PERFORMANCE EVALUATION OF ROUTING PROTOCOLS IN WIRELESS AD HOC NETWORKS WITH SERVICE DIFFERENTIATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

SEMRA YILMAZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

JANUARY 2003

Approval of the Graduate School of Informatics

_____
Prof. Dr. Neşe Yalabık
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____
Prof. Dr. Semih Bilgen
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____	_____
Assist. Prof. Dr. Murat Erten	Dr. Altan Koçyiğit
Co-Supervisor	Supervisor

Examining Committee Members

Prof. Dr.	Semih Bilgen	_____
Assist. Prof. Dr.	Cüneyt Bazlamaçcı	_____
Assist. Prof. Dr.	Erkan Mumcuoğlu	_____
Assist. Prof. Dr.	Murat Erten	_____
Dr.	Altan Koçyiğit	_____

ABSTRACT

Performance Evaluation of Routing Protocols in Wireless Ad Hoc Networks with Service Differentiation

Yılmaz, Semra
MS, Department of Information Systems
Supervisor: Dr. Altan Koçyiğit
Co-Supervisor: Assist. Prof. Dr. Murat Erten

January 2003, 100 Pages

An ad hoc network is a collection of wireless mobile nodes dynamically forming a temporary network without the use of any fixed network infrastructure or centralized administration. Due to the limitations in the wireless environment, it may be necessary for one mobile host to enlist the aid of other hosts in forwarding a packet to its destination. In order to enable communication within the network, a routing protocol is needed to discover routes between nodes. The primary goal of ad hoc network routing protocols is to establish routes between node pairs so that messages may be delivered reliably and in a timely manner.

The basic access method in IEEE 802.11 ad hoc networks is the Distributed Coordination Function (DCF), which provides a fair medium access. Enhanced Distributed Coordination Function (EDCF) has been developed to provide service differentiation among different traffic flows.

In this thesis, we investigate the performance of the EDCF with routing protocols; Direct Sequenced Distance Vector (DSDV) and Dynamic Source Routing (DSR) by simulations.

Keywords: Ad Hoc Networks, Quality of Service for Wireless Networks, Routing Protocols in Ad Hoc Networks, Ad Hoc Network Performance Evaluation.

ÖZ

Servis Ayrımı Yapabilen Özel Amaca Yönelik Ağlarda Yönlendirme Protokollerinin Başarımının Değerlendirilmesi

Yılmaz, Semra
Yüksek Lisans, Bilişim Sistemleri Bölümü
Tez Yöneticisi: Dr. Altan Koçyiğit
Yardımcı Tez Yöneticisi: Yard. Doç. Dr. Murat Erten

Ocak 2003, 100 Sayfa

Özel amaca yönelik ağlar; telsiz, hareketli terminallerden oluşan ve varolan herhangi bir alt yapıyı kullanmadan dinamik olarak geçici ağlar oluşturabilen yapıdır. Telsiz ortamın kısıtlamalarından dolayı, hareketli bir terminal, hedefe göndereceği paketleri aktarabilmek için diğer terminallerin yardımına gereksinim duyabilmektedir. Ağ içinde haberleşmeye imkan verebilmek için ve terminaller arasındaki haberleşme yolunu kurabilmek için yönlendirme protokolleri kullanılmaktadır. Özel amaca yönelik ağlarda yönlendirme protokollerinin asıl amacı, verinin uygun bir şekilde gönderilebilmesi için terminaller arasında doğru ve verimli bir yol kurulmasıdır.

Özel amaca yönelik ağlar için, IEEE 802.11 MAC seviye protokolünde kullanılan temel erişim metodu DCF'tir. DCF, terminallerin ortama eşit şartlarda erişimine imkan vermektedir. Servis ayrımı sağlamak için EDCF metodu kullanılabilmektedir.

Bu tezde, DSDV ve DSR yönlendirme protokolleri için EDCF başarımını değerlendirdik. Başarımı incelemek için çok sayıda simülasyon denemesi yapılmıştır.

Anahtar Kelimeler: Özel Amaca Yönelik Ağlar, Telsiz ağları için Servis Kalitesi, Özel Amaca Yönelik Ağlar için Yönlendirme Protokolleri, Özel Amaca Yönelik Ağlarda Başarım Değerlendirmesi

ACKNOWLEDGEMENTS

I would like to thank my supervisors Dr. Altan Koçyiğit and Assist. Prof. Dr. Murat Erten for their great supervision, guidance, encouragement, patience for the development of this thesis and insights throughout the study.

I would also like to thank to committee members for reading the thesis and for their constructive comments.

I offer sincere appreciation to my family and my friends Ahmet Taştekin and Fatih Türkoğlu for their encouragement and to all my friends for their continuous morale support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS AND ACRONYMS

2G	:	Second Generation
3G	:	Third Generation
ACK	:	Acknowledgement
AEDCF	:	Adaptive Enhanced Distributed Coordination Function
AIFS	:	Arbitration Inter Frame Space
AODV	:	Ad Hoc On-Demand Distance Vector Routing
AP	:	Access Point
ARP	:	Address Resolution Protocol
BrID	:	Broadcast Identification
BSA	:	Basic Service Area
BSS	:	Basic Service Set
CBQ	:	Class Based Queuing
CBR	:	Constant Bit Rate
CFP	:	Contention Free Period
CP	:	Contention Period
CPU	:	Central Processing Unit
CSMA/CA	:	Carrier Sense Multiple Access with Collision Avoidance
CSMA/CD	:	Carrier Sense Multiple Access with Collision Detection
CTS	:	Clear To Send
CW	:	Contention Window
DCF	:	Distributed Coordination Function
DiffServ	:	Differentiated Services
DIFS	:	Distributed Coordination Function Inter Frame Space
DS	:	Distribution System

DSDV	:	Destination Sequenced Distance Vector Routing
DSR	:	Dynamic Source Routing
DSSS	:	Direct Sequence Spread Spectrum
EDCF	:	Enhanced Distributed Coordination Function
FHSS	:	Frequency Hopping Spread Spectrum
FQ	:	Fair Queuing
IBSS	:	Independent Basic Service Set
IEEE	:	Institute of Electrical and Electronics Engineering
IETF	:	Internet Engineering Task Force
IFS	:	Inter Frame Space
IntServ	:	Integrated Services
IP	:	Internet Protocol
IR	:	Infrared
LAN	:	Local Area Network
MAC	:	Medium Access Control
MAN	:	Metropolitan Area Network
Mbps	:	Mega Bit Per Second
MPDU	:	MAC Protocol Data Unit
NAM	:	Network Animator
NAV	:	Network Allocation Vector
NS	:	Network Simulator
ODMRP	:	On-Demand Multicast Routing Protocol
OSI	:	Open Systems Interconnection
OTCL	:	Object Tool Command Language
PAN	:	Personal Area Network
PC	:	Point Coordinator
PCF	:	Point Coordination Function
PF	:	Persistence Factor
PIFS	:	Point Coordination Function Inter Frame Space
QoS	:	Quality of Service
RED	:	Random Early Detection

RREP	:	Route Reply
RREQ	:	Route Request
RTS	:	Request To Send
Rx	:	Receive
SFQ	:	Stochastic Fair Queuing
SIFS	:	Short Inter Frame Space
TCL	:	Tool Command Language
TCP	:	Transmission Control Protocol
TORA	:	Temporarily Ordered Routing Algorithm
Tx	:	Transmit
WAN	:	Wide Area Network
WLAN	:	Wireless Local Area Network
WRR	:	Weighted Round Robin
WT	:	Wireless Terminal

CHAPTER 1

INTRODUCTION

Wireless and mobile communication networks need an architecture that provides Quality of Service (QoS) support along with fast and scalable routing techniques. This is partly due to the proliferation of real-time applications such as voice and video. A big drop in service quality when a call is made as the mobile moves from one region to another may not be acceptable for such applications. It is therefore required to maintain the QoS for these applications, in the presence of user mobility with the use of resource reservation and with fast re-routing techniques.

There are two major concerns related to the communication within wireless and mobile networks. The first issue is QoS support in mobile networks. QoS guarantees in mobile networks may require some form of resource reservation as the mobile moves between regions. Resource reservation in a mobile environment is a challenge since reservations have to be maintained wherever the mobile goes. The second issue is routing in light of mobility.

QoS may be achieved by handling flow priority with preemption, or resource allocation for multiple service classes. If there are critical traffic flows that must be accorded higher priority than other types of flows, a mechanism must be implemented in the network to recognize flow priorities. For a given network load, a high priority flow should be more likely to get a certain QoS from the network than a lower priority flow. This mechanism should be studied for different routing procedures, because behavior of routing hosts takes an important role to improve

the QoS in the network, so routing procedures for flow prioritization can be complex.

In this thesis, we have compared these two issues; medium access and routing protocols. We have investigated the performances of Distributed Coordination Function (DCF) and Enhanced Distributed Coordination Function (EDCF) that correspond to the fair and unfair allocation of the channel to different traffic flows, respectively. EDCF is a QoS mechanism that provides more resources to high priority defined flows. Direct Sequenced Distance Vector (DSDV) and Dynamic Source Routing (DSR) routing protocols are also compared with and without priority assigned to the flows in the network.

1.1. Organization of the Thesis

Following the Introduction, wireless networking is reviewed in the second chapter. Challenges in wireless networks and types of networks are introduced. Importance of ad hoc networking and related issues are also explained.

In the third chapter, IEEE 802.11 standard for wireless local area networks and two types (Distributed Coordination Function and Point Coordination Function) of Medium Access Control (MAC) protocols defined in the standard are presented. QoS mechanism based on Enhanced Distributed Coordination Function is explained.

In the fourth chapter routing in ad hoc networks is investigated and various routing protocols (DSDV, DSR, AODV, and TORA) are explained.

In the fifth chapter, performances of DCF and EDCF together with routing protocols DSDV and DSR are presented and compared with each other.

In Appendix A, the network simulator and network animator tools used to evaluate performance are briefly introduced. In Appendix B, the Tool Command Language (Tcl) scripts used for our simulations are given.

CHAPTER 2

WIRELESS NETWORKING

Wireless networking is an emerging technology providing users with network connectivity without the use of any wired network infrastructure [21]. Wireless networking is being developed to provide high bandwidth to users. Physical and environmental necessities also make wireless networks more popular; for example new buildings might be planned with network connectivity in mind, however; users cannot access the wired networks while they are out of the building. For the networks, that are temporary and operational for a short time, deployment of wired networks may be an impractical solution.

The goals of wireless networks are;

Mobility, provide users with access to information from anywhere.

Installation speed and simplicity, do not need to pull cable through walls and ceilings.

Installation flexibility, allows network to go where wire cannot go.

2.1. Issues in Wireless Networks

Having gained the freedom to move within an area, users expect the same services and capabilities that they have with wired networks. However, to meet these objectives, the wireless community faces constraints [28] that are not imposed on the wired networks:

Bandwidth is scarce

Wireless communication makes use of radio (or infrared) spectrum, and while the number and type of applications are increasing rapidly the available bandwidth is severely constrained. As bandwidth is scarce, it is very expensive in line with the laws of supply and demand. The cost of wired access is falling much faster than the cost of wireless access resulting in increasing cost gap between wired and wireless communications that is currently growing, and is likely to continue to do so.

Complete coverage is hard

Signals fade rapidly through the medium (with the square of the distance from the transmitter at best); this means that many intermediate stations are needed. Also, higher the frequency faster the signals decay, and more base stations are needed increasing the cost. To complicate matters further, higher frequency used makes greater bandwidth available, but causes faster signal decays, and these need to be balanced when designing a network.

The environment is hostile

Copper wire and fiber provide an excellent propagation medium. The air through which wireless signals must travel is a very poor environment due to obstacles, humidity etc. This means that complex error correction schemes are needed which are not only expensive to design and build but also require additional bandwidth to work.

Power consumption in mobile devices

Mobile devices are usually battery powered and this brings some problems. The greater the distance being communicated over, the greater the signal power required and hence the size and weight of battery needed to make it work increases. That places a major constraint on both the signal strength and the processing used to encode and decode signals.

Security

In a wired medium, the transmission medium can be physically secured and access to the network can be easily controlled. A wireless network is more difficult to secure, since the transmission medium is open to any one within the range of a transmitter.

Human Safety

Research is ongoing to determine whether electromagnetic transmissions are linked to human illnesses. Networks should be designed to minimize the power transmitted by network devices. For infrared (IR) system, optical transmitter must be designed to prevent vision impairment.

2.2. Types of Network

Different types of wireless interconnection have been developed. According to coverage area, they can roughly be divided into personal area networks (PAN), local area networks (LAN), metropolitan area networks (MAN) and wide area networks (WAN).

Figure 2-1 illustrates the relative coverage and data rates for 2G and 3G cellular networks, WLAN, and WPANs.

