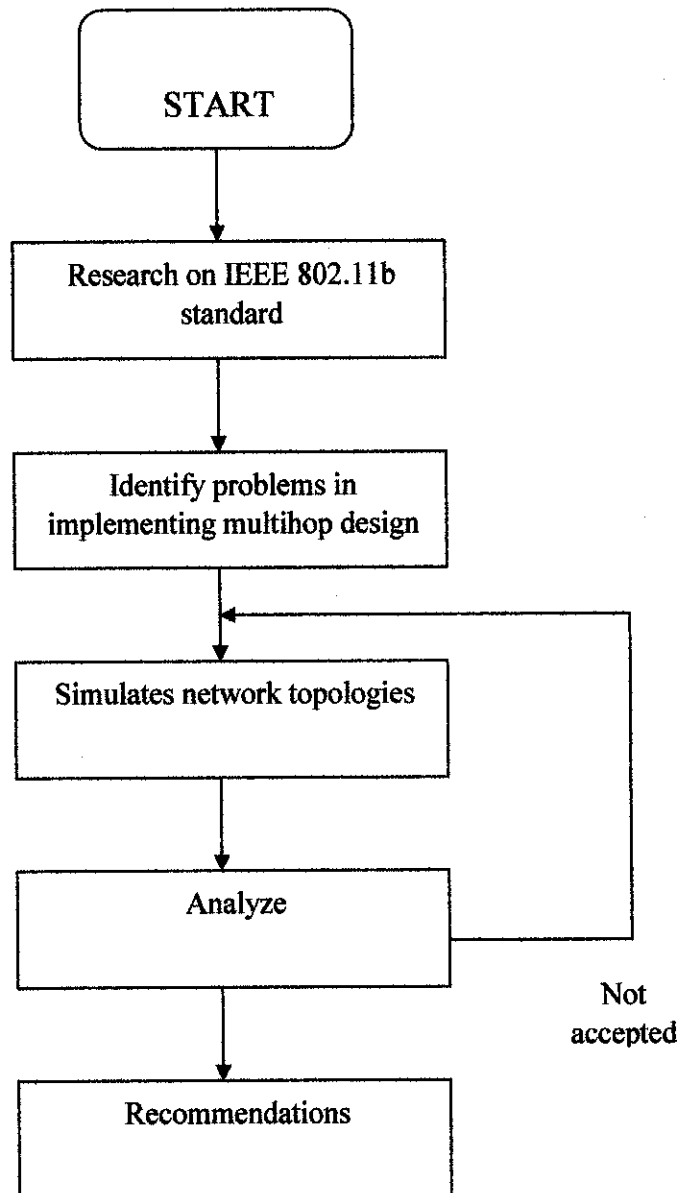


Figure 8 Steps in implementing multihop design in WLANs

3.2 Tools required

The tools that are required in this project are:

- OMNeT++

This network simulator was downloaded from www.omnetpp.org. OMNeT++ is open source software and it is free to use by anyone in the net. Other network simulator like NS2, OPNET and COMNET 3 are available either free based or we have to purchase it. There are several issues that we have to look for before deciding to use a network simulator:

- *Does the simulation tool have the necessary power to express details in the model?*
- *What protocol models are readily available for simulation tools?*
- *How does the simulation tool support defining the network topology?*
- *What is the programming model supported by the simulation environment?*
- *What debugging or tracing facilities does the simulation tool offer?*
- *What performance can be expected from the simulation?*
- *Is the simulation library available in source?*

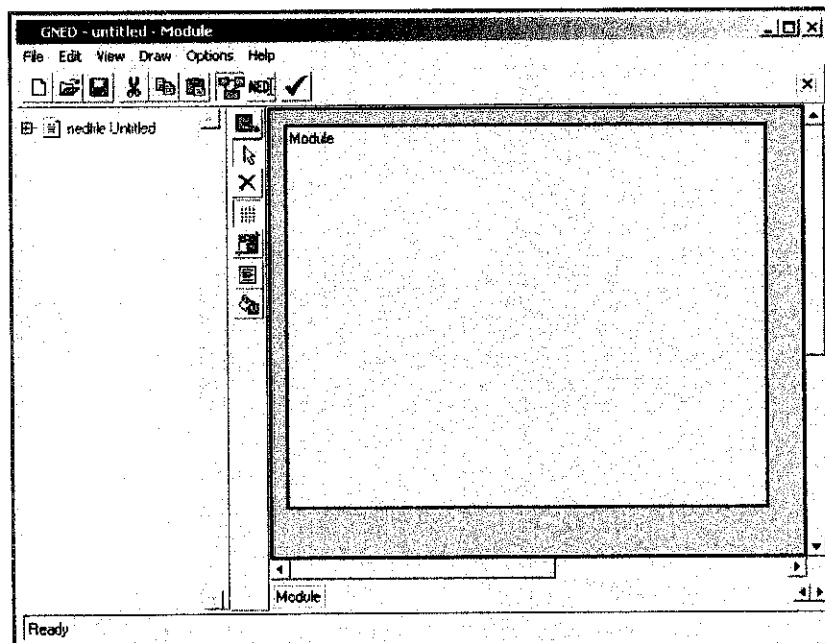


Figure 9 OMNeT++ GNEd

- Microsoft Visual C++ 6.0.

We have to have a good knowledge in C++ language to use OMNeT++. This is because OMNeT++ system construction is made from C++ language. It is obvious also that its name hold the need of C++ (Objective Modular Network Testbed in C++).

3.3 Simulation of ad hoc networks in OMNeT++

The network simulator does not support infrastructure network for WLANs. This limitation does effect the project development as the objectives need to be changed according to the ability of the simulator. As an alternative the simulation that could be simulate using OMNeT++ is ad hoc networks simulation. For this simulation the main reference is a paper written by researches from Department of Telecommunications Budapest University of Technology and Economics. The paper of 'Simulation Environment for Ad-Hoc Networks in OMNeT++' is somewhat the same with the simulation that is intended to do. The objective of the project that is launched on 2000 is to build simulation environment which is able to investigate different medium access, routing, mobility prediction, etc. algorithms. The simulation was developed by several European university teams (BUTE – Hungary, TU Delft – The Netherlands, Monash University – Australia, University of Karlsruhe – Germany, TU Berlin – Germany, etc).

CHAPTER 4

RESULTS AND DISCUSSION

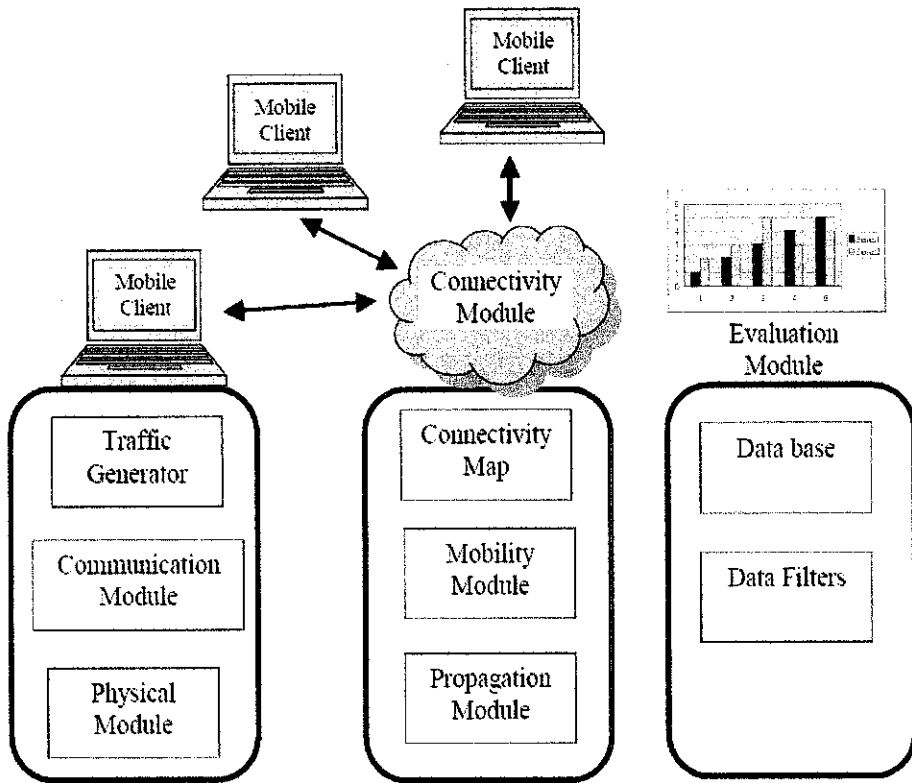


Figure 10 Logical architecture of the ad-hoc simulator

The ad-hoc simulation environment consists of several modules (see Figure 10).

1. **Mobile client module:** This entity represents the user terminals in the system. Several physical and higher layer parameters, such as transmission power, implemented protocols (routing, security), etc. are characterized in each every terminal. In the mobile client module it consists several following submodules:
 - **Traffic generator** that emits packets according to the type of the client (e.g., speech, FTP, etc.)
 - **Physical block** has a sender and a receiver entity with queues where

the received and generated packets are stored.

- **Communication module** implements the rule of the defined protocol stack (i.e., generates signaling messages).

Every submodule has one or more input and output ports, which ensure the connections between the submodules.


2. **Connectivity module:** This module realizes the connections among the mobile clients. This module knows the topology of the network. The network is described by means of a graph. The graph can be generated randomly or according to a predefined topology during the initialization phase. After the initialization the connectivity graph is modified according to the mobility module.
3. **Radio propagation submodule** contains different radio propagation models for indoor and outdoor environment, such as open air, Rayleigh and Rice fading etc. By means of the radio range of a given terminal can be calculated.
4. **Mobility submodule** describes the mobility customs of the users. At this stage of the system our network has fixed topology.
5. **Evaluation module** is responsible to collect and store measurement data during a simulation and the presentation of the results. It contains filters which work on the stored data to provide the required information.

The step by step simulation is as follows:

Step 1 Drawing the sub-module:

There are fixed steps to create a sub module in OMNeT++, which will be applied to all that is to be created:

- Draw menu in the toolbar of the GNED is clicked.
- Submodules/connection drawing mode is choose.

The second is by clicking on this  icon in the icons tool bar.

The cursor then is moved into the module gray rectangle to create a box that is dedicated to a module.

