

**PERFORMANCE EVALUATION OF WiFi  
WITH AND WITHOUT QoS**

By

NUR AZMINA BINTI KAMARUDDIN

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 2007

by

Nur Azmina Binti Kamaruddin, 2007

# **CERTIFICATION OF APPROVAL**

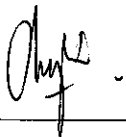
## **PERFORMANCE EVALUATION OF WiFi WITH AND WITHOUT QoS**

by

Nur Azmina Binti Kamaruddin

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:



---

Dr. Lee Sheng Chyan  
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS  
TRONOH, PERAK

December 2007

## **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.

*Azmina*

---

Nur Azmina Binti Kamaruddin

## **ABSTRACT**

Wireless local area networks (WLANs) have been around for a long time but only recently have they become popular. Despite the fact that Wireless LANs have achieved a tremendous amount of growth in recent years, the performance is very poor. Hence, multimedia wireless network QoS support has become one of the most important researches. In order to improve the poor performance of existing system, QoS features and MAC enhancements are needed in the upcoming 802.11e standard. This project aims to evaluate the performance of Wi-Fi systems with and without QoS, and quantify how well the new enhancement can support applications that require certain QoS guarantees. A thorough research on the IEEE 802.11 standards is necessary to the success of this project, as well as an immaculate and extensive study on the QoS performance of the network. All the studies and evaluation is being done with a simulation using OMNeT++. The project requires knowledge of the WiFi architecture, C++ programming language, setting up simulation of a network in OMNeT++, and then evaluating the QoS performance.

## **ACKNOWLEDGEMENTS**

First of all, thank you God, that I have come this far in this project. This project would not have been possible without the support of many people. I would like to express my sincere gratitude to Dr. Lee Sheng Chyan, my supervisor, for his invaluable guidance and encouragement extended throughout the study. His tenacious supervision, helpful suggestion, patience and time deserve a special mention. I would also like to thank my colleagues for their suggestions during the progress of this thesis. To all lecturers and coordinators, thank you for organizing the FYP Talks in order to guide the students in completing the project. I would like to gratefully acknowledge Department of Electrical & Electronics Engineering, Universiti Teknologi PETRONAS for providing the resources and needs during the thesis. Last but not least, I thank my parents for their continued support and encouragement in the course of doing my degree.

## TABLE OF CONTENTS

CERTIFICATION OF APPROVAL .....	ii
CERTIFICATION OF ORIGINALITY .....	iii
ABSTRACT.....	iv
ACKNOWLEDGEMENTS.....	v
LIST OF TABLES .....	viii
LIST OF FIGURES.....	ix
LIST OF ABBREVIATIONS .....	x
CHAPTER 1 INTRODUCTION .....	1
1.1 Background of Study.....	1
1.2 Problem Statement.....	4
1.3 Objective and Scope of Study .....	5
1.3.1 Objective.....	5
1.3.2 Scope of Study .....	5
CHAPTER 2 LITERATURE REVIEW .....	7
2.1 802.11 MAC Layer.....	7
2.2 CDMA/CA .....	10
2.3 Distributed Coordination Function (DCF) .....	11
2.4 EDCF .....	14
2.5 IEEE 802.11 a/b/g Standards.....	17
2.6 QoS.....	19
2.6.1 Problems in Transmission.....	20
CHAPTER 3 METHODOLOGY.....	21
3.1 Procedure Identification .....	21
3.2 Software Required .....	23
CHAPTER 4 RESULTS AND DISCUSSION .....	24
4.1 Preliminary Work in OMNeT++ .....	24
4.2 Basic CDMA/CA simulation.....	29
4.3 Nclients Module .....	31
4.4 Tracing and Analyzing Models Using TCP .....	35
4.5 IEEE 802.11 Model .....	38
4.6 Expected Result .....	40

4.7 Project Difficulties.....	42
CHAPTER 5 CONCLUSION AND RECOMMENDATION.....	43
REFERENCES.....	44
APPENDICES.....	45
Appendix A C++ algorithm for functionality of tex1.....	46
Appendix B NED files for NClients model.....	47

## LIST OF TABLES

Table 2.1: Comparisons of IEEE 802.11 a/b/g Standards.....	18
--	----



## LIST OF FIGURES

Figure 1.1: "IEEE 802.11 and the ISO Model" .....	2
Figure 2.1: CSMA/CA Explanation .....	8
Figure 2.2: DCF .....	11
Figure 2.3: IEEE 802.11e EDCF channel access. ....	15
Figure 2.4: Four access categories (ACs) for EDCF.....	15
Figure 2.5: CFB timing structure. ....	16
Figure 3.1: Flowcharts of Procedures .....	21
Figure 4.1: Properties of a Module.....	24
Figure 4.2: Sub module Properties .....	25
Figure 4.3: Connection Properties.....	25
Figure 4.4: NED File for TicToc1 .....	26
Figure 4.5: Command Prompt Window .....	26
Figure 4.6: Simulation of TicToc 1 .....	28
Figure 4.7: DCF Access Scheme.....	29
Figure 4.8: Example of sending acknowledgement .....	30
Figure 4.9: 802.11 Model in OMNeT++ .....	31
Figure 4.10: NClients Module – Transferring Files .....	32
Figure 4.11: Tkenv file for NClients Simulation .....	32
Figure 4.12: Output Scalar of NClients Simulation .....	33
Figure 4.13: Output Scalar Charts for NClients.....	33
Figure 4.12: FlatNet Simulation.....	34
Figure 4.13: Bulk Transfer Simulation .....	34
Figure 4.14: REDTest Simulation.....	35
Figure 4.15: Plove windows for Round-Trip Time.....	35
Figure 4.16: Round-Trip Time Plot.....	36
Figure 4.17: Sequence Number Plot .....	36
Figure 4.18: IEEE 80211 NiC Models .....	38
Figure 4.19: Throughput vs. DFS Usage.....	40
Figure 4.20: Quite Period Ends With Collision .....	41

## LIST OF ABBREVIATIONS

<b>ACK</b>	Acknowledgement
<b>AP</b>	Access Point
<b>CDMA-CD</b>	Carrier Detection Multiply Access – Collision Detection
<b>CDMA-CA</b>	Carrier Detection Multiply Access – Collision Avoidance
<b>CSMA/CA</b>	Carrier Sense Multiple Access/Collision Avoidance
<b>CSMA/CD</b>	Carrier Sense Multiple Access/Collision Detection
<b>CTS</b>	Clear-to-Send
<b>DCF</b>	Distributed Coordination Function
<b>DIFS</b>	Distributed Interframe Space
<b>DS</b>	Direct-Sequence
<b>EDCF</b>	Enhanced-DCF
<b>ESS</b>	Extended Service Set
<b>FH</b>	Frequency-Hopping
<b>HCF</b>	Hybrid Coordination Function
<b>LAN</b>	Local Area Network
<b>MAC</b>	Media Access Control
<b>PCF</b>	Pattern Compressed File
<b>QoS</b>	Quality of Service
<b>RTS</b>	Request-to-Send
<b>SIFS</b>	Short Interframe Space
<b>SAP</b>	Service Access Point
<b>TCP/IP</b>	Transmission Control Protocol/Internet Protocol
<b>WiFi</b>	Wireless Fidelity, IEEE802.11

# **CHAPTER 1**

## **INTRODUCTION**

This chapter will describe more clearly on the background information and context of QoS Performance Evaluation.

### **1.1 Background of Study**

A wireless LAN (WLAN or WiFi) is a data transmission system designed to provide location-independent network access between computing devices by using radio waves rather than a cable infrastructure.

In the corporate enterprise, wireless LANs are usually implemented as the final link between the existing wired network and a group of client computers, giving these users wireless access to the full resources and services of the corporate network across a building or campus setting.

The first generation 802.11™ wireless market, once struggling to expand, has spread from largely vertical applications such as healthcare, point of sale, and inventory management to become much more broad as a general networking technology being deployed in offices, schools, hotel guest rooms, airport departure areas, airplane cabins, entertainment venues, coffee shops, restaurants, and homes. This has led to the tremendous growth of new sources of IEEE 802.11 devices.

Like all IEEE 802 standards, the 802.11 standards focus on the bottom two levels the ISO model, the physical layer and link layer (see Figure 1 below). Any LAN application, network operating system, protocol, including TCP/IP and Novell NetWare, will run on an 802.11-compliant WLAN as easily as they run over Ethernet.

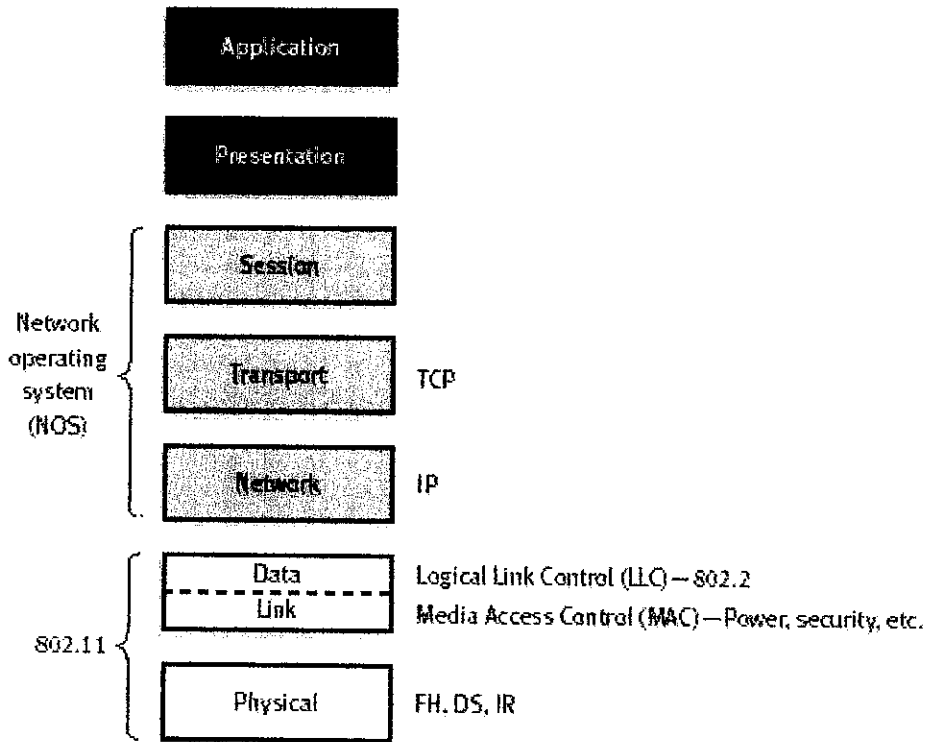


Figure 1.1: "IEEE 802.11 and the ISO Model"

The major motivation and benefit from Wireless LANs is increased mobility. Not confined from conventional network connections, network users can move about almost without restriction and access LANs from nearly anywhere. WLANs liberate users from dependence on hard-wired access to the network backbone, giving them anytime, anywhere network access. [4]

The following IEEE 802.11 standards exist or are in development to support the creation of technologies for wireless local area networking:

- **802.11a** - 54 Mbps standard, 5 GHz signaling (ratified 1999)
- **802.11b** - 11 Mbps standard, 2.4 GHz signaling (1999)
- 802.11c - operation of bridge connections (moved to 802.1D)
- 802.11d - worldwide compliance with regulations for use of wireless signal spectrum (2001)
- **802.11e** - Quality of Service (QoS) support (not yet ratified)
- 802.11F - Inter-Access Point Protocol recommendation for communication between access points to support roaming clients (2003)

- **802.11g** - 54 Mbps standard, 2.4 GHz signaling (2003)
- 802.11h - enhanced version of 802.11a to support European regulatory requirements (2003)
- 802.11i - security improvements for the 802.11 family (2004)
- 802.11j - enhancements to 5 GHz signaling to support Japan regulatory requirements (2004)
- 802.11k - WLAN system management (in progress)
- 802.11l - skipped to avoid confusion with 802.11i
- 802.11m - maintenance of 802.11 family documentation
- 802.11n - future 100+ Mbps standard (in progress)
- 802.11o - skipped
- 802.11p - Wireless Access for the Vehicular Environment
- 802.11q - skipped
- 802.11r - fast roaming support via Basic Service Set transitions
- 802.11s - ESS mesh networking for access points
- 802.11T - Wireless Performance Prediction - recommendation for testing standards and metrics
- 802.11u - internetworking with 3G / cellular and other forms of external networks
- 802.11v - wireless network management / device configuration
- 802.11w - Protected Management Frames security enhancement
- 802.11x - skipped (generic name for the 802.11 family)
- 802.11y - Contention Based Protocol for interference avoidance

## **1.2 Problem Statement**

Despite the fact that Wireless LANs have achieved a tremendous amount of growth in recent years, the performance is very poor. The IEEE 802.11 WLAN legacy standard cannot provide QoS support for multimedia applications. Thus, considerable research efforts have been carried out to enhance QoS support for 802.11.