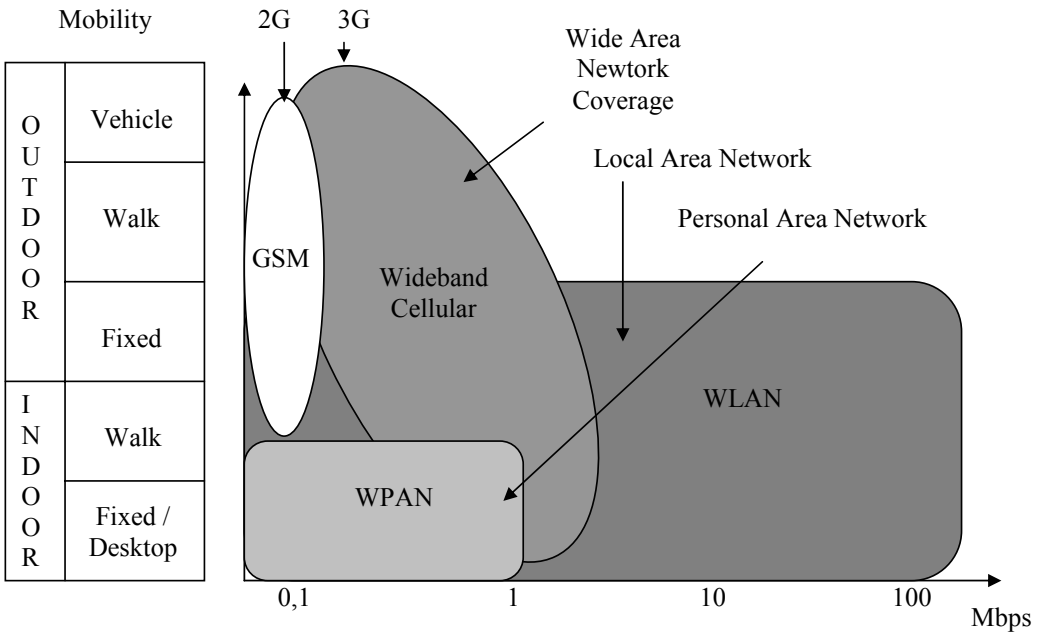


Figure 2-1 Relative Mobility, Coverage and Data Rates of Generations of Cellular System and Local Broadband and Ad Hoc Networks [21]

Personal Area Network

Personal Area Network (PAN) is recently based on a global specification called Bluetooth which uses radio frequency to transmit voice and data over a short range. PAN is a short range wireless network that allows the connectivity among intelligent devices. The Standards Board of the IEEE approved the standard 802.15, as MAC and PHY Specifications for Wireless PANs.

Local Area Network

Devices communicating with each other within short distances form a local area network. Coverage area can be a building, or campus of up to a few hundred meters.

In the conventional wired LAN, a network of computers and peripherals are connected to each other by means of optical fiber or copper cables. Wireless LANs also contain such devices, computers and peripherals, however they communicate through a wireless medium. Although it is possible to construct a pure wireless

LANs in which every component of the LAN is connected by a wireless link; in most environments, wireless LANs are used to complement wired LANs.

Metropolitan Area Network

A Metropolitan Area Network (MAN) connects an area larger than a LAN but smaller than a WAN, such as a city, with dedicated hardware.

Wide Area Network

Wide Area Network (WAN) is a collection of LANs. A network device connects the LANs to a WAN.

The most familiar form of wireless WAN is a mobile cellular network. Millions of people around the world use mobile phones to communicate with each other and to access Public Switched Telephone Network.

2.3. IEEE 802.11 Wireless LANs

IEEE 802.11 is the standard for Wireless Local Area Networks (WLANs) developed by the Institute of Electrical and Electronics Engineers (IEEE). The IEEE 802.11 standard defines two types of networks; ad hoc and infrastructure.

The Basic Service Set (BSS) is the fundamental building block of the IEEE 802.11 architecture. A BSS is defined as a group of stations that are using the same medium access protocol and the same frequency band. The geographical area covered by the BSS is known as the Basic Service Area (BSA), which is analogous to a cell in cellular wireless networks. Conceptually, all stations in a BSS can communicate directly with all other stations in the same BSS.

An *ad hoc network* is a deliberate grouping of stations to form a single BSS for the purpose of internetworked communications without the aid of a base station. An Independent Basic Service Set (IBSS) is the formal name of an ad hoc network in the IEEE 802.11 standard. Any station can establish a direct communications

session with any other station in the BSS, without the requirement of channeling all traffic through a centralized Access Point (AP), Figure 2-2.

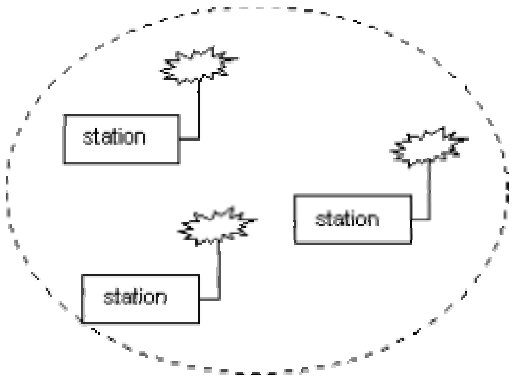


Figure 2-2 Independent Basic Service Set (IBSS)

The major challenge imposed by the ad hoc architecture, is the mobility of network nodes resulting in unpredictable and continuous change in the network topology. In addition to these, because of the limited radio propagation range, routes are mostly multiple hops. These characteristics in combination with bandwidth, energy and memory constraints make ad hoc networking research interesting and challenging.

In contrast to the ad hoc networks, *infrastructure networks* are established to provide some specific services and range extension. Infrastructure networks in the context of IEEE 802.11 are established using Access Points (APs). Wireless infrastructure can be built by connecting many AP to their existing LANs. This can provide wireless coverage for an entire building or campus, each cell created and served by an AP. An AP can also be used to provide a local relay function for the BSS. All stations in the BSS communicate with the access point. All frames are relayed between stations by the access point. This local relay function effectively doubles the range of the IBSS.

AP also provides connection to a distribution system. So, as the terminal moves, it can be handed-off from the current AP to the new one, Figure 2-3.

The distribution system (DS) is the means by which an access point communicates with another access point to exchange frames for stations in their respective BSSs, forward frames to follow mobile stations as they move from one BSS to another, and exchange frames with a wired network.

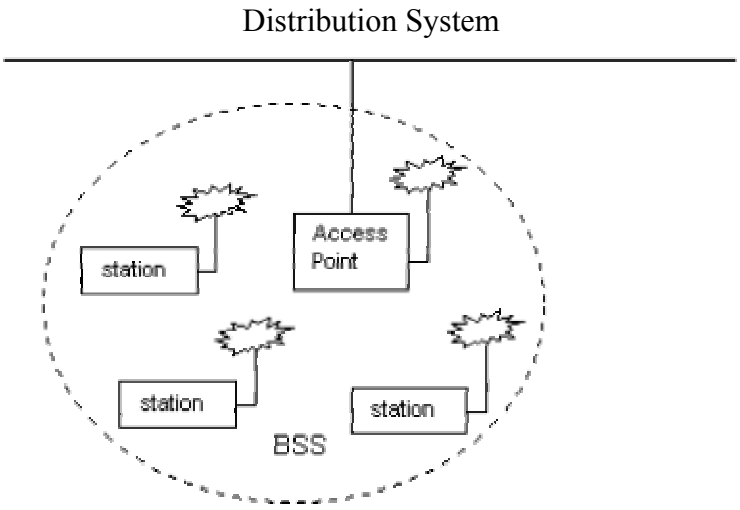


Figure 2-3 Infrastructure Basic Service Set

2.4. Importance of Ad Hoc Networks

A mobile ad hoc network is an autonomous system of mobile routers (nodes) connected by wireless links. The routers are free to move randomly and organize themselves arbitrarily; thus wireless network topology may change rapidly and unpredictably.

The ad hoc architecture has many benefits, such as self-reconfiguration and adaptability to highly variable mobile characteristics. However, such benefits bring new challenges which mainly reside in the unpredictability of network topology due to mobility of nodes, which cause a set of concerns in designing a communication system on top of an ad hoc wireless network. This concern presents a collection of approaches. One of these approaches deals with the IEEE 802.11 MAC protocol, which is standard for wireless LANs. 802.11, was not designed for multiple hop

networks [5]. One of the research areas is to find out whether IEEE 802.11 is appropriate for multiple hop ad hoc networks or not.

The ad hoc wireless networks offer unique benefit for certain applications. Since there is no need for existing fixed infrastructure, they can be created and used any time, anywhere. These are the main reasons of interest in military, police, and rescue operations.

For example in disaster situations, disruption of service may be caused by any reason such as earthquakes, floods, landslides and other natural disasters. Generally, cables and fibers are used for local and long distance transmission of signals. But the failure affects first the fixed telephone service. The mobile system of the operators becomes overloaded, and eventually it breaks down completely. Television services also starts to malfunction by leaving large part of the city without news services so people can no longer receive information on what is going on. These events may cause many problems. People are no longer able to communicate or receive news explaining the situation, and necessary actions to be taken.

Natural disasters led to the development of radio systems that could be used in distress situations in order to co-ordinate rescue work because the fixed network became overloaded by traffic from anxious people.

Some problems are also faced during the military operations. The entire fixed network can be damaged because of the hostile attacks. There must be some command control communication network between people.

All these problems imply the necessity of a network that is independent of any fixed infrastructure. So, communication is the primary issue for the disaster and military applications.

2.4.1. Issues in Ad Hoc Networks

Ad Hoc networks support a wide range of powerful applications; from instant conferencing between notebook PC users to emergency and military services that should survive in the harshest conditions. Among the areas of research in this field, are problems associated with applying the IEEE 802.11 [5], routing [26], power management [9], topology management, and security.

2.4.1.1. Routing in Ad Hoc Networks

In an ad hoc network, it may be necessary for one mobile node to aid the others in forwarding a packet to its destination. That is, two nodes that may wish to exchange packets might not be able to communicate directly with each other, due to the limited propagation range of each mobile host's transmitter. Therefore some form of routing protocol is necessary in such an environment.

For example, Figure 2-4 illustrates a simple ad hoc network of three mobile nodes using wireless network interfaces. Host C is not within the range of host A's transmitter (indicated by circle around A) and host A is not within the range of host C's transmitter. If A and C wish to communicate, they may enlist the service of the host B to forward packets for them, since B is within the overlap between A's range and C's range. The maximum number of hops needed to reach another mobile host in any practical ad hoc network is likely to be small, but often be greater than one as shown here. The routing in a practical ad hoc network may be even more complicated than this example due to the inherent non-uniform propagation characteristics of wireless transmission, since any or all of the hosts involved may move at any time.

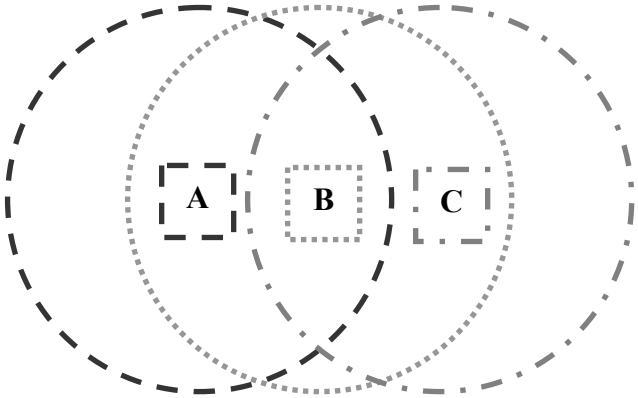


Figure 2-4 Simple Topology

Unlike cellular networks, the lifetime of mobile nodes will deeply impact the performance of the ad hoc mobile network. In a cellular network, a reduction in the number of active mobile users will reduce the amount of signal interference and channel contentions. However, since ad hoc mobile nodes need to relay their messages through other nodes toward their intended destinations, a decrease in the number of mobile users can also degrade network performance. As the number of available nodes decrease, the network may also be partitioned into smaller networks. To make the lifetime of each node longer, ad hoc routing protocols should consider power consumption.

2.4.1.2. Power Consumption in Ad Hoc Networks

Mobile ad hoc devices operate on batteries whose capacity is finite. Every message sent and every computation performed consumes the battery. Hence, the power consumption becomes an important issue. To maximize the lifetime of ad hoc mobile networks, the power consumption rate of each node must be evenly distributed, and the overall transmission power for each connection must be minimized.

Each protocol layer is closely coupled to power consumption. For example, if a routing protocol requires frequent updates of routing information, it is difficult to implement sleep mode at the data link layer. In the following paragraphs there is a brief summary of techniques employed for power conservation in each layer in the protocol stack [9].

Physical Layer: At this layer, transmission power can be adjusted. The use of excess transmission power can increase the interference to other nodes and will cause an increase in transmission power by other nodes. Therefore physical layer functions should ensure transmitting data at the minimum power level to maintain links.

Data Link Layer: At this layer, energy conservation can be achieved by using effective retransmission request schemes and sleep mode operation. The data link layer provides error free communication between two nodes. It detects transmission errors and if necessary request retransmission of packets. In ad hoc networks, due to the presence of mobility and co-channel interference, transmission errors occur frequently, which lead to frequent retransmission requests. Retransmissions lead to extra power consumption and cause higher interference to other users.

Network Layer: Paths are computed based on minimizing hop count or delay. To maximize the lifetime of mobile nodes, routing algorithms must select the best path by taking care of power constraints. Hence, routes requiring lower levels of power transmission are preferred.

CHAPTER 3

IEEE 802.11 MEDIUM ACCESS CONTROL AND QUALITY OF SERVICE

In 1997 the IEEE adopted IEEE Std. 802.11-1997, the first wireless LAN (WLAN) standard. This standard defines the media access control (MAC) and physical (PHY) layers (Figure 3-1), [21]. It addresses local area networking where devices communicate to other devices that are within close proximity to each other.

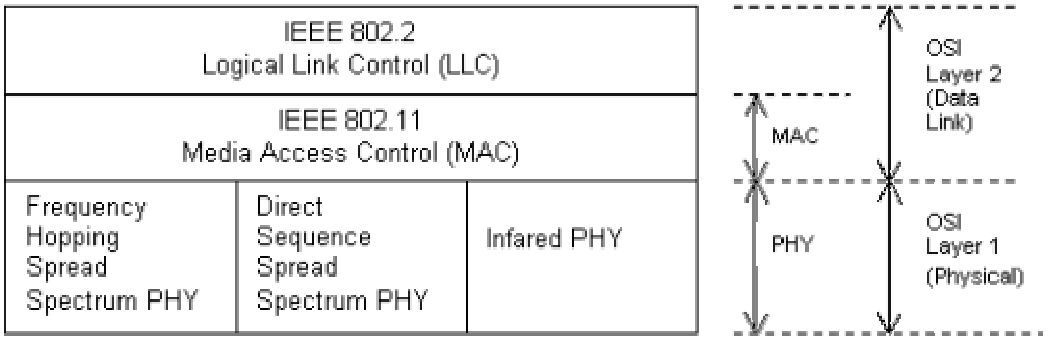


Figure 3-1 IEEE 802.11 Standards Mapped to the OSI Reference Model

The IEEE 802.11 physical layer is the interface between the MAC and the wireless media where frames are transmitted and received. The PHY provides three functions. First, it provides an interface to exchange frames with the upper MAC layer for transmission and reception of data. Secondly, it uses signal carrier and spread spectrum modulation to transmit data frames over the media. Thirdly, it provides a carrier sense indication back to the MAC to verify activity on the media.