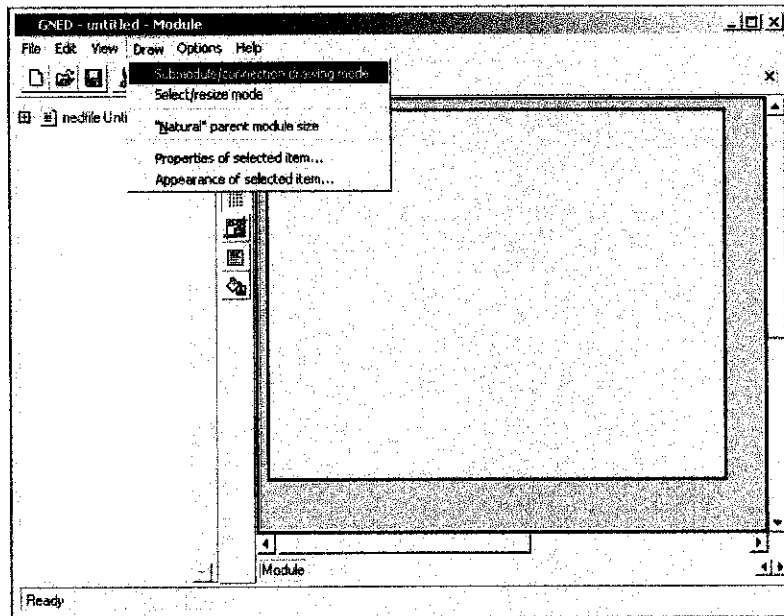


Figure 11 Draw menu

The cursor will be changed to a cross on the module gray rectangle. The cursor is drag according to the size that the users want. The simulator will create a box once the user finished dragging the cursor. The drawing will appear as the Figure 11.

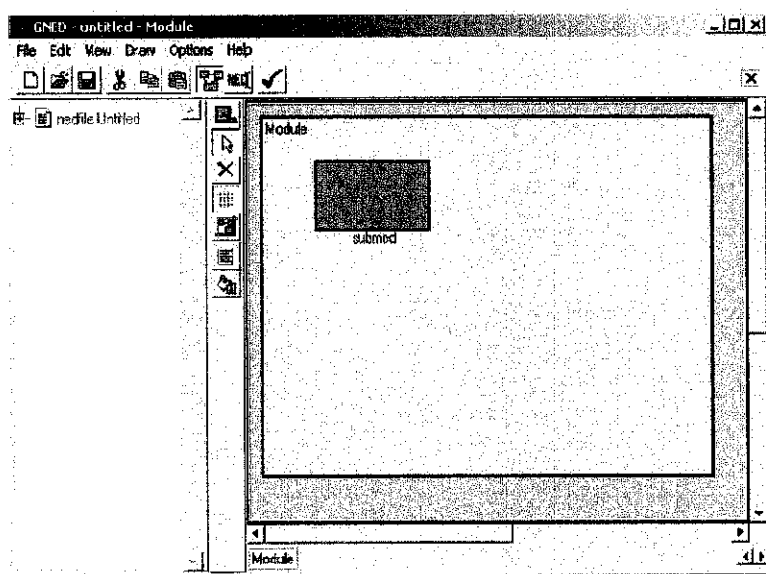


Figure 12 Drawing the first sub module

The sub-module that has just appeared is right click and a sub-menu will appear. There are five menus that will pop up once the user right click on the module:

Appearance: used to change the shape of the sub-module.

Properties: used to set the sub-module parameters.

Show ned code: used to view the ned code of the sub-module

Rename: used to change the sub-module name.

Delete: used to delete the sub-module.

The changing of the sub-module shape is started by clicking on **Appearance** and a small window will appear like in Figure 13.

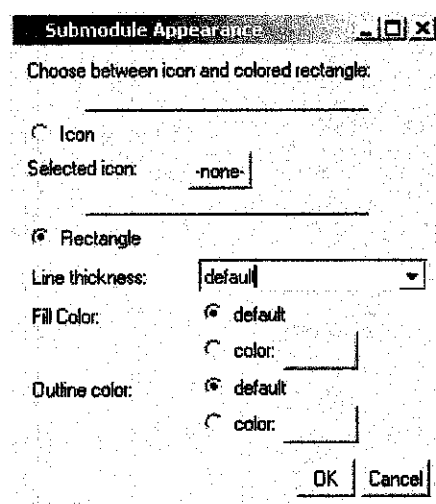


Figure 13 Sub-module appearances window

Air is the first sub module that was needed to build in this ad hoc system so the air icon is click from the list. After the icon is selected then it has to be renamed and it was renamed as 'Air'.

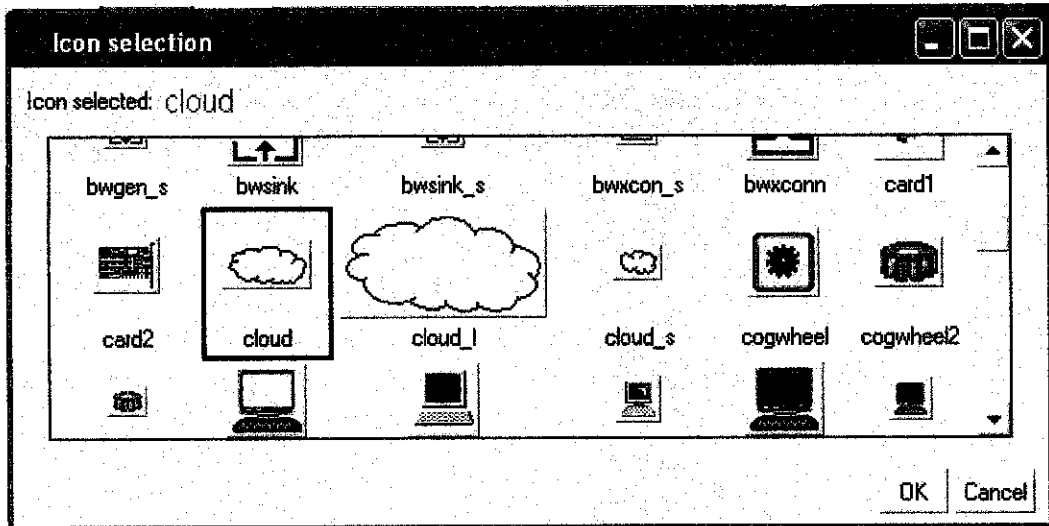


Figure 14 Icon selected

Step 2 Specifying the sub-module parameters:

This sub-module must be given some parameters to work with the desired way.

To set the sub-module parameters:

- The sub-module is **right click**.
- **Properties option** is choose.

The following window should appear:

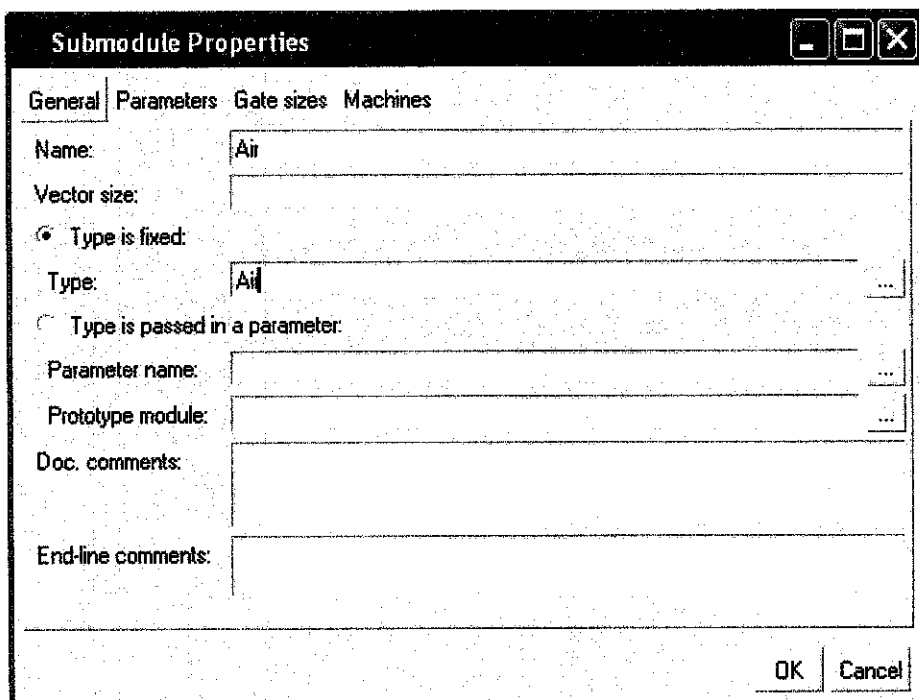


Figure 15 Sub module properties

The sub-module properties window contains several tabs; the important two are **General** and **Parameters**. In the **General** tab as in figure above **Name** and **Type** must be specified for the sub-module. The **Name** that had been specified must be unique. The **Type** must be fixed so **Type is fixed** is selected and the Type will be client (with small c). The type is not unique so it is possible that several sub-modules have the same type. Parameters option is clicked and a window like this should appear:

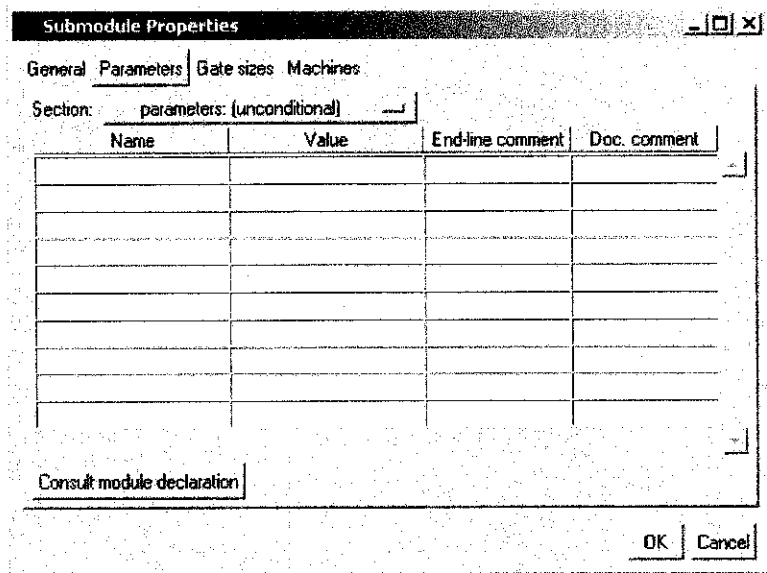


Figure 16 Sub module properties (Parameters tab)

The module parameters must be specified in this tab. However, the steps to this simulation are very dependent with the previous example that has been done in Ethernet simulation so it was assumed that the parameters are the same. The needed parameters for Air model are as follows:

- **No_Ports**: the number of ports in the air.
- **Queue_Len**: maximum queue length in the bridge.

The generated table will be like this:

Name	Value	End-line comment	Doc. comment
No_Ports	6		No of ports in the air.
Queue_Len	input		Max queue length in the bridge.

