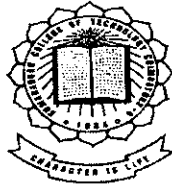


P-3563



**ESTABLISHING ONE TO MANY
CONNECTIONS AND PERFORMANCE
EVALUATION OF TCP IN WIRELESS LINKS**

PROJECT REPORT

Submitted by

T.KANAGASABAPATHY

Reg. No: 0920108007

*In partial fulfillment for the award of the degree
of*

MASTER OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University, Coimbatore)

COIMBATORE – 641 049

APRIL 2011

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University, Coimbatore)

COIMBATORE – 641 049

Department of Computer Science and Engineering

PROJECT WORK

APRIL 2011

This is to certify that the project entitled

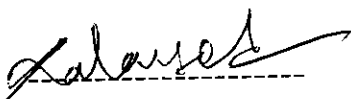
ESTABLISHING ONE TO MANY CONNECTIONS AND PERFORMANCE EVALUATION OF TCP IN WIRELESS LINKS

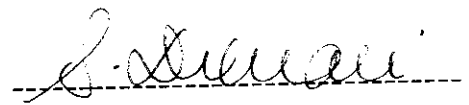
is the bonafide record of project work done by

T.KANAGASABAPATHY

Register No: 0920108007

of M.E. (Computer Science and Engineering) during the year 2010-2011.


Project Guide


Head of the Department

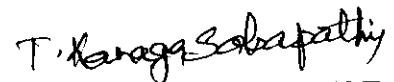
Submitted for the Project Viva-Voce examination held on 25.04.2011





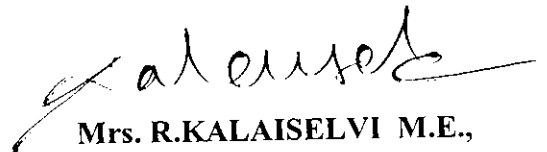
DECLARATION

I affirm that the project work titled “ESTABLISHING ONE TO MANY CONNECTIONS AND PERFORMANCE EVALUATION OF TCP IN WIRELESS LINKS” being submitted in partial fulfillment for the award of M.E degree is the original work carried out by me. It has not formed the part of any other project work submitted for the award of any degree or diploma, either in this or any other University.


KANAGASABAPATHY.T

Register No: 0920108007

I certify that the declaration made above by the candidate is true


Mrs. R.KALAISELVI M.E.,

Assistant Professor,
Department of Computer Science and Engineering,
Kumaraguru College of Technology,
(An Autonomous Institution)
Coimbatore-641 049.

KONGU ENGINEERING COLLEGE

(Autonomous)
PERUNDURAI ERODE 638 052 TAMILNADU INDIA



School of Computer Technology and Applications
Department of Computer Technology - UG

CERTIFICATE

This is to certify that Mr./Ms. T. KANAKA SABAPATHY of
KUMARAGURU COLLEGE OF TECHNOLOGY has Participated / Presented
paper entitled ESTABLISHING ONE TO MANY CONNECTION AND PERFORMANCE EVALUATION
OF TCP IN WIRELESS LINKS in the National Conference held from
0th to 11th March 2011 on Emerging Computing and Communication Technologies (NECCT 2011).


Organizing Secretary


HOD


Principal



ACKNOWLEDGEMENT

First and foremost, I would like to thank the Lord Almighty for enabling me to complete this project.

I express my profound gratitude to our Chairman **Padmabhusan Arutselvar Dr.N.Mahalingam, B.Sc., F.I.E.**, for giving this opportunity to pursue this course.

I would like to thank **Dr.S.Ramachandran, Ph.D.**, *Principal* for providing the necessary facilities to complete my thesis.

I take this opportunity to thank **Mrs.P.Devaki, M.E., (Ph.D)**, *Head of the Department*, Computer Science and Engineering, for her precious suggestions.

I thank all project committee members for their comments and advice during the reviews. Special thanks to **Mrs.V.Vanitha, M.E., (Ph.D)**, *Associate Professor*, Department of Computer science and Engineering, for her valuable advice and the arrangement of brain storming project review sessions.

I register my hearty appreciation to my Guide **Mrs.R.Kalaiselvi, M.E.**, *Assistant Professor*, Department of Computer Science and Engineering, for her encouragement and ideas.

I would like to convey my honest thanks to all **Teaching** and **Non Teaching** staff members of the department for their support. I would like to thank all my classmates for extending technical support whenever I needed it.

I dedicate this project work to my **parents**, without whose love, this work wouldn't have been possible.

TABLE OF CONTENTS

Contents	Page No.
Abstract	viii
Abstract (Tamil)	ix
List of Figures	x
List of Tables	xi
List of Abbreviations	xii
1. Introduction	
1.1 Wireless network	1
1.2 Wireless Connections	1
1.3 Uses of Wireless Network	3
1.4 Types of Wireless Links	3
1.5 Transmission Control Protocol (TCP)	5
2. Literature Survey	
2.1. Transport Protocol Performance	9
2.2. Existing system	10
2.3. Related Works	
2.3.1 Modeling Wireless Links for Transport Protocol	10
2.3.2 TCP in a wireless world	11
2.3.3 Vertical and Horizontal flow controls for TCP Optimization in mobile networks	12
2.3.4 Improving End to End Performance of TCP over Mobile Internetworks	13
2.3.5 Implementation and Performance Evaluation of Indirect TCP	14
2.3.6 Analysis of Congestion Control Strategy for Wireless Network	14

3. Methodology	
3.1 One to Many TCP Connection Establishment	16
3.2 Modules	
3.2.1 WLAN Communication	17
3.2.2 Cellular Communication	17
3.3 Dynamic Source Routing	17
3.4 Ad Hoc On-Demand Distance Vector Routing	20
3.5 System Specification	
3.5.1 Hardware Requirement	22
3.5.2 Software Requirement	22
4. Implementation	
4.1 The Network Simulator	23
4.2 Network Simulator 2 (Ns2)	
4.2.1 Overview	24
4.2.2 Main features	24
4.3 Components of NS	
4.3.1 Network AniMator	25
4.3.2 Pre-processing	25
4.3.3 Post-processing	25
4.4 Goals of NS	26
4.5 NS-2 Programming Languages	
4.5.1 Why Two Languages	27
4.5.2 NAM (Network AniMator)	27
4.5.3 Gnuplot	27
4.5.4 Trace File Formats	28

5. Results	
5.1. Performance Evaluation	
5.1.1 WLAN Communication (one to one connections)	30
5.1.2 WLAN Communication (one to many connections)	32
5.1.3 Cellular Communication (one to one connections)	35
5.1.4 Cellular Communication (one to many connections)	38
6. Conclusion and Future Outlook	
6.1 Conclusion	42
6.2 Future Outlook	42
7. Appendices	
7.1 Source Code	43
7.2 Screen Shots	62
8. REFERENCES	68

ABSTRACT

The performance of the transport protocol in the wireless networks plays vital role in obtaining efficient transmission. Wireless links have intrinsic characteristics that affect the performance of transport protocols; these include variable bandwidth, corruption, channel allocation delays and asymmetry. Certain works have been carried out to improve the performance of the networks. The ineffective performance of the network is due to the high mobility condition of the node. The complete utilization of the bandwidth can improve the network performance. In this project, one to many TCP connections are given in the wireless mesh networks and transmission is carried out. Then the performance of the network is compared with the same network established with one to one TCP connections. The one to many connections established would utilize bandwidth to maximum level. In this transmission other performance parameters are also evaluated and compared.

ஆய்வுச் சுருக்கம்

கம்பி இல்லா இணையத்தில், கடத்தி உடன்பாட்டு நெறீமுறை முக்கிய பங்கு வகிப்பதோடு மட்டும் இல்லாமல் திறமைமிக்க செலுத்துதலிலும் உதவுகிறது. கம்பியில்லா தொடர்பு இணைப்புகளில் மாறுபடும் கற்றை அகலம், கட்டழிவு, செயற்படு தடம் ஒதுக்கீடு, தாமதப்படுதல் மற்றும் சமச்சீரற்றத் தன்மை ஆகிய இயற்கையான பண்பியல்புகள் கடத்தி உடன்பாட்டு நெறீமுறையின் திறனை பாதிக்கிறது. ஒரு சில முயற்சிகள் பிணையத்தின் திறனை அதிகரிக்க கையாளப்படுகிறது. பிணையத்தின் திறனானது முனைப்பின் உயர்நகருத்தன்மையால் பயனற்று விடுகிறது. கற்றை அகலத்தின் முழுமையான பயன், பிணையத்தின் திறனை அதிகரிக்கும்.

இத்தீட்டபணியில், ஒன்றிலிருந்து அதிகப்படியான கடத்தல் கட்டுப்பாட்டு நெறீமுறை இணைப்புகள் கம்பியில்லா கண்ணிப்பிணையத்தில் கொடுக்கப்பட்டு கடத்தல் பணி கையாளப்படுகிறது. பிணையத்தின் இத்திறனை ஒன்றிலிருந்து ஒன்று கடத்தல் கட்டுப்பாட்டு நெறீமுறை இணைப்புகளுடன் ஒப்பிடப்படுகிறது. ஒன்றிலிருந்து அதிகப்படியான இணைப்புகளானது, அதிகப்படியான கற்றை அகலத்தை பயன்படும் அளவிற்கு உபயோகப் படுத்தப்படுகிறது.-

LIST OF FIGURES

FIGURE	TITLE	PAGE NO
Fig 1.1	TCP Header Format	6
Fig 1.2	TCP Connection Establishment and Termination	7
Fig 4.1	Simplified User's View of NS2	25
Fig 4.2	NAM Window	26

LIST OF TABLES

TABLE	TITLE	PAGE NO
Table 4.1	Wired Trace File Format	28

LIST OF ABBREVIATIONS

MANET	: Mobile Ad-hoc Network
QoS	: Quality of Service
MAC	: Medium Access Control
TCP	: Transmission Control Protocol
ACK	: Acknowledgement
UDP	: User Datagram Protocol
AODV	: Ad hoc On-demand Distance Vector
DSR	: Dynamic Source Routing
FEC	: Forward Error Correction
RTT	: Round-Trip Time
MTU	: Maximum Transmission Unit
RTS	: Request To Send
CTS	: Clear To Send
NS	: Network Simulator

CHAPTER 1

INTRODUCTION

1.1 Wireless network

Wireless network refers to one type of computer network that is wireless, and is commonly associated with a telecommunications network whose interconnections between nodes are implemented without the use of wires. Wireless telecommunication networks are generally implemented with some type of remote information transmission system that uses electromagnetic waves such as radio waves.

1.2 Wireless connections

The different types of wireless connections are

- Wireless PAN
- Wireless LAN
- Wireless MAN
- Wireless WAN
- Mobile devices networks

➤ Wireless PAN

Wireless Personal Area Networks (WPANs) interconnect devices within a relatively small area, generally within reach of a person. For example, Bluetooth provides a WPAN for interconnecting a headset to a laptop. ZigBee also supports WPAN applications. Wi-Fi PANs are also getting popular as vendors have started integrating Wi-Fi in variety of consumer electronic devices. Intel My WiFi and Windows 7 virtual Wi-Fi capabilities have made Wi-Fi PANs simpler and easier to set up and configure.

➤ Wireless LAN

A wireless local area network (WLAN) links two or more devices using a wireless distribution method (typically spread-spectrum or OFDM radio), and usually providing a connection through an access point to the wider internet. This gives users the mobility to move around within a local coverage area and still be connected to the network.

Wi-Fi is increasingly used as a synonym for 802.11 WLANs, although it is technically a certification of interoperability between 802.11 devices.

Fixed wireless data implements point to point links between computers or networks at two locations, often using dedicated microwave or laser beams over line of sight paths. It is often used in cities to connect networks in two or more buildings without physically wiring the buildings together.

➤ Wireless MAN

Wireless Metropolitan area Networks are a type of wireless network that connects several wireless LANs.

WiMAX is the term used to refer to wireless MANs and is covered in IEEE 802.16d/802.16e.

➤ Wireless WAN

Wireless wide area networks are wireless networks that typically cover large outdoor areas. These networks can be used to connect branch offices of business or as a public internet access system. They are usually deployed on the 2.4 GHz band. A typical system contains base station gateways, access points and wireless bridging relays. Other configurations are mesh systems where each access point acts as a relay. When combined with renewable energy systems such as photo-voltaic solar panels or wind systems they can be stand alone systems.

➤ Mobile devices networks

With the development of smart phones, cellular telephone networks routinely carry data in addition to telephone conversations.

Global System for Mobile Communications (GSM): The GSM network is divided into three major systems: the switching system, the base station system and the operation and support system. The cell phone connects to the base system station which then connects to the operation and support station; it then connects to the switching station where the call is transferred to where it needs to go. GSM is the most common standard and is used for a majority of cell phones.

Personal Communications Service (PCS): PCS is a radio band that can be used by mobile phones in North America and South Asia. Sprint happened to be the first service to set up a PCS.

D-AMPS: Digital Advanced Mobile Phone Service, an upgraded version of AMPS being phased out due to advancement in technology. The newer GSM networks are replacing the older system.

1.3 Uses of Wireless Network

- Information could be sent globally to all the parts of the world
- The information is being sent through the satellites with least amount of time.
- Internet is the biggest achievement of the wireless network invention. Even the remote areas can remain connected through each other via internet.

1.4 Types of Wireless Links

- Cellular links
- Wireless LANs links
- Satellite links

➤ Cellular links

Most common cellular links are provided today by GPRS (General Packet Radio Service) and CDMA2000 systems and in the future possibly by UMTS (Universal Mobile Telecommunication System). The bandwidth of such links is in the range of 0.01-1 Mbps, with high one-way latency of 0.1-0.5 seconds. The coverage radius of a single cell varies from several hundred meters in urban areas to 30 km in rural areas. We use GPRS as an example of a cellular link. A GPRS link typically has 40 kbps bandwidth and 400 ms latency in downlink and 10 kbps, 200 ms in uplink.

➤ Wireless LANs links

The most commonly used WLAN today is IEEE 802.11b with bandwidth of 2-11 Mbps. In general, WLANs have a low latency of 3-100 ms and bandwidth in the range of 1-50 Mbps. WLAN uplink and downlink channels are not independent as in cellular or satellite, but compete with each other for shared bandwidth. The coverage radius of a single base station varies from tens to hundreds of meters. The link error control of 802.11b is tightly coupled with the MAC mechanism. There are at most three retransmission attempts per data frame. Packet fragmentation is supported for higher efficiency of error recovery, but it is not commonly used.

➤ Satellite links

In general, satellite links are characterized by high latency in the range of 50-300 ms and bandwidth of 0.01-50 Mbps. Today, satellite links are mostly provided by a fixed GEO satellite. Such links typically have a latency of 270 ms, downlink bandwidth of 40 Mbps and uplink bandwidth of 1 Mbps. There is tremendous variation in capacity provided by satellite links. The uplink bandwidth might be only 64 kbps for VSAT terminals. Modern satellite links are generally error-free except for occasional fades. A single GEO satellite can cover an entire continent.

1.5 Transmission Control Protocol (TCP)

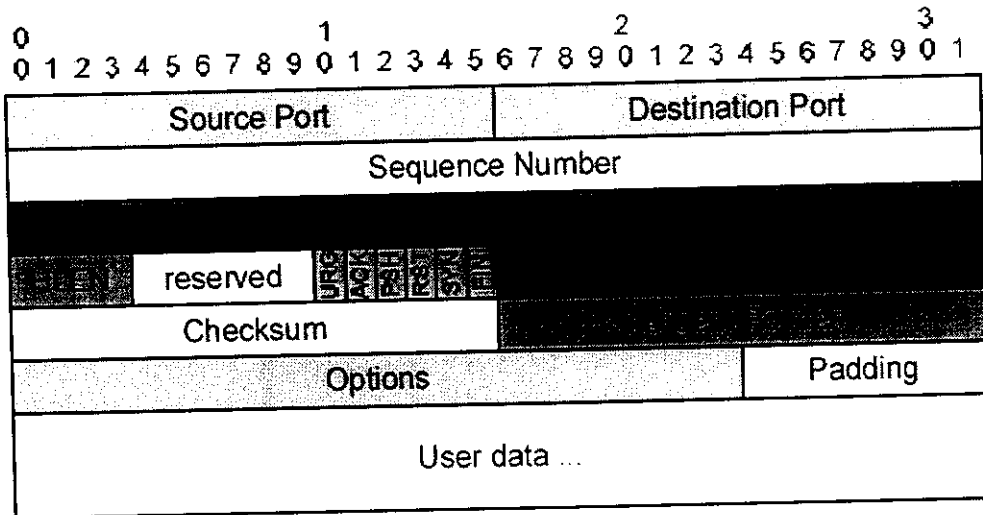
TCP is one of the core protocols of the Internet Protocol Suite. TCP is one of the two original components of the suite complementing the Internet Protocol (IP) and therefore the entire suite is commonly referred to as TCP/IP. TCP provides reliable, ordered delivery of a stream of bytes from a program on one computer to another program on another computer. TCP is the protocol that major Internet applications rely on, applications such as the World Wide Web, e-mail and file transfer. Other applications which do not require reliable data stream service may use the User Datagram Protocol (UDP) which provides a datagram service that emphasizes reduced latency over reliability. TCP provides a point-to-point channel for applications that require reliable communications.

TCP provides a communication service at an intermediate level between an application program and the Internet Protocol (IP). That is when an application program desires to send a large chunk of data across the Internet using IP, instead of breaking the data into IP-sized pieces and issuing a series of IP requests, the software can issue a single request to TCP and let TCP handle the IP details.

IP works by exchanging pieces of information called packets. A packet is a sequence of octets and consists of a header followed by a body. The header describes the packet's destination and optionally for routers to use for forwarding until it arrives at its destination. The body contains the data that IP is transmitting.

Due to network congestion, traffic load balancing or other unpredictable network behavior, IP packets can be lost, duplicated, or delivered out of order. TCP detects these problems, provides retransmission of lost data, rearranges out-of-order data, and even helps minimize network congestion to reduce the occurrence of the other problems. Once the TCP receiver has reassembled the sequence of octets originally transmitted, it passes them to the application program. Thus, TCP abstracts the application's communication from the underlying networking details.

➤ TCP HEADER FORMAT



Source: RFC793 (September 1981)

* Field refers to data transmission in reverse direction

Figure 1.1 TCP Header Format

➤ Connection Establishment / Termination

TCP connection establishment and termination process are given below

• Connection Establishment

- ☐ 3-way handshake (SYN, SYN-ACK, ACK)
- ☐ selection of initial sequence numbers
- ☐ agreement on maximum segment size

- **Connection end**

- ▣ Indicated by FIN flag
- ▣ Signaled for both transmission directions
- ▣ Alternative: RST flag

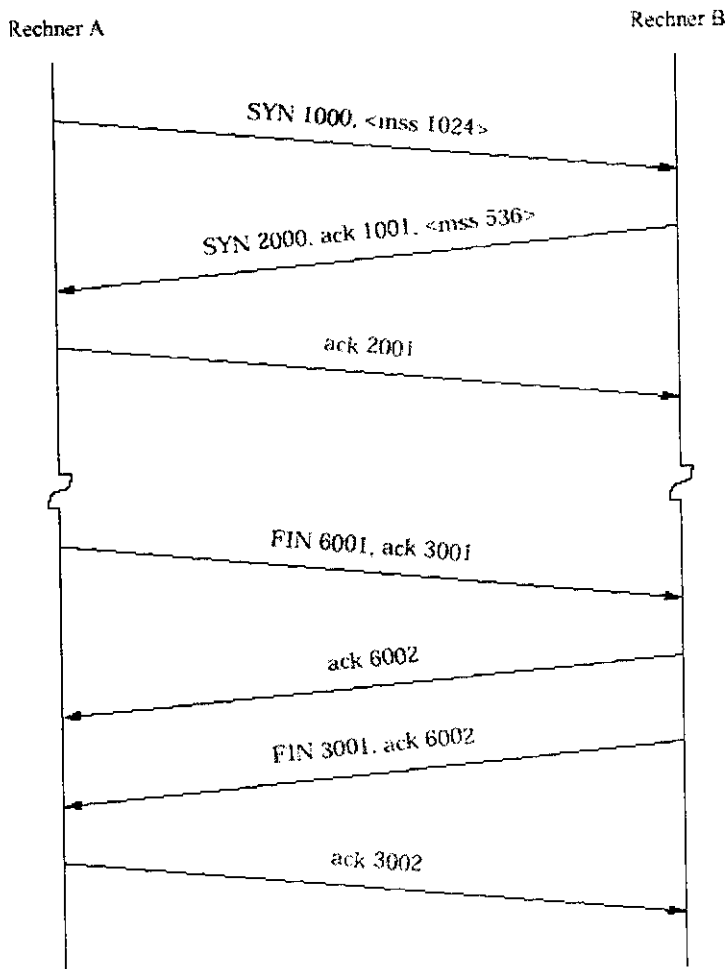


Figure 1.2 TCP Connection Establishment and Termination

➤ TCP Flow/Congestion Control

The TCP Flow control and Congestion control mechanisms are given below

▣ Flow Control

- Adapt sending rate to receiver speed
- Prevent Buffer Overflow at receiver

▣ Congestion Control

- Avoid network overload
- Prevent packet loss at buffers in intermediate systems (routers)
- 'fair' sharing of rates between all TCP connections at bottleneck links

CHAPTER 2

LITERATURE SURVEY

2.1 Transport Protocol Performance

Characteristics of wireless links that affect transport protocol performance

- Packet loss due to corruption
- Delay variation due to link-layer error recovery, handovers and scheduling
- Asymmetric and/or variable bandwidth (e.g., satellite)
- Shared bandwidth (e.g., WLANs)
- Complex link-level buffering (e.g., cellular links)
- Mobility

➤ TCP-Mechanisms in Wireless Settings

TCP assumes congestion in wireless networks if packets are dropped in certain scenario. But here we often have packet loss due to transmission errors. Mobility itself can cause packet loss (handover losses) or temporary connection disruptions (timeouts or even broken TCP connections).