IEEE 802.11 defines three different PHY: Frequency Hopping Spread Spectrum (FHSS), Direct Sequence Spread Spectrum (DSSS) and Infrared (IR), [21].

3.1. Medium Access Control

The 802.11 MAC layer provides functionality to allow reliable data delivery for the upper layers over the wireless PHY media. The 802.11 MAC provides a controlled access method to the shared wireless media based on Carrier-Sense Multiple Access (CSMA).

The primary access method of the IEEE 802.11 MAC protocol for ad hoc networks is the distributed coordination function (DCF). In DCF mode, Wireless Terminals (WTs) have to content for the use of the channel at each data packet transmission. The secondary access method is the Point Coordination Function (PCF). In PCF mode, the medium usage is controlled by the Point Coordinator (PC), polling the WTs to access the medium, thus eliminating contentions. DCF is the only possible function in ad-hoc networks, and can be either exclusive or combined with PCF when used in an infrastructure network (equipped with an AP).

3.1.1. Carrier Sense Multiple Access in WLAN

The ALOHA protocol is the first protocol for multiple access channel and it refers to simple communications scheme in which each source in a network sends data whenever there is a frame to send. If the frame successfully reaches the destination, the next frame is sent. If the frame fails to be received at the destination, it is sent again. ALOHA does not consider if other stations have started transmission, thus result in a lot of collisions. One improvement can be made is to try to find out if others are transmitting by sensing the medium.

- If no signal is detected on the medium, the station starts transmission.
- If it senses a transmission is going on, the station defers its transmission at some late time according to a certain protocol.

This is the basic idea of *carrier sense multiple access (CSMA),* [5].

In CSMA protocol, whenever a station wants to transmit, it first listen the medium. If the medium is busy, the station defers the transmission to a later time. If the medium is free, station transmits. This kind of protocols is effective when the medium is not heavily loaded. But there is always a chance of two stations simultaneously sensing the medium idle and transmitting at the same time. This is the case faced in wired networks for collision. In wireless networks, this problem is defined as hidden node problem, that is, two stations are transmitting at the same time because they don't hear each other.

In order to reduce the probability of two stations colliding due to not hearing each other, the standard defines a virtual carrier sensing mechanism. A hidden node is one that is within the interfering range of the destination but out of the sensing range of the transmitter. Due to signal fading, range of a transmitter is limited, so there is a possibility of a station believing the channel is idle, while there is already a transmitting station.

Besides the hidden node problem, wireless networks also face the exposed node problem. An exposed node is one that is within the sensing range of the sender but out of the interfering range of the destination. In the 802.11 MAC layer protocol, there is no scheme to deal with this problem, [5]. This causes waste of bandwidth when most of the routes are of multihop type in a wireless network. Due to the exposed node problem, although it will not cause any problem, the intermediate node may not transmit a packet to its next hop while there is an ongoing transmission in the medium from other stations. In Figure 2-4, Node B sends to node A and node C wants to send another node out of the range of nodes A and B. Node C senses the medium before it transmits, and finds the medium is busy. This leads to node C to wait. Actually waiting is not necessary for C, because there will be no collision in the medium and the communication in the medium is independent from node C, but it has to wait until the medium is free to transmit. Node C is "exposed node" to node B.

3.1.1.1. Inter Frame Space Time

Not all packet types have the same priority. For example, acknowledgement (ACK) packets should have priority over RTS packets. This is done by associating each packet type a certain Inter Frame Space (IFS) time before which a packet cannot be transmitted once the channel becomes idle. In DCF, two IFSs are used: Short IFS (SIFS) and DCF IFS (DIFS), where SIFS is shorter than DIFS (see Figure 3-2). As a result, if an ACK (affected with SIFS) and a new data packet (RTS affected with DIFS should be sent before data packet is sent) are waiting simultaneously for the channel just after the previous transmit, ACK will be transmitted before the new data packet, since it has to wait for SIFS whereas the RTS has to wait for DIFS which is longer than SIFS.

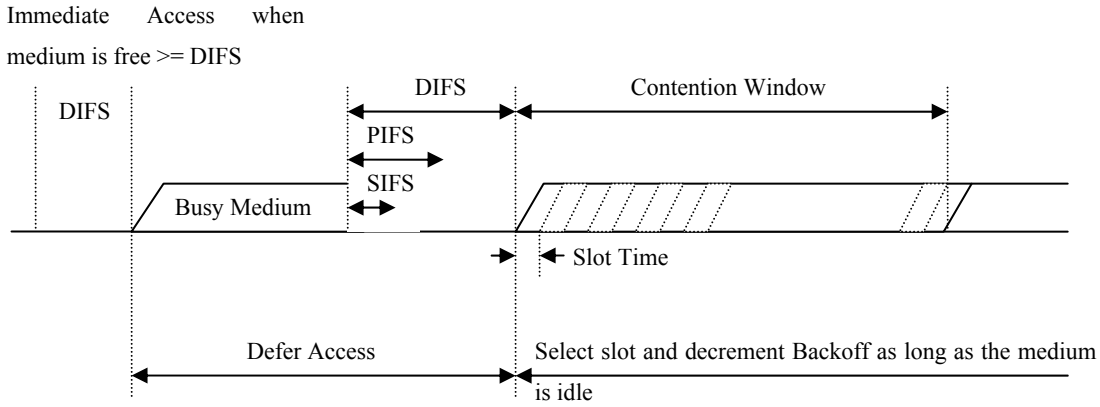


Figure 3-2 Inter Frame Space Time

3.1.2. Distributed Coordination Function

DCF provides a contention based medium access. In DCF, stations employ Carrier Sense Multiple Access (CSMA) to sense medium and start transmission if the medium is idle.

Carrier Sense Multiple Access with Collision Detection (CSMA/CD) is used in Ethernet (IEEE 802.3) wired networks to deal with collision. Whenever a node

detects that the transmitted signal is different from the one on the channel, it aborts transmission, saving useless collision time. This mechanism is not possible in wireless communications as WTs cannot listen to the channel while transmitting, due to the big difference between transmitted and received power levels. The signal received from other stations cannot be distinguished from transmitted signal, because received signals fade through the air and this signal has relatively low power compared to stations own transmitted signal power. Therefore, CSMA/CD is not applicable in wireless communications.

3.1.2.1. Basic Access Mechanism

DCF is based on Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). Before a station starts transmission, it senses the wireless medium to determine if it is idle. If the medium appears to be idle, the transmission may proceed, else the station will wait until the end of current transmission.

The CSMA/CA mechanism requires a minimum specified space between contiguous frame transmissions. A station will ensure that the medium has been idle for the specified inter frame interval before attempting to transmit. The DIFS is used by stations operating under the DCF to transmit data frames. A station using the DCF has to follow two medium access rules: (1) the station will be allowed to transmit only if its carrier sense mechanism determines that the medium has been idle for at least DIFS time; and (2) in order to reduce the collision probability among multiple stations accessing the medium, the station will select a random backoff interval after deferral, or prior to attempting to transmit another frame after a successful transmission.

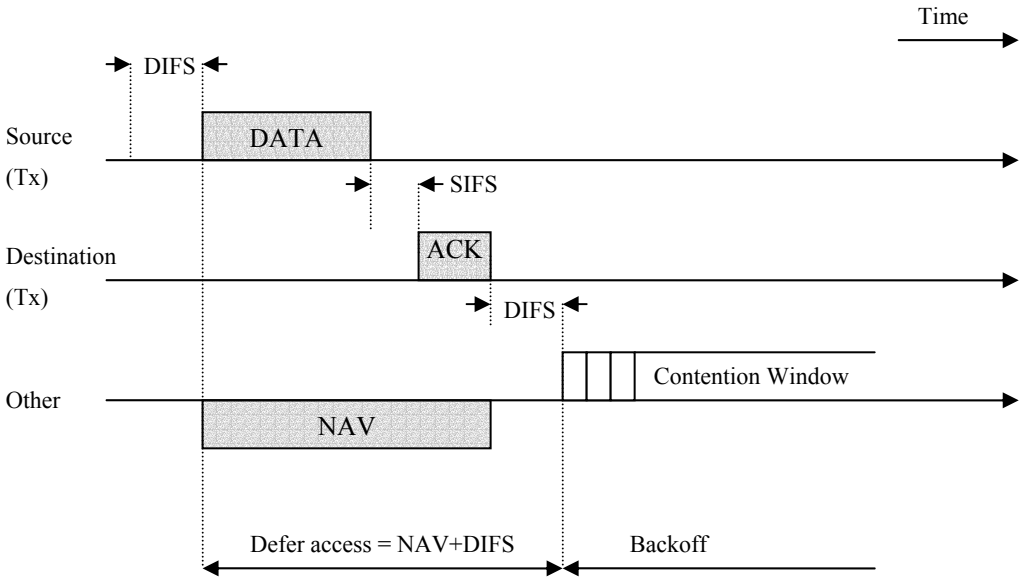


Figure 3-3 Basic Access Mechanism [10]

In Figure 3-3, source axis shows data transmitted by the source. The destination responds by an ACK, represented on the Destination axis. The third axis represents the network state, as seen by other WTs. Note that propagation delays are not shown.

3.1.2.2. RTS/CTS Scheme

Carrier sensing can be performed on both layers. In the physical layer, physical carrier sensing is done by detecting any channel activity caused by other sources. On the MAC sub layer, virtual carrier sensing can be done by updating a local Network Allocation Vector (NAV) with the value of other terminal's transmission duration which is placed in data and control frames. NAV is a virtual carrier sense indicator to designate the amount of time that must elapse until current transmission completes. Using the NAV, a WT MAC knows when the current transmission will end. Virtual carrier sensing mechanism allows for a station that wants to send

something to first send a Request to Send (RTS), which is a short packet that contains the source and destination addresses, as well as the duration of the transmission. If the medium is free, the receiver station will then reply with a short packet called Clear to Send (CTS), which will include the duration of transmission information. Stations that receive the RTS, the CTS, or both packets will set their NAV, so the hidden node problem is avoided. By sending out these short RTS and CTS packets, the likelihood of collision drops because stations that may not normally be able to hear each other will know to consider the medium busy until the end of the transmission. This scheme is shown in Figure 3-4 .

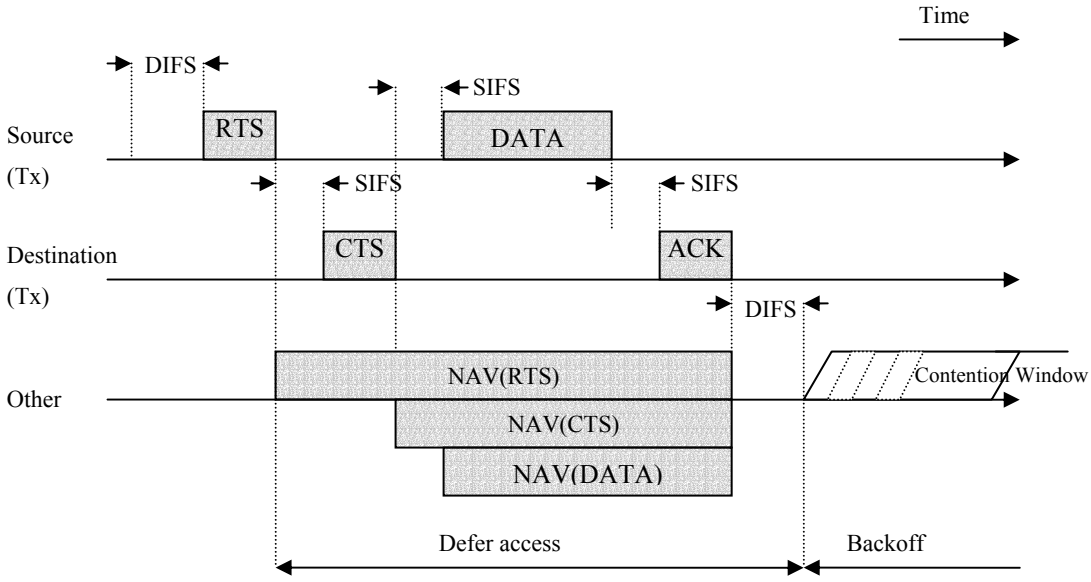


Figure 3-4 RTS/CTS Access Scheme [10]

3.1.2.3. Backoff Time

Station waits for a random time after it senses the channel idle for DIFS duration, before it transmits the packet. This random time is named as backoff time, shown in Figure 3-3.

The DCF adopts a binary exponential backoff mechanism to select the random backoff interval (in unit of slot time). This random number is drawn from a uniform distribution over the interval [0, CW-1], where CW is the contention window size and initial value of CW is CW_{min}. If there is an unsuccessful transmission attempt, i.e. not receiving an acknowledgement, the WT computes a new random backoff time, with a new range. This range increases exponentially as 2^{2+i} where *i* (initially equal to 1) is the transmission attempt number. Therefore, the backoff time equation is [22]:

$$Backoff\_time=[2^{2+i} \times rand()] \times Slot\_time$$

where Slot_time is function of some physical layer parameters, and rand() is a random function with a uniform distribution in [0,1]. There is a higher limit for retransmission attempts i, above which the frame will be dropped. This parameter is defined as 7 by default but can be changed.