Multimedia wireless network QoS support has become one of the most important researches in recent years. Among them, 802.11e is the upcoming QoS-enhanced standard proposed by the IEEE working group. In order to improve the poor performance of existing system, QoS features and MAC enhancements are needed in the upcoming 802.11e standard.

IEEE wireless 802.11e is found as the most hopeful among all the proposed methods. However, 802.11e experiment performance metrics are essentially network-level measures. They represent the network state, not the perceptual quality of the end user.

## 1.3 Objective and Scope of Study

### 1.3.1 Objective

It is known that basic WiFi systems based on 802.11 a/b/g standards have poor performance in terms of QoS provisioning. The *main objective* of this project is to evaluate the performance of WiFi systems with and without QoS, and quantify how well the new enhancement can support applications that require certain QoS guarantees.

### 1.3.2 Scope of Study

This final year project (FYP) will be covered in two semesters. For the first semester for FYP1, the students are expected to come out with a project title, identified the problems and study a few alternatives on how to solve the problem.

During the first semester, as the first step, deeper studies on the WiFi architecture need to be done. A few topics related to WiFi are also being studied, such as the MAC Layer and Handshaking in access methods. Then, the scope will be focusing on the available IEEE 802.11 Standards. Comparison will be made by analyzing the performance of each standard like 802.11 a/b/g. In order to come out with a simulation of a real network, the OMNeT++ software and Microsoft Visual C++ will be much in used. Therefore, for this first semester, a lot of time is needed to learn the both application. A lot of exercises such as the Tutorial exercises need to be done as the preliminary work.

For the second semester, a simulation of IEEE 802.11a with and without QoS will be done in OMNeT++. The performance of the network will be observed and studied. From the simulation, the performance can be observed and comparisons will be made. From the comparison, it will be proved that the WiFi Systems with QoS will perform better.

As an overall, this project requires the programming skills such as C++ to do the simulation of a real-working environment of a WiFi in OMNeT++. It also expects the

knowledge of WiFi architecture, MAC Layer, Handshaking methods, the available IEEE 802.11 standards, and the QoS performance.



## CHAPTER 2

### LITERATURE REVIEW

#### 2.1 802.11 MAC Layer

The 802.11 family uses a MAC layer known as CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance). Classic Ethernet uses CSMA/CD - (Carrier Sense Multiple Access/Collision Detection). CSMA/CA is, like all Ethernet protocols, peer-to-peer, where there is no requirement for a master station [5].

In CSMA/CA a Wireless node that wants to transmit performs the following sequence:

1. Listen on the desired channel.
2. If channel is idle (no active transmitters) it sends a packet.
3. If channel is busy (an active transmitter) node waits until transmission stops then a further **CONTENTION** period. The Contention period is a random period after every transmission on every node and statistically allows every node equal access to the media. To allow tx to rx turn around the contention time is **slotted** 50 micro sec for Frequency-Hopping (FH) systems and 20 micro sec for Direct-Sequence (DS) systems.
4. If the channel is still idle at the end of the **CONTENTION** period the node transmits its packet otherwise it repeats the process defined in 3 above until it gets a free channel.

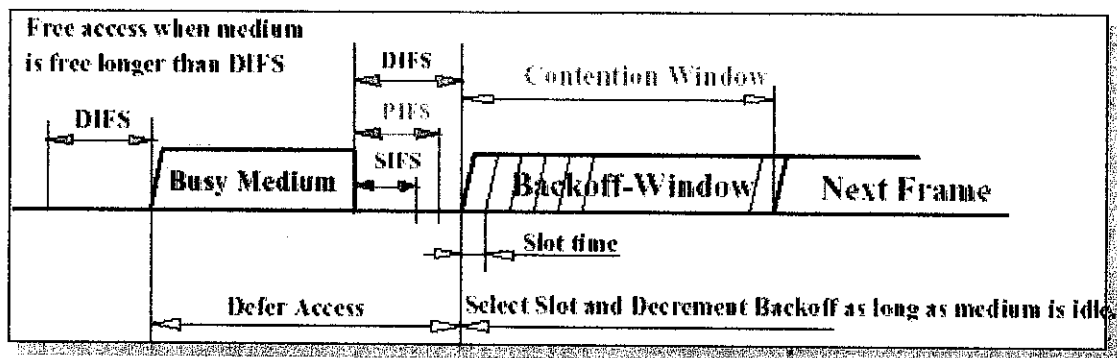


Figure 2.1: CSMA/CA Explanation

CSMA/CA Explanation are as below:

- Reduce collision probability where mostly needed.
  - Stations are waiting for medium to become free.
  - Select Random Backoff after a Defer, resolving contention to avoid collisions.
- Efficient Backoff algorithm stable at high loads.
  - Exponential Backoff window increases for retransmissions.
  - Backoff timer elapse only when medium is idle.
- Implement different fixed priority levels.
  - To allow immediate responses and PCF coexistence.

To improve efficiency additional features are employed:

- i. Positive Acknowledgement (ACK)

At the end of every packet the receiver, if it has successfully received the packet, will return an ACK packet (if not received or received with errors the receiver will NOT respond). The transmit window allows for the ACK i.e. CONTENTION period starts after the ACK should have been sent.

## ii. MAC level retransmission

If no ACK is received the sender will retry to transmit (using the normal CSMA/CA procedures) until either successful or the operation is abandoned with exhausted retries.

## iii. Fragmentation

Bit error rates on wireless systems ( $10^{-5}$ ,  $10^{-6}$ ) are substantially higher than wire-line systems ( $10^{-12}$ ). Large blocks may approach the number of bits where the probability of an error occurring may = 1 i.e. every block could fail including the re-transmission. To reduce the possibility of these happening large blocks may be fragmented by the transmitter and reassembled by the receiver node e.g. a 1500 byte block (12,000 bits) may be fragmented into 5 blocks of 300 bytes (2,400 bits). While there is some overhead in doing this - both the probability of an error occurring is reduced and, in the event of an error, the re-transmission time is also reduced.

## 2.2 CDMA/CA

In newer technologies CDMA is often in use. In CDMA the whole bandwidth is always in use. Signals are separated with codes. Codes are (pseudo)random binary sequences that are used to modulate the signal. Receiver knows the code and can use it to extract the information. Other signals (modulated with different codes) seem like noise to the receiver.

Ethernet uses CDMA-CD (Carrier Detection Multiply Access – Collision Detection), WLAN CDMA-CA (Collision Avoidance). In Ethernet, host first listens to the medium (cable) to find out if it is free. If it is, host can start sending. If someone happens to start sending at the same time, there is collision. Both parties back off and try again little later. There isn't any kind of reservation system.

In CDMA-CA host who wants to send something announces his intent with RTS (Request-to-Send) frame. Receiver replies with CTS (Clear-to-Send) letting other host know who has permission to send. This minimizes the risk of collision and also shortens the wasted time in case of RTS collision. This also solves so called hidden-station problem.

Differences between CDMA-CA and -CD are because they have different medium. Cable is easy to control and everybody knows if there is a collision. Radio networks on the other hand suffer from multitude of disturbances and noise. Because of hidden station problem it is sometimes impossible to notice all the collisions. [6]

## 2.3 Distributed Coordination Function (DCF)

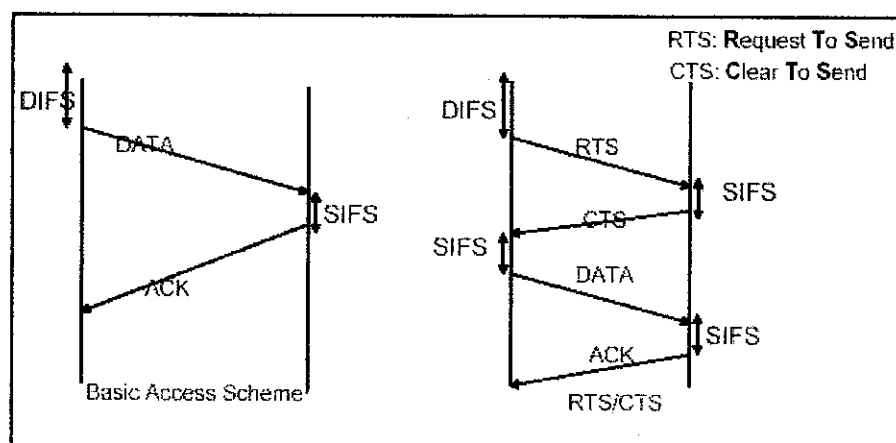


Figure 2.2: DCF

DCF defines a **basic** two-way handshaking access mechanism and an **optional** four-way handshaking mechanism. The basic mechanism is mandatory for all IEEE802.11 implementation. The optional mechanism is not implemented in some wireless card [7].

A station with a *new packet* to transmit monitors the channel activity. If the channel is idle for a period of time equal to a distributed interframe space (DIFS), the station transmits. Otherwise, if the channel is sensed busy (either immediately or during the DIFS), the station persists to monitor the channel until it is measured idle for a DIFS. At this point, the station generates a random backoff interval before transmitting (this is the Collision Avoidance feature of the protocol), to minimize the probability of collision with packets being transmitted by other stations. In addition, to avoid channel capture, a station must wait a random backoff time between *two consecutive new packet transmissions*, even if the medium is sensed idle in the DIFS time.

For efficiency reasons, DCF employs a discrete-time backoff scale. The time immediately following an idle DIFS is slotted, and a station is allowed to transmit only at the beginning of each slot time. The slot time size is set equal to the time needed at any station to detect the transmission of a packet from any other station. The value depends on the physical layer (specified in IEEE802.11 specification), and it accounts for the propagation delay, for the time needed to switch from the receiving

receiving station detects an RTS frame, it responds, after a SIFS, with a clear to send (CTS) frame. The transmitting station is allowed to transmit its packet only if the CTS frame is correctly received. The frames RTS and CTS carry the information of the length of the packet to be transmitted. This information can be read by any listening station, which is then able to update a network allocation vector (NAV) containing the information of the period of time in which the channel will remain busy. Therefore, when a station is *hidden* from either the transmitting or the receiving station, by detecting just one frame among the RTS and CTS frames, it can suitably delay further transmission, and thus avoid collision [7].

receiving station detects an RTS frame, it responds, after a SIFS, with a clear to send (CTS) frame. The transmitting station is allowed to transmit its packet only if the CTS frame is correctly received. The frames RTS and CTS carry the information of the length of the packet to be transmitted. This information can be read by any listening station, which is then able to update a network allocation vector (NAV) containing the information of the period of time in which the channel will remain busy. Therefore, when a station is *hidden* from either the transmitting or the receiving station, by detecting just one frame among the RTS and CTS frames, it can suitably delay further transmission, and thus avoid collision [7].

## 2.4 EDCF

The 802.11 legacy MAC does not support the concept of differentiating frames with different priorities. Basically, the DCF is supposed to provide a channel access with equal probabilities to all stations contending for the channel access in a distributed manner. However, equal access probabilities are not desirable among stations with different priority frames. The emerging EDCF is designed to provide differentiated, distributed channel accesses for frames with 8 different priorities (from 0 to 7) by enhancing the DCF. As distinct from the legacy DCF, the EDCF is not a separate coordination function. Rather, it is a part of a single coordination function, called the Hybrid Coordination Function (HCF), of the 802.11e MAC. The HCF combines the aspects of both DCF and PCF. All the detailed aspects of the HCF are beyond the scope of this paper as we focus on the HCF contention-based channel access, i.e., EDCF.

Each frame from the higher layer arrives at the MAC along with a specific priority value. Then, each QoS data frame carries its priority value in the MAC frame header. An 802.11e STA shall implement four access categories (ACs), where an AC is an enhanced variant of the DCF 0. Basically, an AC uses AIFSD[AC], CWmin[AC], and CWmax[AC] instead of DIFS, CWmin, and CWmax, of the DCF, respectively, for the contention process to transmit a frame belonging to access category AC. AIFSD[AC] is determined by

$$\text{AIFSD[AC]} = \text{SIFS} + \text{AIFS[AC]} \cdot \text{SlotTime},$$

where AIFS[AC] is an integer greater than zero. Moreover, the backoff counter is selected from  $[1, 1 + \text{CW[AC]}]$ , instead of  $[0, \text{CW}]$  as in the DCF.