2.2 Existing system

In wireless networks the performance of the transport protocol is not efficient under all conditions. The performance is largely depended on the number of users, mobility condition, network topology and routing protocol used. It does not efficiently support one to many connections in wireless networks. There is a large delay variation and packet loss in the high mobility condition. Bandwidth utilization is not to the optimum level. Existing works does not use effective routing protocols. These all affect the performance of the transport protocols. Performance of the transport protocol should be improved to have the efficient communication in wireless networks.

2.3 Related Works

Recently, improving the network performance has obtained great interest in research community. However limited work has addressed in the global. In the most of research efforts presented in the literature have discussed the problem of ineffective bandwidth utilization, high delay variation and other parameters.

2.3.1 Modeling Wireless Links for Transport Protocol

Wireless links are an important part of the Internet today. In the future the number of wireless or mobile hosts is likely to exceed the number of fixed hosts in the Internet. Internet access using wireless LANs and cellular links is growing particularly quickly. As an example, in Japan over 40 million users are accessing data services from mobile phones. The main classes of wireless links are wireless LANs, wide-area cellular links, and satellite links. In this project we focus on simulation models of these wireless links for unicast transport protocols. We do not consider ad-hoc and sensor wireless networks, as this is a separate modeling area on its own.

It is well known that the presence of a wireless link can significantly affect the performance of end-to-end transport protocols. Congestion control in today's Internet is based on an assumption that almost all packet losses result from congestion. Packet losses on wireless links that are from corruption rather than congestion violate this assumption. While many link

technologies include FEC (Forward Error Correction) and local retransmission for addressing corruption at the link layer, these mechanisms can introduce their own complications. A high variability of bandwidth and delay on wireless links can reduce the performance of transport protocols (including those transport protocols that consider increased delay as an indication of congestion). Furthermore, because cellular and satellite links have high latency, connections traversing them can be penalized by the TCP congestion control flows with high RTT (Round-Trip Time).

Mechanisms for reducing the bit error rate of cellular links, such as interleaving and FEC, are the main sources of high latency on cellular links. For some link technologies, large link-level queues contribute to unnecessarily high delay.

There is a clear benefit of using the same protocol stack for fixed and wireless links. It enables interoperability between users with an adjacent wireless link and the rest of the Internet. Therefore, existing and new transport protocols should be designed for good performance over both wireless and fixed links. At the same time, wireless links should be designed to minimize negative effects on transport protocols. There are more works on the evaluation and design of transport protocols over wireless links.

The essential aspects of model are types of Wireless Links, Topologies, Wireless Traffic and Performance Metrics.

- The important aspects of modeling wireless links for the evaluation of transport protocols are given in this work. For delay variation, bandwidth on demand and intersystem handovers comparisons are done for simulation models with measurement results. The drawbacks here are mobility presents a major challenge to transport protocols. The performance of the protocol is better only in short range distance communication and support only one to one communication.

2.3.2 TCP in a wireless world

Mobile wireless is one of the more challenging environments for the Internet protocols, and for TCP in particular. Here the transport protocol used within the mobile wireless environment is not TCP, but is instead a transport protocol that has been specifically adapted to mobile wireless. In this model, Internet applications interact with an application gateway to reach



the wireless world, and the application gateway uses a wireless transport protocol and potentially a modified version of the application data to interact with the wireless device.

An alternative is to allow mobile wireless devices to function as any other Internet-connected device. This approach requires some form of end-to-end direct IP continuity and an associated end-to-end TCP functionality, where the TCP path straddles both wired and wireless segments. Ensuring the efficient operation of TCP in this environment becomes integral to the development of the environment itself: The problem is no longer one of adjusting TCP to match the requirements of the wireless environment, but one of providing seamless interworking between the wired and wireless worlds.

TCP was designed for wire-based carriage and the protocol design makes numerous assumptions that are typical of such an environment. For example, it assumes that packet loss is the result of network congestion, that round-trip times (RTT) have some level of stability, that bandwidth is constant, and that session durations will justify the initial TCP handshake overhead. The wireless environment challenges these assumptions and others underlying TCP design. Wireless has significant bit-error rates (BER), often bursting to very high rates.

Wireless links that use forward error-correcting (FEC) codes often have high latency. If the link-level protocol uses a stop and resend mechanism that automatically retransmits corrupted data, this wireless segment latency will also have high variability. Wireless links may also use adaptive coding techniques that adjust to the prevailing signal-to-noise ratio, in which case the bandwidth of the link will vary. If the wireless device is a handheld mobile device, it may be memory constrained. And finally, wireless environments typically support short-duration sessions. The other mechanisms that are carried out by TCP in wireless networks are Forward ACK with Rate Halving, Link-Level Signaling, Managing Payloads, and Managing Link Outages. TCP performance is not efficient in network which high mobility and more congestion.

2.3.3 Vertical and Horizontal flow controls for TCP Optimization in mobile networks

A mobile adhoc network (MANET) is a set of wireless mobile nodes forming a dynamic autonomous network through a fully mobile infrastructure. This network is independent of any fixed infrastructure or centralized administration. A node communicates directly with other nodes within its wireless communication range without the intervention of centralized access point or base station.

The mechanisms of the two algorithms are given below

- Two algorithms are proposed to improve the TCP performance in the network.
- Packet drop would occur and retransmission will be carried out for the node it receives data more than capacity.
- In Vertical flow control algorithm if length of the queue is less than the maximum limit then packets are allowed.
- Then the maximum to be allowed further is difference of maximum limit and the packets allowed before.
- Otherwise TCP will not send packets.
- In horizontal flow control CTS and RTS mechanisms are used.
- If the normal CTS is received for the RTS sent the data can be send.
- If it receives flagged CTS then packet will not be sent and put in hold state.
- The throughput using these algorithms is better than that of the normal TCP.
- Power consumption is less compared to that of normal TCP
- It is not efficient when more users are in the mobility state causing high delay.

2.3.4 Improving End to End Performance of TCP over Mobile Internetworks

Reliable transport protocols such as TCP use end-to-end flow, congestion, and error control mechanism to provide reliable delivery over an inter network. However co-existence of wireless links and mobile hosts with fixed network poses unique problem for transport protocols. In particular, the following communication characteristics of wireless links have significant implications.

First, maximum transmission unit (MTU) on a wireless link is typically much smaller than that over links in the wired network. Small MTU over the first link forces transmissions of smaller packets over the entire end-to-end path even though wired path can accommodate much larger packets.

Second, the error rates on the wireless link are much higher than those experienced over the links in the wired networks. Higher error rates (and resulting intermittent connectivity)

over a wireless link are due to a combination of factors such as multipath fading, terrain and environmental factors, and interference from other transmissions. In addition, this error often causes a burst of packet to be lost.

Third, communication processes during handoffs are also perceived as periods of heavy losses by transport and higher levels of protocols. The connection will be established from mobile host to base station. Then from base station to fixed host. Data transfer is done by fragmenting to match the smaller MTU of wireless links. Selective repeat protocol is used for the error recovery.

The drawback of this work are in high handoff situations packet loss is high. Delay for the transmission is high in such situations.

2.3.5 Implementation and Performance Evaluation of Indirect TCP

The work carried out to improve the performance of Indirect TCP are given below

- Wireless links are slower and less reliable compared to wired links causing loss of signal due to noise and fading.
- Mobility also gives disconnection to the communication.
- Indirect TCP provide intermediate routers to provide backward compatibility to the fixed networks.
- During handoff the new router will wait for handoff request from the old router.
- Old router will make entry of the mobile host that moved out of the connection.
- During mobile host switching between two non overlapped cells the connectivity is lost for certain period of time causing delay to increase.
- Performance throughput is only from small to moderate in local area networks.

2.3.6 Analysis of Congestion Control Strategy for Wireless Network

TCP is widely used in Transmission Control Protocol; however, in the wireless network environment, TCP congestion control typically has some defects such as high error rate, long-latency, low-bandwidth and frequent-movement, etc. In the wireless network, the

implementation difficulty of congestion control mechanism is the degree of congestion, which is not only relevant to the length of queue, but also the wireless channel around the node is busy.

TCP provides reliable and sequence transmission service for hosts. TCP congestion control is important factor which has successfully applied in the internet. However compared with the wired network, wireless environment usually has disadvantages such as high data error rate, long latency, low bandwidth as well as frequent environment etc. Therefore, the traditional wire environment based on TCP technology has been unable to adapt to the wireless environment with relatively poor quality.

The difficulty of implementation congestion control mechanism in wireless network lies mainly in two aspects. Different from wired network, the degree of congestion is not only relevant to the length of the queue, but to whether the wireless channel around the node is busy, so the difficulty is how to measure these indicators quantitatively. Other difficulty is when congestion detected, control strategy can reduce the band width cost of the system as much as possible.

Node state can be divided into two parts, the external state and internal state. The performance of the external state is the state of node's receiving and sending packets, while the internal state is the length of the queue. Sending and receiving packets directly impact on the internal storage of the queue.

CHAPTER 3

METHODOLOGY

3.1 One to Many TCP Connection Establishment

To improve the performance of the wireless network the one to many TCP connections are established and the performance is evaluated. Wireless mesh topology is supported here. For the network initially one to one TCP connections are established. Then the communication is carried out. With the same network the one to many connections are established and then the performance is evaluated. The communication can be carried out in different ranges. Mainly the communications carried out here are,

- Short range communication.
- Long range communication.

Selection routing protocol is important in the communication carried out. The routing protocol varies for the range of communication carried out. The communication will be carried out using increasing the number of the nodes and also specifying the mobility conditions. Then the comparison of the both the communication will be carried out and performance will be evaluated.

3.2 Modules

The modules that are considered in this project are

- WLAN Communication
- Cellular Communication

3.2.1 WLAN Communication

This communication is an example for short range communication. The mobility conditions in this communication will be less when compared to other communication. The routing protocol used here is DSR (Dynamic Source Routing). The network is given with the one to one connections and communication is carried out. Then for the same network the one to many TCP connections are established and communication is carried out. Then the graph is drawn to show the performance of the network.

3.2.2 Cellular Communication

This communication is an example for long range communication. The mobility conditions in this communication will be high when compared to the other communication. The routing protocol used here is AODV. The network is given with one to one connections and the communication is carried out. Then for the same network the one to many TCP connections are established and communication are carried out. In cellular communication base station to the substation connectivity are given. The user can communicate with the substation. If the user moves from one substation to another substation the handoff technique is used to transfer the signal without loss of signal. Later the graph is plotted to show the performance of the network.

3.3 Dynamic Source Routing

Dynamic Source Routing (DSR) is a routing protocol for wireless mesh networks. It is similar to AODV in that it forms a route on-demand when a transmitting computer requests one. However, it uses source routing instead of relying on the routing table at each intermediate device. Many successive refinements have been made to DSR, including DSRFLOW.

Determining source routes requires accumulating the address of each device between the source and destination during route discovery. The accumulated path information is cached by nodes processing the route discovery packets. The learned paths are used to route packets. To accomplish source routing, the routed packets contain the address of each device the packet will traverse. This may result in high overhead for long paths or large addresses, like IPv6. To avoid

using source routing, DSR optionally defines a flow id option that allows packets to be forwarded on a hop-by-hop basis.

This protocol is truly based on source routing whereby all the routing information is maintained (continually updated) at mobile nodes. It has only two major phases, which are Route Discovery and Route Maintenance. Route Reply would only be generated if the message has reached the intended destination node (route record which is initially contained in route request and would be inserted into route reply).

To return the Route Reply, the destination node must have a route to the source node. If the route is in the Destination Node's route cache, the route would be used. Otherwise, the node will reverse the route based on the route record in the Route Reply message header (this requires that all links are symmetric). In the event of fatal transmission, the Route Maintenance Phase is initiated whereby the Route Error packets are generated at a node. The erroneous hop will be removed from the node's route cache then all routes containing the hop will be truncated at that point. Again, the Route Discovery Phase will be initiated to determine the most viable route.

Dynamic source routing protocol (DSR) is an on-demand protocol designed to restrict the bandwidth consumed by control packets in ad hoc wireless networks by eliminating the periodic table-update messages required in the table-driven approach. The major difference between this and the other on-demand routing protocols is that it is beacon-less and hence does not require periodic hello packet (beacon) transmissions, which are used by a node to inform its neighbors of its presence. The basic approach of this protocol (and all other on-demand routing protocols) during the route construction phase is to establish a route by flooding Route Request packets in the network. The destination node, on receiving a Route Request packet, responds by sending a Route Reply packet back to the source, which carries the route traversed by the Route Request packet received.

Consider a source node that does not have a route to the destination. When it has data packets to be sent to that destination, it initiates a Route Request packet. This Route Request is flooded throughout the network. Each node, upon receiving a Route Request packet, rebroadcasts the

packet to its neighbors if it has not forwarded it already, provided that the node is not the destination node and that the packet's time to live (TTL) counter has not been exceeded. Each Route Request carries a sequence number generated by the source node and the path it has traversed. A node, upon receiving a Route Request packet, checks the sequence number on the packet before forwarding it. The packet is forwarded only if it is not a duplicate Route Request. The sequence number on the packet is used to prevent loop formations and to avoid multiple transmissions of the same Route Request by an intermediate node that receives it through multiple paths. Thus, all nodes except the destination forward a Route Request packet during the route construction phase. A destination node, after receiving the first Route Request packet, replies to the source node through the reverse path the Route Request packet had traversed. Nodes can also learn about the neighboring routes traversed by data packets if operated in the promiscuous mode (the mode of operation in which a node can receive the packets that are neither broadcast nor addressed to itself). This route cache is also used during the route construction phase. If an intermediate node receiving a Route Request has a route to the destination node in its route cache, then it replies to the source node by sending a Route Reply with the entire route information from the source node to the destination node.

This protocol uses a reactive approach which eliminates the need to periodically flood the network with table update messages which are required in a table-driven approach. In this type of reactive (on-demand) approach such as this, a route is established only when it is required and hence the need to find routes to all other nodes in the network as required by the table-driven approach is eliminated. The intermediate nodes also utilize the route cache information efficiently to reduce the control overhead. The disadvantage of this protocol is that the route maintenance mechanism does not locally repair a broken link. Stale route cache information could also result in inconsistencies during the route reconstruction phase. The connection setup delay is higher than in table-driven protocols. Even though the protocol performs well in static and low-mobility environments, the performance degrades rapidly with increasing mobility. Also, considerable routing overhead is involved due to the source-routing mechanism employed in DSR. This routing overhead is directly proportional to the path length.

3.4 Ad hoc On-Demand Distance Vector Routing

Ad hoc On-Demand Distance Vector (AODV) is a routing protocol for mobile ad hoc networks (MANETs) and other networks. It is a reactive routing protocol, meaning that it establishes a route to a destination only on demand. In contrast, the most common routing protocols of the Internet are proactive, meaning they find routing paths independently of the usage of the paths. AODV is, as the name indicates, a distance-vector routing protocol. AODV avoids the counting-to-infinity problem of other distance-vector protocols by using sequence numbers on route updates, a technique pioneered by DSDV. AODV is capable of both unicast and multicast routing.

In AODV, the network is silent until a connection is needed. At that point the network node that needs a connection broadcasts a request for connection. Other AODV nodes forward this message, and record the node that they heard it from, creating an explosion of temporary routes back to the needy node. When a node receives such a message and already has a route to the desired node, it sends a message backwards through a temporary route to the requesting node. The needy node then begins using the route that has the least number of hops through other nodes. Unused entries in the routing tables are recycled after a time.

When a link fails, a routing error is passed back to a transmitting node, and the process repeats. Much of the complexity of the protocol is to lower the number of messages to conserve the capacity of the network. For example, each request for a route has a sequence number. Nodes use this sequence number so that they do not repeat route requests that they have already passed on. Another such feature is that the route requests have a "time to live" number that limits how many times they can be retransmitted. Another such feature is that if a route request fails, another route request may not be sent until twice as much time has passed as the timeout of the previous route request.

The advantage of AODV is that it creates no extra traffic for communication along existing links. Also, distance vector routing is simple, and doesn't require much memory or calculation. However AODV requires more time to establish a connection, and the initial communication to establish a route is heavier than some other approaches. The AODV Routing protocol uses an on-

demand approach for finding routes, that is, a route is established only when it is required by a source node for transmitting data packets. It employs destination sequence numbers to identify the most recent path. The major difference between AODV and Dynamic Source Routing (DSR) stems out from the fact that DSR uses source routing in which a data packet carries the complete path to be traversed. However, in AODV, the source node and the intermediate nodes store the next-hop information corresponding to each flow for data packet transmission. In an on-demand routing protocol, the source node floods the Route Request packet in the network when a route is not available for the desired destination. It may obtain multiple routes to different destinations from a single Route Request. The major difference between AODV and other on-demand routing protocols is that it uses a destination sequence number (DestSeqNum) to determine an up-to-date path to the destination. A node updates its path information only if the DestSeqNum of the current packet received is greater than the last DestSeqNum stored at the node.

A Route Request carries the source identifier (SrcID), the destination identifier (DestID), the source sequence number (SrcSeqNum), the destination sequence number (DestSeqNum), the broadcast identifier (BcastID), and the time to live (TTL) field. DestSeqNum indicates the freshness of the route that is accepted by the source. When an intermediate node receives a Route Request, it either forwards it or prepares a Route Reply if it has a valid route to the destination. The validity of a route at the intermediate node is determined by comparing the sequence number at the intermediate node with the destination sequence number in the Route Request packet. If a Route Request is received multiple times, which is indicated by the BcastID-SrcID pair, the duplicate copies are discarded. All intermediate nodes having valid routes to the destination, or the destination node itself, are allowed to send Route Reply packets to the source. Every intermediate node, while forwarding a Route Request, enters the previous node address and its BcastID. A timer is used to delete this entry in case a Route Reply is not received before the timer expires. This helps in storing an active path at the intermediate node as AODV does not employ source routing of data packets. When a node receives a Route Reply packet, information about the previous node from which the packet was received is also stored in order to forward the data packet to this next node as the next hop toward the destination.

The main advantage of this protocol is that routes are established on demand and destination sequence numbers are used to find the latest route to the destination. The connection setup delay is lower. One of the disadvantages of this protocol is that intermediate nodes can lead to inconsistent routes if the source sequence number is very old and the intermediate nodes have a higher but not the latest destination sequence number, thereby having stale entries. Also multiple Route Reply packets in response to a single Route Request packet can lead to heavy control overhead. Another disadvantage of AODV is that the periodic beaconing leads to unnecessary bandwidth consumption.

3.5 System Specification

3.5.1 Hardware Requirement

Processor: Pentium IV

Clock speed: 550 MHz

Hard Disk: 20 GB

RAM: 128 MB

Cache Memory: 512 KB

Monitor: Color Monitor

Keyboard: 104 keys

Mouse: 3 Buttons

3.5.2 Software requirement

Operating System: Fedora 8/Linux

Simulation tool: NS2

CHAPTER 4

IMPLEMENTATION

4.1 NETWORK SIMULATOR

Network simulators are used by people from different areas such as academic researchers, industrial developers, and Quality Assurance (QA) to design, simulate, verify, and analyze the performance of different networks protocols. They can also be used to evaluate the effect of the different parameters on the protocols being studied. Generally a network simulator will comprise of a wide range of networking technologies and protocols and help users to build complex networks from basic building blocks like clusters of nodes and links. With their help, one can design different network topologies using various types of nodes such as end-hosts, hubs, network bridges, routers, optical link-layer devices, and mobile units.