Once CW reaches CW_{max}, it will remain at this value until it is reset to CW_{min}. In the case of a successful transmission, the CW value is reset to CW_{min} before the random backoff interval is selected. Each station decrements its backoff counter every slot time interval after the wireless medium is sensed to be idle for DIFS time. When the counter finally reaches zero, the station starts its transmission.

Figure 3-5 illustrates such an operation of decrementing the backoff counter. After the successful transmission and acknowledgment of frame A1, station A waits for DIFS time and selects a backoff interval equal to 6, before attempting to transmit the next frame A2. Assume that station B selects a smaller backoff interval equal to 3 after it has sensed the medium to be idle for DIFS time. Since the backoff counter of station B reaches zero before that of station A, frame B1 is transmitted after exchanging the RTS and CTS frames. As a result of the medium sensed busy, the backoff counter of station A is frozen at 3, and decrements again after the medium is sensed idle for DIFS time.

The stations that deferred from channel access during the channel busy period do not select a new random backoff time, but continue to count down the time of the

deferred backoff in progress after sensing a channel as being idle again. In this manner, stations, that deferred from channel access because their random backoff time was larger than the backoff time of other stations, are given a higher priority when they resume the transmission attempt.

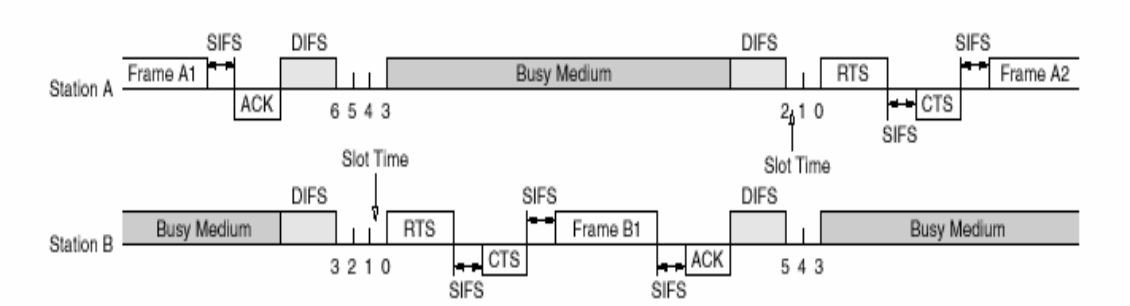


Figure 3-5 Backoff Decrements under 802.11 DCF [27]

3.1.3. Point Coordination Function

PCF is a centralized, polling-based access mechanism which requires the presence of a base station, or access point, that acts as Point Coordinator (PC). The PCF may be required to coexist with the DCF and logically sits on the top of the DCF (Figure 3-6) [22]. If PCF is to be used, time is divided into super frames where each superframe consists of a contention period where DCF is used, and a contention-free period (CFP) where PCF is used (Figure 3-7). Base station starts the CFP with beacon frame, using the ordinary DCF access method with higher priority by using PIFS. Therefore, the CFP may be shortened since the base station has to contend for the medium.

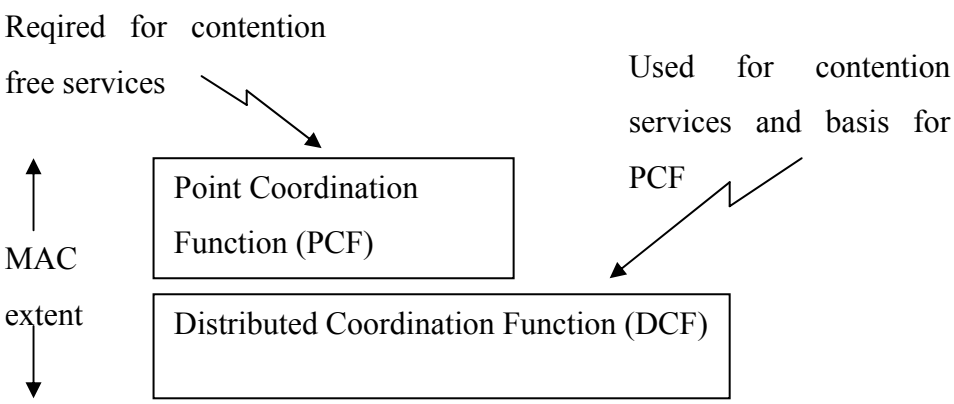


Figure 3-6 MAC Architecture

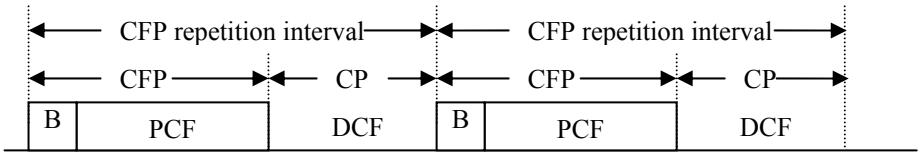


Figure 3-7 Coexistence of PCF and DCF

The CFP repetition interval is initiated by a beacon frame, which is transmitted by the AP. The duration of the CFP repetition interval is a manageable parameter. The maximum size of the CFP is determined by CFP_Max_Duration. At a minimum, time must be allotted for at least one MPDU (MAC Protocol Data Unit, an MPDU is a complete data unit that is passed from the MAC sub layer to the physical layer) to be transmitted during the CP.

During the CFP, the PC polls each station in its polling list, to check whether they are clear to access the medium. To ensure that no DCF stations are able to interrupt this mode of operation, the Inter Frame Space (IFS) between PCF data frames is shorter than the DCF IFS (DIFS). This time is called a PCF Inter Frame Space (PIFS).

At the nominal beginning of each CFP repetition interval, all stations in the BSS update their NAV to the maximum length of the CFP (i.e., CFP_Max_Duration). During the CFP, the only time stations are permitted to transmit is in response to a poll from the PC or for transmission of an acknowledgment a SIFS interval after receipt of an MPDU.

At the nominal start of the CFP, the PC senses the medium. If the medium remains idle for a PIFS interval, the PC transmits a Beacon frame to initiate the CFP. The PC starts CF transmission a SIFS interval after the beacon frame is transmitted by sending a CF-Poll (no data), Data, or Data+CF-Poll frame. The PC can immediately terminate the CFP by transmitting a CF-END frame, which is common if the network is lightly loaded and the PC has no traffic buffered. If a CF-Aware station receives a CF-Poll (no data) frame from the PC, the station can responds to the PC after a SIFS idle period, with a CF-ACK frame (no data) or a Data+CF-ACK frame. If the PC receives a Data+CF-ACK frame from the station, the PC can send a Data+CF-ACK+CF-Poll frame to a different station, where CF-ACK portion of the frame is used to acknowledge receipt of the previous data frame. If the PC transmits a CF-Poll (no data) frame and the destination station does not have a data frame to transmit, the station sends a Null Function (no data) frame back to the PC. If the PC fails to receive an ACK for a transmitted data frame, the PC waits a PIFS interval and continues transmitting to the next station in the polling list (Figure 3-8).

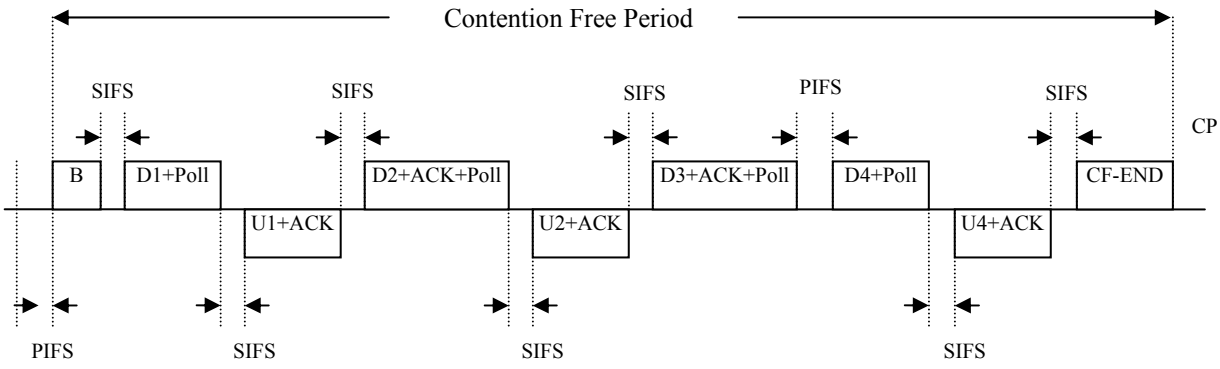


Figure 3-8 PC to Station

3.2. Quality of Service

IP-based internets were designed for applications that are relatively delay insensitive, can tolerate variations in throughput, and can tolerate packet loss, and they were initially deployed using relatively low capacity links. Today, IP-based internets are being asked to support high volumes of traffic over high capacity links, and the traffic includes real-time or near real-time applications that are sensitive to delay, packet loss, and require certain throughput. There are some arguments that have been stated against any need for complicated system to provide some QoS, [26].

Bandwidth will be infinite

There will be a large carrying capacity (especially by the use of optical fibers) and it will be cheap. There will be no communication delays other than delays due to bounded speed of light. However, this argument is not valid for wireless communication because bandwidth available in usable radio frequencies is limited.

Simple priority is sufficient

Prioritizing traffic would lead to better support for real-time traffic. Problem is where to assign the priority; users may abuse the priority mechanism. The user terminals cannot generally be trusted to give "fair" priorities for different traffic flows. So there should be some control over the priorities of traffic flows.

3.2.1. Issues in QoS

Three major considerations of QoS support are bandwidth, delay and delivery guarantee. Voice, data, image, and video have different bandwidth requirements. Some classes of traffic, such as voice, are also much more sensitive to delay than background classes, such as data. QoS provisioning in a wireless network is a particularly difficult issue because physical layer problems; such as path loss, fading, and multipath; can make the communication links unreliable. This makes

delivery guarantee a necessary feature in wireless ad hoc network QoS provisioning.

The challenge is to first prescribe a feasible QoS scheme for different classes of traffic, and then to optimize the use of network resources, mainly link capacities and transmitter powers, to satisfy QoS requirements for all classes while maximizing either the total network performance. Unlike cellular wireless networks, ad hoc networks have no fixed infrastructure, and long range communications require multihop transmissions where a packet is routed through the network by other transceivers that act as relay nodes.

Some of the requirements [2], [14] can be identified for a multihop wireless networks as below:

Bandwidth reservation: The wireless network must implement network in order to support real time connections and allocate bandwidth to them at call setup time. Network bandwidth reservation in a mobile network is a major challenge since reservations have to be maintained wherever the mobile goes.

QoS routing: Traditional minimum hop or minimum delay routing algorithms are not adequate. To support QoS for real time traffic we need to know not only the minimum delay path to destination, but also the bandwidth available on it. For example, at call setup time, a circuit should be accepted only if there is enough available bandwidth. Routing with QoS indication is thus needed in order to efficiently manage bandwidth resources [26].

Consider the Figure 3-9 where the numbers next to the radio represents their respective bandwidth in megabits per seconds. To minimize delay and better use of network resources, minimizing the number of intermediate hops is one of the principle objectives in determining the suitable routes. However, suppose that the packet flow from A to E requires a bandwidth guarantee of 3Mbps. QoS routing will then select the route A-B-C-E over route A-D-E, although the A-D-E route has fewer hop.

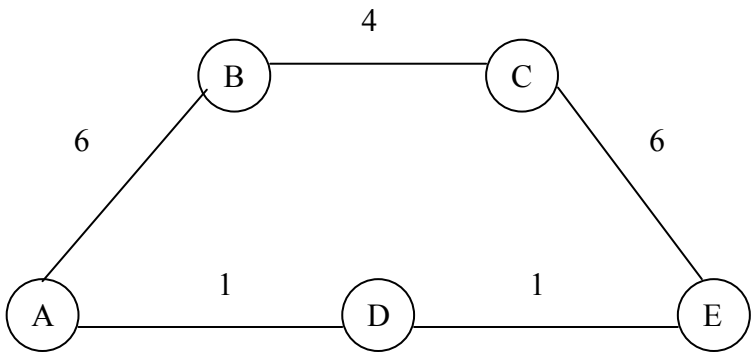


Figure 3-9 An Ad Hoc Network

Once a route has been selected for a specific flow, the necessary resources (bandwidth, buffer space, etc) must be reserved for the flow.

Congestion control: Network congestion can be faced due to the dynamics of mobility and of traffic patterns. Some of the concepts should be used in wireless networks, i.e. carrier sensing.

Mobility: The mobility of the terminals is a critical issue. Networks formed by mobile terminals are weaker than the fixed networks due to the difficulty of maintaining of the routes. Reservation of resources and the maintenance of QoS for the mobile as it moves from one region to another create a new set of challenges. Apart from resource reservation, there is a need for routing techniques which help in faster routing updates so that QoS can be maintained.

Besides these, medium access control can be identified as a quality of service mechanism. MAC enhancements are found in the current 802.11e draft specification by emphasizing the differences from the legacy 802.11 standard. New mechanisms for QoS support, namely Enhanced Distributed Coordination Function (EDCF) is defined in 802.11e draft.

3.2.2. Traditional Methods for QoS

To respond to the new demands on IP-based networks, an elaborate set of mechanisms are being developed. Two complementary efforts have been pursued: integrated services (IntServ) and differentiated services (DiffServ), [24].

IntServ framework is concerned with providing an integrated, or collective, service to the set of the traffic demands placed on a given domain. IntServ provider can be thought as consisting of the network elements within the domain. The IntServ provider views the totality of the current traffic demand and; limits the demand that is satisfied to that which can be handled by the current capacity of the network and, reserve resources within the domain to provide a particular QoS to particular portion of the satisfied demand.

DiffServ framework does not attempt to view the total traffic demand in any overall or integrated sense, nor does it attempt to reserve network capacity in advance. Rather, in the DiffServ framework, traffic is classified into a number of traffic groups. Each group is labeled appropriately, and the service provided by network elements depends on group membership, with packets belonging to different groups being handled differently.

The complexity of services for QoS architecture is shown in Figure 3-10.