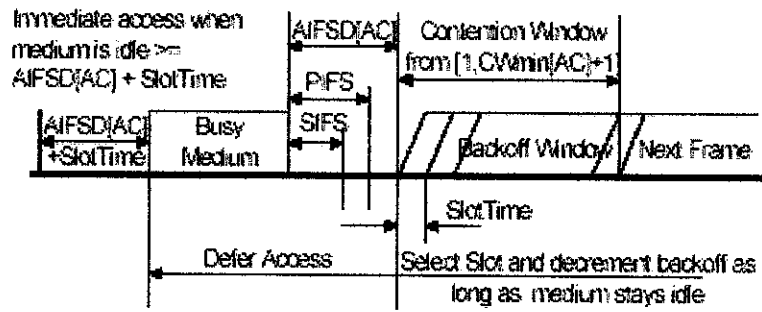


Figure 2.3: IEEE 802.11e EDCF channel access.

Figure 4 shows the timing diagram of the EDCF channel access. The values of  $AIFS[AC]$ ,  $CWmin[AC]$ , and  $CWmax[AC]$ , which are referred to as the EDCF parameters, are announced by the AP via beacon frames. The AP can adapt these parameters dynamically depending on network conditions. Basically, the smaller  $AIFS[AC]$  and  $CWmin[AC]$ , the shorter the channel access delay for the corresponding priority, and hence the more capacity share for a given traffic condition. However, the probability of collisions increases when operating with smaller  $CWmin[AC]$ . These parameters can be used in order to differentiate the channel access among different priority traffic.

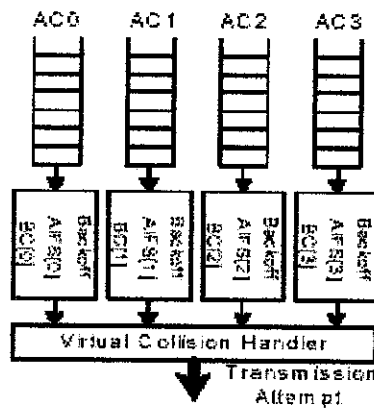


Figure 2.4: Four access categories (ACs) for EDCF.

Figure 5 shows the 802.11e MAC with four transmission queues, where each queue behaves as a single enhanced DCF contending entity, i.e., an AC, where each queue has its own AIFS and maintains its own Backoff Counter BC. When there is more than one AC finishing the backoff at the same time, the collision is handled in a virtual manner. That is, the highest priority frame among the colliding frames is chosen and transmitted, and the others perform a backoff with increased CW values.

The IEEE 802.11e defines a transmission opportunity (TXOP) as the interval of time when a particular STA has the right to initiate transmissions. Along with the EDCF parameters of  $AIFS[AC]$ ,  $CWmin[AC]$ , and  $CWmax[AP]$ , the AP also determines and announces the limit of an EDCF TXOP interval for each AC, i.e.,  $TXOPLimit[AC]$ , in beacon frames. During an EDCF TXOP, a STA is allowed to transmit multiple MPDUs from the same AC with a SIFS time gap between an ACK and the subsequent frame transmission. This multiple MPDU transmission is referred to as “Contention-Free Burst (CFB).”

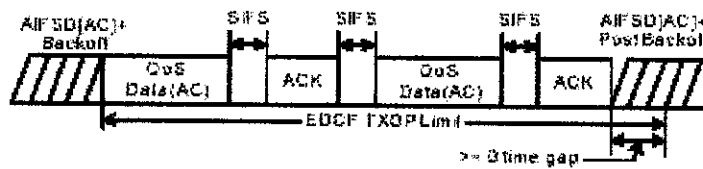


Figure 2.5: CFB timing structure.

Figure 6 shows the transmission of two QoS data frames during an EDCF TXOP, where the whole transmission time for two data and ACK frames is less than the EDCF TXOP limit announced by the AP. As multiple MSDU transmission honors the TXOP limit, the worst-case delay performance is not be affected by allowing the CFB. We show below that CFB increases the system throughput without degrading other system performance measures unacceptably as long as the EDCF TXOP limit value is properly determined. [8]

## 2.5 IEEE 802.11 a/b/g Standards

Before proceeding with evaluation of 802.11a, all the existing 802.11 a/b/g standards are being studied first. Here are some findings on the 802.11 a/b/g [9]:

IEEE expanded on the original 802.11 standard in July 1999, creating the **802.11b** specification. 802.11b supports bandwidth up to 11 Mbps, comparable to traditional Ethernet. 802.11b uses the same radio signaling frequency - 2.4 GHz - as the original 802.11 standard. Being an unregulated frequency, 802.11b gear can incur interference from microwave ovens, cordless phones, and other appliances using the same 2.4 GHz range. However, by installing 802.11b gear a reasonable distance from other appliances, interference can easily be avoided. Vendors often prefer using unregulated frequencies to lower their production costs.

When 802.11b was developed, IEEE created a second extension to the original 802.11 standard called **802.11a**. Because 802.11b gained in popularity much faster than did 802.11a, some people believe that 802.11a was created after 802.11b. In fact, 802.11a was created at the same time. Due to its higher cost, 802.11a is usually found on business networks whereas 802.11b better serves the home market.

802.11a supports bandwidth up to 54 Mbps and signals in a regulated frequency spectrum around 5 GHz. This higher frequency compared to 802.11b limits the range of 802.11a networks. The higher frequency also means 802.11a signals have more difficulty penetrating walls and other obstructions. Because 802.11a and 802.11b utilize different frequencies, the two technologies are incompatible with each other. Some vendors offer hybrid **802.11a/b** network gear, but these products simply implement the two standards side by side (each connected devices must use one or the other).

In 2002 and 2003, WLAN products supporting a newer standard called **802.11g** began to appear on the scene. 802.11g attempts to combine the best of both 802.11a and 802.11b. 802.11g supports bandwidth up to 54 Mbps, and it uses the 2.4 GHz frequency for greater range. 802.11g is backwards compatible with 802.11b, meaning

that 802.11g access points will work with 802.11b wireless network adapters and vice versa.

Table 2.1: Comparisons of IEEE 802.11 a/b/g Standards

<b>IEEE 802.11 Standards</b>	<b>PROS</b>	<b>CONS</b>
802.11a	<ul style="list-style-type: none"> <li>fastest maximum speed; supports more simultaneous users; regulated frequencies prevent signal interference from other devices</li> </ul>	<ul style="list-style-type: none"> <li>highest cost; shorter range signal that is more easily obstructed</li> </ul>
802.11b	<ul style="list-style-type: none"> <li>lowest cost; signal range is best and is not easily obstructed</li> </ul>	<ul style="list-style-type: none"> <li>slowest maximum speed; supports fewer simultaneous users; appliances may interfere on the unregulated frequency band</li> </ul>
802.11g	<ul style="list-style-type: none"> <li>fastest maximum speed; supports more simultaneous users; signal range is best and is not easily obstructed</li> </ul>	<ul style="list-style-type: none"> <li>costs more than 802.11b; appliances may interfere on the unregulated signal frequency</li> </ul>

## 2.6 QoS

QoS stands for "Quality of Service". It is a means to prioritize network traffic, to help ensure that the most important data gets through the network as quickly as possible.

QoS works by slowing unimportant packets down, or in the cases of extreme network traffic, throwing them away entirely. This leaves room for important packets to reach their destination as quickly as possible. Basically, once router is aware of how much data it can enqueue on the modem at any given time, it can "shape" traffic by delaying unimportant packets and "filling the pipe" with important packets FIRST, then using any leftover space to fill the pipe up in descending order of importance.

Since QoS cannot possibly speed up a packet, basically what it takes total available upstream bandwidth, calculate how much of the highest priority data it has, put that in the buffer, then go down the line in priority until it runs out of data to send or the buffer fills up. Any excess data is held back or "requeued" at the front of the line, where it will be evaluated in the next pass.

"Importance" is determined by the priority of the packet. Priorities range from "Low" or "Bulk" (depending on the router) to "High" or "Premium". The number of levels and the exact terminology depends on router.

QoS packets may be prioritized by a number of criteria, including generated by applications themselves, but the most common techniques with Consumer grade routers are MAC Address, and TCP/IP Port.

**MAC Address** prioritizes network devices by their Media Access Address (MAC Address). This is a long string associated with network card or other network device.

Simply enter the MAC address and the priority and the router takes care of the rest.

**TCP/IP Port** allows some level of control over applications, rather than devices. For example, web browsing (port 80) should get priority over FTP (ports 20 and 21).

### 2.6.1 Problems in Transmission

When looking at packet-switched networks, Quality of service is affected by various factors, which can be divided into "human" and "technical" factors. Human factors include: stability of service, availability of service, delays, user information. Technical factors include: reliability, scalability, effectiveness, maintainability, Grade of Service, etc .Many things can happen to packets as they travel from origin to destination, resulting in the following problems as seen from the point of view of the sender and receiver:

- **Dropped packets**
  - When the buffers are already full, some, none, or all of the packets might be dropped. The retransmission will cause severe delays.
  
- **Delay**
  - It is caused by holding up in long queues, or taking a less direct route to avoid congestion. Excessive delay can render an application, such as VoIP, and it is impracticable.
  
- **Jitter**
  - A packet's delay varies with its position in queues of the routers along the path. This variation in delay is known as jitter and can seriously affect the quality of streaming
  
- **Out-of-order delivery**
  - different routes → different delay → packets arrive in a different order
  - This necessitates special additional protocols for rearranging out-of-order packets. Quality of video and VoIP streams is dramatically affected by both latency and lack of isochronicity.
  
- **Error**
  - Sometimes packets are misdirected/combined together/or corrupted, while *en route*. The receiver has to detect this and ask for retransmission.

## CHAPTER 3 METHODOLOGY

### 3.1 Procedure Identification

Below is the flow of methodology approach in completing the project:

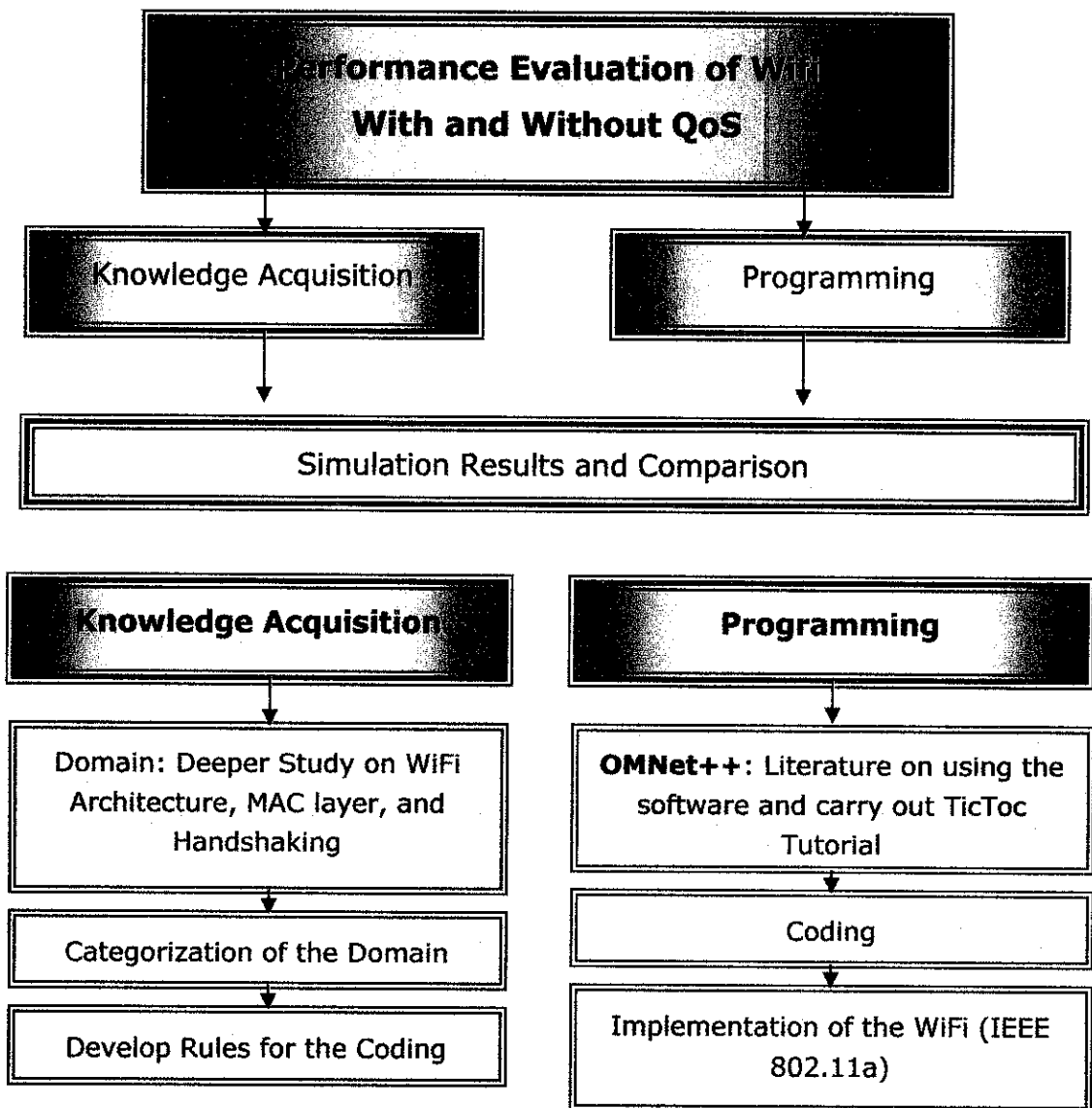


Figure 3.1: Flowcharts of Procedures

Figure 3.1 show the flowchart representation of the steps undertaken in the implementation that have evolved during this project. This project starts with problem identification, where the problem is analyzed and the possible solution is proposed. Then, the project has been divided into two parts which is Knowledge Acquisition and Programming.