Generally speaking, network simulators try to model the real world networks. The principal idea is that if a system can be modeled, then features of the model can be changed and the corresponding results can be analyzed. As the process of model modification is relatively cheap than the complete real implementation, a wide variety of scenarios can be analyzed at low cost (relative to making changes to a real network). However, network simulators are not perfect. They cannot perfectly model all the details of the networks. But when the simulator is well modeled, they will be close enough so as to give the researcher a meaningful insight into the network under test and how changes will affect its operation.

Types of Network Simulators

- Commercial: OPNET, QualNet
- Open source: NS2, NS3, OMNeT++, SSFNet, J-Sim

4.2 Network Simulator 2 (NS2)

NS2 is one of the most popular open source network simulators. The original NS is a discrete event simulator targeted at networking research. In this section, we will give a brief introduction to the NS2 system.

4.2.1 Overview

NS2 is the second version of NS (Network Simulator). NS is originally based on REAL network simulator. The first version of NS was developed in 1989 and evolved a lot over the past few years. The current NS project is supported through DARPA. The current second version NS2 is widely used in academic research and it has a lot of packages contributed by different non-benefit groups.

4.2.2 Main features

First and foremost, NS2 is an object-oriented, discrete event driven network simulator which was originally developed at University of California-Berkeley. The programming it uses is C++ and OTcl (Tcl script language with Object-oriented extensions developed at MIT). The usage of these two programming language has its reason. The biggest reason is due to the internal characteristics of these two languages. C++ is efficient to implement a design but it is not very easy to be visual and graphically shown. It's not easy to modify and assembly different components and to change different parameters without a very visual and easy-to-use descriptive language. Moreover, for efficiency reason, NS2 separates control path implementations from the data path implementation. The event scheduler and the basic network component objects in the data path are written and compiled using C++ to reduce packet and event processing time. OTcl happens to have the feature that C++ lacks. So the combination of these two languages proves to be very effective. C++ is used to implement the detailed protocol and OTcl is used for users to control the simulation scenario and schedule the events. A simplified user's view of NS2 is shown in figure 4.1. The OTcl script is used to initiate the event scheduler, set up the network topology, and tell traffic source when to start and stop sending packets through event scheduler.

The scenes can be changed easily by programming in the OTcl (Object Tool Command Language) script. When a user wants to make a new network object, he can either write the new object or assemble a compound object from the existing object library, and plumb the data path through the object. This plumbing makes NS2 very powerful.

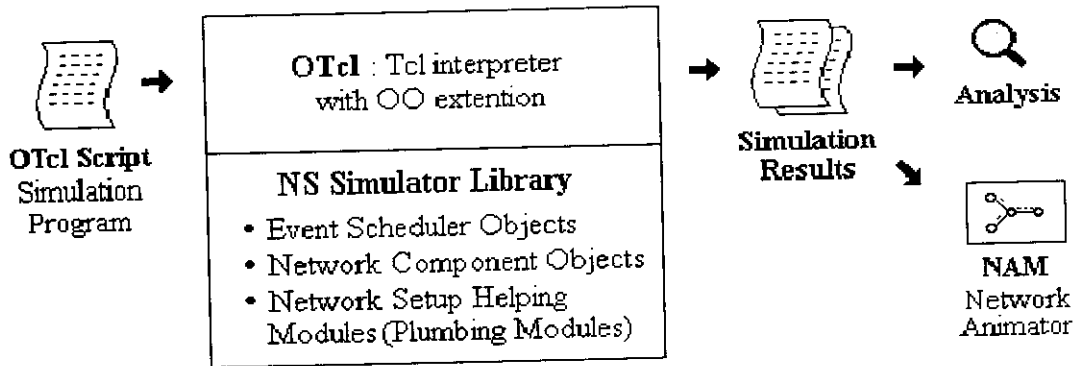


Figure 4.1 Simplified User's View of NS2

Another feature of NS2 is the event scheduler. In NS2, the event scheduler keeps track of simulation time and release all the events in the event queue by invoking appropriate network components. All the network components use the event scheduler by issuing an event for the packet and waiting for the event to be released before doing further action on the packet.

4.3 Components of NS

4.3.1 Nam, the Network Animator

Visualize ns (or other) output

GUI input simple ns scenarios

4.3.2 Pre-processing

Traffic and topology generators

4.3.3 Post-processing

Simple trace analysis, often in Awk, Perl, or Tcl

4.4 Goals of NS

- Support networking research and education
 - Protocol design, traffic studies, etc.
 - Protocol comparison
- Provide a collaborative environment
 - Freely distributed, open source
 - Share code, protocols, models, etc.
 - Allow easy comparison of similar protocols
 - Increase confidence in results
 - Models provide useful results in several situations
- It covers multiple layers
 - Application layer, transport layer, network layer and link layer.
- Supports the simulation of Intserv/diffserv, Multicast, Transport, Applications
Wireless(fixed, mobile, satellite)

4.5 NS-2 Programming Languages

- NS-2

It is an object oriented simulator, written in C++, with an OTcl (Object Tool Command Language) interpreter as a front-end.

- Back-end C++
 - Defining new agents, protocols and framework.
 - Manipulations at the byte/bit levels.
 - If you have to change the behavior of an existing C++ class in ways that weren't anticipated.

- Front-end Otcl
 - Topologies, scenarios, simulations
 - Script language (easy topology modifications)
 - If you can do what you want by manipulating existing C++ objects.

4.5.1 Why Two Languages

- Simulator had two distinct requirements
 - Detailed simulation of Protocol (Run-time speed)
 - Varying parameters or configuration (Change model & rerun)
- C++ is fast to run but slower to change
- Otcl runs much slower but can be changed quickly.

4.5.2 Network AniMator

Nam is a Tcl/TK based animation tool for viewing network simulation traces and real world packet traces.

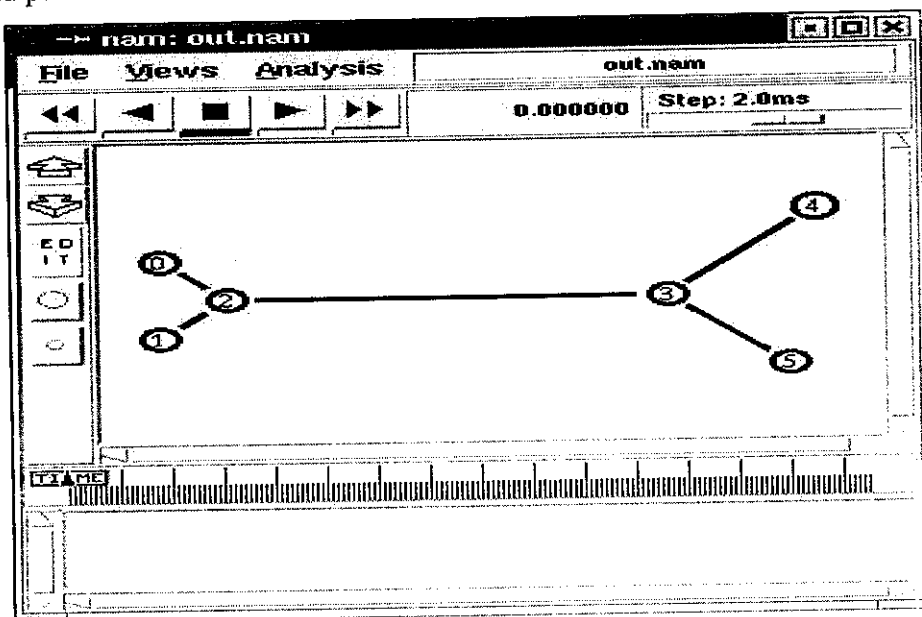


Figure 4.2 Nam window

4.5.3 Gnuplot

Gnuplot is a command-driven, interactive, function and data plotting program. Gnuplot supports many types of plots in either **2D** or **3D**. It can draw using lines, points, boxes,

contours, vector fields, surfaces, and various associated text. It also supports various specialized plot types.

Eg: Gnuplot> plot 'graph.dat'.

4.5.4 Trace File Formats

Used to trace packets on all links

Set tracefd [open simple.tr w]

\$ns_trace-all \$tracefd

1) Wired trace file format:

This trace format is normally used for normal wired operations.

Event	Abbreviation	Type	Value
		double	Time
		int	(Link-layer) Source Node
		int	(Link-layer) Destination Node
		string	Packet Name
Normal Event	r Receive	int	Packet Size
	d Drop	string	Flags
	e Error	int	Flow ID
	+ Enqueue	int	(Network-layer) Source Address
	- Dequeue	int	Source Port
		int	(Network-layer) Destination Address
		int	Destination Port
		int	Sequence Number
		int	Unique Packet ID

Table 4.1. Wired Trace File Format

2) Wireless trace file format:

An example of wireless trace output is,

r 40.639943289 _1_ AGT --- 1569 tcp 1032 [a2 1 2 800] -----

- The first field is a letter that can have the values r, s, f, D for “received”, “sent”, “forwarded” and dropped. Also M for movement indication.
- The second field is the time.
- The third field is the node number.

- The fourth field is MAC to indicate if the packet concerns a MAC layer, AGT to indicate transport layer, or RTR if it concerns routed packet.
- After the dashes comes global sequence number indicator.
- Next field comes for more information on packet type. (tcp, ack, or udp).
- Then comes packet size in bytes.
- Next concerns with MAC layer information. a2 specifies expected time to send data packet, 1 stands for MAC-id, 2 is that for receiving node, and 800 specifies that MAC type is ETHERTYPE_IP.
- The next numbers concern with IP source and destination address.
- The last numbers concerns the tcp information: its sequence number and acknowledgement number.

CHAPTER 5

RESULTS

5.1 Performance Evaluation

The one to many TCP connections are given and then performance of the TCP is compared with the one to one connections. Certain performance parameters are considered.

The performances analyzed are,

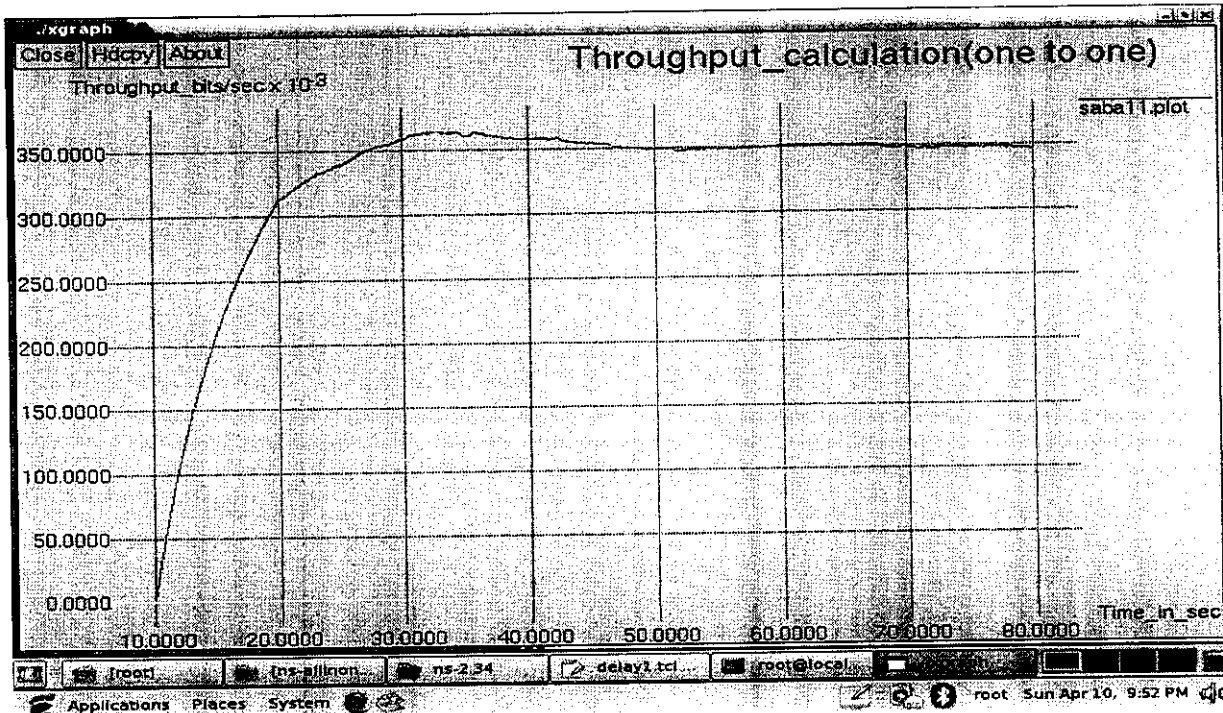
Throughput – Number of packets transmitted by total number of packets received by all members.

Delay – Number of packets sent by Average time received by all receivers.

Routing Overhead – Total number of packets sent as routing messages in the overall transmission of packets in the network.

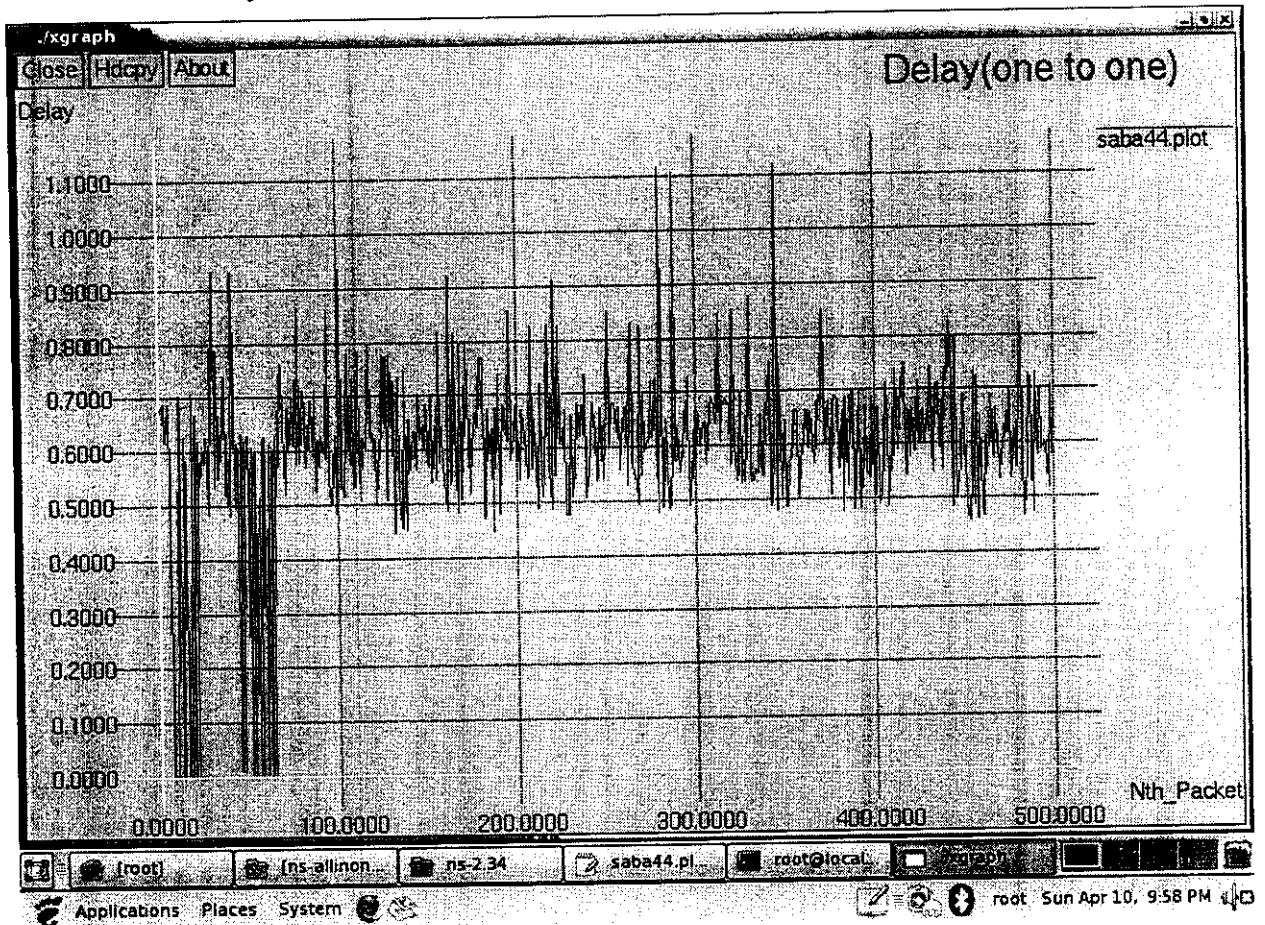
5.1.1 WLAN Communication (one to one connections)

The wireless network is given one to one TCP connections and communication is carried out. The throughput graph is given below for the communication carried out.



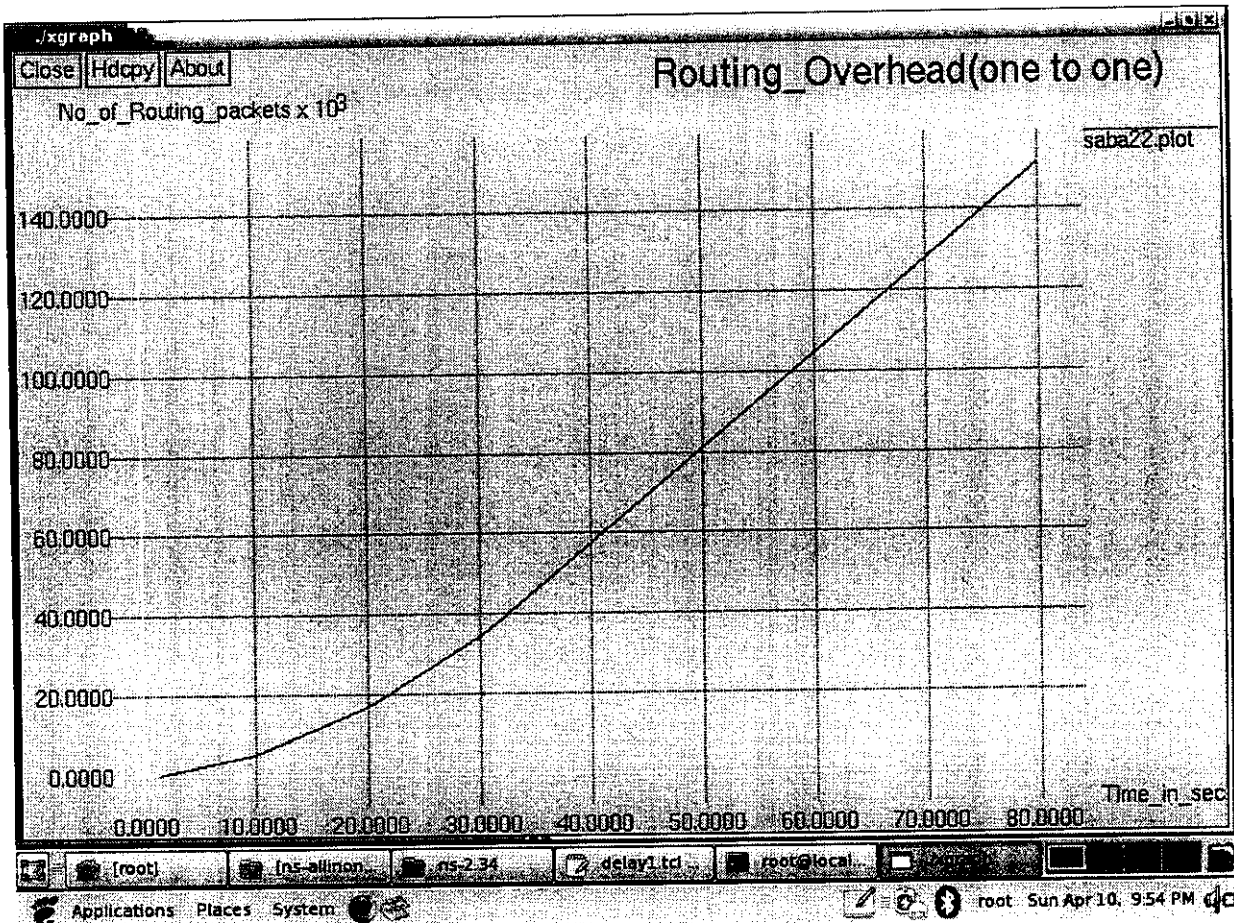
Then the delay carried out during the communication is given. In the one to one communication carried out in WLAN the mobility conditions are not given. It is also carried out in the short range communication.