Table 3 Sub module parameters

Step 3 Specifying the gate size properties:

The gate size is used for the sub-modules that contain more than one input or one output gate. The number of gates must be specified. The gate size tab in the sub module properties is clicked and the window will appeared as in Figure 17:

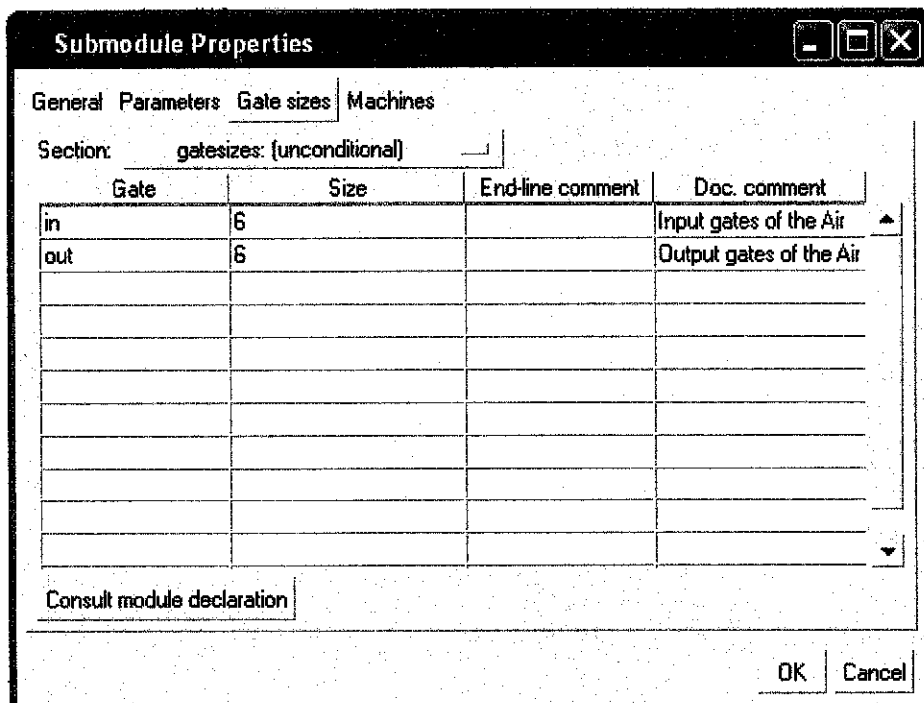


Figure 17 Gate size tab

The following data is needed to key in the values in the gate size tab:

Gate	Size	End-line comment	Doc. comment
in	6		Input gates of the air.
out	6		Output gates of the air.

Table 4 Air gate size

After adding the parameters and gate size then the module development is finished. The finished window will be as Figure 17:

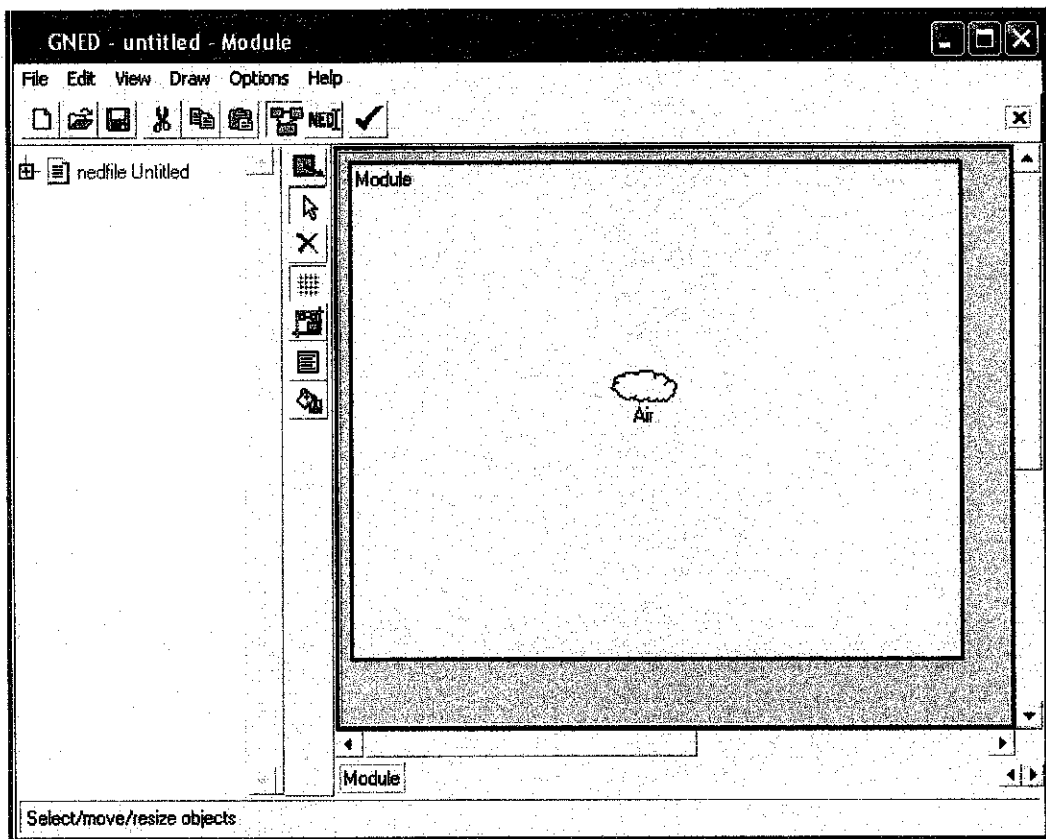


Figure 18 Finished creating Air module

Next is to create client module around the air module that was created before. The network model consists of several clients to create the environment for ad-hoc networks simulation. The steps to creating the client is the same as creating the air module but several changes must be done like the parameters need to be changed.

Step 4 Creating the client module

The step to create the client module is the same when to make the air module. The differences are the icon and parameters specified. The icon selected for the client module is shown in Figure 19:

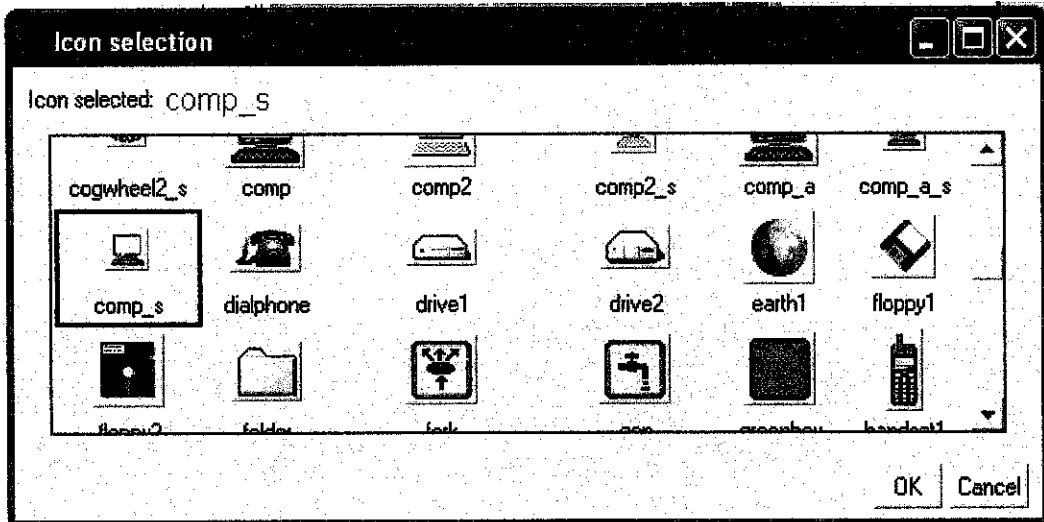


Figure 19 Icon selected

Next the parameters of this module must be specified just like the air module that is built before. The needed parameters to be defined in client module are:

- **address:** the unique address of the node.
- **frame_length:** the length of the generated frames.
- **ia_time:** the inter-arrival time of the generated frames.

The generated table will be like this:

Name	Value	End-line comment	Doc. comment
address	1		Address of the node
frame_length	input		Length of the generated frame
ia_time	input		ia time of the generated frame

Table 5 Sub module parameters

The same step is repeated for six clients. The finished topology of this ad hoc network is shown in Figure 20.

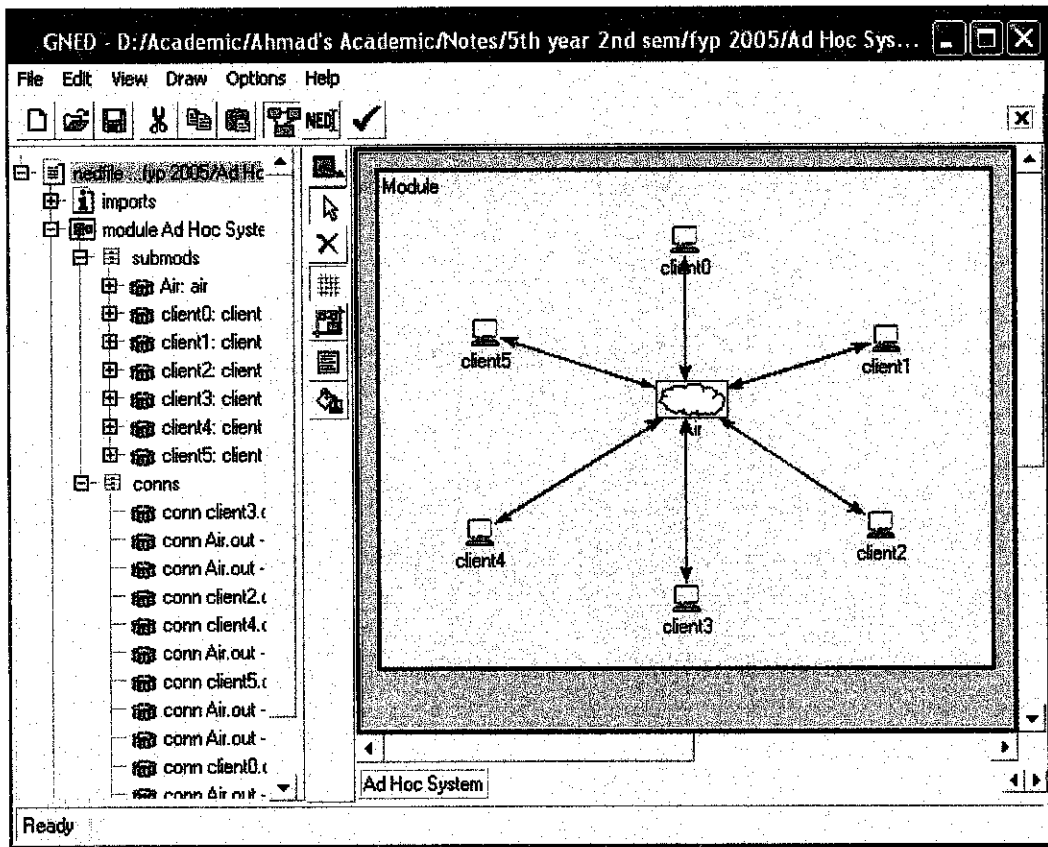


Figure 20 Ad Hoc network topology

Network Definition

The next step is to define the network name, parameters, and the needed sub-module files. This is done by clicking on the plus sign in the menu as shown in Figure 21:

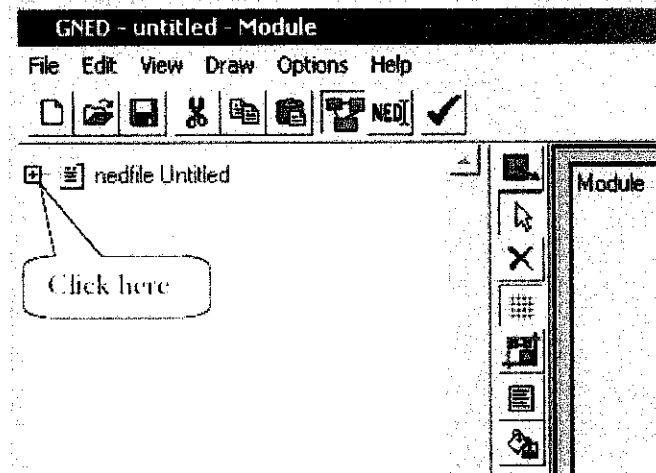


Figure 21 Opening the module contents

A sub-menu with the name **module Module** will appear. The next plus sign is clicked to be appeared like below:

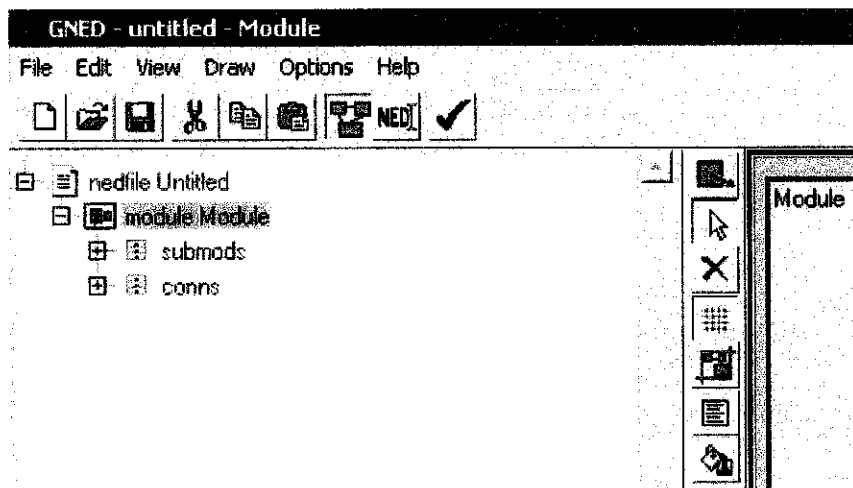


Figure 22 The submods and conns tabs

In the **submods** we can find all information about our sub-modules and their parameters.

In the **conns** tab we can know the connection between the sub-modules. It is shown in Figure 23:

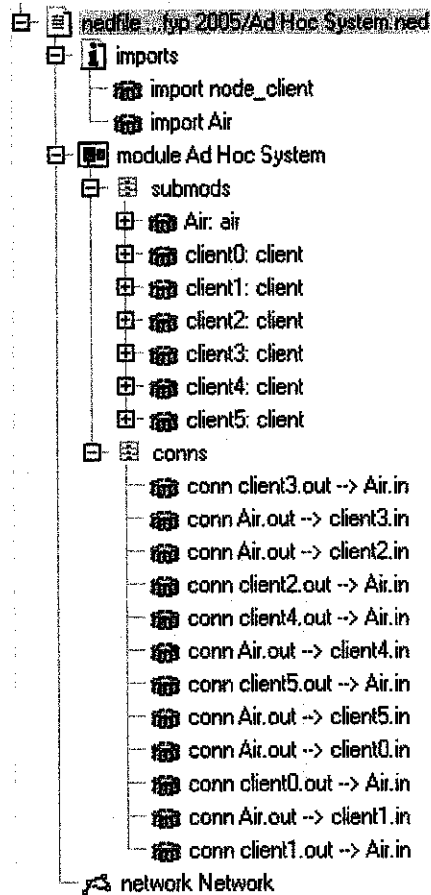


Figure 23 All model tabs

A menu will appear by right clicking on **module Module**. **Properties** is selected and a menu will appear:

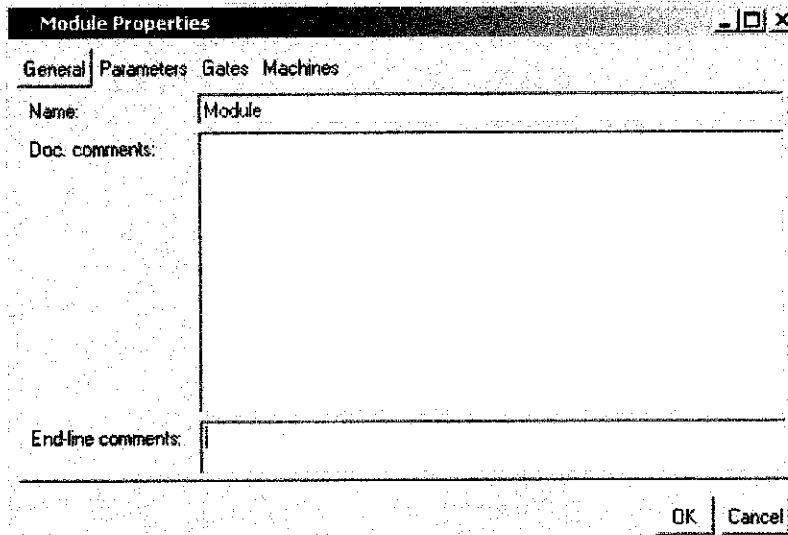


Figure 24 Module properties

The **Name** is changed to Ad Hoc System. The **module Module** will change to module Ad Hoc system:

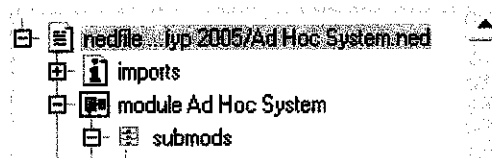


Figure 25 Changing the module name to Ad Hoc System

The next step is to add the sub-modules definition files. We follow this sequence

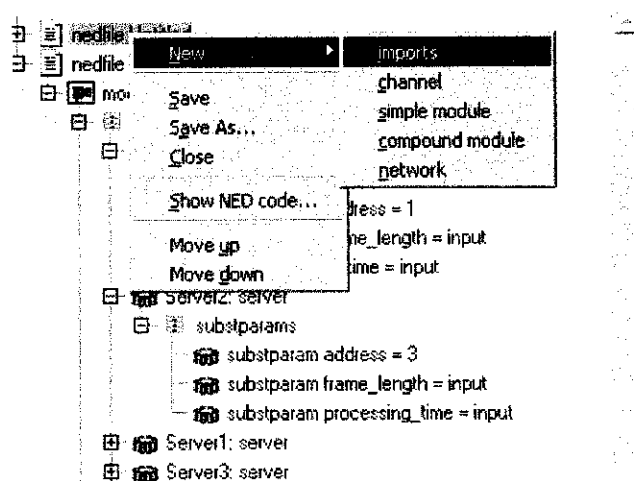


Figure 26 Adding the imports tab

Nedfile Untitled is right click and a sub-menu will appear, New is chosen and an import is selected:

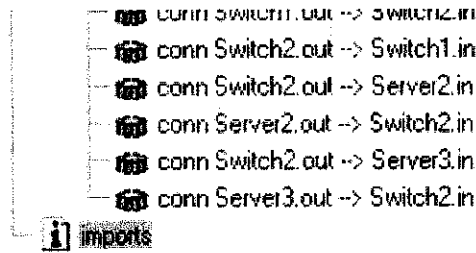


Figure 27 The model menu after adding the import tab

This **imports** is right click and properties is chosen and a window like following figure will appear:

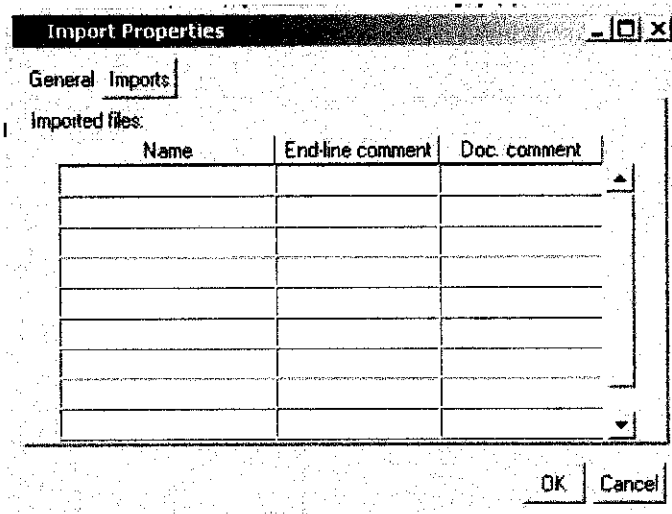


Figure 28 Import properties menu

This window will be used to add the name of the files, which the definition of the sub-modules model can be found. The window is filled like the following:

Name	End-line comment	Doc. comment
node_client		The client model file
node_server		The server model file
Switch		The switch model file

Table 6 The imported files

Imports tab is right click and **Move up** is choose to put the model definitions file at the top of the menu to prevent undefined modules error.

Each node must be specified a port in the switch. NED code must be edited to do this.

The NED button is click as shown in the figure:

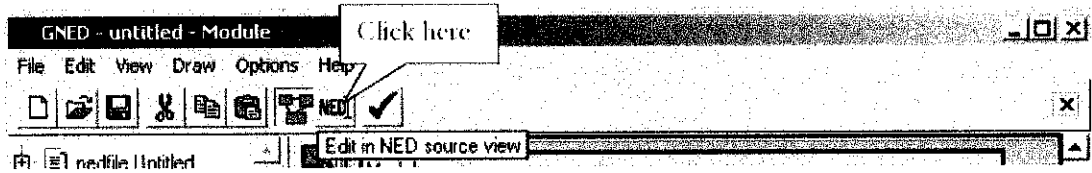


Figure 29 Switch to NED code mode

The last step to finish this model is to add a network to call the model using OMNeT++. To do this:

- nedfile Untitled is right click and a sub-menu will appear, **New** is choose and the **network** is selected. A tab with the name **network Network** will be added to the menu like the following figure:

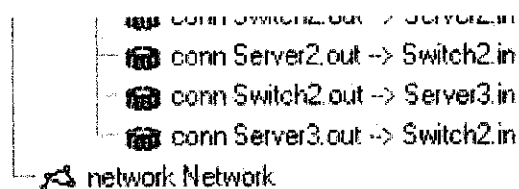


Figure 30 The network tab added

This **network Network** is right click and properties option is choose that should show the following figure:

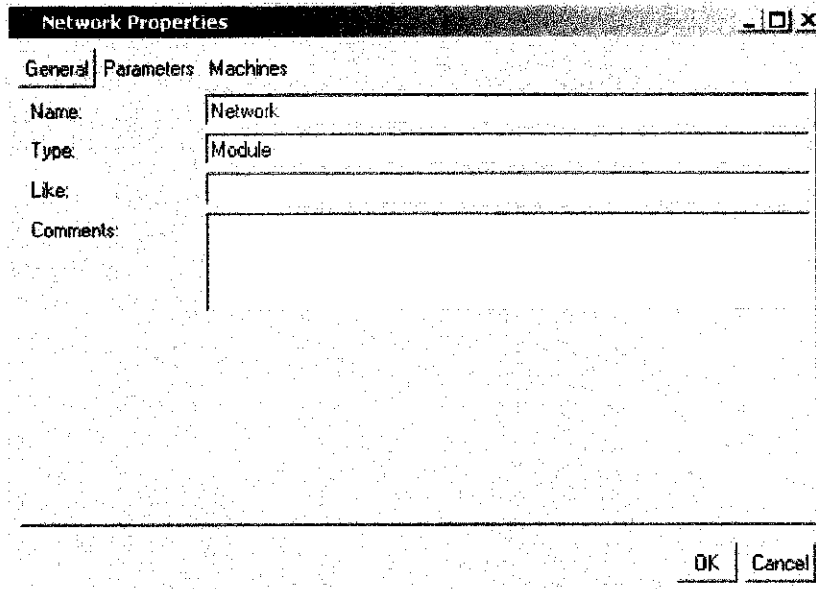


Figure 31 Network properties menu

The Type field is changed to Ad Hoc System. Nedfile Untitled is saved with the name Ad Hoc System.