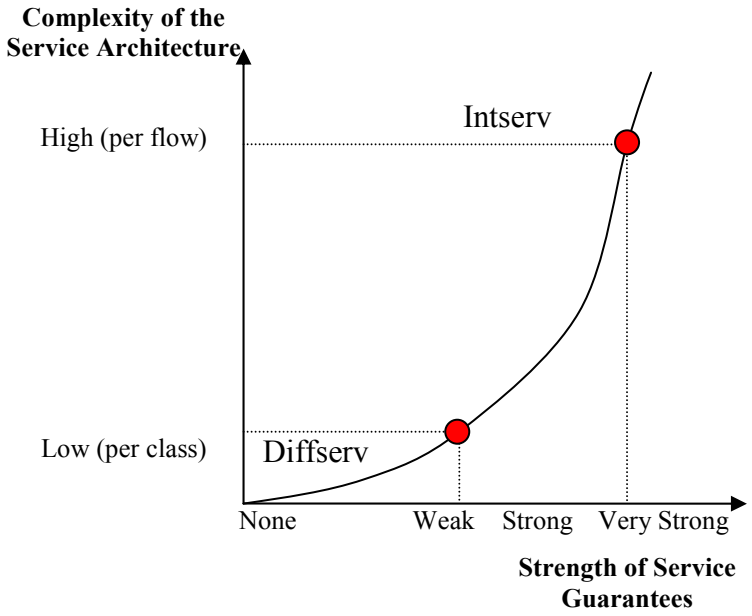


Figure 3-10 Tradeoff between Complexity of QoS Architecture and Strength of Guarantees

Traditional QoS methods are used in a fair medium access of stations. There can be multiple stations contending for the same channel, this leads a need for a mechanism that provides an unfair medium access. So, QoS enhancement can be supported by adding service differentiation into the MAC layer. This can be achieved by modifying the parameters that define how a station or flow should access the medium.

3.2.3. Enhanced Distributed Coordination Function

IEEE 802.11e – Enhanced DCF Task group E of the IEEE 802.11 working group are currently working on an extension to the IEEE 802.11 standard called IEEE 802.11e. The goal of this extension is to enhance the access mechanisms of IEEE 802.11 and provide a distributed access mechanism that can provide service differentiation. All the details have not yet been finalized, but a new access

mechanism called Enhanced DCF (EDCF), which is an extension of the basic DCF mechanism, has been selected.

The goal of this scheme is to enhance DCF access mechanism of IEEE 802.11 and to provide a differentiated access approach that can support service differentiation. EDCF combines two measures to provide differentiation, [4]. The proposed scheme provides capability for up to eight types of traffic classes. It assigns a short CW to high priority classes in order to ensure that in most cases, high-priority classes will be able to transmit before the low-priority ones. More specifically, the CW_{min} parameter is set differently for different priority classes, yielding high priority classes with small CW_{min}. For further differentiation, different inter frame spaces can be used by different traffic classes. Instead of DIFS, an inter frame space called Arbitration Inter Frame Space (AIFS) is used. The AIFS for a given class should be a DIFS plus some time slots. Classes with smaller AIFS will have higher priority as shown in Figure 3-11.

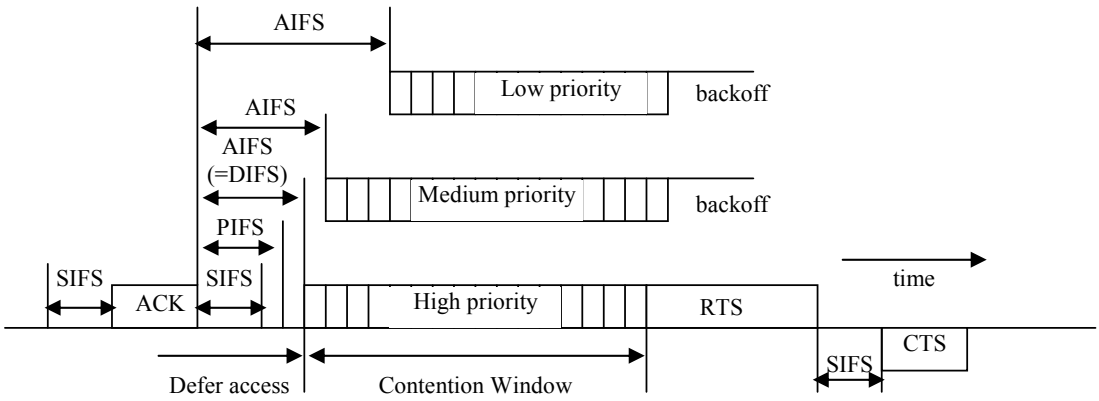


Figure 3-11 Different AIFS [16]

In this thesis, we have used EDCF implementation (by Ni Qiang) for service differentiation and used different CW_{min} approach for different classes which have different priorities. CW_{min} value is set to 31 slots as default. For service differentiation, we have changed this value to a smaller value as 7 slots for high

priority traffic flow and we haven't change the CW_{min} value for low priority traffic flow, set as 31 slots. The duration of each slot is set to 20 μsec.

The source code of the Ns-2 with EDCF module can be reached from Ni Qiang's web page [29].

CHAPTER 4

ROUTING AND MOBILITY

A natural method for trying to provide routing in an ad hoc network is to simply treat each mobile node as a router and to run a conventional routing protocol between them. In effect, node B in Figure 2-4 acts as a router between the network directly reachable by A and the network directly reachable by C. Node A transmits its packets for C to B, which then forwards them on to C. Conventional routing protocols are mainly based on either *distance vector* or *link state algorithms*.

In *distance vector routing*, each router maintains a table giving the distance from itself to all possible destinations. Each router periodically broadcasts this information to each of its neighbor routers, and uses the values received from its neighbors to compute updated values for its own table. By comparing the distances received from each destination from each of its neighbors, a router can determine which of its neighbors is the correct "next hop" on the shortest path toward each destination. When presented a packet for forwarding to some destination, each router simply forwards the packet to the correct next hop router. By transmitting routing table updates more frequently such as when any information in the table changes, the algorithm converges more quickly to the correct path, but the overhead in CPU time and network bandwidth for transmitting routing updates increases.

In *link state routing*, each router maintains a complete picture of the topology of the entire network. Each router monitors the cost of the link to each of its neighbor routers, and periodically broadcast an update of this information to all other routers in the network. Given this information of the cost of each link in the network, each

router computes the shortest path to each possible destination. When presented a packet for forwarding to some destination, each router forwards the packet to the next hop router based on its current best path to that destination. Link state routing protocols converge much more quickly as conditions in the network change, but generally require more CPU time (to compute the complete shortest path to each possible destination) and more network bandwidth (to broadcast the routing update from each router to all other routers in the entire network) than distance vector algorithms.

Actually there are some other routing algorithms as source routing, broadcast, etc.

Source Routing is a technique whereby the sender of a packet can specify the route that a packet should take through the network. It may be used to map the network to find all routes between points on the network.

In broadcast routing, source node sends a packet to all receivers. There are two algorithms for broadcast routing: point-to-point and flooding. In point-to-point, source sends packets to every destination. This requires knowledge of all destinations and also wastes the bandwidth. In flooding, when a node receives a broadcast packet, it sends the packet to every link. Nodes may receive many copies of the broadcast packets, hence must be able to detect duplicates.

4.1. Challenges in Routing

Although using either type of conventional routing protocol in an ad hoc network, treating each mobile node as a router, may often work, there are a number of problems with this approach:

- Transmission between two nodes over a wireless network does not necessarily work equally well in both directions. Even though node A in Figure 2-4 may receive a routing update from B indicating that B is closest to C and thus would be the first hop on A's shortest path to C, node A may in fact be unable to transmit a packet back to B. Figure represents the transmission range of all nodes as equal and uniform on all sides of the

node, but radio propagation does not always work so nicely in reality. Thus, some routes determined by conventional routing protocols may not work in some environments.

- Many links between routers seen by the routing algorithm may be redundant. Rather than a single router (node B) between A and C, there may be many mobile nodes within A's range, all perhaps equally good for forwarding packets to C. Wired networks, on the other hand, are usually explicitly configured to have only one (or a small number) of routers connecting to any two networks. The redundant paths in the wireless environment unnecessarily increases the size of routing updates that must be sent over the network, and increases the CPU overhead required to process each update and to compute new routes.

- Periodically sending routing updates wastes network bandwidth. Often, nothing will change from one routing update to the next, but each router must continue to send periodic updates so that other routers will continue to consider routes through that router as valid. Routing updates from mobile nodes outside each other's transmission range will not interfere with each other, but where many mobile nodes are within transmission range of each other, their routing updates will consume each other's network bandwidth.

- Periodically sending routing updates wastes battery power. Most mobile nodes in ad hoc network are operating on battery, and transmitting a packet expends some amount of battery power. Although receiving a packet generally requires less power than sending one, the need to receive these periodic routing updates effectively prevents a node from conserving its own battery power by putting itself into "sleep" or "standby" mode when not otherwise busy.

4.2. Approaches to Routing in Ad Hoc Networks

The problem of routing can be divided into the two sub-problems of *route discovery* and *route maintenance.* In order for one node to communicate with another, it must initially discover a suitable route to use in sending packets to that destination. As long as conditions remain unchanged, this route continues to work. However, as the status of different links or routers used in this route change, changes in the route may be necessary, or a new route may need to be discovered.

4.2.1. Route Discovery

A very simple method of route discovery suitable for use directly in some ad hoc networks is the internet's Address Resolution Protocol (ARP). ARP is designed for dynamically translating a node's network protocol address (IP address) to its MAC level address (Ethernet address). A node attempting to translate another node's address broadcast a query packet onto its local network, which is answered by the target node giving its MAC address; other nodes on the local network receiving the query do not reply. The returned MAC address is then cached by the node for use in sending future packets to this destination.

In an ad hoc network, if the source and target mobile nodes are both within transmission range of each other, a simple ARP query is all that is needed to find a route to the target node; the returned MAC address may be used directly to transmit packet to that node. In this case no periodic routing updates are needed, providing substantial saving in network bandwidth and battery power. But source and destination may not be within range of each other. In this case, a general solution should be provided for route discovery.

One possible solution is to send a request packet but to propagate the request using some form of flooding, in order to reach other mobile nodes in the sender's transmission range. As the request propagates, each node adds its own address to a route being recorded in the packet, before broadcasting the request to its neighbors.

When a request is received and the node finds its own address recorded in the route, it discards that copy of the request and does not forward that copy further.

Since many mobile nodes may be within transmission range of each other, there may be many duplicate copies of each request. To largely eliminate these duplicates, each request should contain a unique *request id* from the original sender; each node keeps a cache giving the request id and sender address of recently forwarded request, and discard a request if it has already transmitted an earlier copy of the same request id. Thus, each node will only transmit the first copy of each request that it receives. This scheme could easily be extended to include the length of the path in the request id cache and to transmit a later copy of the same request if it somehow arrived over a shorter path than the earlier copy.

Although more than one network hop may be needed to reach another mobile node in an ad hoc network, the maximum number of hops needed it is likely to be limited. The number of duplicate requests transmitted can thus be further reduced by limiting the maximum number of hops over which any route discovery packet can be propagated. When processing a received route discovery request, a mobile node should discard the request rather than forwarding it if it is not the target of the request and if the route recorded in the packet has already reached to its maximum length.

When the query packet reaches the target node, the complete route from the original sender to this node will have been recorded in the packet. In order to be of use to the original sender, the route information must then be returned to the sender. The target node may attempt to reserve the recorded route to reach the original sender, or may use the same route discovery procedure to find a route back to the original sender; the route from the original sender to this target should be returned to the sender in the new query packet used for its own route discovery.

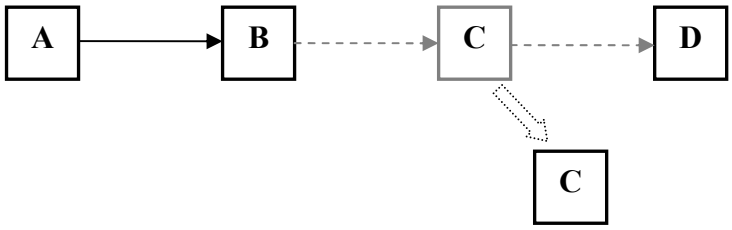
Mobile nodes should cache route discovered for use in sending future packets to that same destination. If a mobile node has cached a route listing some number of hops to a destination mobile node, then the shortest route to each of the hops listed

on that route is naturally a prefix of that route. Also, as a mobile node forwards packets, it will be able to observe many other routes to other mobile nodes, since each packet contains a route. By examining the routes on packets that it forwards, a mobile node may be able to cache routes to new destinations or to obtain updated information to destinations already in its cache. Furthermore, since transmissions in a wireless network are necessarily broadcast transmissions, a mobile node may be able to learn new routing information from the route contained in any packet that it can receive, even if the packet is not explicitly addressed to this node.

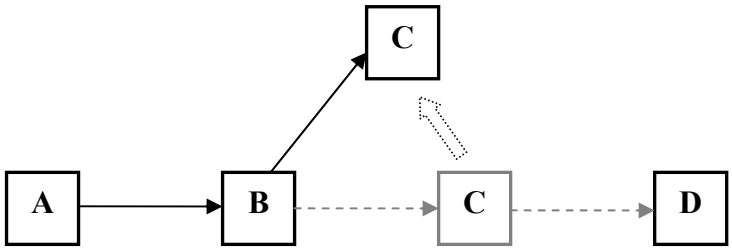
4.2.2. Route Maintenance

Conventional routing protocols integrate route discovery with route maintenance by continuously sending the periodic routing updates. If the status of a link or router changes, the periodic updates will eventually reflect the changes to all other routers, resulting in the computation of new routes. When a link or router goes down, it will cause the route to stop working with no feedback to the sender. The role of the route maintenance protocol is to provide this feedback, and to allow the route to be modified or a new route to be discovered in this case.