For the first part, knowledge acquisition, literature review or research has been done based on the solution proposed. This includes reviewing journals and articles in the library and the internet. The domain has covered on WiFi Architecture, MAC layer, and Handshaking. After some reviews, the domain is categorized. At this stage, some coding rules or parameters for the OMNeT++ can be developed.

By the end of first part, some ideas on how to implement tasks will be gathered. However, since this project involves modeling as well as programming. The skills and knowledge on using the OMNeT++ and Visual C++ are very important. In the second part of the project, the main focus is to learn the new software and exercise using the TicToc Tutorial. Then, it is followed by building simulations. It will require some coding in .NED, .ini, and C++ files.

Finally, after the system has been implemented, a comparison will be made according to the performance.



## 3.2 Software Required

Here are the tools needed for this project:

- Microsoft Visual C++
- Ghostscript
- OMNeT ++

**OMNeT++** is an object-oriented modular discrete event simulator. The name itself stands for Objective Modular Network Tested in C++. OMNeT++ has its distant roots in OMNet, a simulator written in Object Pascal by Dr. György Pongor. The simulator can be used for:

- ✓ *traffic modeling of telecommunication networks*
- ✓ *protocol modeling*
- ✓ *modeling queueing networks*
- ✓ *modeling multiprocessors and other distributed hardware systems*
- ✓ *validating hardware architectures*
- ✓ *evaluating performance aspects of complex software systems*
- ✓ *modeling any other system where the discrete event approach is suitable.*

It can be said that OMNet++ has enough features to make it a good alternative to most network simulation tools, and it has a strong potential to become one of the most widely used network simulation packages in academic and research environments.

All simulations of a real-working environment of a WiFi will be built in OMNET++.

## CHAPTER 4

### RESULTS AND DISCUSSION

#### 4.1 Preliminary Work in OMNeT++

For a software familiarization, the exercises in TicToc Tutorial have been done. It took quite a while to get familiarized with the software for a first time user. Here are some of basic examples from the exercises in TicToc Tutorial.

The first basic simulation is connecting one node to another node in a network. For the simulation in OMNeT++, the connection between the two nodes can be created by using Graphics or NED source. NED source requires writing a command or algorithm in the process of creating the network. However, it is easier to use the Graphics than the NED Source. From the Graphics, a network can be created through drawings, as there are icons to draw the module, sub-module and the connection as well. On the graphic it self, the properties for each function can be add. Figure 4.1 below shows the properties of module called Tictoc 1

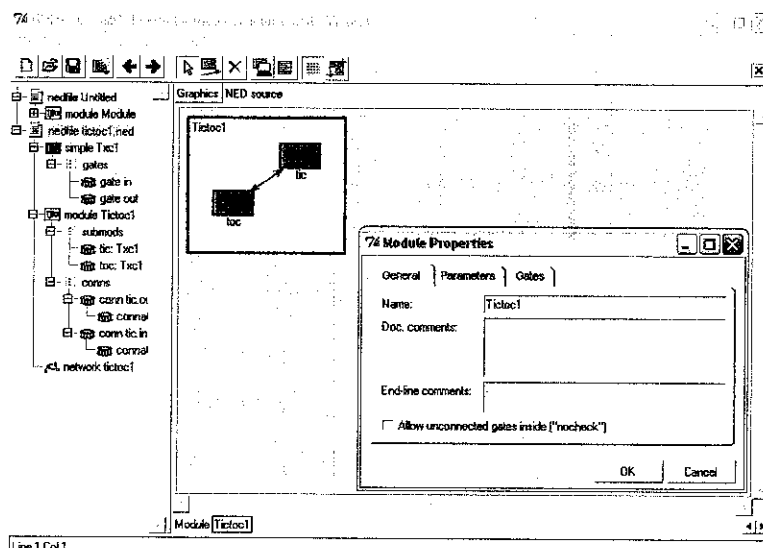


Figure 4.1: Properties of a Module

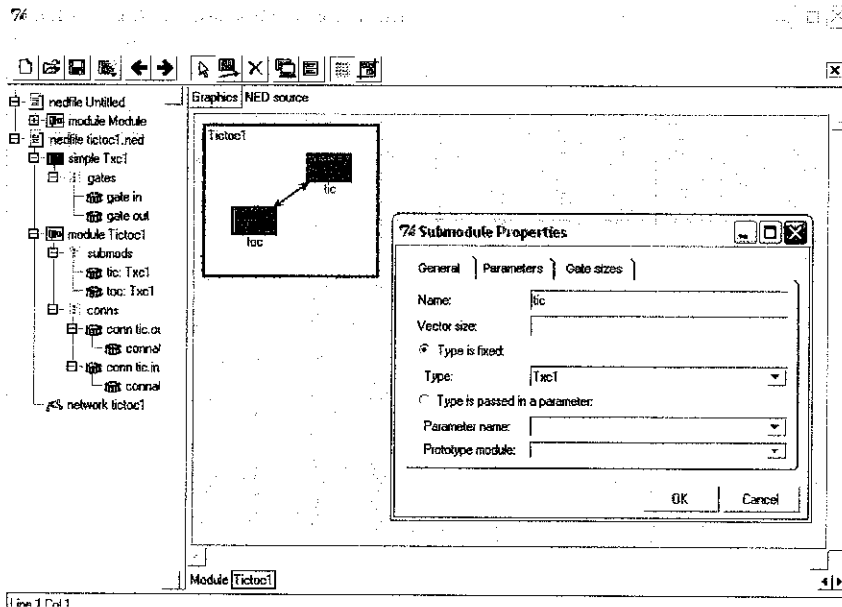


Figure 4.2: Sub module Properties

Tictoc 1 is a compound model, which is assembled from two sub-modules, Tic and Toc. Each sub-module has its own properties. However, for this simulation both sub-models have the same properties. Like Figure 4.2 above, that is the way in creating a property for Tic. The Tic Graphic was highlighted and the properties icon was pressed. In the properties box, the sub-model was name as Tic and the module was called Txcl. Txcl is a simple module type, it is atomic on NED level and will be implemented in C++ (refer Appendix A for the C++ algorithm). The same steps were taken in creating the properties for Toc.

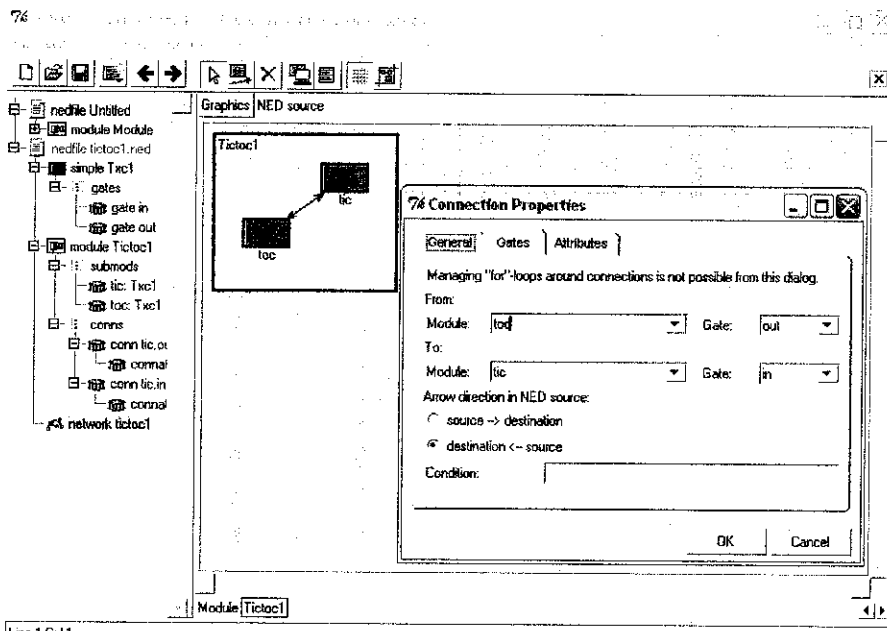


Figure 4.3: Connection Properties

In creating a connection between Tic and Toc, some information regarding the connection was put in the Connection Properties box, where the delay for the connection was indicated; as well as the gates for the connection between the Tic and Toc.

```

7% NED code -- nedfile tictoc1.ned
simple Txc1
  gates:
    in: in;
    out: out;
endsimple

module Tictoc1
  submodules:
    tic: Txc1;
    display: "p=124,48;b=40,24";
    toc: Txc1;
    display: "p=55,96;b=40,24";
  connections:
    tic.out --> delay 100ns --> toc.in;
    tic.in <-- delay 100ns <-- toc.out;
endmodule

network tictoc1 : Tictoc1
endnetwork
  
```

Figure 4.4: NED File for TicToc1

After creating the network by graphic, the NED Source tab was clicked and an algorithm was written automatically at the NED Source.

With the existence of .ned, .ini, and c++ files, the simulation is ready to be run by following the next steps in the command prompt windows.

```

Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Azmain\My Documents> cd C:\OMNeT++\tictoc\tictoc1
C:\OMNeT++\tictoc\tictoc1> opp_nmake -f
Creating Makefile.vc in C:\OMNeT++\tictoc\tictoc1...
Makefile.vc created.
Please type 'nmake -f Makefile.vc depend' NOW to add dependencies!
C:\OMNeT++\tictoc\tictoc1> nmake -f Makefile.vc

Microsoft (R) Program Maintenance Utility   Version 6.00.8168.0
Copyright (C) Microsoft Corp 1988-1998. All rights reserved.

C:\OMNeT++\bin\nedtool -s _n.cpp tictoc1.ned
cl.exe /nologo -c /EHsc /GR /FD /Zm250 /O2 /DNDEBUG /D_CRT_SECURE_NO_DEP
RECAIE -IC:\OMNeT++\include /Ip tictoc1_n.cpp
tictoc1_n.cpp
cl.exe /nologo -c /EHsc /GR /FD /Zm250 /O2 /DNDEBUG /D_CRT_SECURE_NO_DEP
RECAIE -IC:\OMNeT++\include /Ip txc1.cpp
txc1.cpp
  
```

Figure 4.5: Command Prompt Window

A Makefile is done to help compiling and linking the program in creating the executable tictoc. The command is:

```
opp_nmake, and it will create Makefile.vc.
```

The very first simulation is compiled and linked by issuing the command:

```
-f Makefile.vc
```

To tell the simulation program in which network is to be simulated, the following command is written in the .ini file:

```
[General]
network = tictoc1

[General]
preload-ned-files=*.ned
network=tictoc1 # this line is for Cmdenv, Tkenv will still
let you choose from a dialog

[Parameters]
tictoc4.toc.limit = 5

# argument to exponential() is the mean; truncnormal() returns
values from

# the normal distribution truncated to nonnegative values
tictoc6.tic.delayTime = exponential(3)
tictoc6.toc.delayTime = truncnormal(3,1)
```

To launch the simulation, the command below is written:

```
tictoc.
```

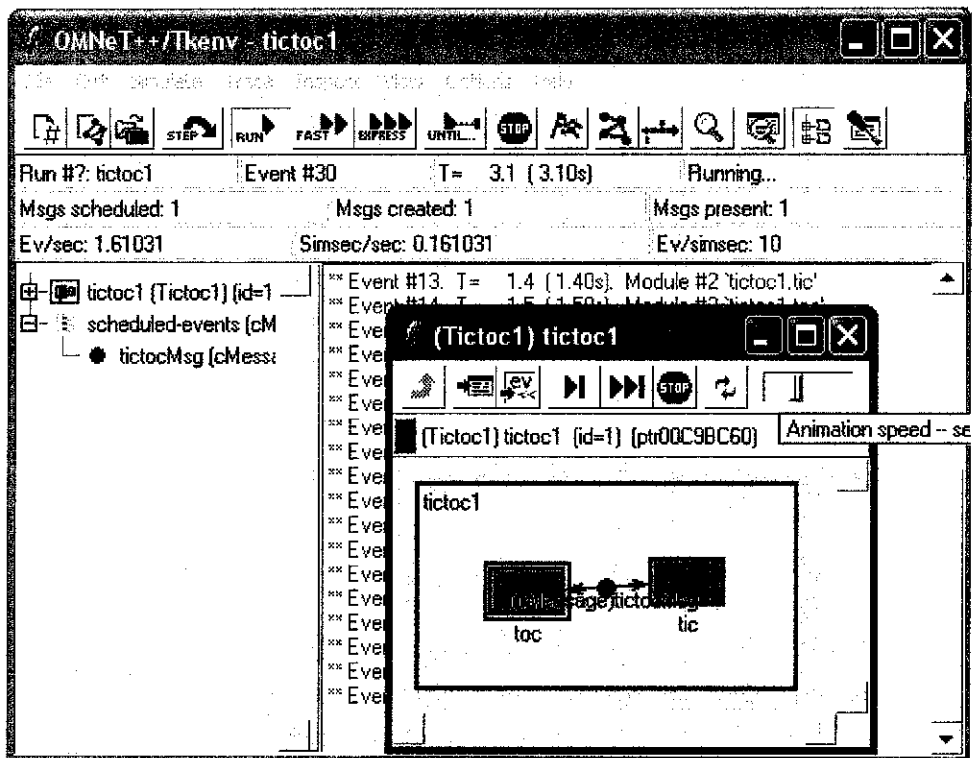


Figure 4.6: Simulation of TicToc 1

The simulation windows will pop out as in Figure 4.5. When the Run button on the toolbar is pressed, it will start the simulation. In the window, it is observed that the tic and toc nodes are exchanging messages with one another. The simulation for the TicToc1 is now successful.