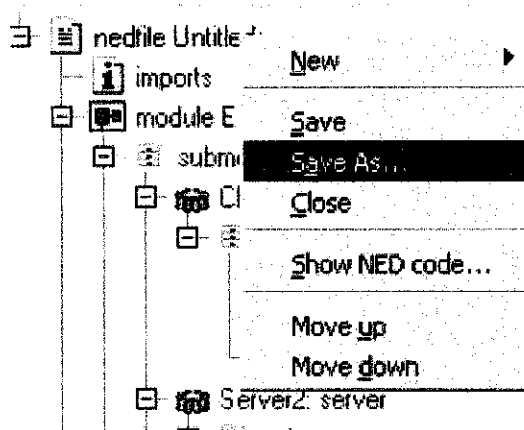


Figure 32 Saving the model file

4.1 Linking and Compiling the Model Files

Microsoft Visual C++ is used to link the entire model files. In this section the linking and compiling of the model will be described.

Visual C++ is started and on the **File** then **New** is chosen.

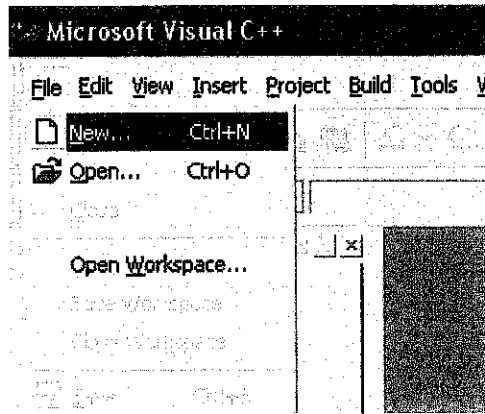


Figure 33 A new project created

A window like in the Figure 34 should appear.

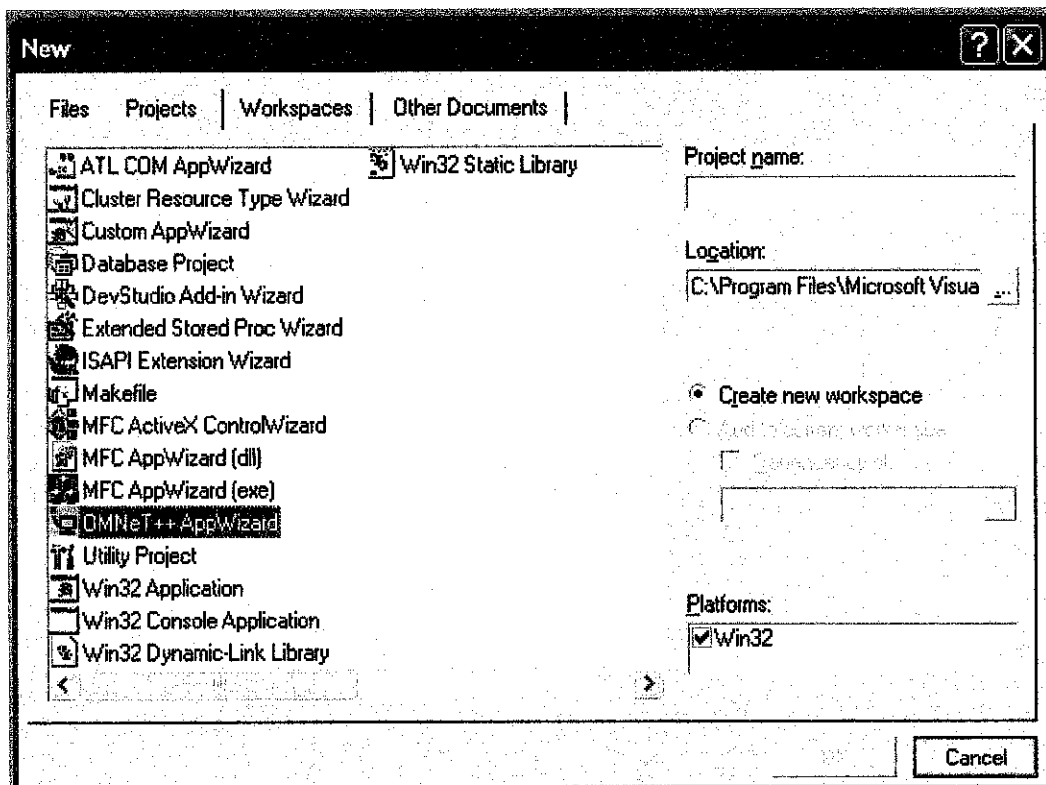


Figure 34 New project window

There are few steps before the user could proceed to the next step.

Step 1 OMNeT++ AppWizard is chosen

Step 2 The project name is plugged in as Ad Hoc System the same name of our NED model.

Step 3 On the location space, the path must be chosen the same folder which the user

saved the model NED files in.

Step 4 After that click **OK**.

After clicking OK the next window will appear as shown in Figure 35:

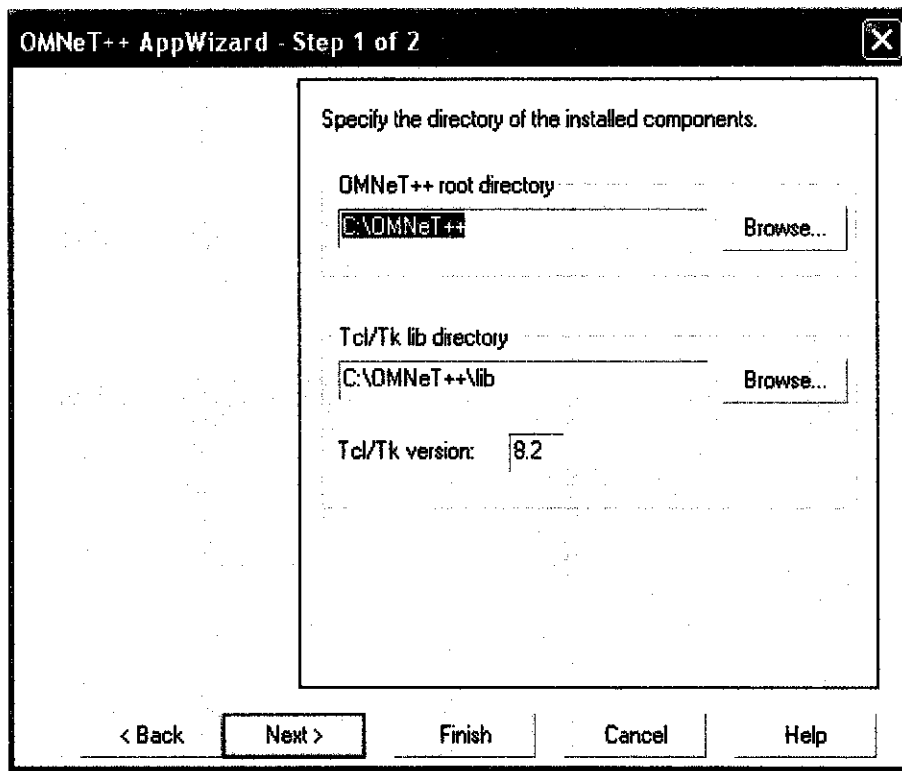


Figure 35 OMNeT++ AppWizard Page 1

The Tcl/Tk is changed to 8.4. After that Next button is click. Another following window like in Figure 36 should appear.

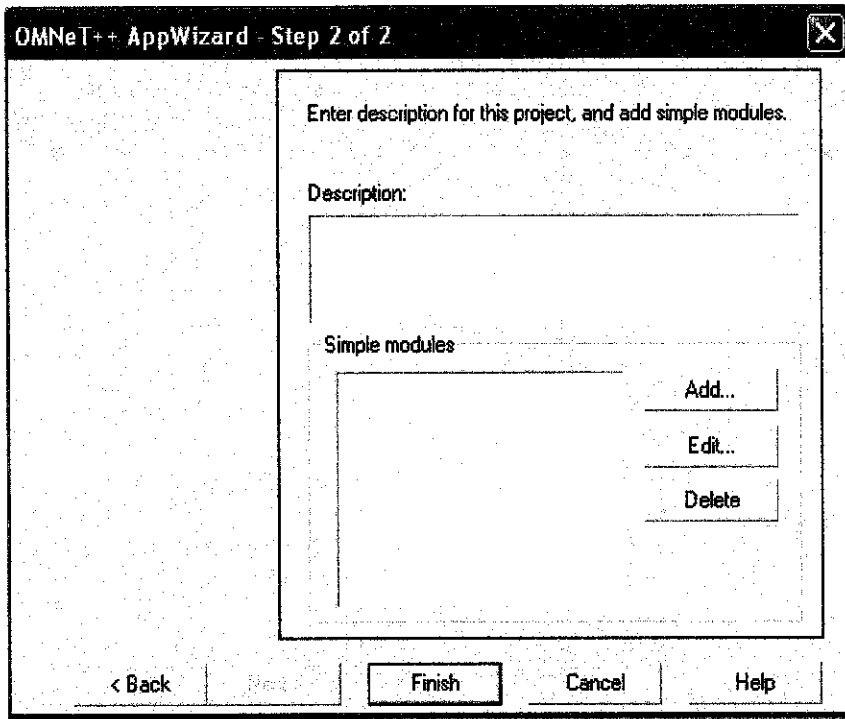


Figure 36 OMNeT++ AppWizard Page 2

The window is leave just it is and Finish is click. Then the following message should appear