The delay for this communication carried out is given below



Delay variation is only less in the one to one communication. Because of the short range distance and mobility conditions are rare.

Then the Routing overhead transmission is considered. The connection is established between eight nodes. The one to one communication is established between them. Thus source will send packets to same single destination from the start to the end of the transmission. The total numbers of routing messages that are sent in the transmission are given below.

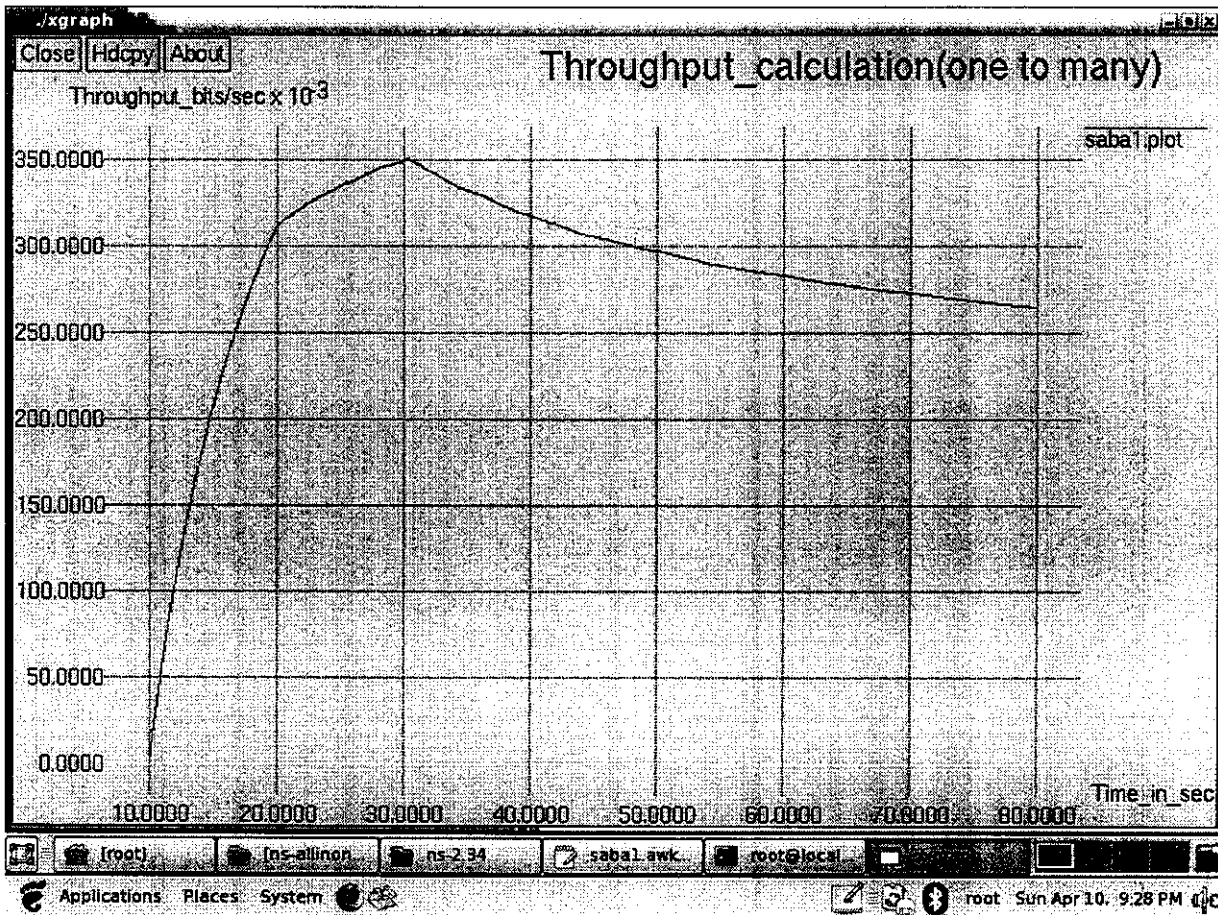


5.1.2 WLAN Communication (one to many connections)

The wireless network is given one to many TCP connections and communication is carried out. The throughput graph is given below for the communication carried out.

The graph is plotted with the different communication carried out. The four communications is carried out in this transmission. Thus the throughput for the all communications is given.

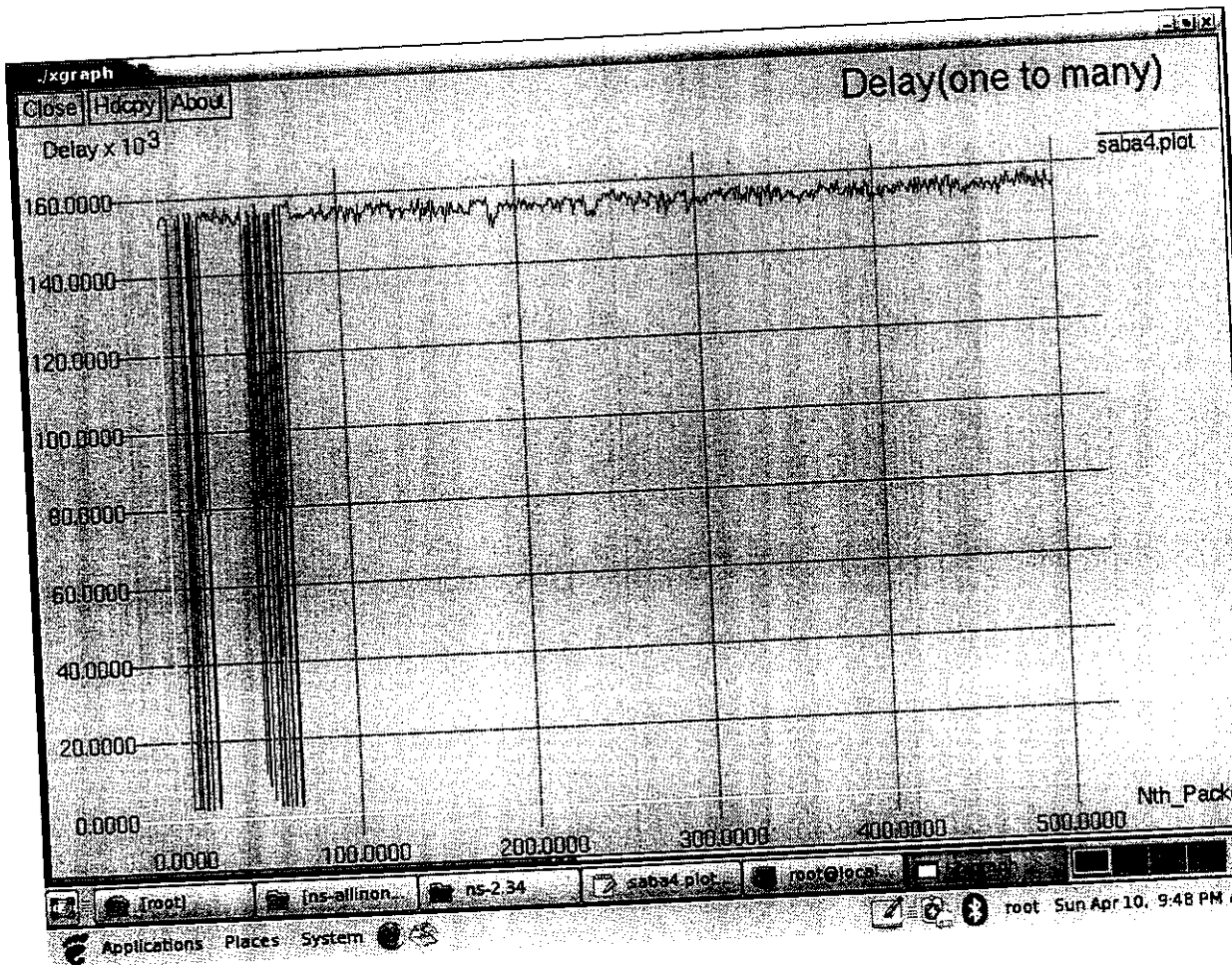
X axis represents the time in seconds. Y axis represents the throughput packets.



Then the delay carried out during the communication is given. In the one to many communication carried out in WLAN the mobility conditions are not given. It is also carried out in the short range communication.

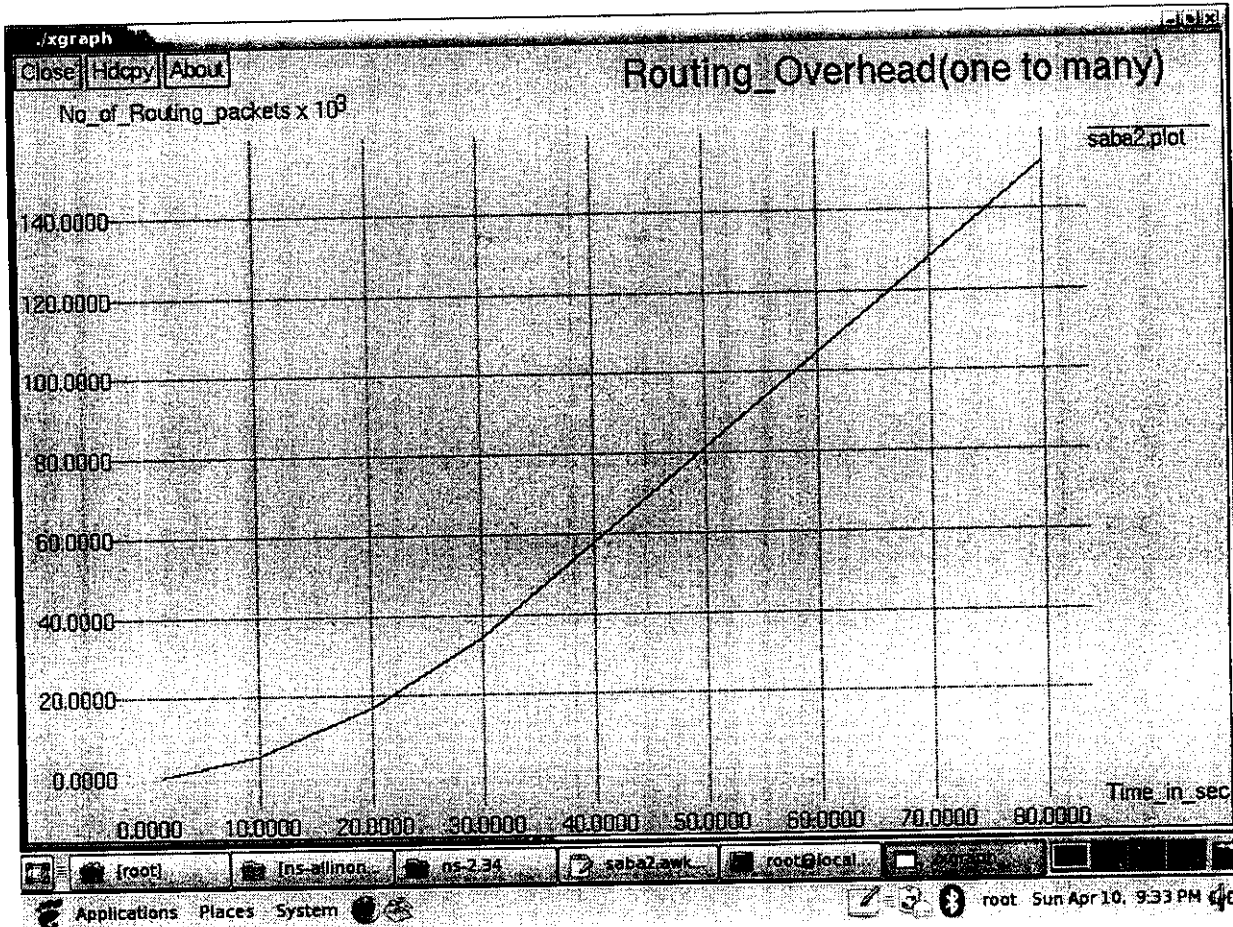
The delay is marginally high compared to one to one communication. The short range distance and without mobility conditions are only given. This helps in reducing the delay carried out in the transmission.

The graph is plotted with the different communication carried out with the respective delays. The graph is given as follows.



Then the Routing overhead is considered. The connection is established between eight nodes. The one to many communication is established between them. Thus source will send packets to different destination from the start to the end of the transmission. The total numbers of routing messages that are sent in the transmission are given below.

In the graph plotted time is represented in the X axis and the numbers of packets are represented in the Y axis.

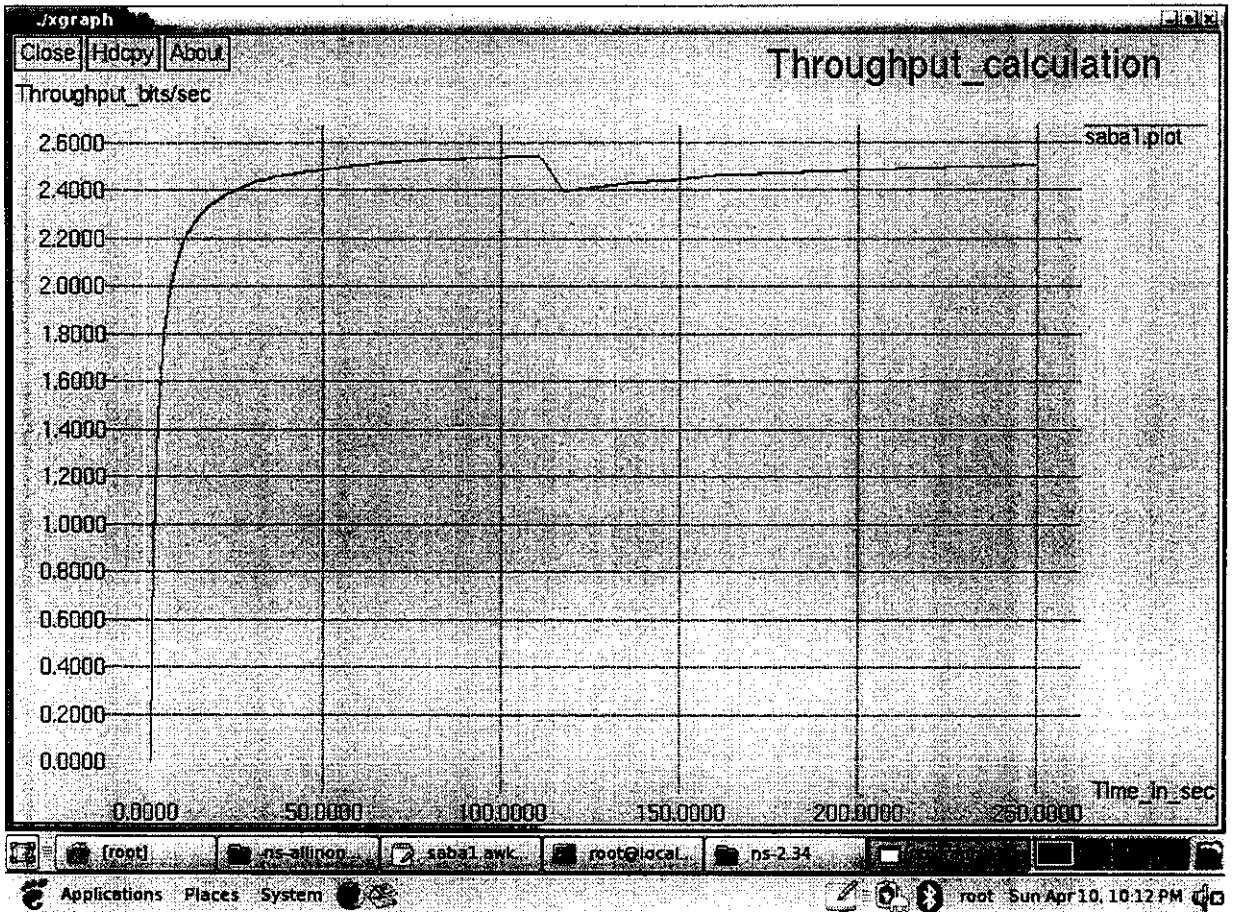


5.1.3 Cellular Communication (one to one connections)

The wireless network is given one to one TCP connections and communication is carried out. Here the mobility conditions are given. The base station and substation concept is used here. Long range communication is carried out here.

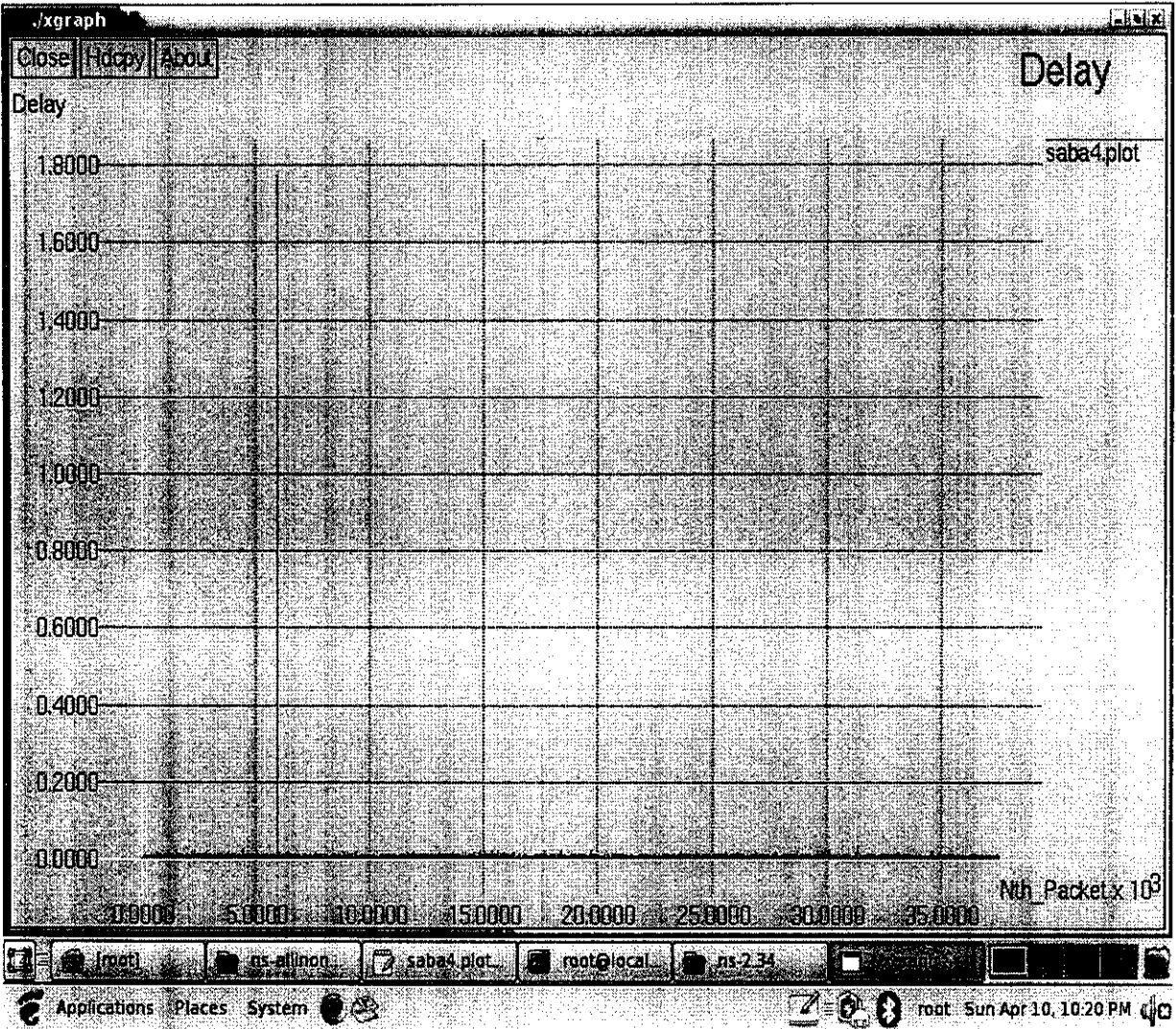
The throughput graph is given for the communication carried out. The throughput obtained for the transmission carried out using the one to one connection is high. Later the same communication is given for the one to many connections and performance is evaluated.

The graph showing the throughput for the one to one communication is given below.