## 4.2 Basic CDMA/CA simulation

Based on what have been learnt from the TicToc Tutorial, a simple simulation can be built. As a start, a basic CDMA/CA simulation is done with 1 AP and 1 SS.

These are the access scheme of DCF that should be followed by AP and SS;

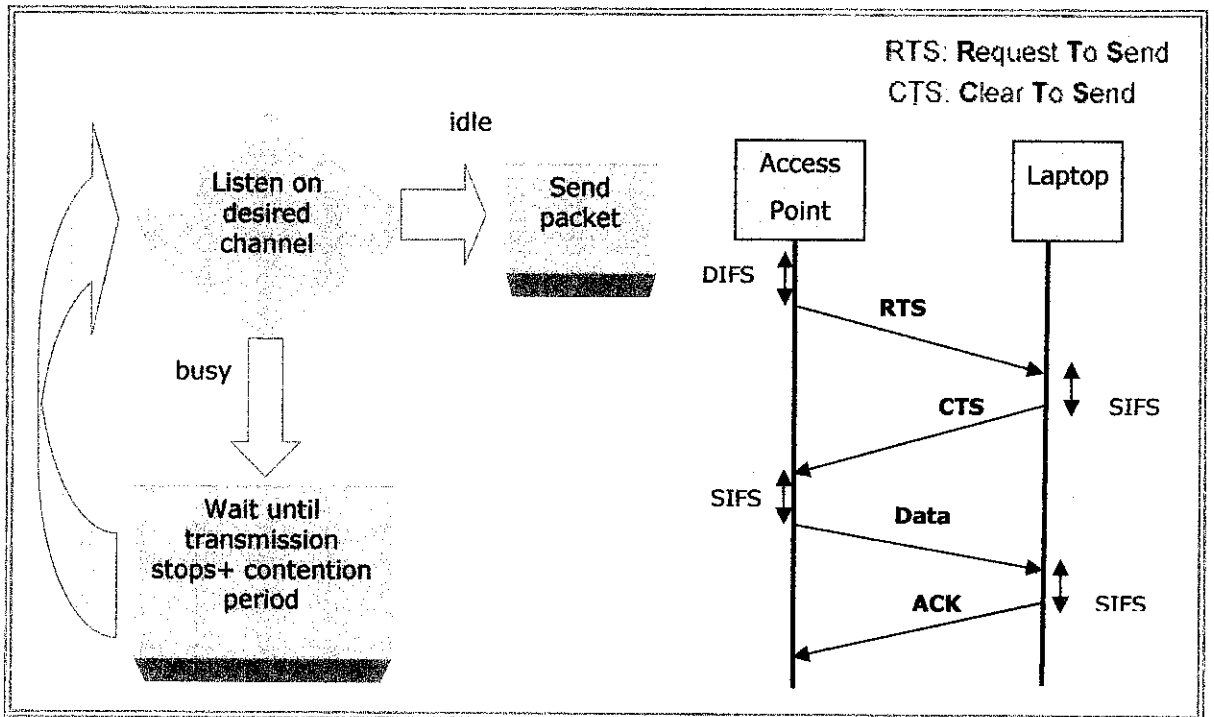


Figure 4.7: DCF Access Scheme

For the first part, by using only basic access scheme, the coding could be done by modifying the tictoc 5 and tictoc 7 and tictoc8.

For the second part with RTS/CTS, it will be more complicated. It is like repeating the process two times.

These AP and SS can listen and send data to each other. It is a bi-directional traffic and it will be based on contention period.

Below is one of the examples how the sending message occur when it involves sending data and acknowledgement:

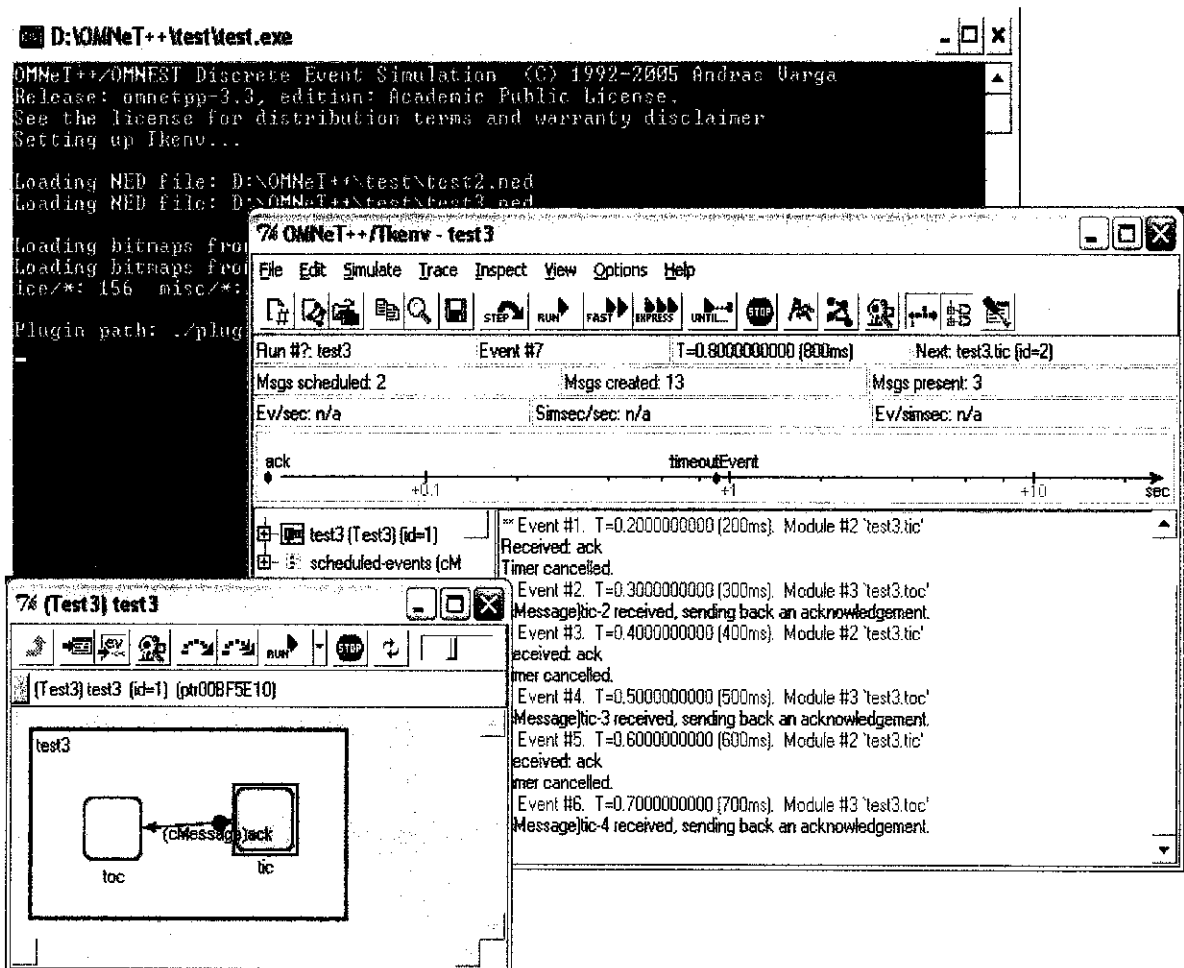


Figure 4.8: Example of sending acknowledgement



### 4.3 NClients Module

The same network (consisting of three routers, a server and several clients) can be simulated with three different kinds of traffic:

- Telnet sessions, using the TelnetApp and TCPGenericSrvApp modules;
- web users issuing HTTP requests, using the TCPBasicCliApp and TCPGenericSrvApp modules;
- file transfer sessions, using the TCPBasicCliApp and TCPGenericSrvApp modules with a different configuration.

IP addresses and routing tables are set up automatically, by using the FlatNetworkConfigurator module.

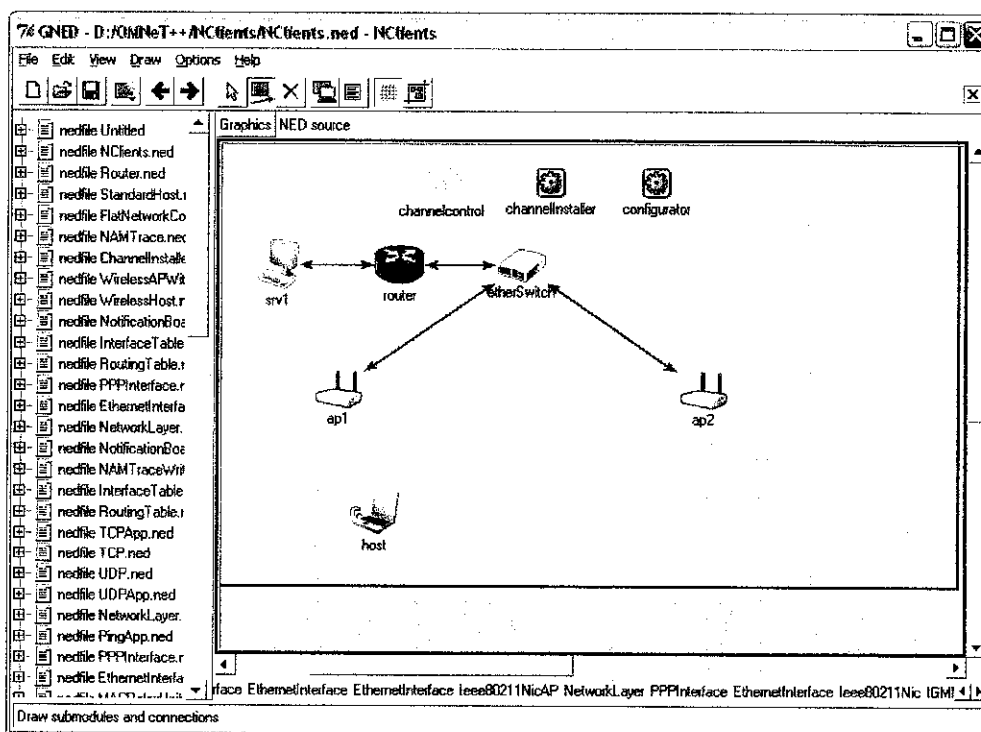


Figure 4.9: 802.11 Model in OMNeT++

Above figure is the example of 802.11 Model in the form of Graphical User Interface (GUI). This is only the look from outmost interface. There are a lot of sub modules in each of the component above (as shown on the left hand side of the window). The NED files behind this model is attach on Appendix B.