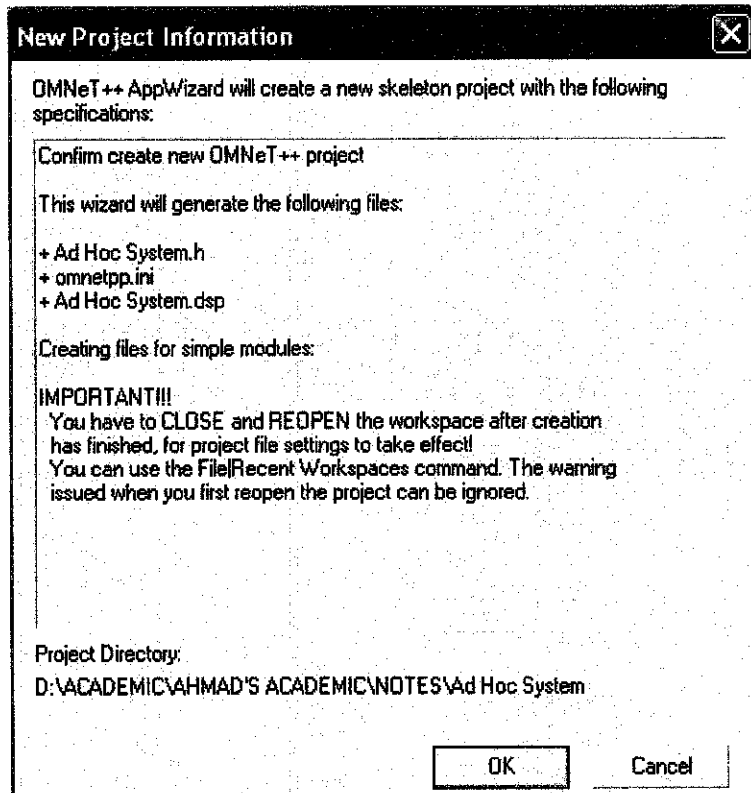


Figure 37 OMNeT++ Note

Then the project will have to close and reopen for the files settings to take effect. To close a project it is shown in the Figure 38 below:

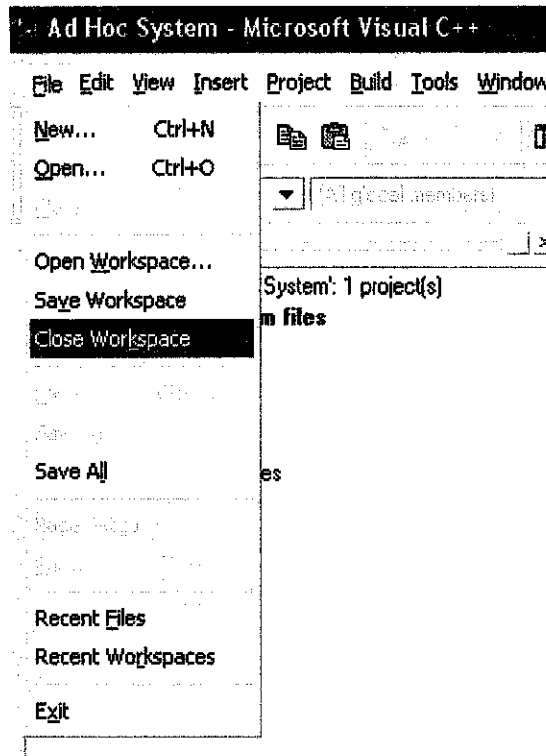


Figure 38 Closing a project in Visual C++

The linking process is described in the figure below. After the project is reopened again the following figure will appear to continue the linking process:

Step 1: File view is clicked.

Step 2: Plus sign of the Ad Hoc System is clicked and a list of files will appear as a source files.

Step 3: All the three files is deleted as shown in Figure 39.

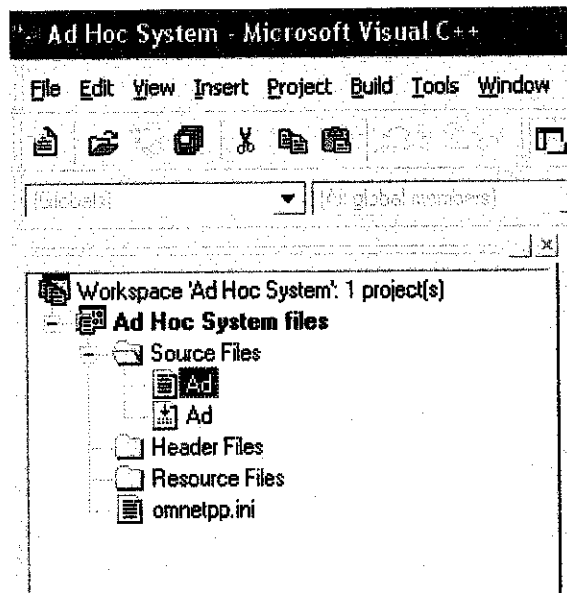


Figure 39 Ad Hoc System project file menu

As a rule of thumb to run the simulation all the model files in the Ad Hoc System must be copied in the Ad Hoc System directory and then add the file to the project. The file is prepared earlier using the NED compiler and some programming in C++. The files that are needed to build the project are as below:

From the Client directory: application. ned, application_client.cpp, and node_client.ned.

From the Air directory: application_air.ned, application_air.cpp, and node_air.cpp.

From the MAC directory: mac_in.ned, mac_in.cpp, mac_out.ned, and mac_out.cpp.

From the message_definition directory pack.msg.

And the last file that had been created earlier Ad Hoc System.ned.

These are the files that are needed in this project. After copying all this files to the Ad Hoc System project directory, next step is to add them to the project by right clicking on Ad Hoc System files. A small menu will appear and user will choose Add Files to Project as shown in the Figure 40:

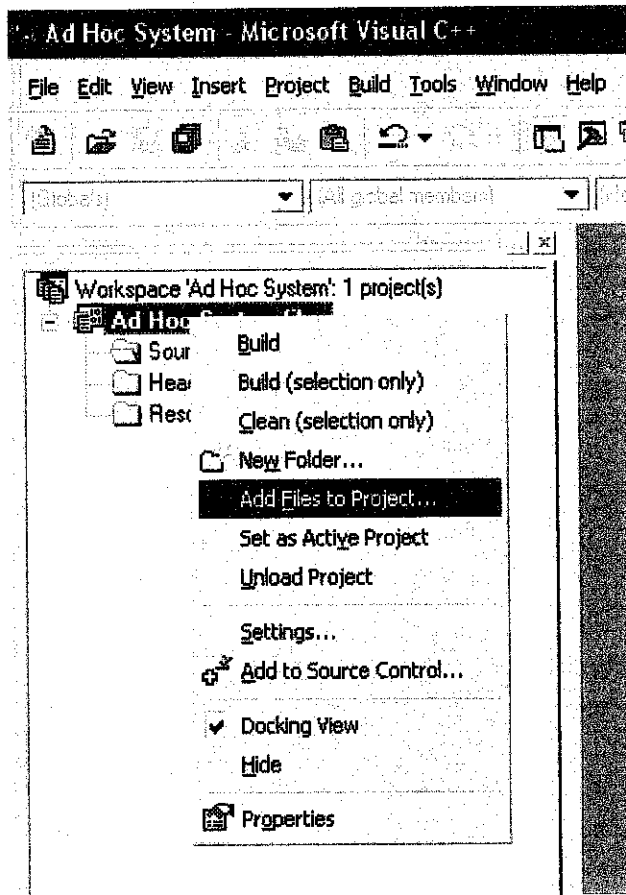


Figure 40 Importing files to the project

After adding all the files in the project a list of files in the visual C++ like in Figure 41 will appear:

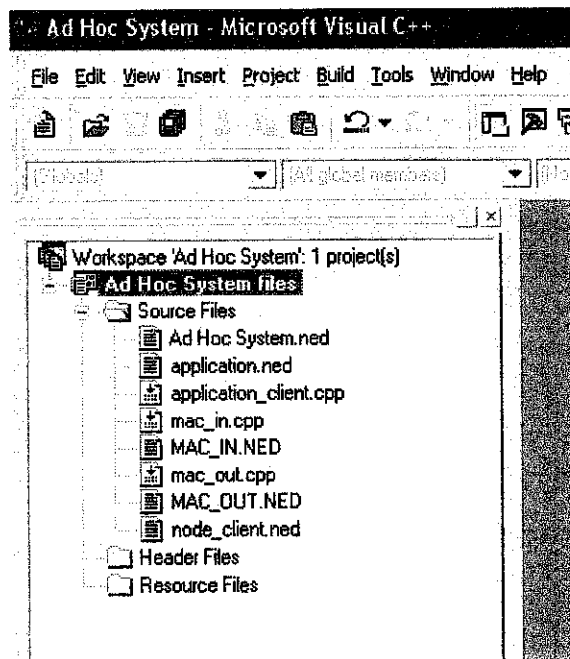


Figure 41 The files menu after importing the files

The following step is to set the properties of the project: To do this is by clicking on Project then followed by Settings or by pressing Alt+F7 on the keyboard.

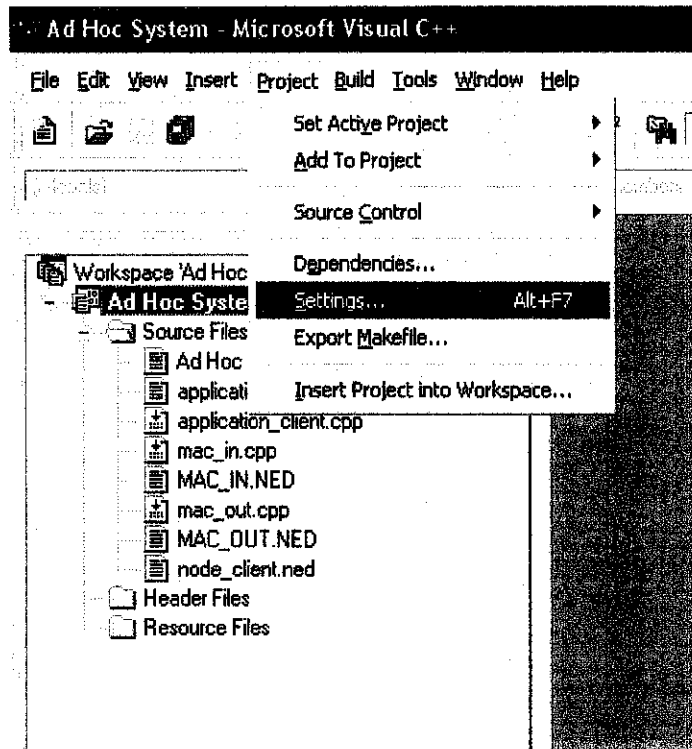


Figure 42 Setting the project properties

The following window of the Project properties will appear. To continue all configuration tab is selected and followed by the C/C++ tab as in the following figure shown:

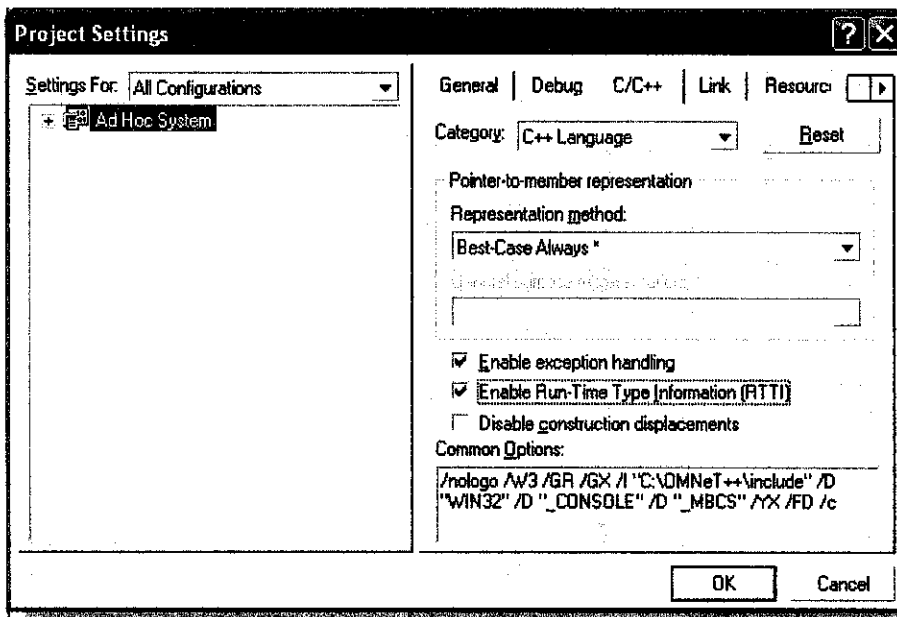


Figure 43 Setting the C/C++ properties

After that the Ethernet tab is opened to show all the sources files of the project. All the ned files is selected using the Ctrl key in the keyboard and mouse select. After selecting them Custom Build tab is selected. A window like in Figure 44 will appear:

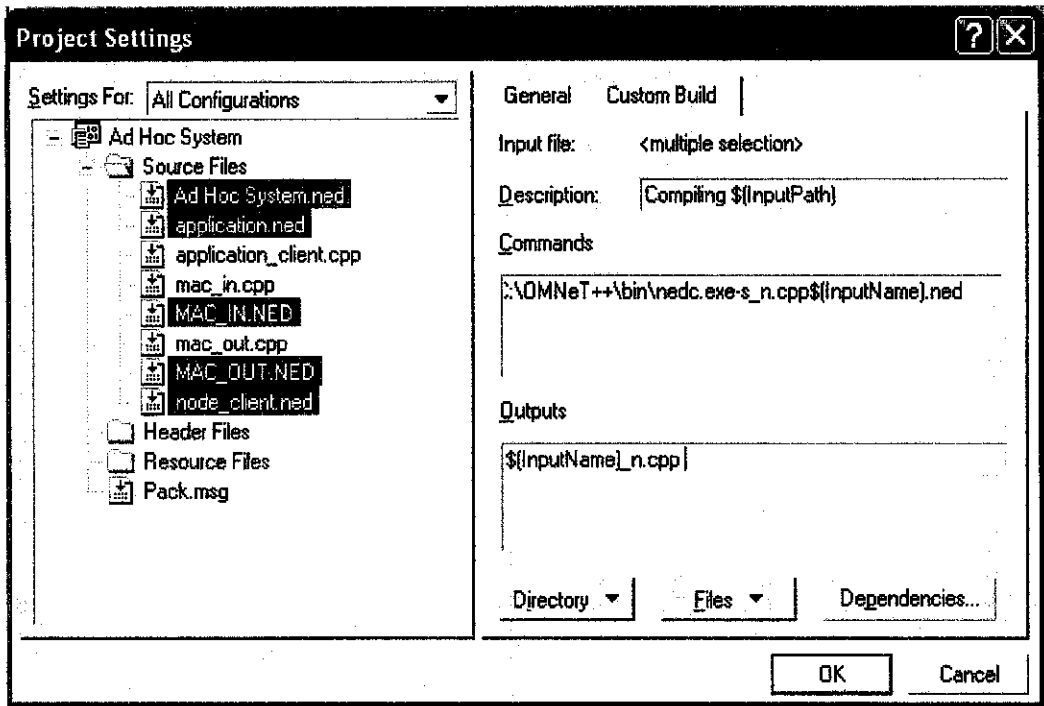


Figure 44 Setting the NED files compiler properties

This is a window after the properties have been added. The following lines are added:

In the Description **NED Compiling $\$(InputPath)$** is added.

In the Commands **C:\OMNeT++\bin\nedc.exe -s_n.cpp $\$(InputName).ned$** is added.

In the Outputs **$\$(InputName)_n.cpp$** is added.

The same procedure is applied to the Pack.msg file. The following line is added:

In the Description this line is added **Processing $\$(InputPath)$ with opp_msgc.**

In the Commands this line is added **C:\OMNeT++\bin\opp_msgc-s_m.cpp $\$(InputName).msg$.**

In the Outputs this line is added **$\$(InputName)_m.cpp$**

$\$(InputName)_m.h$

After the properties are added the following window should be like this:

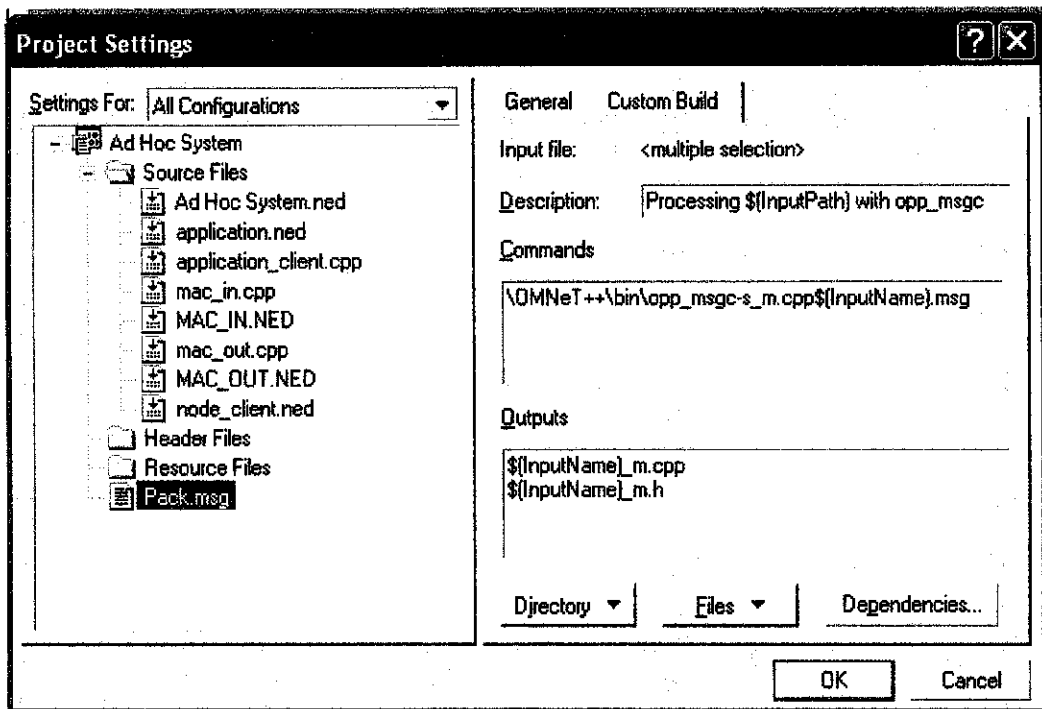


Figure 45 Setting the msg compiler properties

After this step is finished then all the ned files and Pack.msg is compile. The window will appear as in Figure 46:

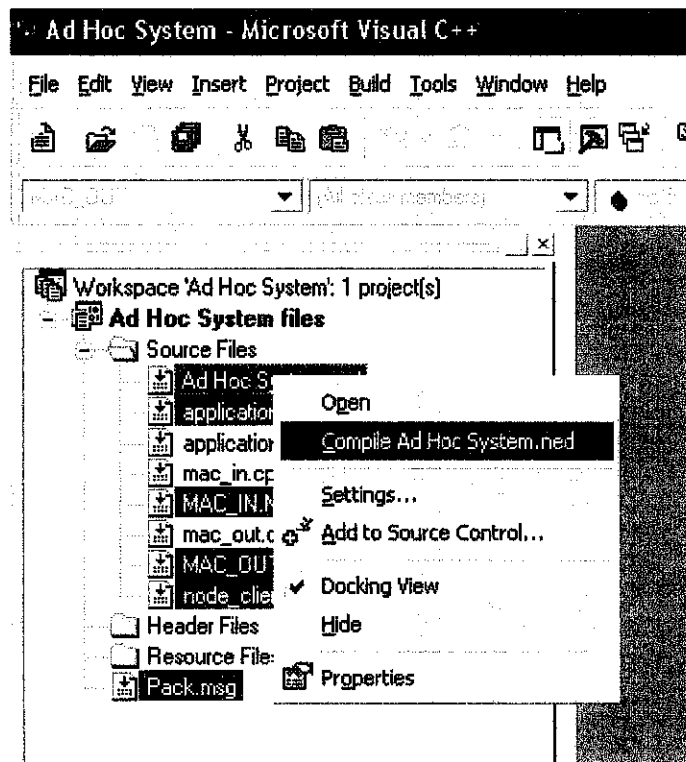


Figure 46 Compiling the .ned and .msg files

The last step is to produce the executable file of the simulation is to link all the files together by clicking on F7 on the keyboard. The output of the simulation is shown in Figure 47:

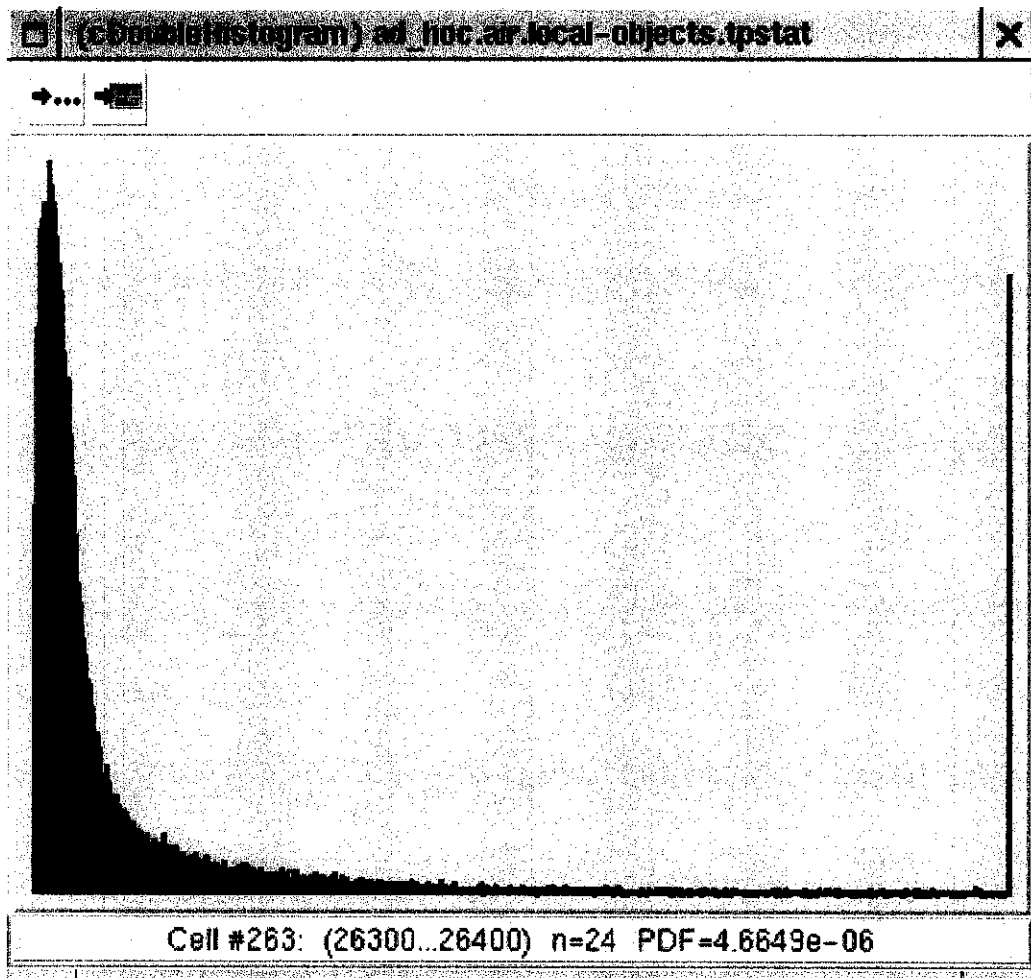


Figure 47 Histogram of the network throughput

CHAPTER 5

CONCLUSION AND RECOMMENDATION

As for conclusion this project serves as a basis for the implementation of multihop design in WLAN for future work. The works done to search and find the ways to implement the simulation is clearly shown and achieved in this project. All the materials needed for the simulation of wireless network like the instructions to simulate ad hoc networks. Because of the limitations in the network simulator only ad hoc model could be simulated while this is different from what we want study which is access point based network. The simulation of Ethernet network is the most significant achievement in this project. The simulation is the main reference to continue the wireless simulation. It shows how to use the network simulator and finally come out with a complete simulation. This is the main achievement in this project and the wireless simulation will have to continue in the future based on this Ethernet simulation. Other achievement from this project is the discovery of the enhanced MAC protocol for multihop wireless design in WLAN. The study shows how we can implement the common channel and dedicated channel to transmit data from sender to receiver. This idea could be implemented to enhanced the performance of the omnipresent MAC protocol so we could decrease the installation cost and extend the coverage.

5.1 Recommendations

1. To improve further the outcome of this project an example that presented clearly the mechanics of this project should be available because the title is new. The paper that relates to the project should be clear enough to tell how to simulate the network in OMNeT++. The multihop concept is not new in the field of communications but the implementation on the simulation is limited. Given the time for six month to learn the new software is not enough and moreover the simulator is very new to students unlike PsPice, MATLAB and

others.

2. Student must have very strong knowledge of C++ programming. Without it the example cannot be manipulated easily. To use and just run the simulation is not enough because our objectives are different from what the founder of this simulator has offered.

REFERENCES

- [1] Prasad, R., "CDMA for Wireless Personal Communications", Norwood, MA: Artech House, 1998.
- [2] Neeli Prasad, Anand Prasad., 2002, "WLAN Systems and Wireless IP for Next Generation Communication" London, Artech House.
- [3] William Stallings, 2004, "Data and Computer Communications" New Jersey, Pearson Prentice Hall.
- [4] Ramjee Prasad, Luis Munoz, 2003, "WLANs and WPANs towards 4G wireless" London, Artech House.
- [5] OMNeT++ User Manual by Andras Varga.
- [6] Benny Bing, "*Wireless Local Area Networks*", Department of Electrical and Computer Engineering, University of Maryland, USA.
- [7] Kenichi Mase, Yasunori Owada, "A Study on Protocol, Implementation and Throughput Evaluation for Multihop Wireless LAN", Proceedings of Vehicular Technology Conference, 2003. VTC 2003-Spring. The 57th IEEE Semiannual, Volume: 3, 22-25 April 2003 Pages: 1773 - 1777 vol.3.
- [8] Hao Zhu, Gouhong Cao, "On Improving the Performance of IEEE 802.11 with Multi-hop Concept", Proceedings of Computer Communications and Networks, 2003. ICCCN 2003. Proceedings. The 12th International Conference on, 20-22 Oct. 2003 Pages: 151 – 156

Manual

J. Jasperneite, E. Elsayed: Tutorial to Simulate Ethernet Network Using OMNeT++
V1.1

APPENDICES

Simulation Codes

MAC_in.cpp

```
# include <omnetpp.h>
# include "Pack_m.h"
class MAC_IN : public cSimpleModule
{
    public:
    Module_Class_Members(MAC_IN,cSimpleModule,8*1024)
    virtual void activity();

    int from_net_gate_id;

};

Define_Module(MAC_IN);

void MAC_IN::activity()
{

    cMessage *msg;
    from_net_gate_id = findGate("from_net");
    int my_address = par("address");

    for(;;)
    {
        msg = receive();
        // reception of the frame from the network.

        if(msg->arrivedOn(from_net_gate_id) )
        {

            Pack *pk = check_and_cast<Pack *>(msg);

            if(pk->getDestAddress() == my_address)
            {
                ev << "I'm client.I received pkt from server["<<pk-
                >getSrcAddress() <<"]" << endl;
                send(msg,"to_sink");
            }
            else
            {
                delete msg;
            }
        }
    }
}
```

MAC_out.cpp

```
// file eth_mac_client_out
```

```
# include <omnetpp.h>
```

```
# include "Pack_m.h"
```

```
class MAC_OUT : public cSimpleModule
```

```
{
```

```
    public:
```

```
    Module_Class_Members(MAC_OUT,cSimpleModule,8*1024)
```

```
    virtual void activity();
```

```
    int from_net_gate_id,to_net_gate_id;
```

```
};
```

```
Define_Module(MAC_OUT);
```

```
void MAC_OUT::activity()
```

```
{
```

```
    cQueue Aqueue ( "ALL Queue");
```

```
    cMessage *msg;
```

```
    to_net_gate_id = findGate("to_net");
```

```
    double datarate = 100000000;
```

```
    double service_time = 0;
```

```
    int msg_length = 0;
```

```
    int my_address = par("address");
```

```
    for (;;) 
```

```
    {
```

```
        while ( !Aqueue.empty() )
```

```
        {
```

```
            msg = (cMessage *)Aqueue.pop();
```

```
            Pack *pk = check_and_cast<Pack *>(msg);
```

```
            pk->setSrcAddress(my_address);
```

```
            msg_length=msg->length();
```

```
            service_time = (msg_length+96)/datarate; //interframe gap
```

```
            added
```

```
            waitAndEnqueue(service_time,&Aqueue);
```

```
            send(pk,to_net_gate_id);
```

```
        }
```

```
msg = receive();
```

```
Aqueue.insert(msg);
```

```
}
```

```
}
```

Application client.cpp

```
// file eth_app ( messages generator ).
```

```
# include <omnetpp.h>
```

```
# include "Pack_m.h"
```

```
class eth_app_client : public cSimpleModule
```

```
{
```

```
    Module_Class_Members(eth_app_client,cSimpleModule,8192)  
    virtual void activity();
```

```
};
```

```
Define_Module(eth_app_client)
```

```
void eth_app_client::activity()
```

```
{
```

```
    for(;;)
```

```
    {
```

```
        double ia_time = par("ia_time");  
        wait( (double) ia_time );
```

```
        int destAddress = par("dest_address");  
        int frame_length = par("frame_length");
```

```
        Pack *pk = new Pack("eth_frame");  
        pk->setLength( frame_length );  
        pk->setDestAddress(destAddress);  
        pk->setKind(0);  
        pk->setTimestamp();
```

```
        send(pk,"out");
```

```
    }
```

```
}
```

Application air.cpp

```
// file eth_app ( messages generator ).

# include <omnetpp.h>
# include "Pack_m.h"

class eth_app_server : public cSimpleModule
{
    Module_Class_Members(eth_app_server,cSimpleModule,16*1024)
    virtual void activity();
};

Define_Module(eth_app_server)

void eth_app_server::activity()
{
    double Processing_time = par("processing_time");

    cMessage *msg;
    int destAddress ;
    int frame_length = par("frame_length");
    long this_frame_length = frame_length;
    cOutVector Dtime("One-way Delay time vector");

    for(;;)
    {
        msg = receive();
        Pack *pk = check_and_cast<Pack *>(msg);
        destAddress = pk->getSrcAddress();
        double d = simTime()- msg->timestamp();
        Dtime.record( d );
        double e = msg->timestamp();

        //sprintf(buf,"To:%li
        Length:%li",dest_address,this_frame_length);
        ev<<"reply To client from server:["<< pk->getDestAddress()
        << "], Length:["<< this_frame_length<<"]"<<endl;
        delete msg;

        Pack *RPmsg = new Pack("eth_frame");
        RPmsg->setLength( this_frame_length );
        RPmsg->setDestAddress(destAddress);
        RPmsg->setKind(1);
        RPmsg->setTimestamp();
        RPmsg->setTime(e);
        wait(Processing_time);
    }
}
```



```
send(RPmsg,"out");
```

```
}
```

Pack_m.h

```
#ifndef PACK_M_H_
#define PACK_M_H_
```

```
#include <omnetpp.h>
```

```
class Pack : public cMessage
{
protected:
    int srcAddress;
    int destAddress;
    double Time;
    int modID;
public:
    Pack(const char *name=NULL, int kind=0);
    Pack(const Pack& other);
    virtual ~Pack();
    Pack& operator=(const Pack& other);
    virtual cObject *dup() const {return new Pack(*this);}

    // field getter/setter methods
    virtual int getSrcAddress() const;
    virtual void setSrcAddress(int srcAddress);
    virtual int getDestAddress() const;
    virtual void setDestAddress(int destAddress);
    virtual double getTime() const;
    virtual void setTime(double Time);
    virtual int getModID() const;
    virtual void setModID(int modID);
};
```

```
#endif // PACK_M_H_
```

OMNeT++.ini

[General]

```
network = Network
ini-warnings = no
random-seed = 1
warnings = yes
output-vector-file = EtherNet.vec
sim-time-limit = 3s
cpu-time-limit = 6000s
total-stack-kb = 2048 ; 2MByte, increase if necessary
```

[Tkenv]

default-run=1
use-mainwindow = yes
print-banners = yes
slowexec-delay = 300ms
update-freq-fast = 10
update-freq-express = 100
breakpoints-enabled = yes

[DisplayStrings]

[Parameters]

Network.Client.frame_length=1000
Network.Client.dest_address=intuniform(2,4)
Network.Client.ia_time=0.01
Network.Switch1.Queue_Len=20
Network.Switch1.forwarding_delay=0
Network.Switch2.Queue_Len=20
Network.Switch2.forwarding_delay=0
Network.Server1.processing_time=0
Network.Server1.frame_length=1000
Network.Server2.processing_time=0
Network.Server2.frame_length=1000
Network.Server3.processing_time=0
Network.Server3.frame_length=1000