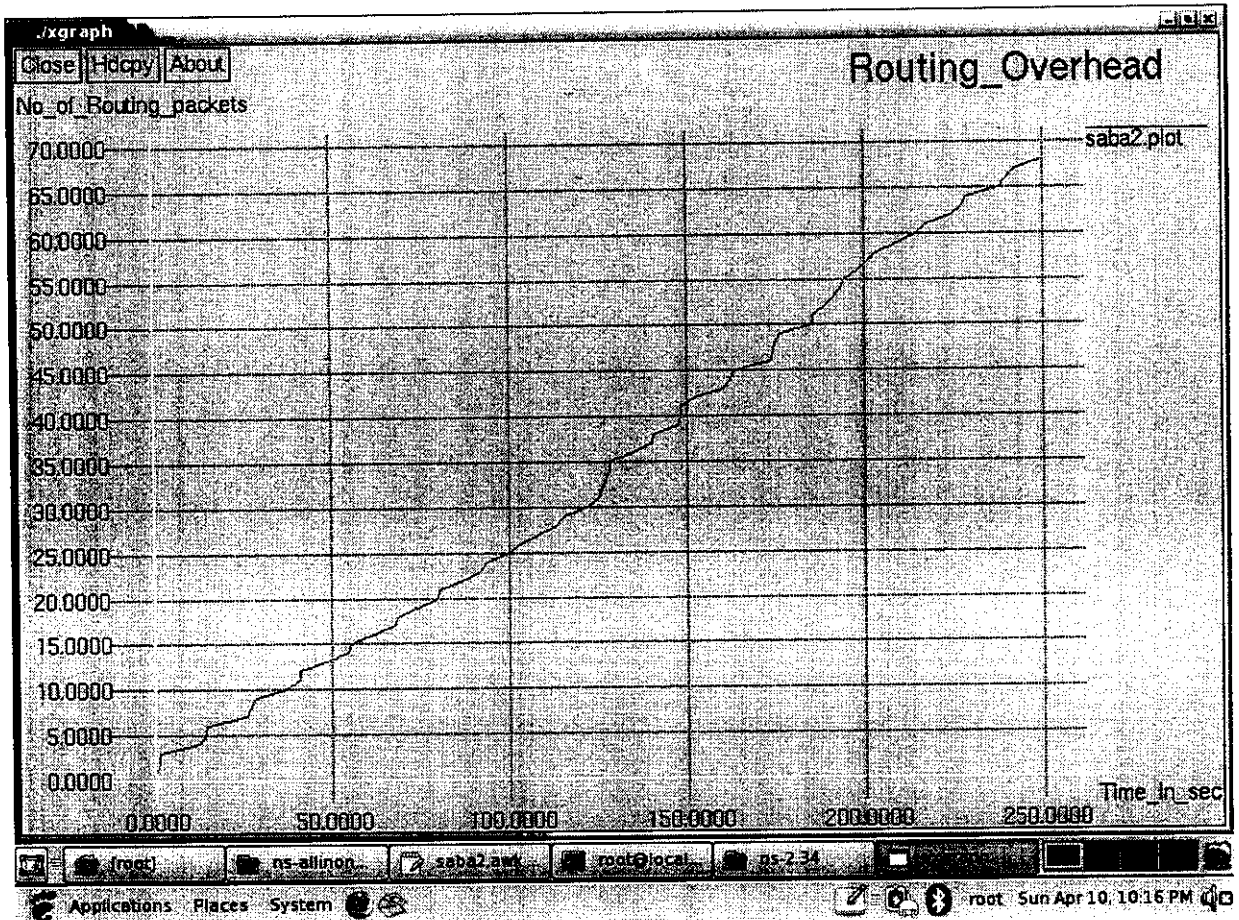
The delay variation in one to one communication carried out in the network is considered. In the long range communication mobility condition will be high. Thus delay variation in the network will vary according to the number of stations that switch and the range of distance the communication is carried out.

In the X axis the Nth packets is represented and Y axis represents the delay in seconds.



The Routing overhead in one to one communication carried out in the network is considered. The number of routing messages that are totally sent in the transmission is represented below.

The graph plotted shows the time in seconds as X axis and the number of routing messages in the Y axis.

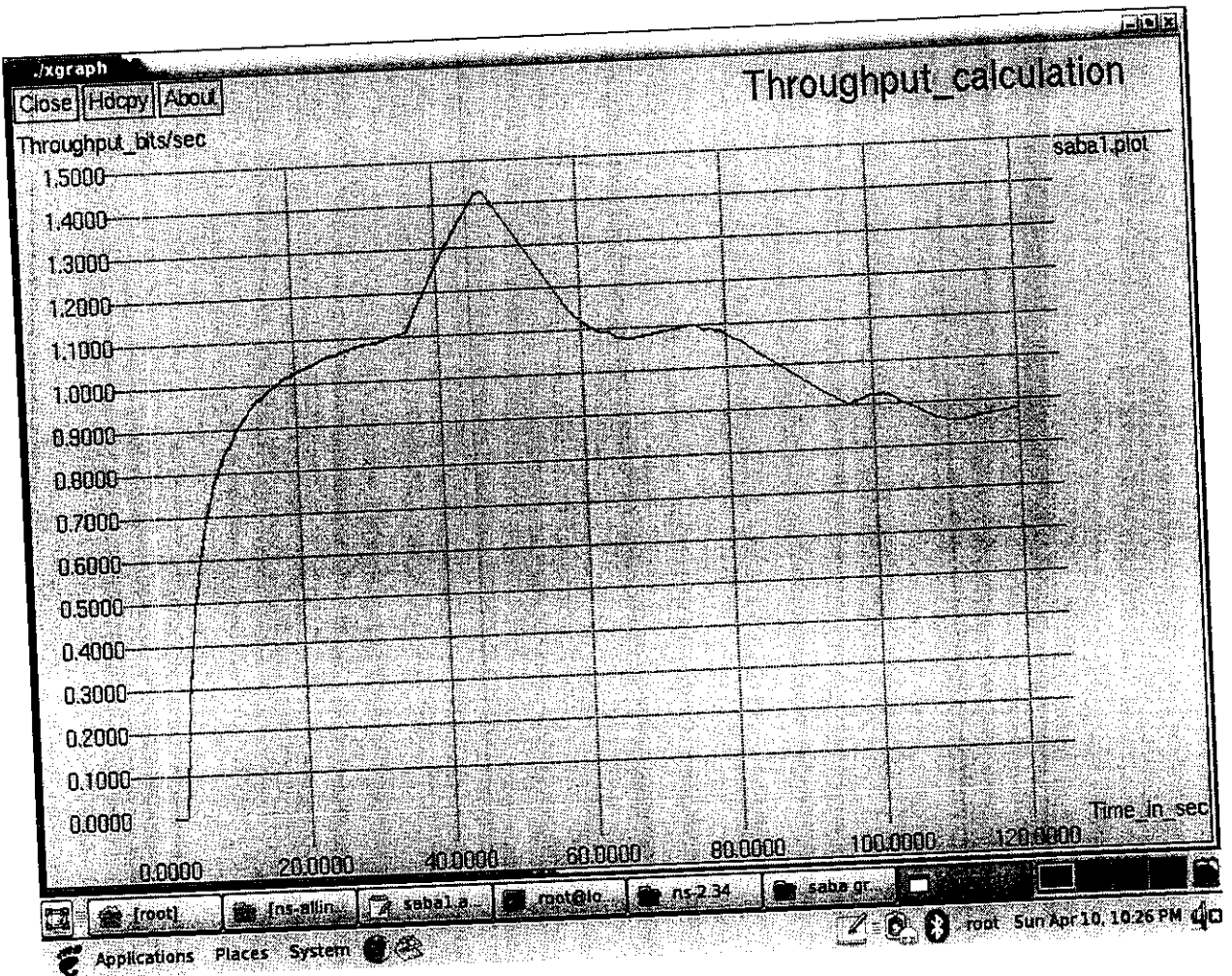


5.1.4 Cellular Communication (one to many connections)

The wireless network is given one to many TCP connections and communication is carried out. Here the mobility conditions are given. The base station and substation concept is used here. Long range communication is carried out here.

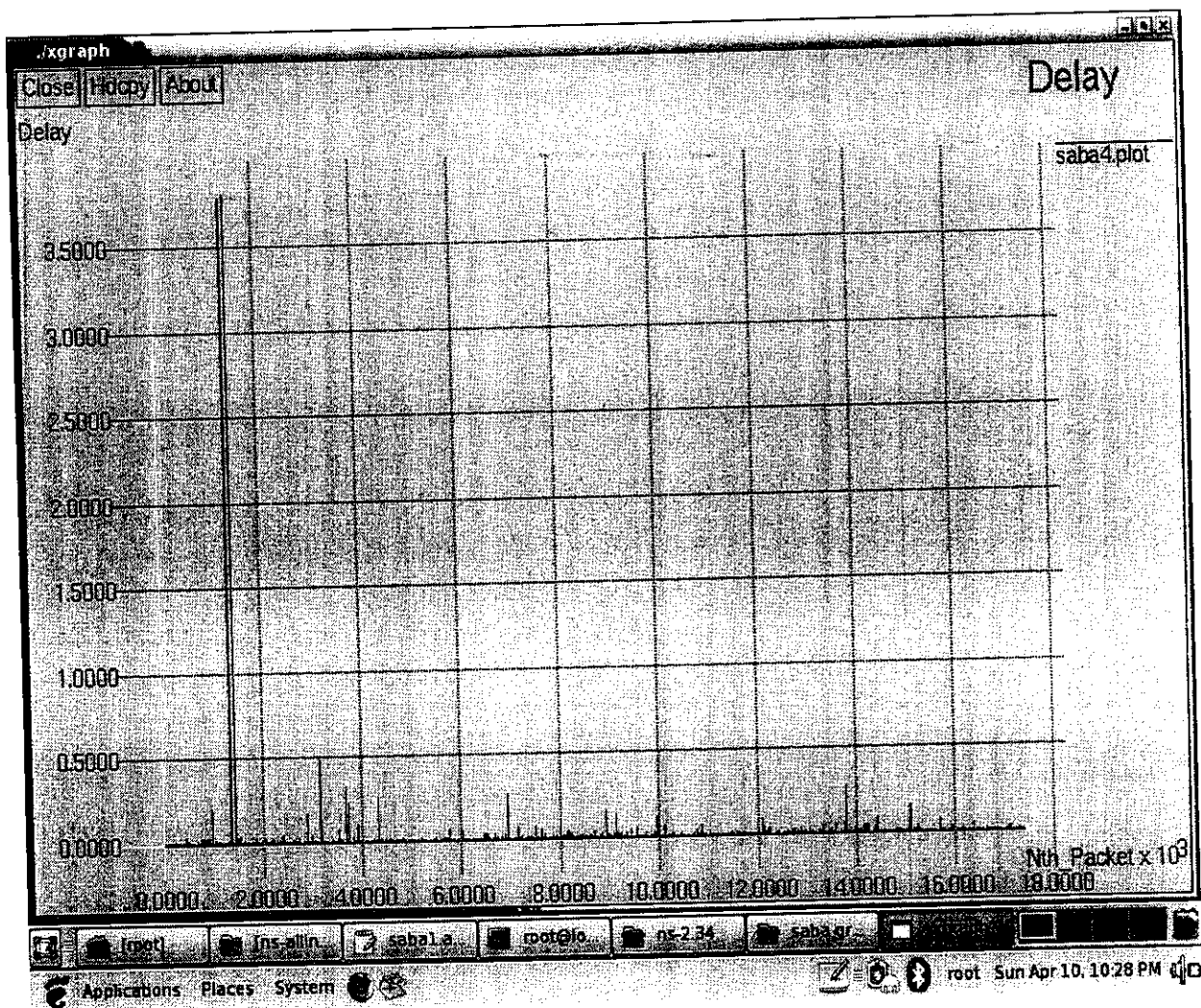
The throughput graph is given for the communication carried out. The throughput obtained for the transmission carried out using the one to many connections depends on the number of nodes communicates in the transmission.

The graph showing the throughput for the one to many communications is given below.



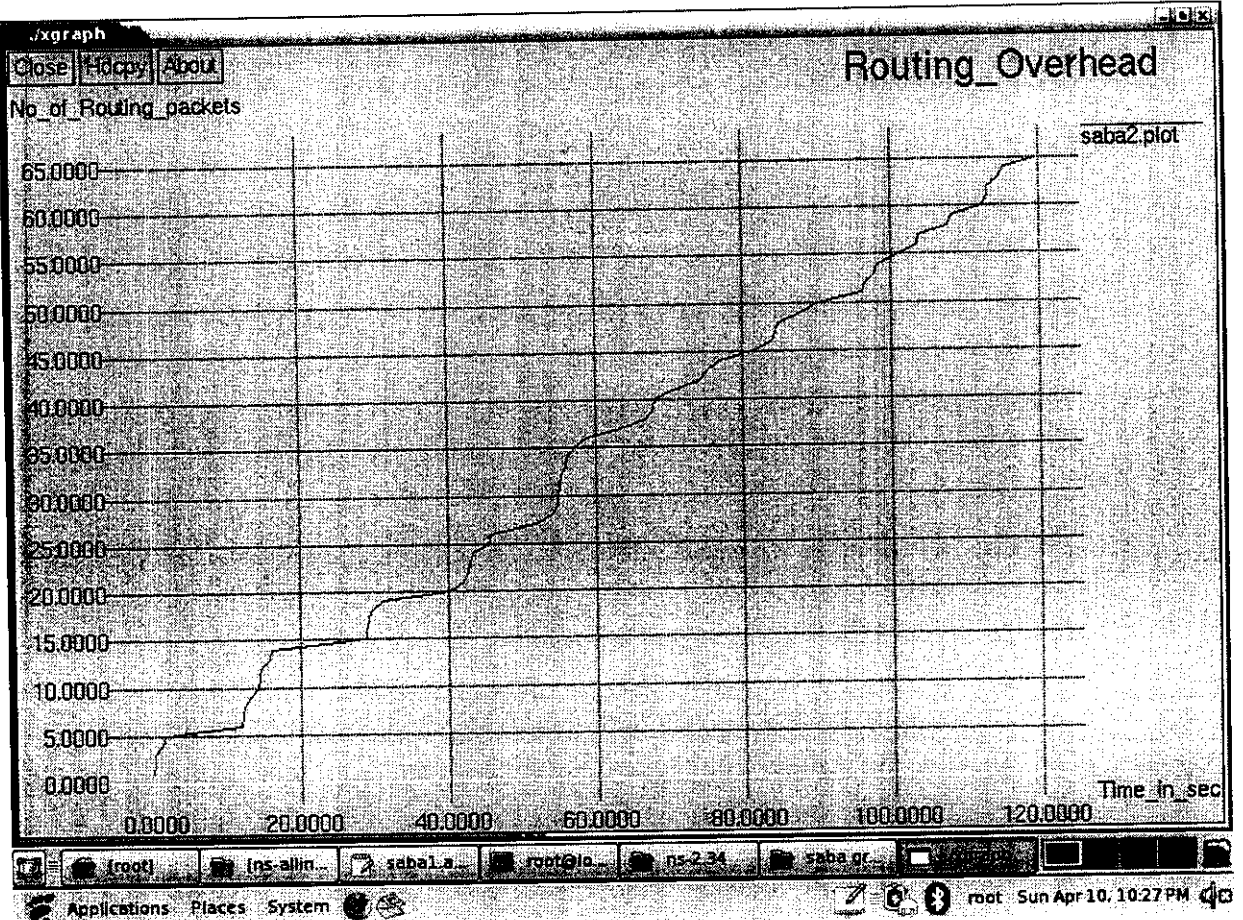
The delay variation in one to many communication carried out in the network is considered. In the long range communication mobility condition will be high. Thus delay variation in the network will vary according to the number of stations that switch and the range of distance the communication is carried out.

In the X axis the Nth packets is represented and Y axis represents the delay in seconds.



The Routing overhead in one to many communication carried out in the network is considered. The number of routing messages that are totally sent in the transmission is represented below.

The graph plotted shows the time in seconds as X axis and the number of routing messages in the Y axis.



CHAPTER 6

CONCLUSION AND FUTURE OUTLOOK

6.1 Conclusion

The performance of the transport protocol is important for the effective transmission in the wireless networks. We have seen the detailed study of the previous work which were proposed to improve the performance of the network. However, the work does not guarantee the effective network performance. To solve this problem, we proposed establishment of the one to many TCP connections in the wireless networks and performance of the TCP in different communications are evaluated. One to many TCP connections better utilizes the bandwidth. The other parameters such as delay in the wireless network is also compared with the one to one connections established. Simulation results, shows the effective bandwidth utilization during mobility conditions also. This means that one to many connection establishment performs transmission task with improved bandwidth utilisation.

6.2 Future Outlook

The proposed work is highly effective in the short range communication (WLAN) and in less mobility condition of long range communication. When large hand off situations occur then packet drop is possible. In future enhancements connection establishment in wireless networks under high mobility condition without packet drop should be implemented.

APPENDICES

7.1 Source Code

#coding for WLAN Communication supporting one to one communication

Define Node Configuration paramaters

```

=====
#
set val(chan)      Channel/WirelessChannel    ;# channel type
set val(prop)      Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)     Phy/WirelessPhy           ;# network interface type

set val(mac)       Mac/802_11                ;# MAC type
set val(ifq)       Queue/DropTail/PriQueue   ;# interface queue type

set val(ll)        LL                        ;# link layer type
set val(ant)       Antenna/OmniAntenna      ;# antenna model
set val(ifqlen)    50                        ;# max packet in ifq
set val(nn)        8                         ;# number of mobilenodes
set val(rp)        DSR                       ;# routing protocol
set val(x)         500                       ;# X dimension of the topography
set val(y)         500                       ;# Y dimension of the topography

```

```

Mac/802_11 set RTSThreshold_ 3000
Mac/802_11 set basicRate_ 1Mb
Mac/802_11 set dataRate_ 2Mb

```

=====

=

Initialize trace file descriptors

=====

=

*** Throughput Trace ***

```

set f0 [open out02.tr w]
set f1 [open out12.tr w]
set f2 [open out22.tr w]
set f3 [open out32.tr w]

```

*** Packet Loss Trace ***

```

set f4 [open lost02.tr w]
set f5 [open lost12.tr w]
set f6 [open lost22.tr w]
set f7 [open lost32.tr w]

```



```

# *** Packet Delay Trace ***
set f8 [open delay02.tr w]
set f9 [open delay12.tr w]
set f10 [open delay22.tr w]
set f11 [open delay32.tr w]

# *** Initialize Simulator ***
set ns_ [new Simulator]

# *** Initialize Trace file ***
set tracefd [open trace2.tr w]
$ns_ trace-all $tracefd

# *** Initialize Network Animator ***
set namtrace [open sim12.nam w]
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)

# *** set up topography object ***
set topo [new Topography]
$topo load_flatgrid 500 500

# Create General Operations Director (GOD) object. It is used to store global information about
the state of the environment, network, or nodes that an
# omniscient observer would have, but that should not be made known to any participant in the
simulation.

create-god $val(nn)

# configure nodes
$ns_ node-config -adhocRouting $val(rp) \
                -llType $val(ll) \
                -macType $val(mac) \
                -ifqType $val(ifq) \
                -ifqLen $val(ifqlen) \
                -antType $val(ant) \
                -propType $val(prop) \
                -phyType $val(netif) \
                -channelType $val(chan) \
                -topoInstance $topo \
                -agentTrace ON \
                -routerTrace ON \
                -macTrace OFF \
                -movementTrace OFF