Below is the simulation demo for NClients/HTTP. The web users issuing HTTP requests:

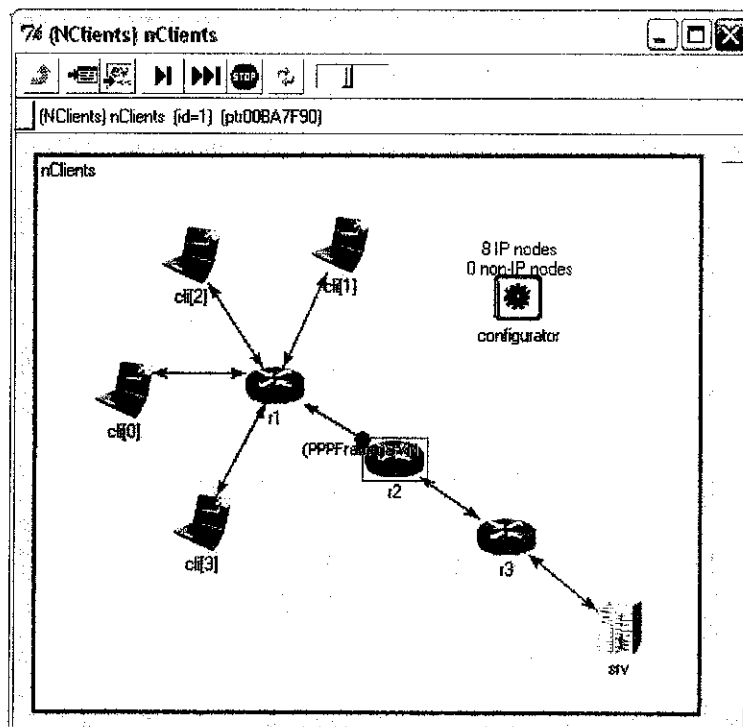


Figure 4.10: NClients Module – Transferring Files

The screenshot shows the OMNeT++ Tkenv interface for the nClients simulation. The window title is '7% OMNeT++/Tkenv - nClients'. The menu bar includes File, Edit, Simulate, Trace, Inspect, View, Options, and Help. The toolbar contains various simulation control icons. The status bar shows 'Run #1: nClients', 'Event #464', 'T=4.5360763 (4.53s)', and 'Next: nClients.r2.ppp[1] [id=27]'. Below the status bar, it displays 'Msgs scheduled: 7', 'Msgs created: 201', and 'Msgs present: 62'. The main area shows simulation statistics: 'Ev/sec: n/a', 'Simsec/sec: n/a', and 'Ev/simsec: n/a'. The log window displays the following events:

```

** Event #454. T=4.5360129 (4.53s). Module #60 `nClients.cli[1].ppp[0]`
Transmission finished.
** Event #455. T=4.536013 (4.53s). Module #15 `nClients.r1.ppp[1]`
** Event #456. T=4.536013 (4.53s). Module #19 `nClients.r1.networkLayer.ip`
** Event #457. T=4.536023 (4.53s). Module #19 `nClients.r1.networkLayer.ip`
Packet destination address is: 145.236.0.8, output port is 4
** Event #458. T=4.536023 (4.53s). Module #18 `nClients.r1.ppp[4]`
Received (IPDatagram)ACK for transmission
Starting transmission of (PPPFrame)ACK
** Event #459. T=4.5360446 (4.53s). Module #18 `nClients.r1.ppp[4]`
Transmission finished.
** Event #460. T=4.5360447 (4.53s). Module #26 `nClients.r2.ppp[0]`
** Event #461. T=4.5360447 (4.53s). Module #28 `nClients.r2.networkLayer.ip`
** Event #462. T=4.5360547 (4.53s). Module #28 `nClients.r2.networkLayer.ip`
Packet destination address is: 145.236.0.8, output port is 1
** Event #463. T=4.5360547 (4.53s). Module #27 `nClients.r2.ppp[1]`
Received (IPDatagram)ACK for transmission
Starting transmission of (PPPFrame)ACK

```

Figure 4.11: Tkenv file for NClients Simulation

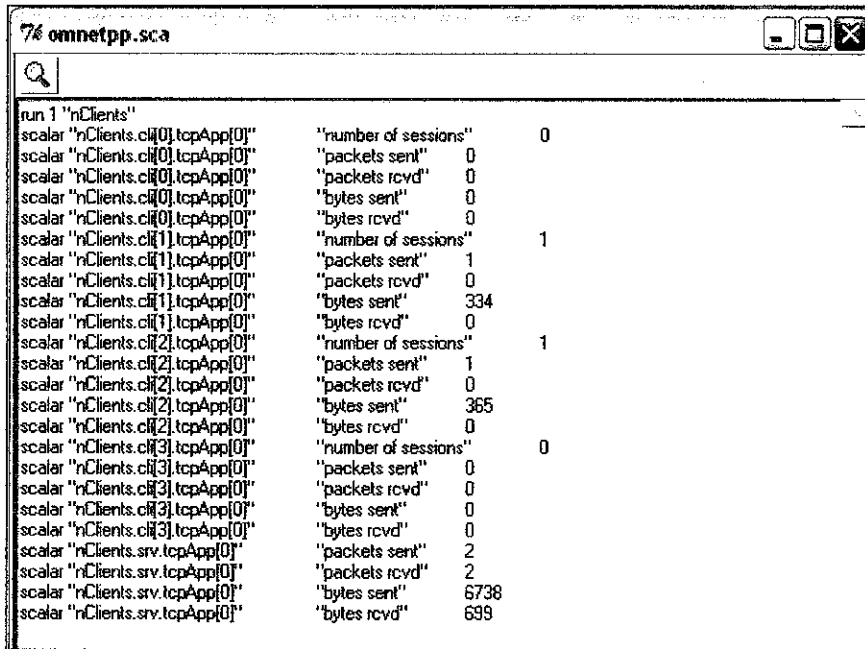


Figure 4.12: Output Scalar of NClients Simulation

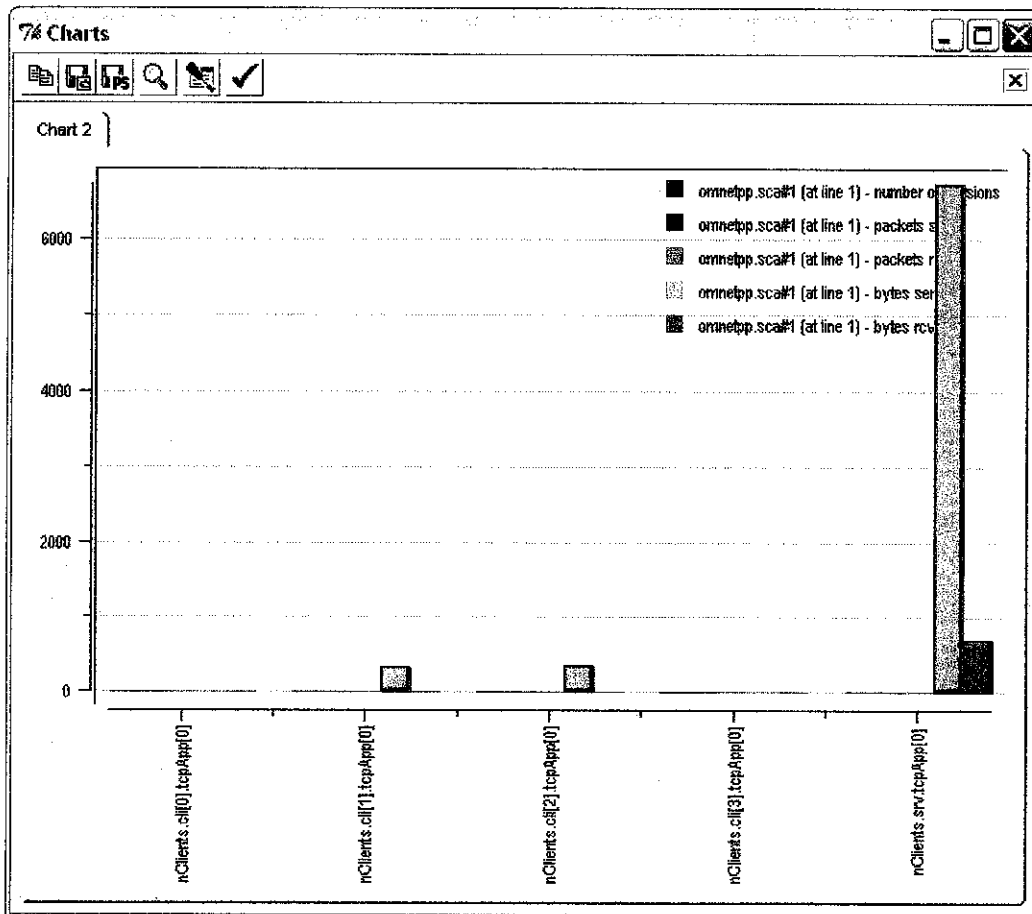


Figure 4.13: Output Scalar Charts for NClients

Following is the FlatNet Demo. It demonstrates automatic IP address assignment and automatic routing table setup using the FlatNetworkConfigurator module.

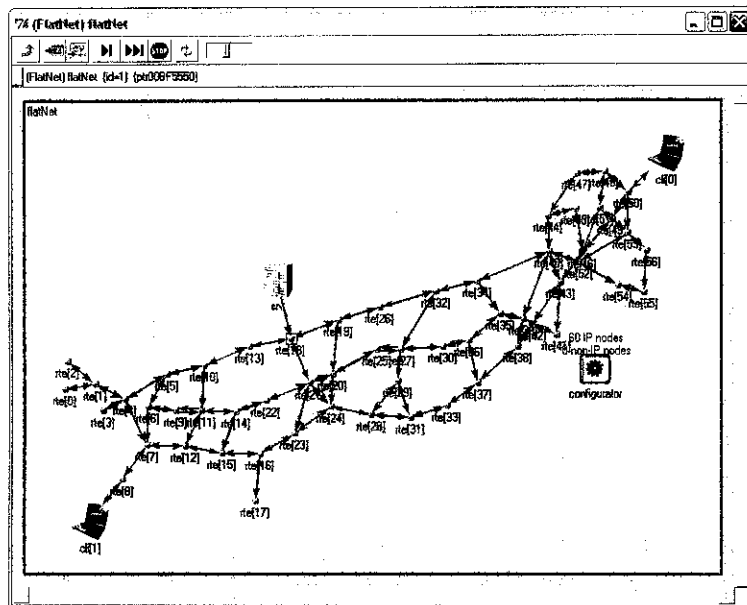


Figure 4.12: FlatNet Simulation

This example shows the use of TCP client/server models. IP addresses and routing tables are set up automatically, using FlatNetworkConfigurator.

The network consists of four hosts and one router. One of the clients has a direct connection to the server; others connect via the router. The application models bulk file transfer.

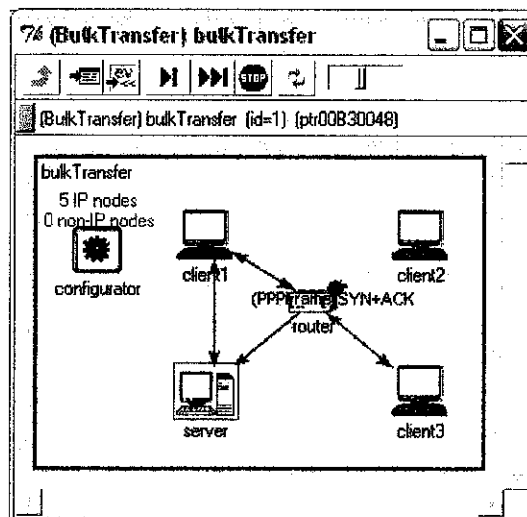


Figure 4.13: Bulk Transfer Simulation

## 4.4 Tracing and Analyzing Models Using TCP

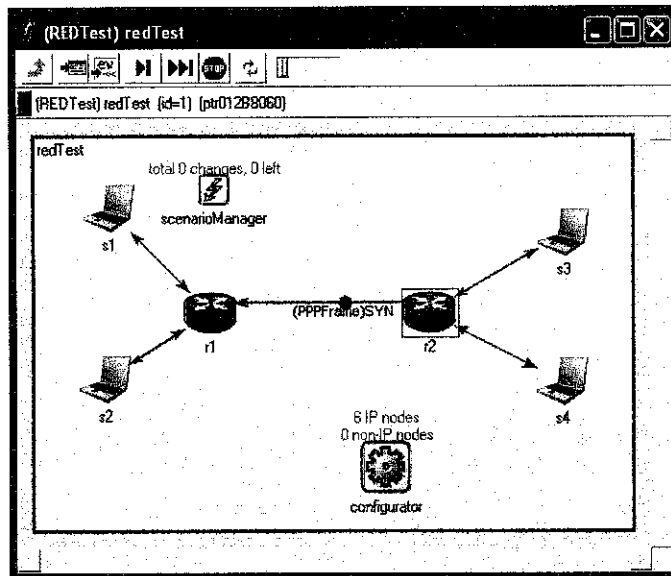


Figure 4.14: REDTest Simulation

After running the REDTest simulation, here are the ploved windows where we can plot the graph for Round-Trip Time