In an ad hoc network, a route may also stop working if one or more of the mobile nodes along the route simply move. For example, Figure 4-1 illustrates two possible scenarios in which the movement of a mobile node causes an existing route to stop working. Assume that mobile node A has been sending packets to mobile node D using a route through mobile nodes B and C. Figure 4-1 (a) shows the case in which C has moved out of range of B, breaking the route on to D. Figure 4-1 (b) shows a different scenario in which C has moved such that it is now out of range of its next hop on to D; in this case, C after moving is still within range of B, but it has led the route away from D.



(a) Host C has moved, breaking the route at B



(b) Host C has moved, leading the route away from D

Figure 4-1 (a), (b) Route Changes due to Node Movement

In many wireless networks, route maintenance can be provided with a very little overhead. Since wireless networks are inherently less reliable than wired networks, many wireless networks utilize a hop-by-hop acknowledgement at the data link level in order to provide early detection and retransmission of lost or corrupted packets. In these networks, the route maintenance is simple, since at each hop, the sender can determine if that hop of the route is still working. If data link level reports a transmission problem for which it cannot recover, all that is needed is to report this error back to the original sender to cause that node to start the route discovery procedure again to find a new route.

If the wireless network does not support such lower level acknowledgements, an equivalent acknowledgement signal may be available in many environments. After sending a packet to the next hop mobile node, the sender may be able to hear that

node transmitting the packet again, on its way further along the path. For example, in Figure 2-4, node A may be able to hear B's transmission of the packet on to C. In addition, existing packets coming in the reverse direction along the same link could also be used as an acknowledgement that the route is still working.

The performance of route discovery and route maintenance protocols depends on a number of factors such as how often mobile nodes in such an environment attempt to communicate with other mobile nodes and how often mobile nodes move to cause existing routes to stop working.

4.3. Routing Protocols

Since some receiving nodes may be out of the direct range of a sending node, intermediate nodes have to act as routers to forward the data to the receiving nodes. Ad-hoc routing protocols have been developed to provide the route discovery and maintenance mechanisms for each mobile node in the network to communicate with all other nodes of the network.

Ad-hoc routing protocols can be divided into two categories:

Table-driven routing protocols

In table driven routing protocols, consistent and up-to-date routing information to all nodes is maintained at each node. Each node has one or more tables that contain the latest information of the routes to any node in the network. Each row has the next hop for reaching a node and the cost (for example, number of hop) of this route.

On-Demand routing protocols

In On-Demand routing protocols, the routes are created when required. When a source wants to send to a destination, it invokes the route discovery mechanism to find a path to that destination. These protocols take a lazy approach to routing. They do not maintain or constantly update their routing tables with the latest topology.

There are four multi-hop wireless ad hoc network routing protocols that are proposed to the IETF [15], [20]:

- Destination Sequenced Distance Vector (DSDV)
- Dynamic Source Routing (DSR)
- Temporally Ordered Routing Algorithm (TORA)
- Ad Hoc On-Demand Distance Vector Routing (AODV).

While DSDV is a table-driven routing protocol, TORA, DSR, AODV, fall under the on-demand routing protocols category.

4.3.1. Destination Sequenced Distance Vector Routing

DSDV is table-driven route discovery protocol; each node maintains a routing table with the next hop entry for each destination and the associated cost. In addition, each node has a sequence number associated with it. This sequence number is periodically incremented by the destination node for the link. Other nodes then choose the route with highest sequence number, as that is the more recent route established to the destination. If a node detects that a link has broken, it sets the metric to infinity, and issues a route update to the other nodes. Other nodes repeat this action until they receive an update with a higher sequence number to provide it with a fresh route again.

The following information for each node is kept in the routing table:

- The destination's address
- Next hop address to reach the destination
- The number of hops required to reach the destination
- The sequence number created by the transmitter

When a mobile node receives new routing information, that information is compared to the information already available from previous routing information packets. Any route with a more recent sequence number is used. Routes with older sequence numbers are discarded. A route with a sequence number equal to an existing route is chosen if it has a better metric, and the existing route discarded, or stored as less preferable.

Consider the Figure 4-2. Table 4-1 shows all possible structure of all forwarding table which is maintained at node 4.

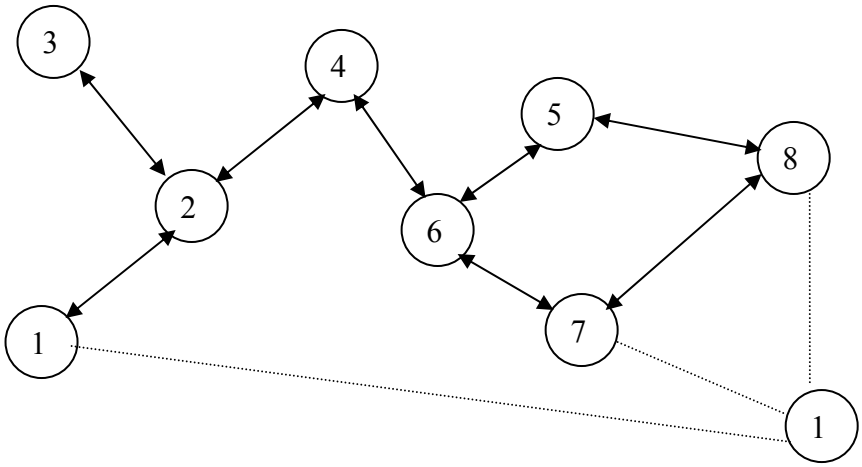


Figure 4-2 Movement in Ad Hoc Network

Table 4-1 Structure of Node 4 Forwarding Table

Destination	Next Hop	Metric	Sequence
Node1	Node2	2	S406_N1
Node2	Node2	1	S128_N2
Node3	Node2	2	S564_N3
Node4	Node4	0	S510_N4
Node5	Node6	2	…
Node6	Node6	1	…
Node7	Node6	2	..
Node8	Node6	3	..

The main advantage to DSDV is that it maintains a loop-free fewest-hop path to every destination in the network. However, this protocol also contains both periodic and triggered route updates. While the triggered updates tend to be small (allowing quick discovery of invalid links), each node's periodic update includes its entire routing table.

4.3.2. Dynamic Source Routing

DSR is one of the more generally accepted ad-hoc routing protocols. It utilizes source-based routing rather than table-based, and it is source-initiated rather than hop-by-hop. When a node wishes to establish a route, it issues a Route Request (RREQ) to all of its neighbors. Each neighbor rebroadcasts this Request, adding its own address in the header of the packet. When the Request is received by the destination or by a node having a route to the destination, a Route Reply (RREP) is generated and sent back to the sender along with the addresses accumulated in the Request header (see Figure 4-3).

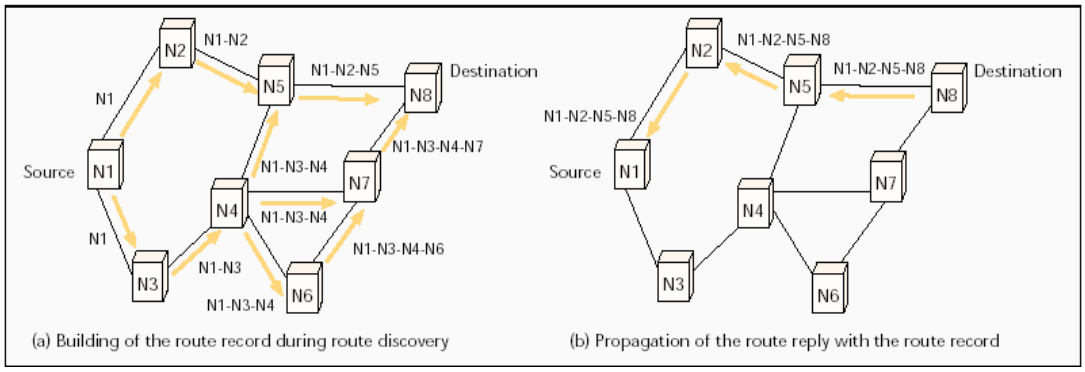


Figure 4-3 Creation of Route in DSR [20]

Each node is responsible for the status of the route. Each must insure that packets successfully cross the link to the next node. If it doesn't receive an acknowledgement, it reports the error back to the source, and leaves it to the source to establish a new route. While this process could use a lot of bandwidth, DSR gives each node a route cache for them to use aggressively to reduce the number of

control messages sent. If it has a cache entry for any destination request received, it uses the cached copy rather than forward the request.

DSR has the advantage that no routing tables must be kept to route a given packet, since the entire route is contained in the packet header. The caching of any initiated or overheard routing data can significantly reduce the number of control messages being sent, reducing overhead. Using only triggered updates furthers that same goal.

DSR is not scalable to large networks. The Internet Draft acknowledges that the protocol assumes that the diameter of the network is no greater than 10 hops. Additionally, DSR requires significantly more processing resources than most other protocols, because it must establish the entire route before sending the data packets.

4.3.3. Temporarily Ordered Routing Algorithm

TORA is a distributed routing protocol based on a "link reversal" algorithm. It is designed to discover routes on demand, provide multiple routes to a destination, establish routes quickly, and minimize communication overhead by localizing algorithmic reaction to topological changes when possible. Route optimality (shortest-path routing) is considered of secondary importance, and longer routes are often used to avoid the overhead of discovering newer routes. For a given destination, TORA uses a somewhat arbitrary 'height' parameter to determine the direction of a link between any two nodes.

For a node to initiate a route, it broadcasts a query to its neighbors. This is rebroadcast through the network until it reaches either the destination, or a node that has a route to the destination. This node replies with an update that contains its height with respect to the destination, which is propagated back to the sender. Each node receiving the update sets its own height to one greater than that of the neighbor that sent it. This forms a series of directed links from the sender to the destination in order of decreasing height. When a node discovers link failure, it sets its own height higher than that of its neighbors, and issues an update to that effect reversing the direction of the link between them. If it finds that it has no neighbors,

the destination is presumed lost, and it issues a clear packet to remove the invalid links from the rest of the network. Figure 4-4 shows the route creation and route maintenance for TORA.

An advantage to TORA is that it supports multiple routes to any source/destination pair. Failure or removal of one node is quickly resolved without source intervention by switching to an alternate route.

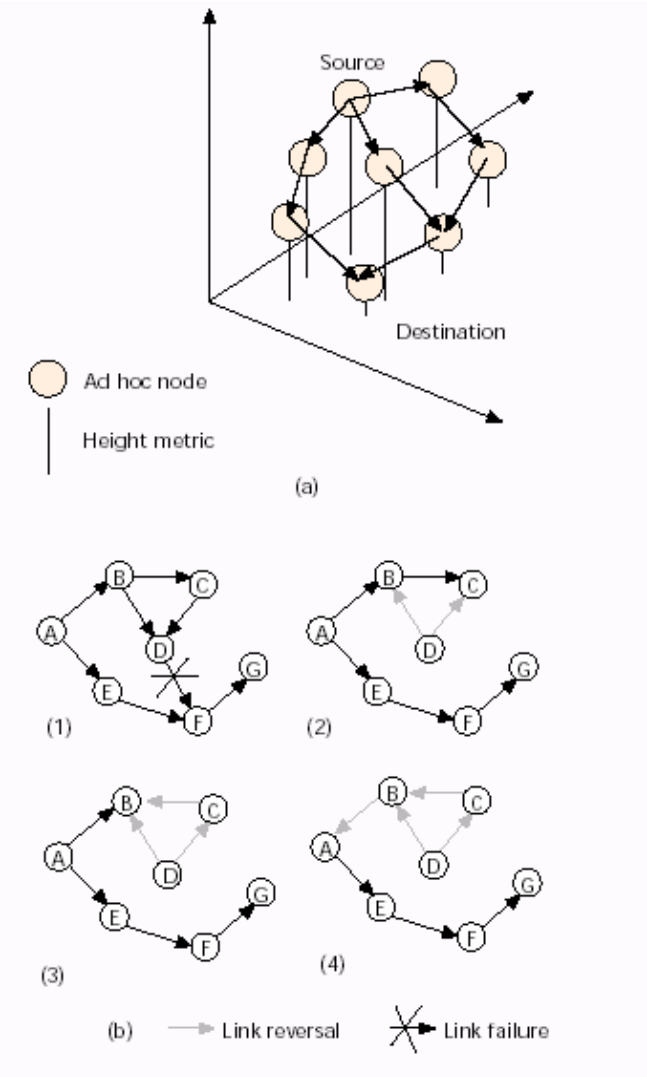


Figure 4-4 (a) Route Creation and (b) Route Maintenance in TORA [20]

4.3.4. Ad Hoc On-Demand Distance Vector Routing

AODV is a modification of the DSDV algorithm. When a source node desires to establish a communication session, it initiates a path-discovery process to locate the other node. The source node broadcasts a RREQ packet with its IP address, Broadcast ID (BrID), and the sequence number of the source and destination. Receiving nodes set the backward pointer to the source and generates a RREP unicast packet if it is the destination or contains a route to the destination with a sequence number greater than or equal to the destination sequence number contained in the original RREQ. As the RREP is routed back to the source, forward pointers are setup by the intermediate nodes in their routing tables. The deletion of a route would occur if an entry was not used within a specified lifetime. An optional feature of AODV is the use of hello messages to maintain the connectivity of neighboring nodes. The hello protocol yields a greater knowledge of the network and can improve the route discovery process.

AODV is a pure distributed on-demand approach that minimizes routing table information. However, this also means that more route requests are generated. This compounded with the periodic 'hellos' can increase routing overhead. The advantage of AODV is its use of destination numbers and replies to the first arriving RREQ implies that AODV favors the least congested route instead of the shortest route (see Figure 4-5).

AODV uses timers to monitor the utilization of routing information. A routing table entry is expired if it is not used for a period of time.