```

```
# Create Nodes
```

```
    for {set i 0} {$i < $val(nm) } {incr i} {
        set node_($i) [$ns_node]
        $node_($i) random-motion 0           ;# disable random motion
    }
```

```
# Initialize Node Coordinates
```

```
$node_(0) set X_ 5.0
$node_(0) set Y_ 5.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 200.0
$node_(1) set Y_ 5.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 5.0
$node_(2) set Y_ 50.0
$node_(2) set Z_ 0.0

$node_(3) set X_ 200.0
$node_(3) set Y_ 50.0
$node_(3) set Z_ 0.0

$node_(4) set X_ 5.0
$node_(4) set Y_ 100.0
$node_(4) set Z_ 0.0

$node_(5) set X_ 200.0
$node_(5) set Y_ 100.0
$node_(5) set Z_ 0.0

$node_(6) set X_ 2.0
$node_(6) set Y_ 150.0
$node_(6) set Z_ 0.0

$node_(7) set X_ 200.0
$node_(7) set Y_ 150.0
$node_(7) set Z_ 0.0
```

```
# Setup traffic flow between nodes
```

```
# UDP connections between node_(0) and node_(1)
```

```
# Create Constant four Bit Rate Traffic sources
```

```

set agent1 [new Agent/UDP]           ;# Create UDP Agent
#$agent1 set prio_ 0                 ;# Set Its priority to 0
set sink [new Agent/LossMonitor]     ;# Create Loss Monitor Sink in order to be able to trace the
number obytes received
$ns_ attach-agent $node_(0) $agent1  ;# Attach Agent to source node
$ns_ attach-agent $node_(1) $sink    ;# Attach Agent to sink node
$ns_ connect $agent1 $sink           ;# Connect the nodes
set app1 [new Application/Traffic/CBR] ;# Create Constant Bit Rate application
$app1 set packetSize_ 512            ;# Set Packet Size to 512 bytes
$app1 set rate_ 600Kb                 ;# Set CBR rate to 200 Kbits/sec
$app1 attach-agent $agent1           ;# Attach Application to agent

set agent2 [new Agent/UDP]           ;# Create UDP Agent
#$agent2 set prio_ 1                 ;# Set Its priority to 1
set sink2 [new Agent/LossMonitor]    ;# Create Loss Monitor Sink in order to be able to trace the
number obytes received
$ns_ attach-agent $node_(2) $agent2  ;# Attach Agent to source node
$ns_ attach-agent $node_(3) $sink2   ;# Attach Agent to sink node
$ns_ connect $agent2 $sink2          ;# Connect the nodes
set app2 [new Application/Traffic/CBR] ;# Create Constant Bit Rate application
$app2 set packetSize_ 512            ;# Set Packet Size to 512 bytes
$app2 set rate_ 600Kb                 ;# Set CBR rate to 200 Kbits/sec
$app2 attach-agent $agent2           ;# Attach Application to agent

set agent3 [new Agent/UDP]           ;# Create UDP Agent
#$agent3 set prio_ 2                 ;# Set Its priority to 2
set sink3 [new Agent/LossMonitor]    ;# Create Loss Monitor Sink in order to be able to trace the
number obytes received
$ns_ attach-agent $node_(4) $agent3  ;# Attach Agent to source node
$ns_ attach-agent $node_(5) $sink3   ;# Attach Agent to sink node
$ns_ connect $agent3 $sink3          ;# Connect the nodes
set app3 [new Application/Traffic/CBR] ;# Create Constant Bit Rate application
$app3 set packetSize_ 512            ;# Set Packet Size to 512 bytes
$app3 set rate_ 600Kb                 ;# Set CBR rate to 200 Kbits/sec
$app3 attach-agent $agent3           ;# Attach Application to agent

set agent4 [new Agent/UDP]           ;# Create UDP Agent
#$agent4 set prio_ 3                 ;# Set Its priority to 3
set sink4 [new Agent/LossMonitor]    ;# Create Loss Monitor Sink in order to be able to trace the
number obytes received
$ns_ attach-agent $node_(6) $agent4  ;# Attach Agent to source node
$ns_ attach-agent $node_(7) $sink4   ;# Attach Agent to sink node
$ns_ connect $agent4 $sink4          ;# Connect the nodes
set app4 [new Application/Traffic/CBR] ;# Create Constant Bit Rate application
$app4 set packetSize_ 512            ;# Set Packet Size to 512 bytes
$app4 set rate_ 600Kb                 ;# Set CBR rate to 200 Kbits/sec

```

```

$app4 attach-agent $agent4           ;# Attach Application to agent

# defines the node size in Network Animator

for {set i 0} {$i < $val(nn)} {incr i} {
    $ns_ initial_node_pos $node_($i) 20
}

# Initialize Flags
set holdtime 0
set holdseq 0

set holdtime1 0
set holdseq1 0

set holdtime2 0
set holdseq2 0

set holdtime3 0
set holdseq3 0

set holdrate1 0
set holdrate2 0
set holdrate3 0
set holdrate4 0

# Function To record Statistics (Bit Rate, Delay, Drop)

proc record {} {
    global sink sink2 sink3 sink4 f0 f1 f2 f3 f4 f5 f6 f7 holdtime holdseq holdtime1 holdseq1
    holdtime2 holdseq2 holdtime3 holdseq3 f8 f9 f10 f11 holdrate1 holdrate2 holdrate3 holdrate4

    set ns [Simulator instance]

    set time 0.9 ;#Set Sampling Time to 0.9 Sec

    set bw0 [$sink set bytes_]
    set bw1 [$sink2 set bytes_]
    set bw2 [$sink3 set bytes_]
    set bw3 [$sink4 set bytes_]

    set bw4 [$sink set nlost_]
    set bw5 [$sink2 set nlost_]
    set bw6 [$sink3 set nlost_]
    set bw7 [$sink4 set nlost_]

```

```
set bw8 [$sink set lastPktTime_]
set bw9 [$sink set npkts_]
```

```
set bw10 [$sink2 set lastPktTime_]
set bw11 [$sink2 set npkts_]
```

```
set bw12 [$sink3 set lastPktTime_]
set bw13 [$sink3 set npkts_]
```

```
set bw14 [$sink4 set lastPktTime_]
set bw15 [$sink4 set npkts_]
```

```
set now [$ns now]
```

```
# Record Bit Rate in Trace Files
```

```
puts $f0 "$now [expr (($bw0+$holdrate1)*8)/(2*$time*1000000)]"
puts $f1 "$now [expr (($bw1+$holdrate2)*8)/(2*$time*1000000)]"
puts $f2 "$now [expr (($bw2+$holdrate3)*8)/(2*$time*1000000)]"
puts $f3 "$now [expr (($bw3+$holdrate4)*8)/(2*$time*1000000)]"
```

```
# Record Packet Loss Rate in File
```

```
puts $f4 "$now [expr $bw4/$time]"
puts $f5 "$now [expr $bw5/$time]"
puts $f6 "$now [expr $bw6/$time]"
puts $f7 "$now [expr $bw7/$time]"
```

```
# Record Packet Delay in File
```

```
if { $bw9 > $holdseq } {
    puts $f8 "$now [expr ($bw8 - $holdtime)/($bw9 - $holdseq)]"
} else {
    puts $f8 "$now [expr ($bw9 - $holdseq)]"
}
```

```
if { $bw11 > $holdseq1 } {
    puts $f9 "$now [expr ($bw10 - $holdtime1)/($bw11 - $holdseq1)]"
} else {
    puts $f9 "$now [expr ($bw11 - $holdseq1)]"
}
```

```
if { $bw13 > $holdseq2 } {
    puts $f10 "$now [expr ($bw12 - $holdtime2)/($bw13 - $holdseq2)]"
} else {
    puts $f10 "$now [expr ($bw13 - $holdseq2)]"
}
```

```
if { $bw15 > $holdseq3 } {
```

```

    puts $f11 "$now [expr ($bw14 - $holdtime3)/($bw15 - $holdseq3)]"
  } else {
    puts $f11 "$now [expr ($bw15 - $holdseq3)]"
  }

```

```

# Reset Variables
$sink set bytes_ 0
$sink2 set bytes_ 0
$sink3 set bytes_ 0
$sink4 set bytes_ 0

```

```

$sink set nlost_ 0
$sink2 set nlost_ 0
$sink3 set nlost_ 0
$sink4 set nlost_ 0

```

```

set holdtime $bw8
set holdseq $bw9

```

```

set holdrate1 $bw0
set holdrate2 $bw1
set holdrate3 $bw2
set holdrate4 $bw3

```

```

}
$ns at [expr $now+$time] "record"; # Schedule Record after $time interval sec
}

```

```

# Start Recording at Time 0
$ns_ at 0.0 "record"

```

```

$ns_ at 1.4 "$app1 start"           ;# Start transmission at time t = 1.4 Sec

```

```

$ns_ at 10.0 "$app2 start"         ;# Start transmission at time t = 10 Sec

```

```

$ns_ at 20.0 "$app3 start"         ;# Start transmission at time t = 20 Sec

```

```

$ns_ at 30.0 "$app4 start"         ;# Start transmission at time t = 30 Sec

```

```

# Stop Simulation at Time 80 sec
$ns_ at 80.0 "stop"

```

```

# Reset Nodes at time 80 sec

```

```

for {set i 0} {$i < $val(nn)} {incr i} {
  $ns_ at 80.0 "$node_($i) reset";
}

```

```

}

# Exit Simulatoion at Time 80.01 sec
$ns_ at 80.01 "puts \"NS EXITING...\" ; $ns_ halt"

proc stop {} {
    global ns_ tracefd f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 f10 f11

    # Close Trace Files
    close $f0
    close $f1
    close $f2
    close $f3

    close $f4
    close $f5
    close $f6
    close $f7

    close $f8
    close $f9
    close $f10
    close $f11

    # Plot Recorded Statistics
    exec ./xgraph out02.tr out12.tr out22.tr out32.tr -geometry 800x400 &
    exec ./xgraph lost02.tr lost12.tr lost22.tr lost32.tr -geometry 800x400 &
    exec ./xgraph delay02.tr delay12.tr delay22.tr delay32.tr -geometry 800x400 &

    # Reset Trace File
    $ns_ flush-trace
    close $tracefd
    exec ./nam sim12.nam &
    exit 0
}

puts "Starting Simulation..."
$ns_ run

```

#coding for WLAN Communication supporting one to many communication

Setup traffic flow between nodes

UDP connections between node_(0) and node_(1)

Create Constant four Bit Rate Traffic sources

```
set agent1 [new Agent/UDP]           ;# Create UDP Agent
#$agent1 set prio_ 0                 ;# Set Its priority to 0
set sink [new Agent/LossMonitor]     ;# Create Loss Monitor Sink in order to be able to trace the
number obytes received
$ns_ attach-agent $node_(2) $agent1  ;# Attach Agent to source node
$ns_ attach-agent $node_(1) $sink    ;# Attach Agent to sink node
$ns_ connect $agent1 $sink           ;# Connect the nodes
set app1 [new Application/Traffic/CBR] ;# Create Constant Bit Rate application
$app1 set packetSize_ 512           ;# Set Packet Size to 512 bytes
$app1 set rate_ 600Kb                ;# Set CBR rate to 200 Kbits/sec
$app1 attach-agent $agent1          ;# Attach Application to agent
```

```
set agent2 [new Agent/UDP]           ;# Create UDP Agent
#$agent2 set prio_ 1                 ;# Set Its priority to 1
set sink2 [new Agent/LossMonitor]    ;# Create Loss Monitor Sink in order to be able to trace the
number bytes received
$ns_ attach-agent $node_(2) $agent2  ;# Attach Agent to source node
$ns_ attach-agent $node_(3) $sink2   ;# Attach Agent to sink node
$ns_ connect $agent2 $sink2          ;# Connect the nodes
set app2 [new Application/Traffic/CBR] ;# Create Constant Bit Rate application
$app2 set packetSize_ 512           ;# Set Packet Size to 512 bytes
$app2 set rate_ 600Kb                ;# Set CBR rate to 200 Kbits/sec
$app2 attach-agent $agent2          ;# Attach Application to agent
```

```
set agent3 [new Agent/UDP]           ;# Create UDP Agent
#$agent3 set prio_ 2                 ;# Set Its priority to 2
set sink3 [new Agent/LossMonitor]    ;# Create Loss Monitor Sink in order to be able to trace the
number obytes received
$ns_ attach-agent $node_(2) $agent3  ;# Attach Agent to source node
$ns_ attach-agent $node_(5) $sink3   ;# Attach Agent to sink node
$ns_ connect $agent3 $sink3          ;# Connect the nodes
set app3 [new Application/Traffic/CBR] ;# Create Constant Bit Rate application
$app3 set packetSize_ 512           ;# Set Packet Size to 512 bytes
```




```

$app3 set rate_ 600Kb                ;# Set CBR rate to 200 Kbits/sec
$app3 attach-agent $agent3          ;# Attach Application to agent

set agent4 [new Agent/UDP]           ;# Create UDP Agent
#$agent4 set prio_ 3                 ;# Set Its priority to 3
set sink4 [new Agent/LossMonitor]    ;# Create Loss Monitor Sink in order to be able to trace the
number obytes received

$ns_ attach-agent $node_(2) $agent4  ;# Attach Agent to source node
$ns_ attach-agent $node_(7) $sink4   ;# Attach Agent to sink node
$ns_ connect $agent4 $sink4          ;# Connect the nodes
set app4 [new Application/Traffic/CBR] ;# Create Constant Bit Rate application
$app4 set packetSize_ 512            ;# Set Packet Size to 512 bytes
$app4 set rate_ 600Kb                ;# Set CBR rate to 200 Kbits/sec
$app4 attach-agent $agent4          ;# Attach Application to agent

```

#coding for cellular communication supporting one to one communication

```

#
=====
#
#           Options
#
=====
#

set opt(nn)      1      ;# number of mobilenodes
set num_wired_nodes 2

set opt(x)      870     ;# x coordinate of topology
set opt(y)      870     ;# y coordinate of topology

set opt(ftp1-start) 0.0
set opt(stop) 250      ;# time to stop simulation

set opt(tr-ns)   out.tr
set opt(tr-nam) out.nam

#
=====
#
#           MAC
#
=====
#

```

```
Mac/802_11 set dataRate_ 2.0e6
Mac/802_11 set RTSThreshold_ 3000
```

```
#
=====
=
# Nodes
#
=====
=
```

```
set ns_ [new Simulator]
$ns_ node-config -addressType hierarchical
```

```
AddrParams set domain_num_ 3 ;# number of domains
lappend cluster_num 2 1 1 ;# number of clusters in each domain
AddrParams set cluster_num_ $cluster_num
lappend eilastlevel 1 1 2 1 ;# number of nodes in each cluster
AddrParams set nodes_num_ $eilastlevel ;# of each domain
```

```
#
=====
=
# MOD
#
=====
=
```

```
set tracefd [open $opt(tr-ns) w]
set namtrace [open out.nam w]
$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $opt(x) $opt(y)
```

```
set topo [new Topography]
$topo load_flatgrid $opt(x) $opt(y)
```

```
create-god [expr $opt(nn) + 2]
# 2 for HA and FA
```

```
#-----
# wired nodes
#-----
```

```
set temp {0.0.0 0.1.0} ;# hierarchical addresses
for {set i 0} {$i < $num_wired_nodes} {incr i} {
    set W($i) [$ns_ node [lindex $temp $i]]
}
```

```

#-----
#          hybrid and wireless nodes
#-----

$ns_ node-config -mobileIP ON \
    -adhocRouting DSDV \
    -llType LL \
    -macType Mac/802_11 \
    -ifqType Queue/DropTail/PriQueue \
    -ifqLen 50 \
    -antType Antenna/OmniAntenna \
    -propType Propagation/TwoRayGround \
    -phyType Phy/WirelessPhy \
    -channelType Channel/WirelessChannel \
        -topoInstance $topo \
    -wiredRouting ON \
        -agentTrace ON \
    -routerTrace OFF \
    -macTrace ON

# Create Home Agent and Foreign Agent
set HA [$ns_ node 1.0.0]
set FA [$ns_ node 2.0.0]

$SHA set X_ 1.00
$SHA set Y_ 2.00
$SHA set Z_ 0.00

$FA set X_ 350.00
$FA set Y_ 300.00
$FA set Z_ 0.00

# create links between wired and BaseStation nodes
$ns_ duplex-link $W(0) $W(1) 5Mb 2ms DropTail
$ns_ duplex-link $W(1) $SHA 5Mb 2ms DropTail
$ns_ duplex-link $W(1) $FA 5Mb 2ms DropTail

$ns_ duplex-link-op $W(0) $W(1) orient down
$ns_ duplex-link-op $W(1) $SHA orient left-down
$ns_ duplex-link-op $W(1) $FA orient right-down

# create a mobilenode (in the domain of the HA)
# that is moving between HA and FA.
$ns_ node-config -wiredRouting OFF

set MH [$ns_ node 1.0.1]

```

```
set HAaddress [AddrParams addr2id [$SHA node-addr]]
[$MH set regagent_] set home_agent_ $HAaddress
```

```
$MH set Z_ 0.00
$MH set Y_ 2.00
$MH set X_ 2.00
```

```
# MH starts to move towards FA
$ns_ at 100.00 "$MH setdest 550.00 510.00 20.00"
# goes back to HA
$ns_ at 130.00 "$MH setdest 2.00 2.00 20.00"
```

```
# Define initial node position in nam
$ns_ initial_node_pos $MH 20
```

```
#####
#                               Agents
#####
```

```
set tcp1 [new Agent/TCP]
$tcp1 set class_ 2
set sink1 [new Agent/TCPSink]
$ns_ attach-agent $W(0) $tcp1
$ns_ attach-agent $MH $sink1
$ns_ connect $tcp1 $sink1
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ns_ at $opt(ftp1-start) "$ftp1 start"
```

```
#Create a UDP agent and attach it to node n0
set udp [new Agent/UDP]
$udp set packetSize_ 1500
```

```
# Create a CBR traffic source and attach it to udp0
set cbr [new Application/Traffic/CBR]
$cbcr set packetSize_ 1000
$cbcr set interval_ 0.1
$cbcr attach-agent $udp
```

```
#create an sink into the sink node
```

```
# Create the Null agent to sink traffic
set null [new Agent/Null]
$ns_ attach-agent $MH $null
```

```
# Attach the 2 agents
$ns_connect $udp $null
```

```
#####
#                               End
#####
```

```
$ns_ at $opt(stop).0 "$MH reset";
$ns_ at $opt(stop).0 "$HA reset";
$ns_ at $opt(stop).0 "$FA reset";
```

```
$ns_ at $opt(stop).0002 "puts \"NS EXITING...\" ; $ns_ halt"
$ns_ at $opt(stop).0001 "finish"
```

```
proc finish {} {
    global ns_ tracefd namtrace
    close $tracefd
    close $namtrace
    exec nam out.nam &
}
```

```
puts "Starting Simulation..."
$ns_run
```

#coding for cellular communication supporting one to many communication

```
#####
#                               Options
#####
```

```
set opt(nn)      3      ;# number of mobilenodes
set num_wired_nodes  2

set opt(x)      870     ;# x coordinate of topology
set opt(y)      870     ;# y coordinate of topology

set opt(ftp1-start) 0.0
```

```
set opt(stop) 120           ;# time to stop simulation
```

```
set opt(tr-ns) out.tr
set opt(tr-nam) out.nam
```

```
#
```

```
=
```

```
#           MAC
```

```
#
```

```
=
```

```
Mac/802_11 set dataRate_    2.0e6
Mac/802_11 set RTSThreshold_ 3000
```

```
#
```

```
=
```

```
#           Nodes
```

```
#
```

```
=
```

```
set ns_ [new Simulator]
$ns_ node-config -addressType hierarchical
```

```
AddrParams set domain_num_ 3           ;# number of domains
lappend cluster_num 2 1 1             ;# number of clusters in each domain
AddrParams set cluster_num_ $cluster_num
lappend eilastlevel 1 1 2 1          ;# number of nodes in each cluster
AddrParams set nodes_num_ $eilastlevel ;# of each domain
```

```
#
```

```
=
```

```
#           MOD
```

```
#
```

```
set tracefd [open $opt(tr-ns) w]
set namtrace [open out.nam w]
$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $opt(x) $opt(y)
```

```

set topo [new Topography]
$Stopo load_flatgrid $opt(x) $opt(y)

create-god [expr $opt(nn) + 2]
# 2 for HA and FA

#-----
#                wired nodes
#-----
set temp {0.0.0 0.1.0}      ;# hierarchical addresses
for {set i 0} {$i < $num_wired_nodes} {incr i} {
    set W($i) [$ns_node [lindex $temp $i]]
}
#-----
#                hybrid and wireless nodes
#-----
$ns_node-config -mobileIP ON \
    -adhocRouting DSDV \
    -llType LL \
    -macType Mac/802_11 \
    -ifqType Queue/DropTail/PriQueue \
    -ifqLen 50 \
    -antType Antenna/OmniAntenna \
    -propType Propagation/TwoRayGround \
    -phyType Phy/WirelessPhy \
    -channelType Channel/WirelessChannel \
    -topoInstance $topo \
    -wiredRouting ON \
    -agentTrace ON \
    -routerTrace OFF \
    -macTrace ON

# Create Home Agent and Foreign Agent
set HA [$ns_node 1.0.0]
set FA [$ns_node 2.0.0]

$SHA set X_ 1.00
$SHA set Y_ 2.00
$SHA set Z_ 0.00

$FA set X_ 350.00
$FA set Y_ 300.00
$FA set Z_ 0.00

# create links between wired and BaseStation nodes
$ns_duplex-link $W(0) $W(1) 5Mb 2ms DropTail