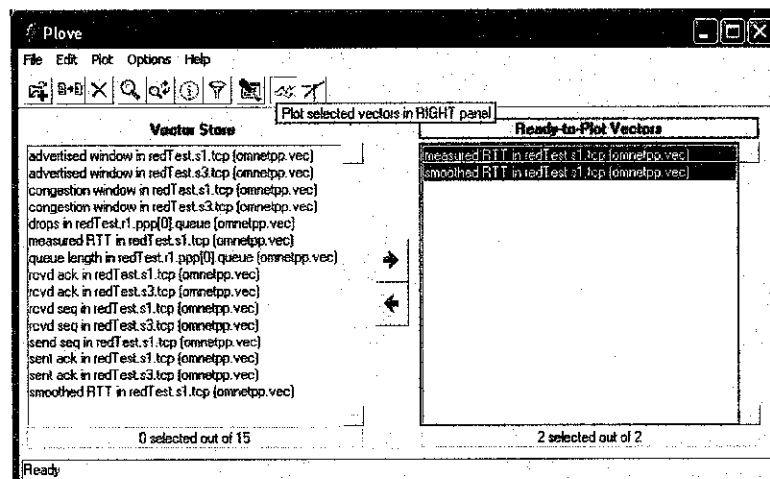


Figure 4.15: Plove windows for Round-Trip Time

Using the toolbar button create a plot. After some zooming and customizing (right-clicking the chart will present a context menu for setting line styles, axis labels, etc) something like the following will appeared:

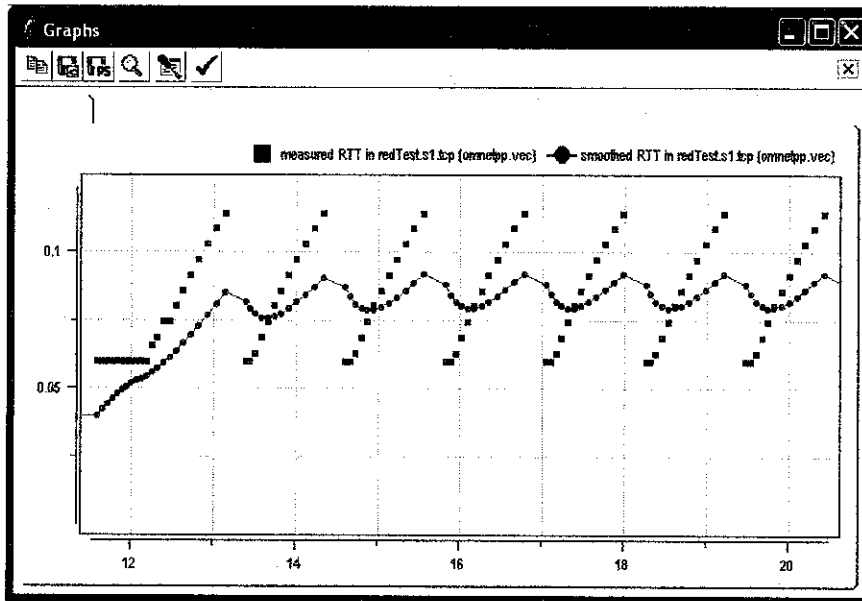


Figure 4.16: Round-Trip Time Plot

A sequence number plot from the same simulation also can be seen:

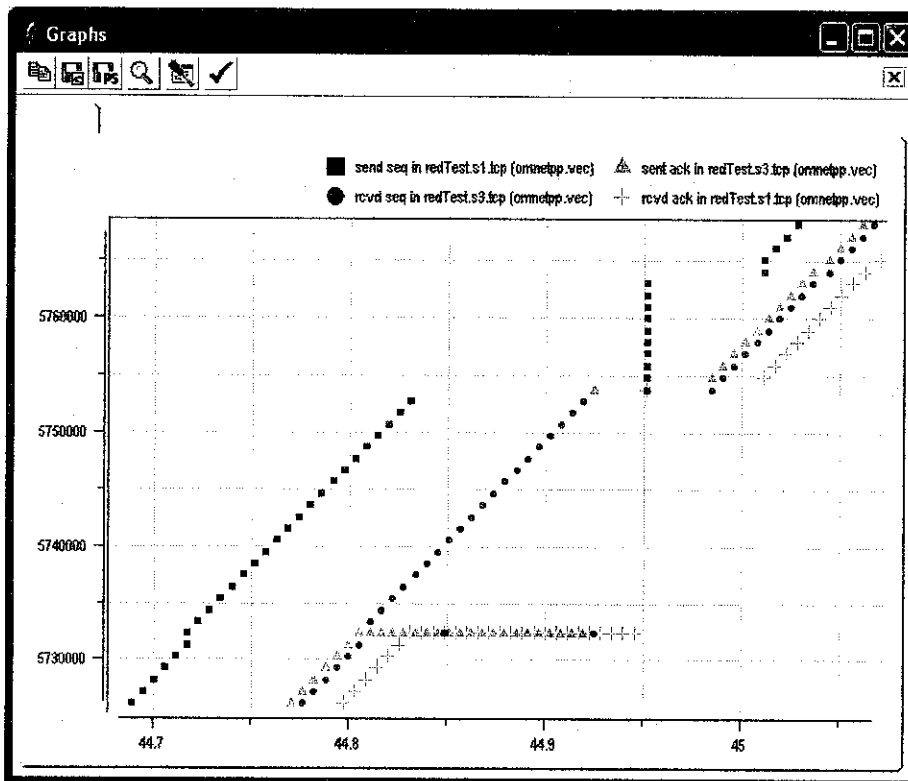


Figure 4.17: Sequence Number Plot

This chart tells a whole story by itself. The horizontal axis is time, and the vertical is TCP sequence numbers. It shows that TCP in "s1" was sending data to "s2". In the beginning, the sequence numbers in sent segments (blue) are growing steadily. These segments also arrive at "s2" after some delay (see red dots by about  $.1s = 100ms$  to the right of the corresponding blue dots). These segments get acknowledged by "s2" (green triangles -- they are one MSS = 1024 bytes above the red dots, because red dots represent the first bytes of full-sized (1024 byte) segments that arrived, while the acknowledgements carry the next expected sequence number as ack.) These acknowledgements arrive at "s1" about 30ms later (yellow plus signs). The asymmetry in delays (100ms vs 30ms) is because acks are smaller, which results in smaller queueing delay at bottleneck links.

Then a segment gets lost: one red dot at around  $t=44.81s$  is missing, meaning that that segment was not received by "s2" -- it was probably dropped in a router. The sender "s1" TCP doesn't know this, so it keeps transmitting until the window lasts. But "s2" keeps sending the same ack number (green triangles become horizontal), saying that it still wants the missing segment. When these acks arrive at "s1", at the 4th ack (3rd duplicate ack) it recognizes that something is wrong, and re-sends the missing segment (solitary blue dot). Finally, after 100ms this segment arrives at "s2" (solitary red dot), but until then "s2" sends further duplicate acks for all the segments that were still in the queue.

When the missing segment arrives at "s2", ack numbers sent by "s2" jump up (solitary green triangle at about  $t=44.93s$ ), signalling that the segment filled in the gap. (This also indicates that received segments above the gap were not discarded by "s2", but rather preserved for the future).

When this ack arrives at "s1" (yellow cross under blue dot at  $t=44.95s$ ), "s1" knows that all is well now, and spits out several segments at same time (blue dots in vertical line). This is because those duplicate acks inflated the congestion window (by indicating that above-sequence segments were received properly by "s2") -- this is the Fast Recovery algorithm. Of course these segments arrive at "s2" one after another (sequence of red dots starting at about  $t=44.98s$  is slanting), because they individually have to queue up for transmission. "s1" TCP gets permission to send further segments when further acks arrive (yellow crosses after  $t=45s$ ).

## 4.5 IEEE 802.11 Model

An IEEE 802.11 interface (NIC) comes in several flavors, differing in their role (ad-hoc station, infrastructure mode station, or access point) and their level of detail:

- Ieee80211Nic: a generic (configurable) NIC
- Ieee80211NicAdhoc: for ad-hoc mode
- Ieee80211NicAP, Ieee80211NicAPSimplified: for use in an access point
- Ieee80211NicSTA, Ieee80211NicSTASimplified: for use in an infrastructure-mode station

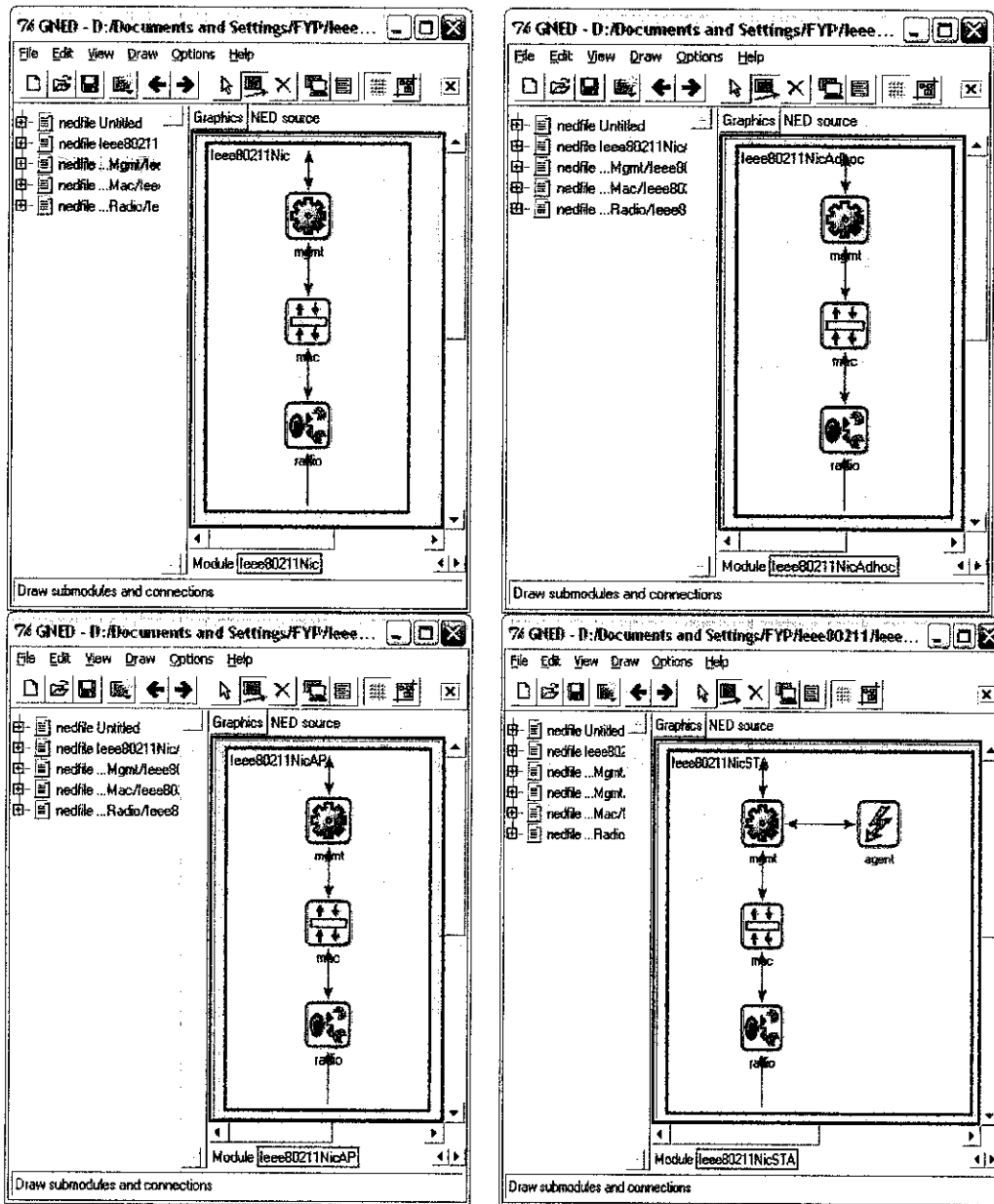


Figure 4.18: IEEE 80211 NIC Models



NICs consist of four layers, which are the following (in top-down order):

- Agent
- Management
- MAC
- physical layer (radio)

#### The agent:

It instructs the management layer to perform scanning, authentication and association. The management layer itself just carries out these commands by performing the scanning, authentication and association procedures, and reports back the results to the agent. The agent layer is currently only present in the Ieee80211NicSTA NIC module, as an Ieee80211AgentSTA module. The management entities in other NIC variants do not have as much freedom as to need an agent to control them. By modifying or replacing the agent, one can alter the dynamic behavior of STAs in the network, for example implement different handover strategies.