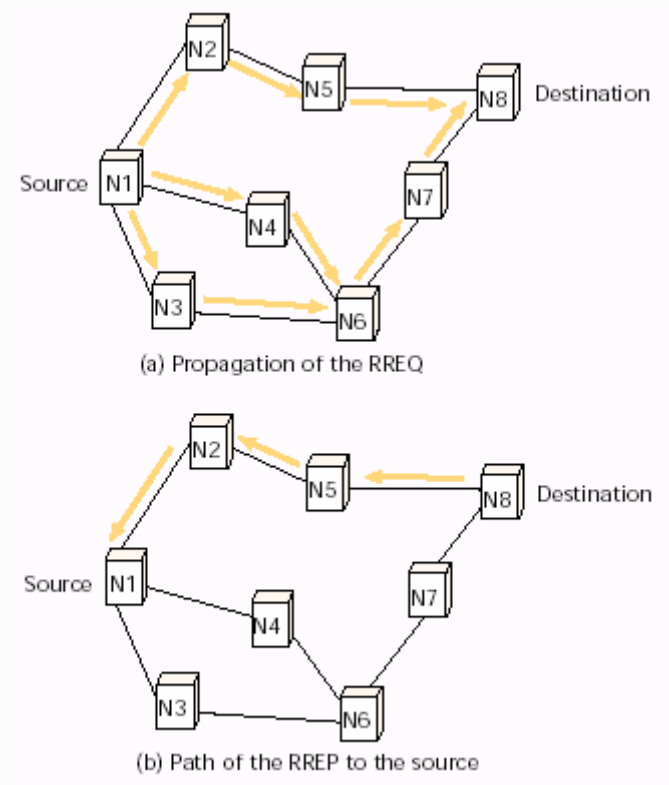


Figure 4-5 AODV Route Discovery [20]

CHAPTER 5

PERFORMANCE EVALUATION OF EDCF AND ROUTING PROTOCOLS

We have simulated and evaluated the performance of two routing protocols; DSDV and DSR, for fair (DCF) and unfair (EDCF) cases. Routing protocols were also studied in [15] and [20] and DCF and EDCF were studied in [16].

5.1. Simulation Framework

In the evaluation, we have used ns-2 environment as network simulator, [30]. Ns-2 is a very effective tool for simulating all kinds of wired and wireless networks (see Appendix A for details).

Ni Qiang [29], who is the implementer of EDCF module to ns-2, wrote a simulation code for a scenario that has two nodes and a traffic flow between them. In his scenario there is no mobility in the network. We have used his simulation code as a basis. We have modified this code for our topology and scenario. Our simulation script codes are given in Appendix B. In these codes, the lines started with "|" mark are the lines that we have written, and other lines are taken from original simulation script codes. We have two different Tool Command Language (TCL) script files; first script file mainly defines the mobility in the network. Besides mobility it defines the node configurations, and wireless parameters. Second script defines the traffics scenario between the nodes and priorities for EDCF. To run the simulation scripts, it is required to write the command line below:

ns dcf.tcl dcf

Where dcf.tcl is the mobility script and dcf which the file name is actually dcf-scenario.tcl under the same directory with mobility script is the traffic scenario script.

We have evaluated two routing protocols by changing the data rates and pause times in mobile environment. We have analyzed average drop rate (number of dropped packets / number of sent packets) and average delay in the networks as the performance metrics. We have found the dropped packets and sent packets for each traffic flow in the network. Then we have calculated the drop rate for each traffic flow by dividing number of dropped packets to number of sent packets. By taking the average of all drop rates, we have defined the average drop rate. Average delay is calculated with same way; delay is calculated for each packet. That is, packet sending time is subtracted from packet receiving time. Then we have calculated the average delay for each traffic flow. By taking the average of delays for each traffic flow, we have defined average delay.

We have used Constant Bit Rate (CBR) traffic. We have used different data rates and packet sizes for different CBR traffics between nodes. Radio propagation range is 250 meters and channel capacity is 36Mbps. Our evaluations are based on traffic *data rates* and *pause times*. Pause time actually indicates mobility of network. If a node is defined as mobile, it first moves to a destination and waits there for defined pause time. When pause time elapses, node starts to move to another destination point and waits for a pause time again and so on. If pause time is set to a high value, node waits longer. This means that a node is not as mobile as when pause time is set to a lower value. The velocity of mobile nodes in network is 10 meters per second.

5.2. Network Topologies

We have used two different topologies. The reason for using two different topologies is to make DCF/EDCF comparison and routing protocols with DCF/EDCF comparison separately, since there are different issues for DCF/EDCF comparison and routing protocols comparison. Although mobility is an important issue for routing protocols comparison, it is not an issue for DCF/EDCF

comparison. Beside this, number of traffic flows and data rates of traffic flows are important issues for DCF and EDCF comparison, they are not important for routing protocols comparison.

We first wanted to check that EDCF performs better than DCF as stated by Ni Qiang. We checked this by using the topology shown in Figure 5-1. In this topology, there is no mobility and there are traffic flows from node 1 to node 0 and node 2 to node 0.

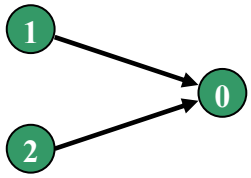


Figure 5-1 Network Topology for Comparison of DCF and EDCF

Our second topology is shown in Figure 5-2 used to compare routing protocols with DCF and EDCF. We designed such a topology since it may be a topology in a disaster situation. Since mobility is an important issue for routing protocols, we have defined node 1 as a mobile terminal. Movement of the node 1 is shown as dotted lines in Figure 5-2. In this topology, there is a bi-directional traffic flow between node 0 and node 1.

The default queue size defined in ns-2 is 50 packets, and we have used same queue size.

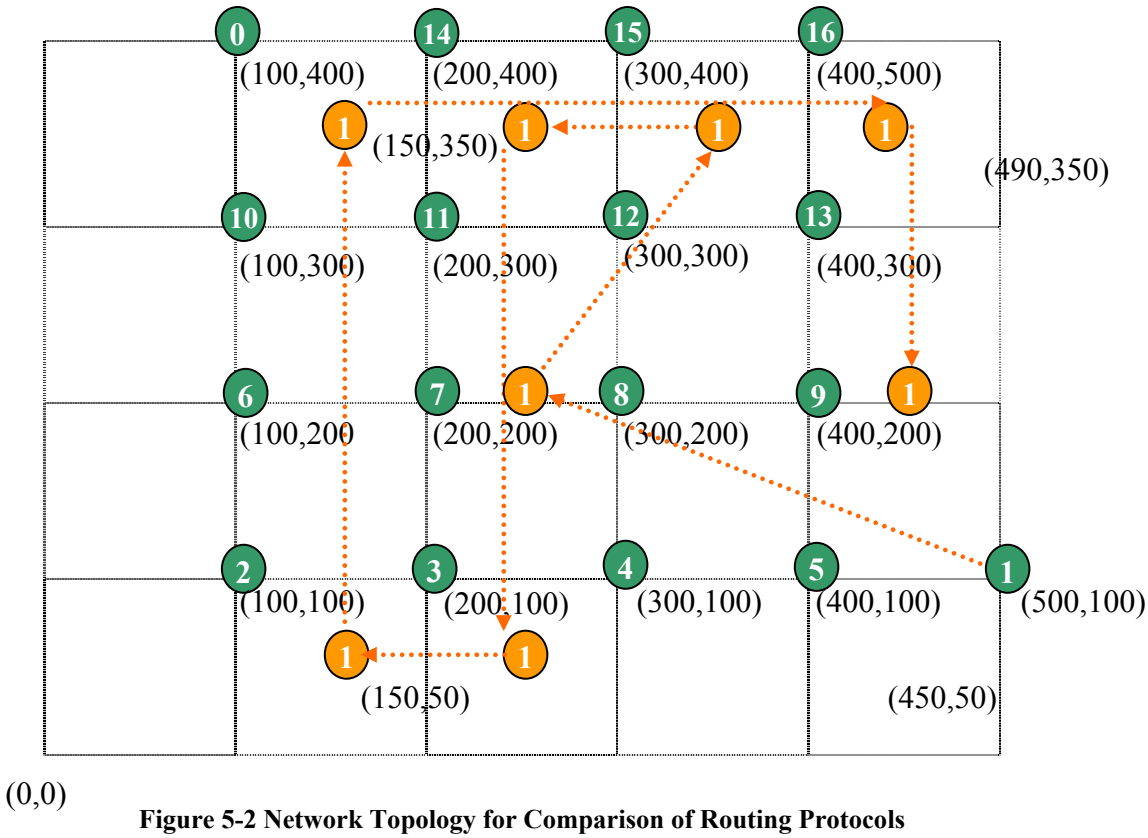


Figure 5-2 Network Topology for Comparison of Routing Protocols

5.3. Verification of the Implementation of 802.11 DCF on the Ns-2 Simulator

Before giving details of our performance evaluations, we will present how we tested whether ns-2 simulation script code operates correctly. For this purpose, we have created two nodes and a traffic flow between these two nodes as single hop traffic. We thought that, if we find the maximum data rate which causes "0" drop analytically and causes practically "0" drop on ns-2 simulation for this scenario, the simulation code is verified. We have calculated a data rate analytically and then we have applied this data rate in simulation scenario script. In the result of the simulation we have got "0" drop for this data rate. When we increase this data rate, we have started to get some drops. That was the first criteria to check whether the simulation code performs correctly. We have presented in Figure 5-3 the analytically expected and practically observed drop rates. In ns-2 simulator, the

buffer size is set to 50 packets as default and we have used the same value for buffer size.

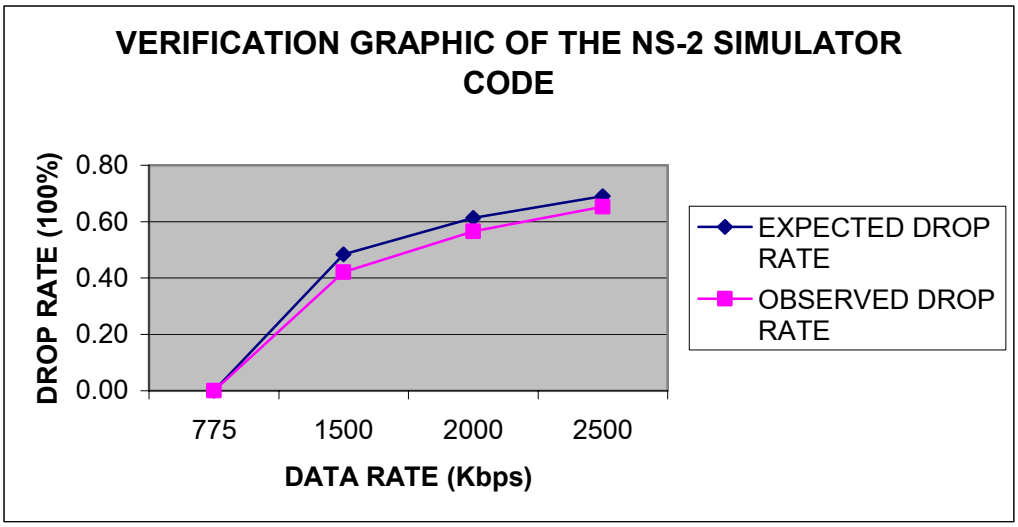


Figure 5-3 Verification Graphic of Ns-2 Simulator

The second criterion was the average delay for "0" drop on the network. We have calculated the average delay which is 26msec, for data rate giving "0" drop and get almost same delay in simulation results.

These calculations verify that the simulation script code is working correctly. After that phase we have started to consider our defined topologies in Figure 5-1 and Figure 5-2.

5.4. Performance of DCF and EDCF

For the topology in Figure 5-1, there are traffic flows from node 1 to node 0 and node 2 to node 0. There is no mobility in the network, because mobility is not an issue for comparison of DCF and EDCF.

For the evaluation of DCF and EDCF, our result is characterized by data rate, since data rate is an issue for comparison of DCF and EDCF. Performance metrics are average drop rate and average delay in the network. By increasing the data rate for

both traffic flows, drop rate is also increased, because buffer size becomes insufficient when data rate increases. To compare the performance of DCF and EDCF, we have conducted two simulations. In the first simulation both nodes uses DCF access mechanism. In second both nodes uses EDCF. The results of these two simulations are shown in Figure 5-4 and Figure 5-5. For DCF evaluation, CW_{min} for each traffic flow is set to 31 slots. For EDCF, we have used the same topology but we have changed CW_{min}; as the CW_{min} is higher for a traffic flow, the priority of that traffic flow is lower since it has to wait longer than other traffic flow which has smaller CW_{min}. CW_{min} is set to 7 slots for high priority traffic flow, which is the traffic flow from node 1 to node 0 and 31 slots for low priority traffic flow, which is the traffic flow from node 2 to node 0. The result of EDCF performance is shown in Figure 5-4 and Figure 5-5 with light colored points.