```

```

Sns_ duplex-link $W(1) $SHA 5Mb 2ms DropTail
Sns_ duplex-link $W(1) $FA 5Mb 2ms DropTail

Sns_ duplex-link-op $W(0) $W(1) orient down
Sns_ duplex-link-op $W(1) $SHA orient left-down
Sns_ duplex-link-op $W(1) $FA orient right-down

# create a mobilenode (in the domain of the HA)
# that is moving between HA and FA.
Sns_ node-config -wiredRouting OFF

set MH [$Sns_ node 1.0.1]
set HAaddress [AddrParams addr2id [$SHA node-addr]]
[$MH set regagent_] set home_agent_ $HAaddress

$MH set Z_ 0.00
$MH set Y_ 2.00
$MH set X_ 2.00

# MH starts to move towards FA
Sns_ at 30.00 "$MH setdest 500.00 470.00 40.00"
# goes back to HA
Sns_ at 70.00 "$MH setdest 2.00 2.00 40.00"

# Define initial node position in nam
  Sns_ initial_node_pos $MH 20

set MH1 [$Sns_ node 1.0.2]
set HAaddress [AddrParams addr2id [$SHA node-addr]]
[$MH1 set regagent_] set home_agent_ $HAaddress

$MH1 set Z_ 0.00
$MH1 set Y_ 50.00
$MH1 set X_ 2.00

# MH starts to move towards FA
Sns_ at 40.00 "$MH1 setdest 500.00 260.00 30.00"
# goes back to HA
Sns_ at 90.00 "$MH1 setdest 20.00 20.00 30.00"

# Define initial node position in nam
  Sns_ initial_node_pos $MH1 20

set MH2 [$Sns_ node 1.0.3]
set HAaddress [AddrParams addr2id [$SHA node-addr]]

```



```
[$MH2 set regagent_] set home_agent_ $SHAaddress
```

```
$MH2 set Z_ 0.00
$MH2 set Y_ 449.00
$MH2 set X_ 440.00
```

```
# MH starts to move towards FA
$ns_ at 50.00 "$MH2 setdest 25.00 25.00 30.00"
# goes back to HA
$ns_ at 95.00 "$MH2 setdest 478.00 400.00 30.00"
```

```
# Define initial node position in nam
```

```
$ns_ initial_node_pos $MH2 20
```

```
#####
#
# Agents
#####
```

```
set tcp1 [new Agent/TCP]
$tcp1 set class_ 2
set sink1 [new Agent/TCPSink]
$ns_ attach-agent $W(0) $tcp1
$ns_ attach-agent $MH $sink1
$ns_ connect $tcp1 $sink1
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ns_ at $opt(ftp1-start) "$ftp1 start"
```

```
set tcp1 [new Agent/TCP]
$tcp1 set class_ 2
set sink1 [new Agent/TCPSink]
$ns_ attach-agent $W(0) $tcp1
$ns_ attach-agent $MH1 $sink1
$ns_ connect $tcp1 $sink1
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ns_ at $opt(ftp1-start) "$ftp1 start"
```

```
set tcp1 [new Agent/TCP]
$tcp1 set class_ 2
set sink1 [new Agent/TCPSink]
$ns_ attach-agent $W(0) $tcp1
$ns_ attach-agent $MH2 $sink1
$ns_ connect $tcp1 $sink1
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ns_ at $opt(ftp1-start) "$ftp1 start"
```

```

#Create a UDP agent and attach it to node n0
set udp [new Agent/UDP]
$udp set packetSize_ 1500

# Create a CBR traffic source and attach it to udp0
set cbr [new Application/Traffic/CBR]
$cbcr set packetSize_ 1000
$cbcr set interval_ 0.1
$cbcr attach-agent $udp

#create an sink into the sink node

# Create the Null agent to sink traffic
set null [new Agent/Null]
$ns_ attach-agent $MH $null

set null [new Agent/Null]
$ns_ attach-agent $MH1 $null

set null [new Agent/Null]
$ns_ attach-agent $MH2 $null

# Attach the 2 agents
$ns_ connect $udp $null
#=====
#                               End
#=====
$ns_ at $opt(stop).0 "$MH reset";
$ns_ at $opt(stop).0 "$SHA reset";
$ns_ at $opt(stop).0 "$FA reset";

$ns_ at $opt(stop).0002 "puts \"NS EXITING...\" ; $ns_ halt"
$ns_ at $opt(stop).0001 "finish"

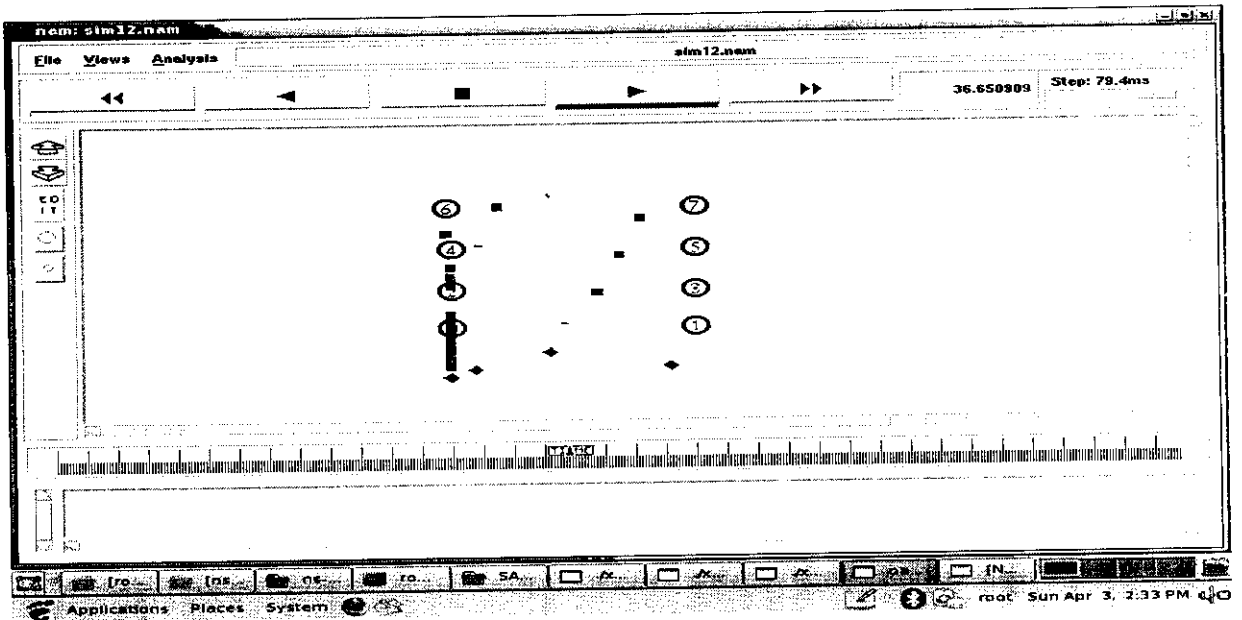
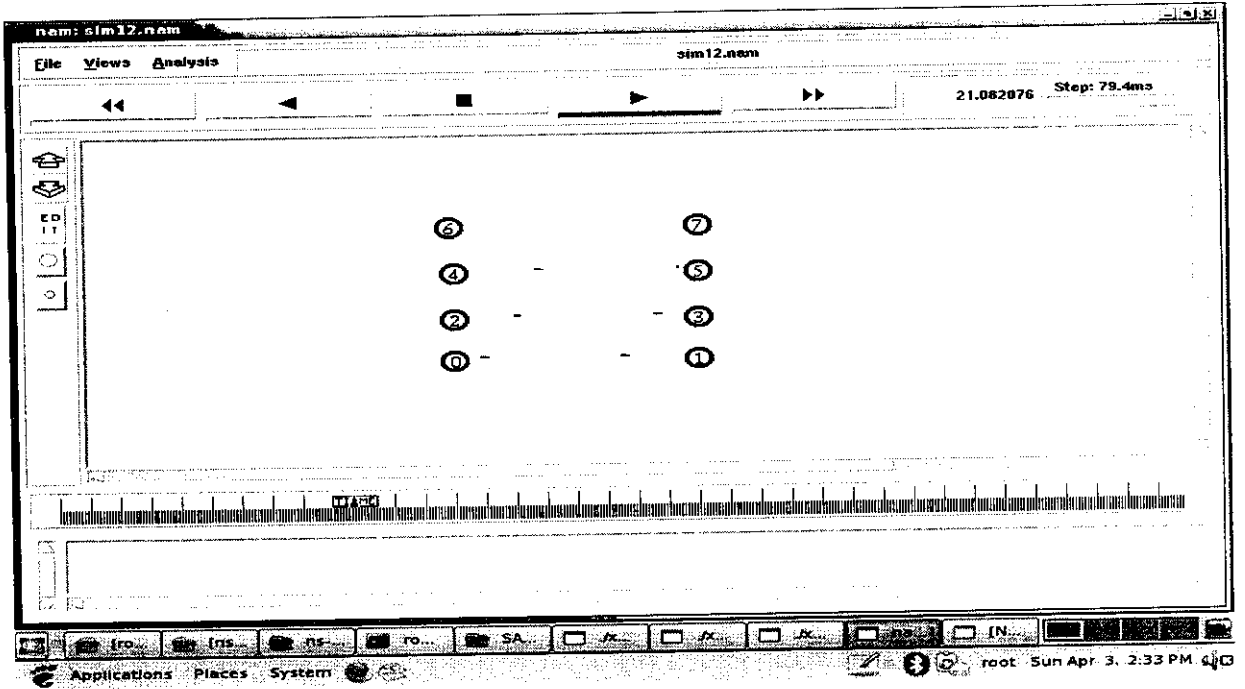
proc finish {} {
    global ns_ tracefd namtrace
    close $tracefd
    close $namtrace
    exec ./nam out.nam &
}

puts "Starting Simulation..."
$ns_ run

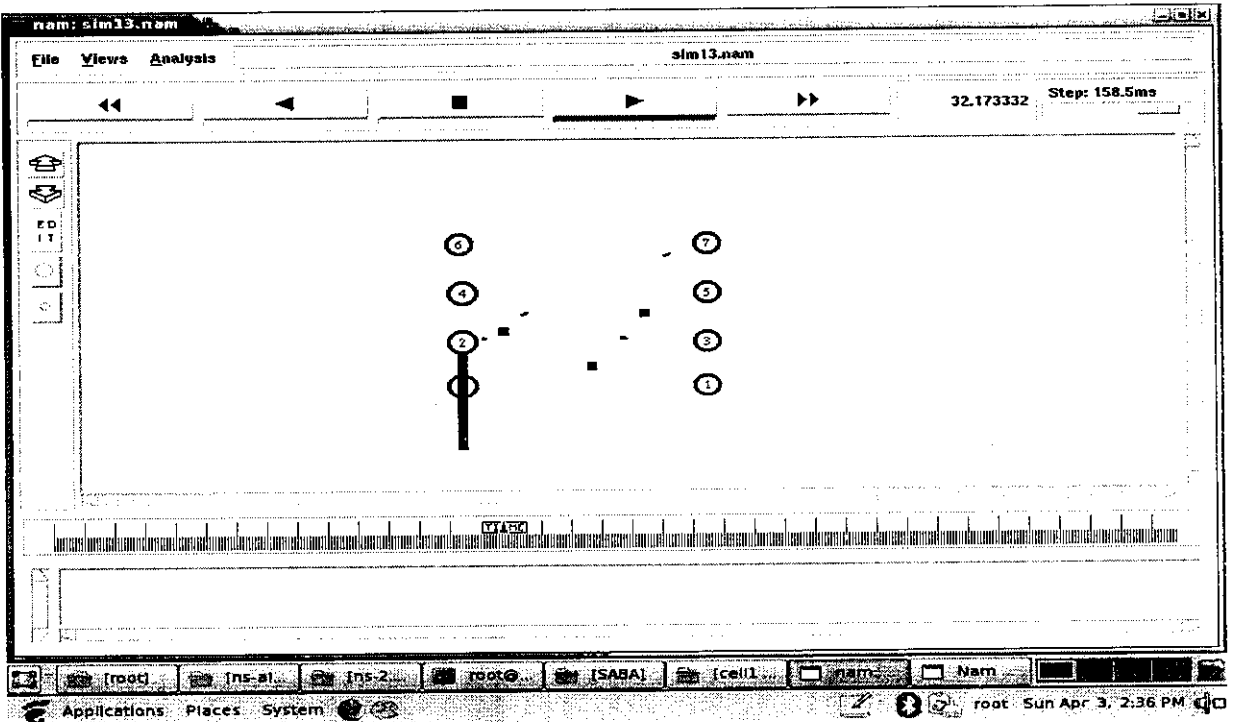
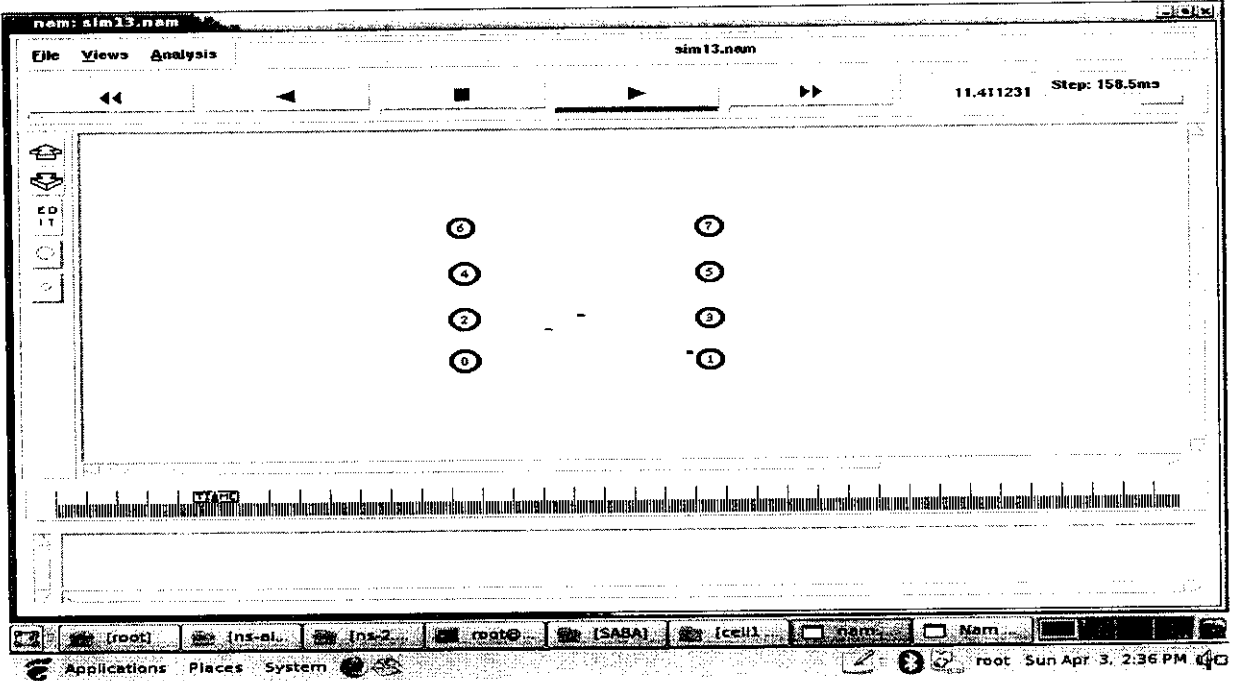
```

7.2 screen shots

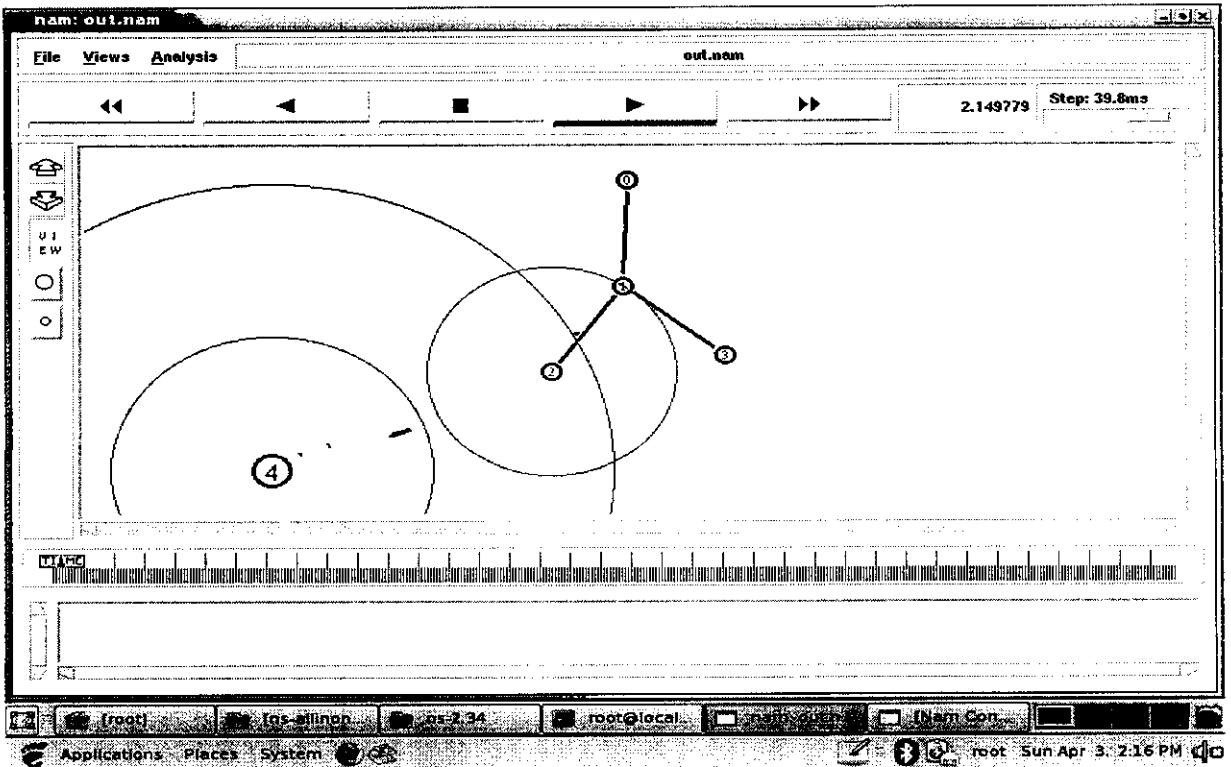
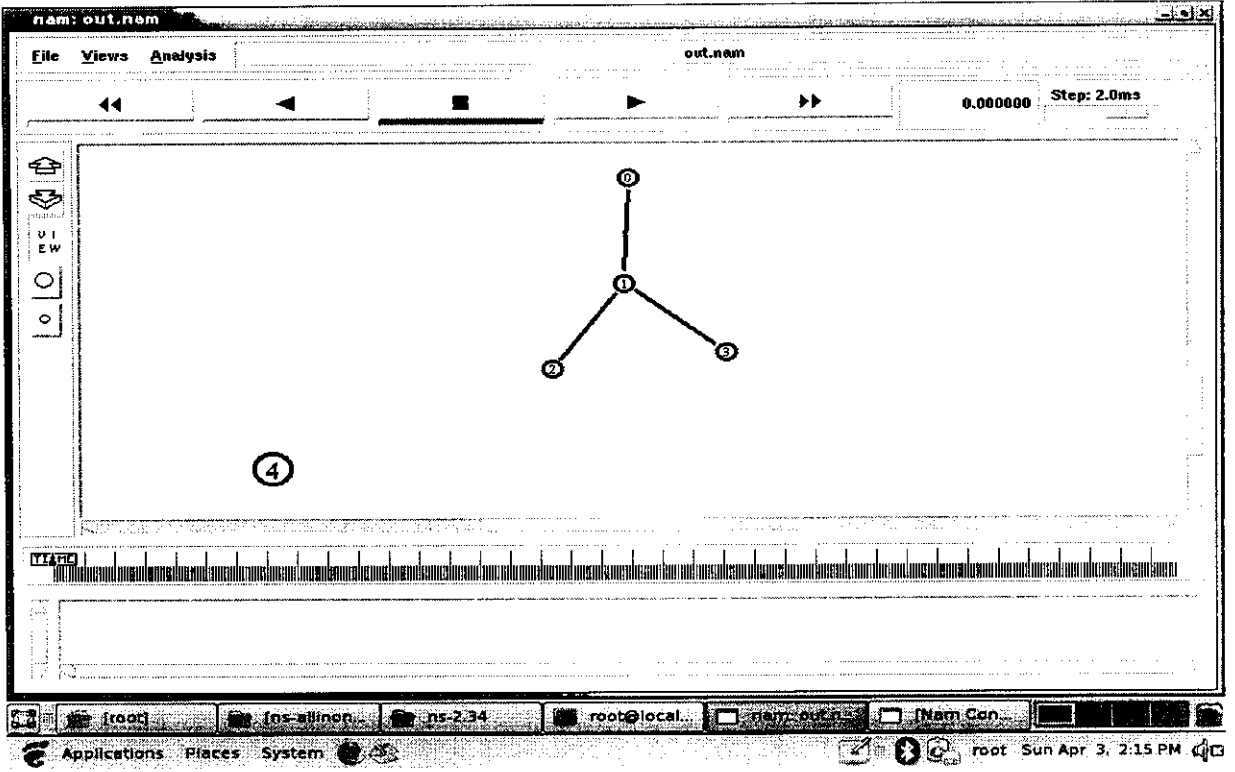
#Screen shots for WLAN Communication supporting one to one communication