#### The management layer

It performs encapsulation and decapsulation of data packets for the MAC, and exchanges management frames via the MAC with its peer management entities in other STAs and APs. Beacon, Probe Request/Response, Authentication, Association Request/Response etc frames are generated and interpreted by management entities, and transmitted/received via the MAC layer. During scanning, it is the management entity that periodically switches channels, and collects information from received beacons and probe responses. The management layer has several implementations which differ in their role (STA/AP/ad-hoc) and level of detail: Ieee80211MgmtAdhoc, Ieee80211MgmtAP, Ieee80211MgmtAPSimplified, Ieee80211MgmtSTA, Ieee80211MgmtSTASimplified.

#### The MAC layer

(Ieee80211Mac) performs transmission of frames according to the CSMA/CA protocol. It receives data and management frames from the upper layers, and transmits them.

#### The physical layer modules

(Ieee80211Radio; with some limitations) deal with modeling transmission and reception of frames. They model the characteristics of the radio channel, and determine if a frame was received correctly (that is, it did not suffer bit errors due to low signal power or interference in the radio channel). Frames received correctly are passed up to the MAC. The implementation of these modules is based on the Mobility Framework.

## 4.6 Expected Result

AP using DFS experiences more poor transmissions due to quiet periods. Figure below shows the difference of having DFS [10]:

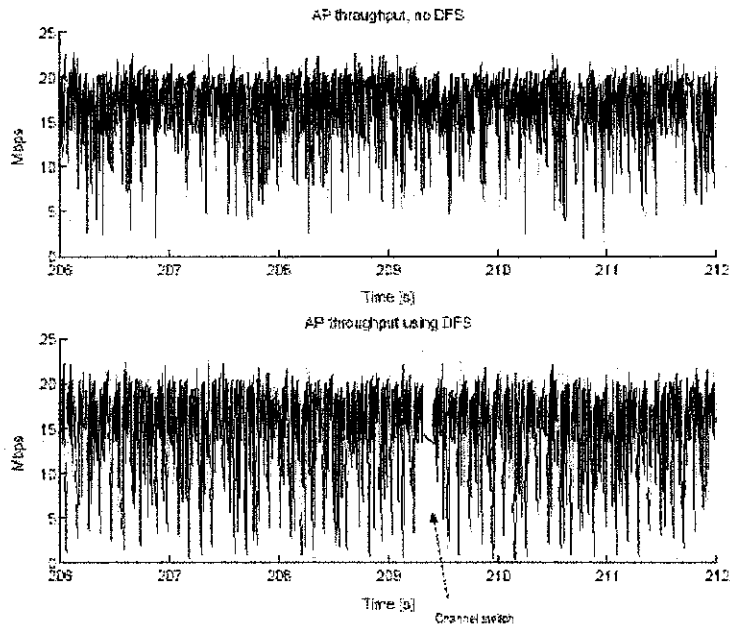


Figure 4.19: Throughput vs. DFS Usage

As previously mentioned, quiet periods also introduce a collision problem. With increasing number of STAs participating in the quiet measurements, the ACK failure ratio increases as well because of these collisions.

This problem can be seen in following figure, showing a situation directly following a quiet period. Collisions are marked with dashed vertical lines, and colliding frames are encircled. In this WLAN cell, all 10 STAs participates in (and reports results of) quiet measurements. Consequently, the AP has 10 measurement request frames to send regarding the next quiet period. This is an obvious problem with this algorithm. The MAC backoff procedure, which is performed at all devices at the quiet period end, handles the increased competition of the medium accordingly.

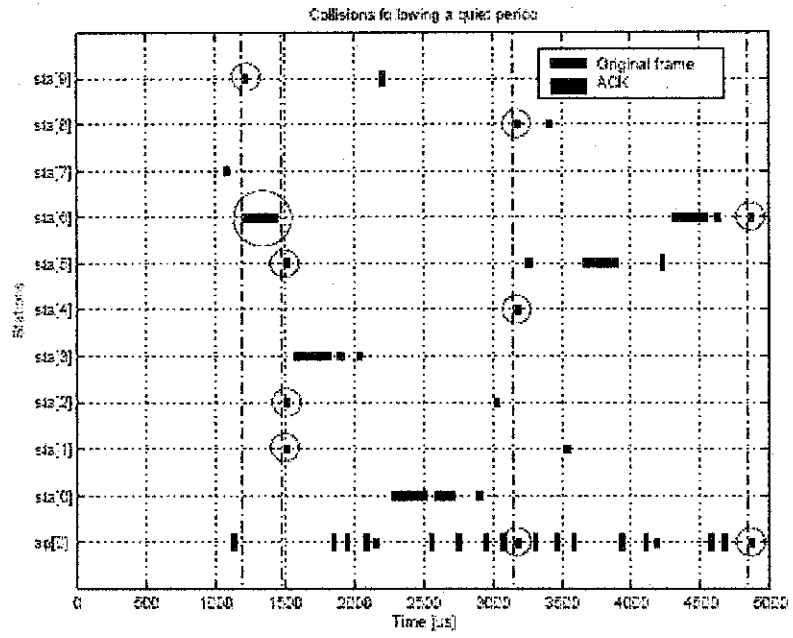


Figure 4.20: Quiet Period Ends With Collision

## 4.7 Project Difficulties

Throughout the project, there are some difficulties that have been faced. This has led to insufficient results at the end of the project. Some of them can be resolved after a long of hardwork, however there are is still some of them unresolved.

Stated here are the problems:

- Due to time consuming and lack of facilities, it is impossible to set up and run a real network environment. Therefore, the best solution is by using a simulator like OMNeT++.
- To get a better grasp on the software a lot of exercises are needed.
- The configuration setting of the simulation is a bit complicated as it is depending on other application such as C++ compiler, and INET Framework.
- OMNeT++ can work better on Linux Platform.
- Insufficient of guidance on how to use the software and how to encounter the errors in simulating the model.
- Lack of expertise that can help to configure the software.

All the coding and files needed for the complete simulation is already prepared. Unfortunately, due to this technical problem, it is hard to come out with the end result.

## **CHAPTER 5**

### **CONCLUSION AND RECOMMENDATION**

The research part takes half the time of this project, in order to fully understand the operation of the Wi-Fi, MAC, Handshaking and other things. Then, a new software need to be learnt to perform the simulation of a real system that is OMNET++. It took quite some times to get started with the OMNET++. Some outcome was already achieved for the last 1 semester. A better overview of WiFi and its performance was being grabbed. The problem or weaknesses of the existence standard were being studied too. After all, a better picture on how to solve the problem has been identified. There are some specific parameters in the network of existence standard need to be improved to get a better performance.

For this semester, a simulation for the IEEE 802.11a standard is built in OMNeT++. The simulation of 802.11a is tested with and without QoS. From the simulation the performance of the system can be evaluated by observing the delay and throughput. As expected the system with QoS provisioning will perform better.

As already known, the latest research is already focusing on the 802.11e standard. This 802.11e will definitely provide much better quality in terms of QoS. The system was implemented by using EDCF (Enhanced-DCF). For the future recommendation, a further research and studies should be focusing on the new upcoming 802.11e standard.

## REFERENCES

- [1] Gilbert, 'Deploying Wireless LANs', *Concepts, Operation, and Utilization*, 2002.
- [2] Asuncion santamaria and Francisco J. Lopez-Hernandez, 'Wireless LAN Standards and Application', 2001.
- [3] [http://shop.ieee.org/ieeestore/product.aspx?product\\_no=SP1136&site=google](http://shop.ieee.org/ieeestore/product.aspx?product_no=SP1136&site=google) - IEEE 802.11 Handbook: A Designer's Companion, Second Edition, (February, 2007)
- [4] [http://www.tutorial-reports.com/wireless/wlanwifi/introduction\\_wifi.php](http://www.tutorial-reports.com/wireless/wlanwifi/introduction_wifi.php) - Introduction to Wireless LAN and IEEE 802.11, (February, 2007)
- [5] [http://www.zytrax.com/tech/wireless/802\\_mac.htm](http://www.zytrax.com/tech/wireless/802_mac.htm) - 802.11 MAC (Medium Access Control), (February, 2007)
- [6] <http://www.tml.tkk.fi/Studies/T-110.300/2004/Homeworks/model5.txt> - July, 2007.
- [7] <http://www.csc.uvic.ca/~wkui/Courses/topic/Lecture3.pdf> - Part3: MediumAccess Layer Protocol, (March, 2007)
- [8] Sunghyun Choi, Javier Del Prado, Sai Shankar N, Stefan Mangold2: 'IEEE 802.11e Contention-Based Channel Access (EDCF) Performance Evaluation'. Journal.
- [9] <http://compnetworking.about.com/cs/wireless80211/a/aa80211standard.htm> - 802.11 standards-802.11a 802.11b 802.11g, (February, 2007)
- [10] Magnus Janson, Magnus Karlsson, 'A WOK Simulation Model for DFS and Link Adaptation in IEEE 802.11a WLAN'. Linköping University, January 2004.
- [11] <http://ctiware.eng.monash.edu.au/twiki/bin/attach/Simulation/> - OMNetpp Comparison, (April 2007)
- [12] <http://www.omnetpp.org/doc/tictoc-tutorial/index.html> - TicToc Tutorial for OMNeT++ 3.2
- [13] <http://www.slrc.kyushu-u.ac.jp/~ishihara/CREST/RWS2006.pdf> - Slides of "Low Power Design for IEEE 802.11 WLAN at the Medium Access Control Layer", (April, 2007)
- [14] Changhua He, John C Mitchell, 'Anaysis of the 802.11 4-Way Handshake'. Standford University, Stanford.
- [15] Hector Velayos, Gunnar Karlsson, 'Techniques to Reduce IEEE 802.11b MAC Layer Handover Time'. Royal Institute of Technology, Stolckholm, April 2003.
- [16] C M Chin, C M Tan and M L Sim: 'Future Trends in Radio Resource Management for Wireless Communications'. BT Technology Journal, Vol 24 No.2, pp 103 – 110 (April 2006)

## **APPENDICES**

## APPENDIX A

### C++ ALGORITHM FOR FUNTIONALITY OF TCX1.

```
#include <string.h>
#include <omnetpp.h>
class Txcl : public cSimpleModule
{
    protected:
        // The following redefined virtual function holds the algorithm.
        virtual void initialize();
        virtual void handleMessage(cMessage *msg);
};
// The module class needs to be registered with OMNeT++
Define Module(Txcl);
void Txcl::initialize()
{
    // Initialize is called at the beginning of the simulation.
    // To bootstrap the tic-toc-tic-toc process, one of the modules needs
    // to send the first message. Let this be `tic'.
    // Am I Tic or Toc?
    if (strcmp("tic", name()) == 0)
    {
        // create and send first message on gate "out". "tictocMsg" is an
        // arbitrary string which will be the name of the message object.
        cMessage *msg = new cMessage("tictocMsg");
        send(msg, "out");
    }
}
void Txcl::handleMessage(cMessage *msg)
{
    // The handleMessage() method is called whenever a message arrives
    // at the module. Here, we just send it to the other module, through
    // gate `out'. Because both `tic' and `toc' does the same, the message
    // will bounce between the two.
    send(msg, "out");
}
```



## APPENDIX B

### NED FILES FOR NCLIENTS MODEL

```
import
    "Router",
    "StandardHost",
    "FlatNetworkConfigurator",
    "NAMTrace",
    "ChannelInstaller",
    "WirelessAPWithEth",
    "WirelessHost";
channel ethernetline
    delay 0.1us;
    datarate 10*1000000;
endchannel
module Nclients

    submodules:

        channelInstaller: ChannelInstaller;
            parameters:
                channelClass = "ThruputMeteringChannel",
                channelAttrs = "format=u";
            display: "p=274,34;i=block/cogwheel_s";
        configurator: FlatNetworkConfigurator;
            parameters:
                moduleTypes = " WirelessHost SDR ",
                nonIPModuleTypes = " EtherSwitch
                WirelessAPWithEth";
                networkAddress = "163.180.116.0",
                netmask = "255.255.255.0";
            display: "p=361,34;i=block/cogwheel_s";
        srv1: StandardHost;
            parameters:
                routingFile = "srv1.irt";
            display: "p=50,100;i=device/pc";
        router: Router;
            parameters:
                routingFile = "r3.irt";
            display: "p=150,100;i=abstract/router";
        etherSwitch: EtherSwitch;
            parameters:
                relayUnitType = "MACRelayUnitNP"; //TBD
            display: "p=250,100;i=device/switch";
```

```
ap1: WirelessAPWithEth;
    parameters:
        relayUnitType = "MACRelayUnitNP"; //TBD
        display: "p=100,200;i=device/accesspoint";
ap2: WirelessAPWithEth;
    parameters:
        relayUnitType = "MACRelayUnitNP"; //TBD
        display: "p=400,200;i=device/accesspoint";
host: WirelessHost;
    display: "p=130,300;i=device/wifilaptop";
```