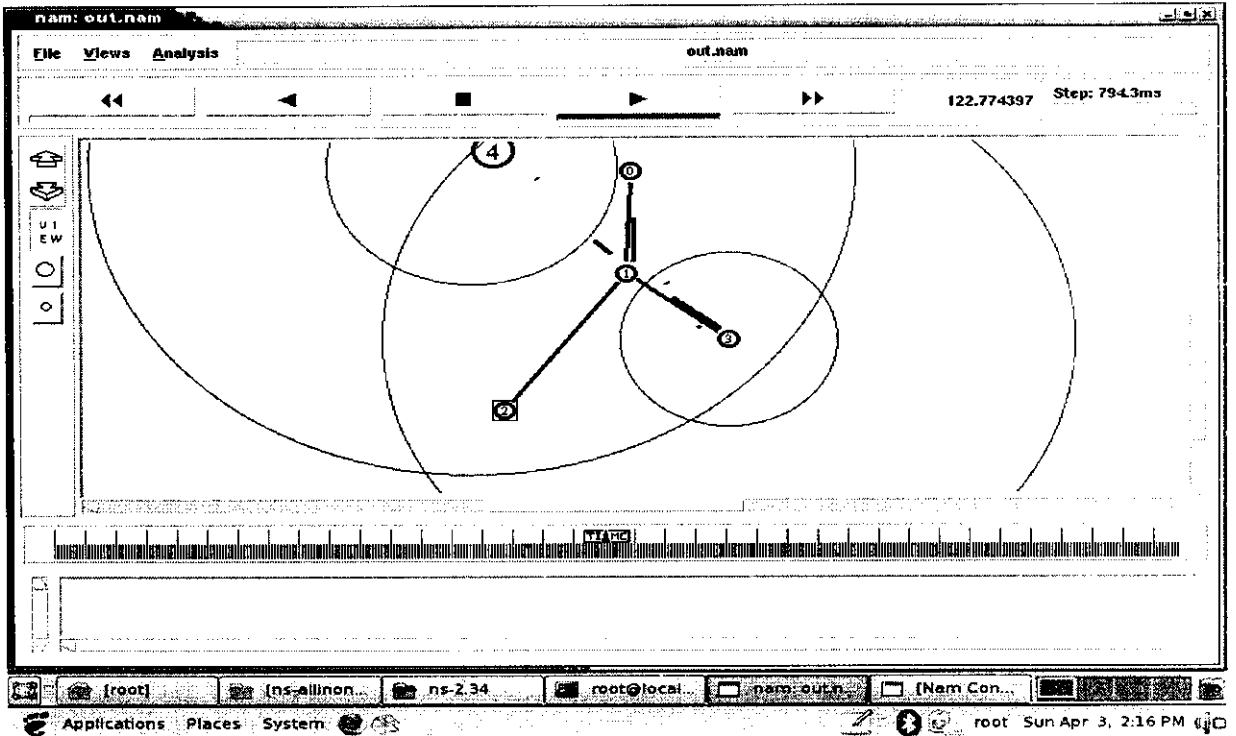


#Screen shots for WLAN Communication supporting one to many communication

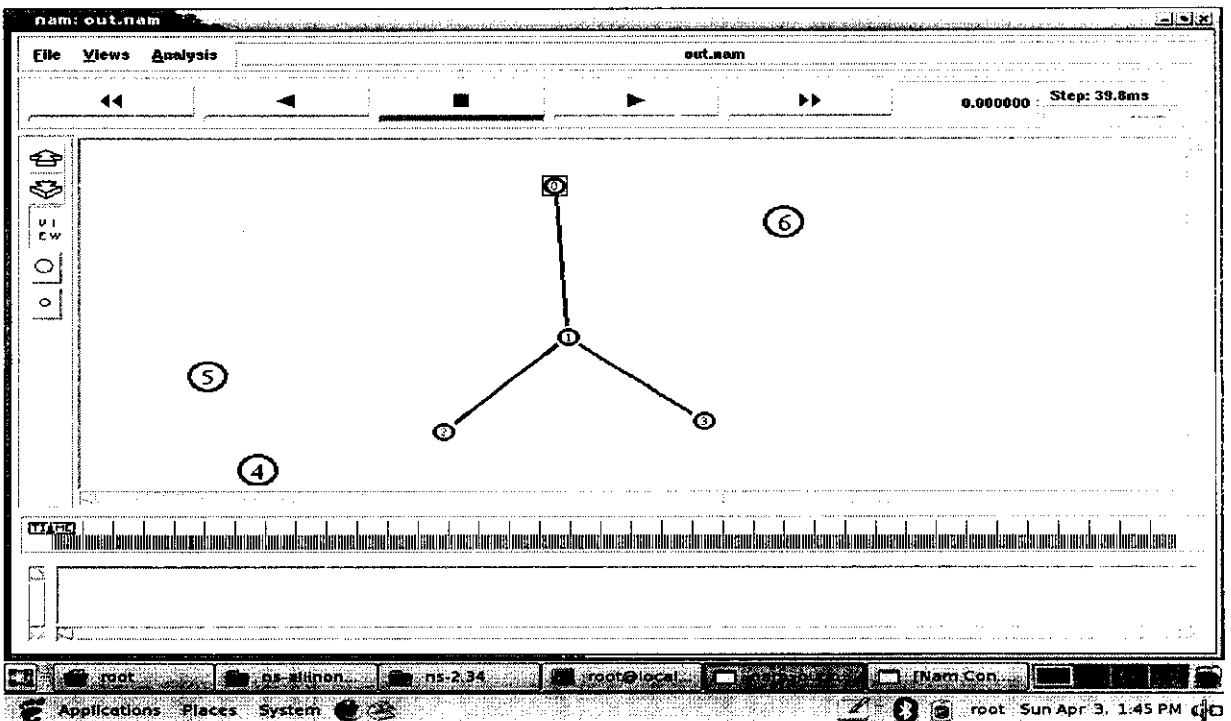


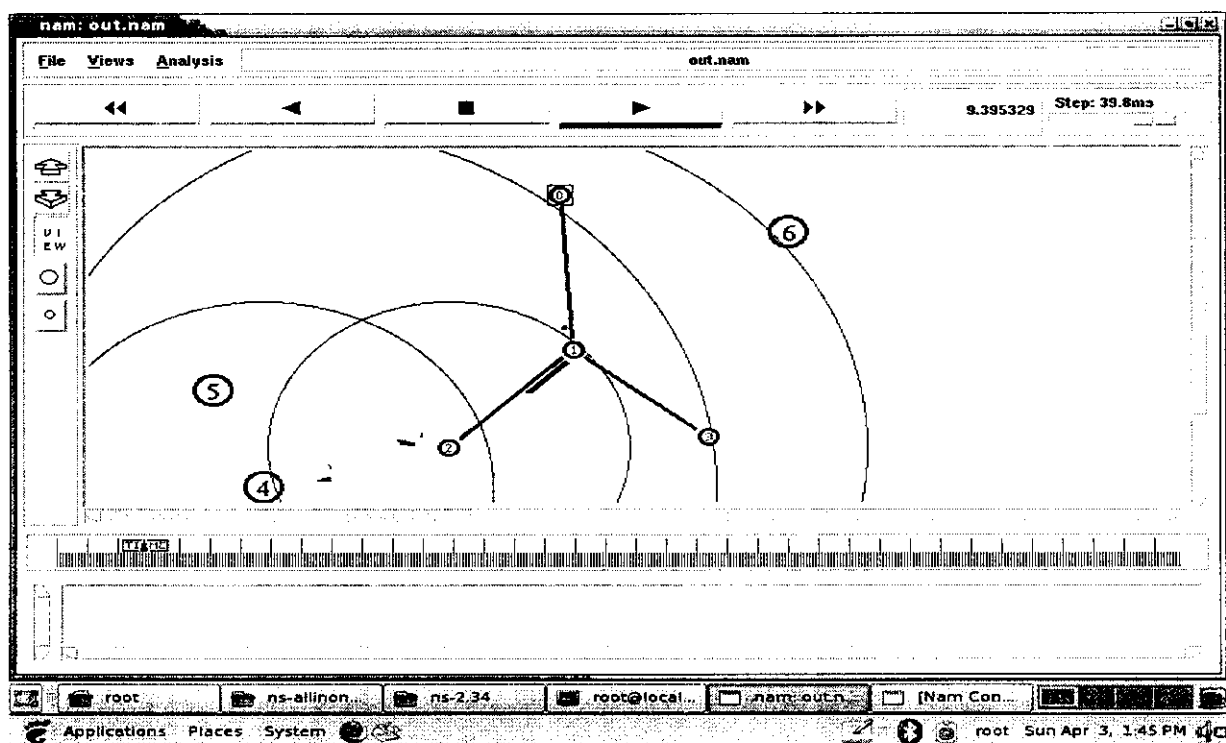
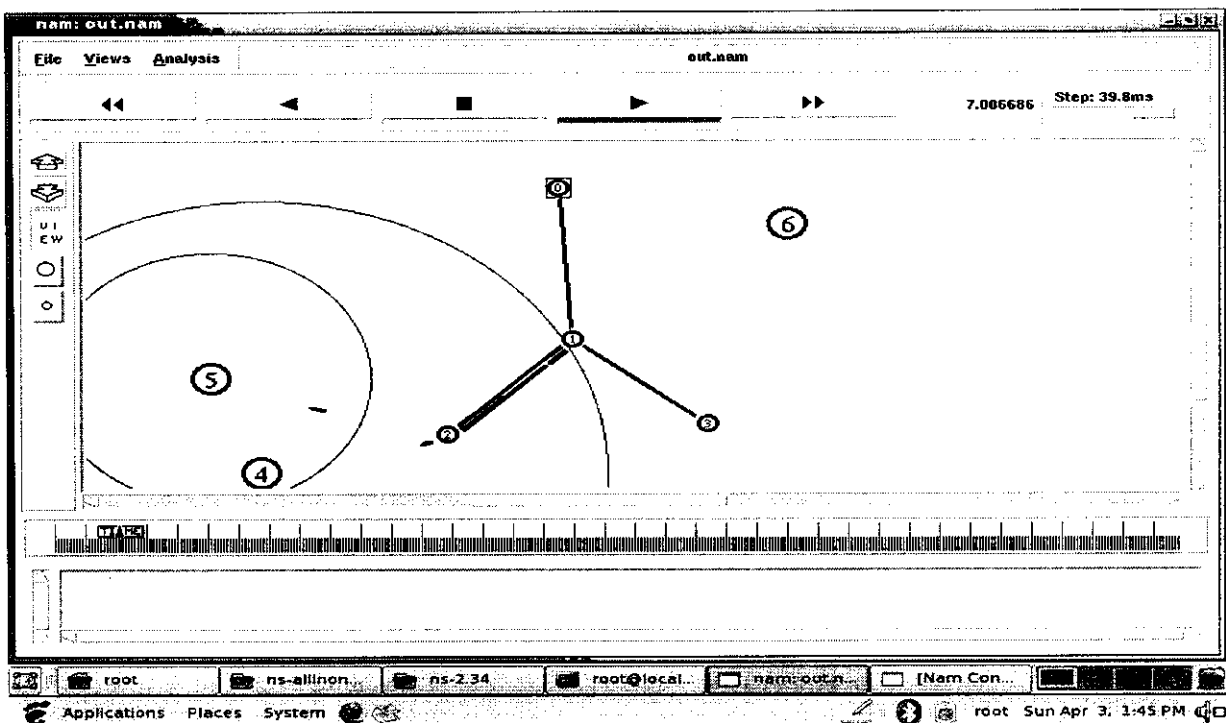
#Screen shots for Cellular Communication supporting one to one communication

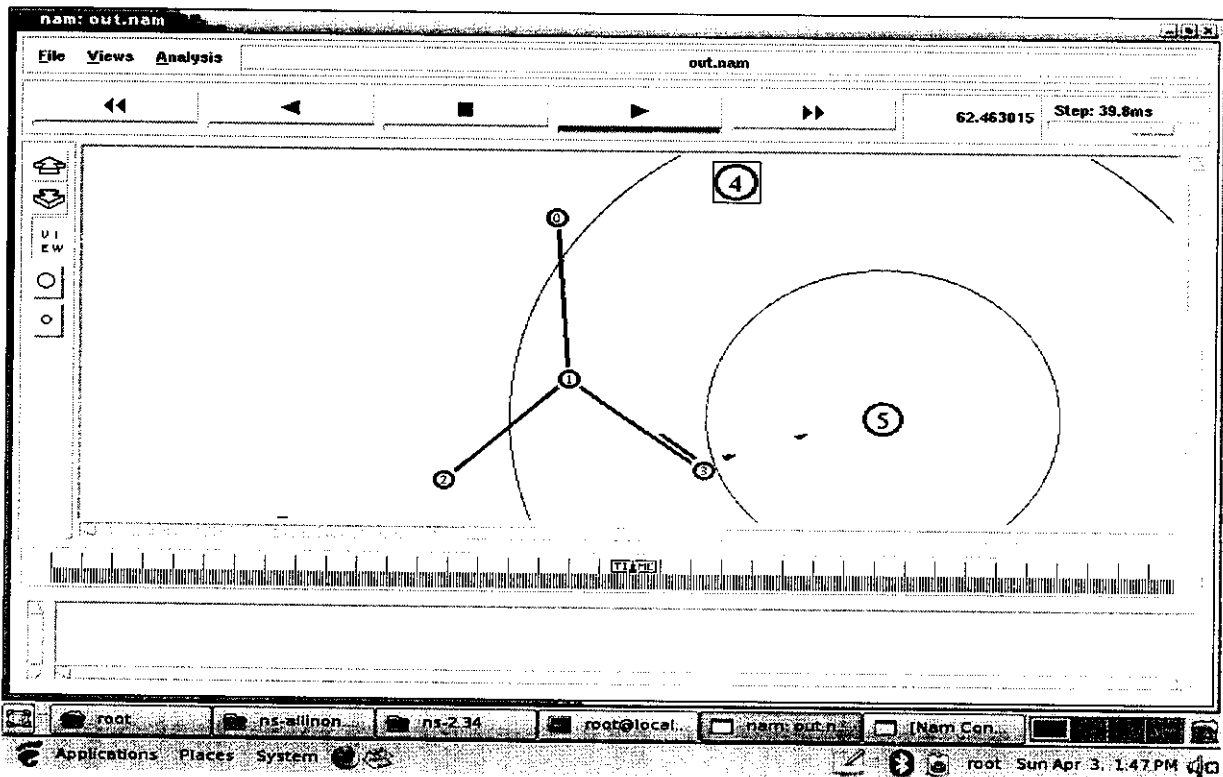
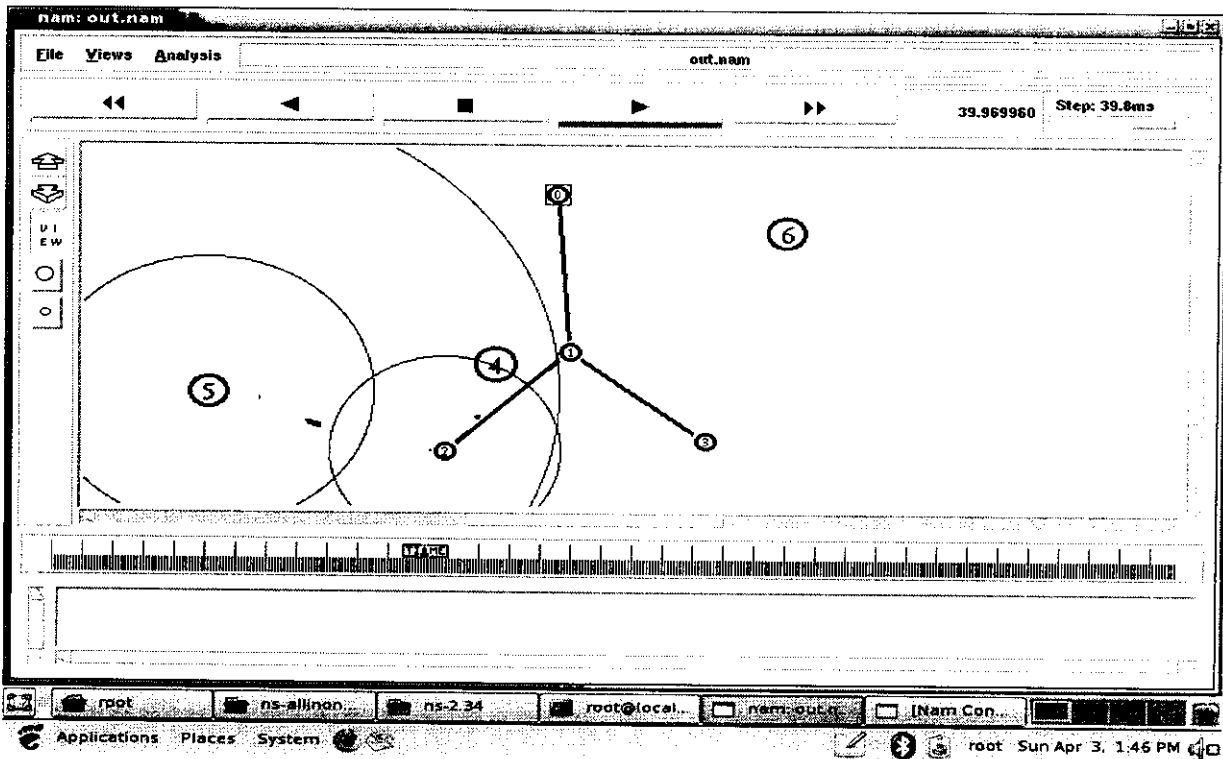




#Screen shots for Cellular Communication supporting one to one communication







REFERENCES

1. Andrei Gurtov and Sally Floyd, "Modeling wireless links for transport protocols". In *Proceedings of IEEE Symposium ASSET'08*, March 2008.
2. Yongho Seok, Youngsam Park and Yanghee Choi, "Vertical and Horizontal Flow Controls for TCP Optimization in the mobile Ad Hoc Networks", *IEEE INFOCOM*, June 2007.
3. H. Balakrishnan, V. R. H.Katz, N.Padmanabha and S.Seshan "A comparison of mechanisms for improving TCP performance over wireless links", *IEEE/ACM Transaction on Networking*, 5(6):756–769, Dec. 2006.
4. A.V.Bakre and B.R.Badrinath, "Implementation and Performance Evaluation of Indirect TCP", *IEEE TRANSACTIONS ON COMPUTERS*, March 2005.
5. Geoff Huston and Telstra, "TCP in a Wireless World", *IEEE INTERNET COMPUTING*, April 2006.
6. Yavatkar R. and Bhagawat N, "Improving end-to-end performance of TCP over mobile internetworks". In *Proceedings of IEEE*, 2005.
7. Hala ElAarag and Mostafa Bassiouni, "Simulation of Transport Protocols Over Wireless Communication Networks". In *Proceeding of the Winter Simulation Conference, IEEE*, Aug. 2002.
8. Charles E. Perkins, "The Effects of Mobility on Reliable Transport protocols", *IEEE*, March 2006.

9. G. Xylomenos, "Multi Service Link Layers: An Approach to Enhancing Internet Performance over Wireless Links", PhD thesis, University of California at San Diego, 2006.
10. A. Chockslingam, M. Zai and V. T, "Wireless TCP performance with link layer FEC/ARQ". In *Proceedings of IEEE*, pp.1212-1216, 2007.
11. T. Henderson and R. Katz, "Transport protocols for Internet-compatible satellite networks", *IEEE Journal on Selected Areas in Communications*, 17(2):345-359, Feb.2006.
12. Len An and Ning Jia, " Analysis of congestion Control Strategy for Wireless Network", *IEEE publications*, November 2008.