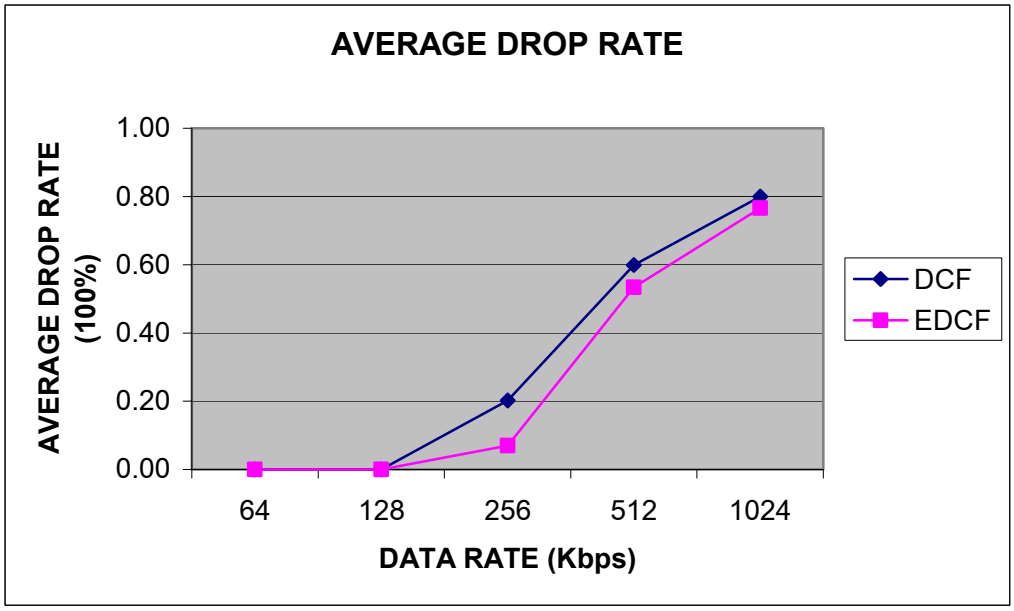


Figure 5-4 Average Drop Rate Characterized by Data Rate for Comparison of DCF and EDCF

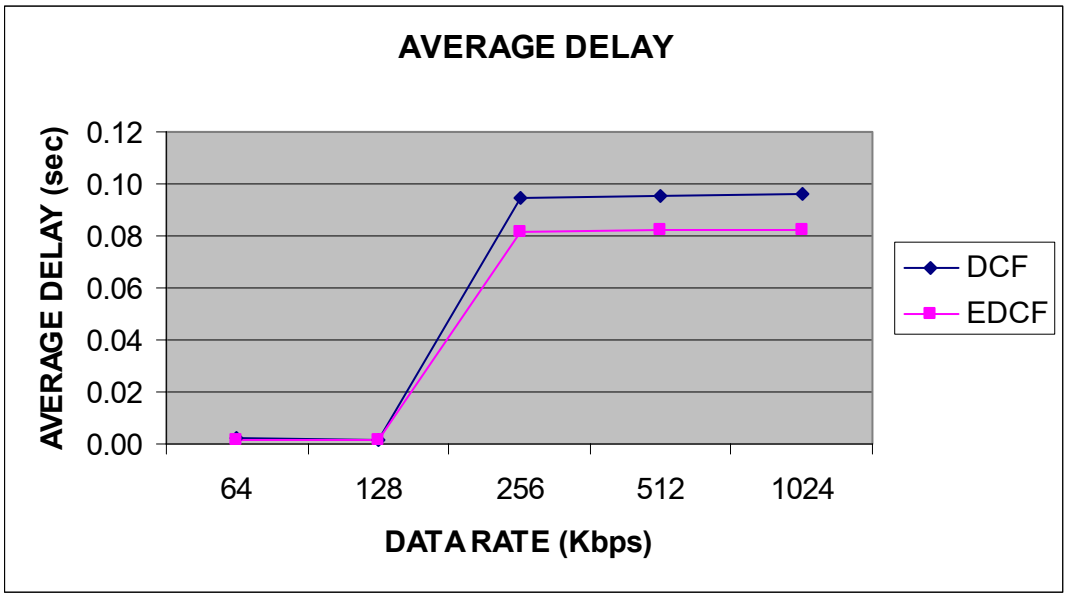


Figure 5-5 Average Delay Characterized by Data Rate for Comparison of DCF and EDCF

As shown in Figure 5-4 and Figure 5-5, EDCF has better drop rate and delay characteristics than DCF. In lower data rates, drop rate and delay are close to zero, because the bandwidth is sufficient for these data rates. Besides, queue size is sufficient for getting nearly zero drop rate and very small duration of delay. But when the data rate increases, even the bandwidth is sufficient, queue size may not be sufficient to get the same performance of lower data rates. When CBR is generated, packets are first sent to the queue and if the medium is free, packets in the queue are transmitted. But if the medium is busy, packets wait in the queue to be transmitted and CBR packets still continue to be generated by the traffic source. When the queue is full with 50 packets, every arriving packet will be dropped. When the data rate is increased, more packets are dropped. For delay, after 256Kbps, there is almost a stable delay characteristic. This is because queue of the node is going to be full for 256Kbps and higher data rates.

5.5. Performance of Routing Protocols with DCF and EDCF

For the evaluation of routing protocols, we have used the topology in Figure 5-2 and our results are characterized by data rate and pause time. The performance metrics are again average drop rate and average delay in the network. We have compared DCF and EDCF for two routing protocols; DSDV and DSR. In ns-2, the default period to update the routing table is 3 seconds. In our simulations we have changed it as 1.5 seconds. If 3 seconds were used for DSDV routing updates, this would cause too much packet drops in the networks, because the update duration is long and the stations will not be informed about any change in the network between update periods. Packets are tried to send 7 times until they reach their destination. After 7^{th} retransmission if the packets did not reach the destination, they drop. Since packets are retransmitted many times until drop and every event is written in the output trace file of the simulator, the output file size gets very large when DSDV update period is 3 seconds and difficult to open the file in the analyzer program.

For DCF evaluations, we have used CW_{min} value as 31 slots for both traffic flows. For EDCF evaluations, we have used CW_{min} value as 7 slots for high priority traffic flow which is from node 1 to node 0, and 31 slots for low priority traffic flow which is from node 0 to node 1.

In this topology node 1 is mobile node with the velocity of 10 meters per second.

5.5.1. Simulation Results According to Data Rates

Node 0's traffic data rate is fixed (64Kbps) for the topology in Figure 5-2. We have changed node 1's traffic data rate from 64Kbps to 512Kbps and we have evaluated average drop rate and delay characteristics for each data rate. Drop and delay calculations were done for DSDV and DSR with respect to DCF and EDCF.

This topology is highly mobile topology. That is, node 1 is moving along the path which is shown in Figure 5-2 with dotted lines without pausing. When it reaches the end location, it stops and stays there until the end of simulation.

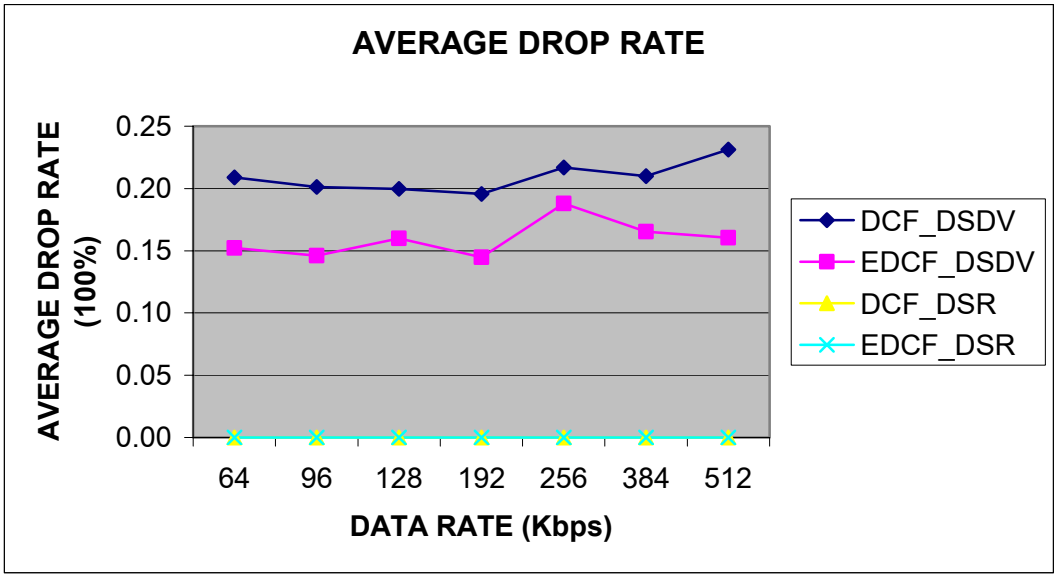


Figure 5-6 Average Drop Rate Characterized by Data Rate for Comparison of Routing Protocols

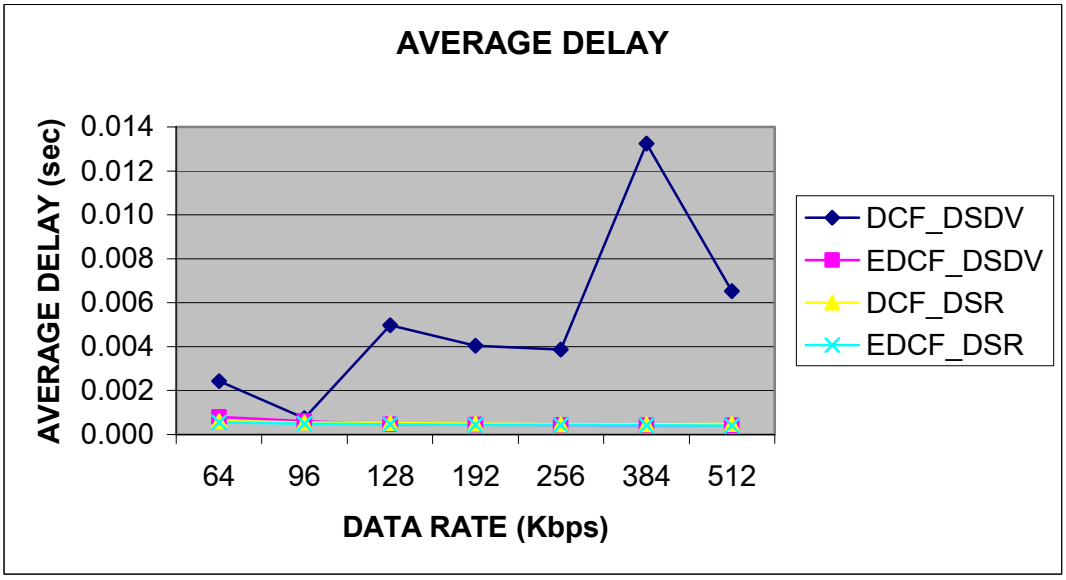


Figure 5-7 Average Delay Characterized by Data Rate for Comparison of Routing Protocols

For the drop rate, DSR routing protocol performs better than DSDV. The reason why DSDV is worse than DSR is that; this topology is a mobile topology and in mobile network, DSR routing updates works better than DSDV. In DSDV, there is an update period for each node to send its information to its neighbors. If there is any motion between these periods, since the node may go out of the range of its neighbor and there are no routing updates between the update periods, the packets may drop. But for DSR, if there is any motion, the node immediately informs the neighbor about the mobility condition and every neighbor nodes updates their routes quickly. If node goes out of the range, a new route is quickly established. Therefore, for mobile environment DSR performs better that DSDV.

For the delay, DSR again performs better. During the motion, nodes send their update information periodically in DSDV. During these periods when the node moves while transmitting packet and cannot find any route during the defined time period, the node will buffer the packet and sent the packet whenever it finds a route to the destination.

For EDCF evaluation, it performs better than DCF, because EDCF gives priorities to the traffics to access the medium quicker than DCF, and this improves the overall network performance. Ni Qiang only evaluated EDCF for single hop networks. There is no comparison for multiple hop case. When we evaluate the performance of EDCF for multiple hop networks (more than two hops), EDCF does not perform well, because EDCF gives priorities to the traffic not to packets. When there is multiple hops (more than two) routing in the network, there will be no service differentiation for the traffics after first hop. The router (intermediate) node processes the packets as in normal DCF. In our topology, there are maximum two hops. It performs well for two hops, because in the router (intermediate) nodes there will actually be more packets which belong to higher priority traffic.

5.5.2. Simulation Results According to Pause Times

As explained in section 5.1, pause time gives the information about the mobility. When the pause time reaches to zero, the network is highly mobile. When the pause

time increases, mobility decreases. In topology Figure 5-2, we have defined the pause time 20, 40, 80, 160, and 320 seconds. There is bi-directional traffic between Node 0 and Node1 with the data rate 32Kbps. We have selected the data rate small (32Kbps), because we wanted to analyze the drop rates caused by mobility (routing information) not by bandwidth or queue size limitation.

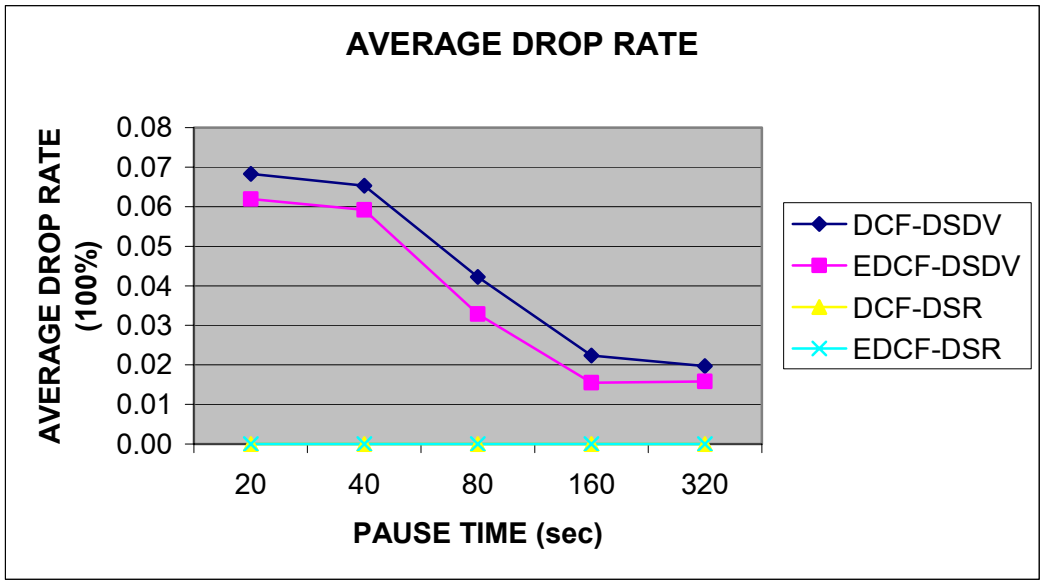


Figure 5-8 Average Drop Rate Characterized by Pause Time for Comparison of Routing Protocols

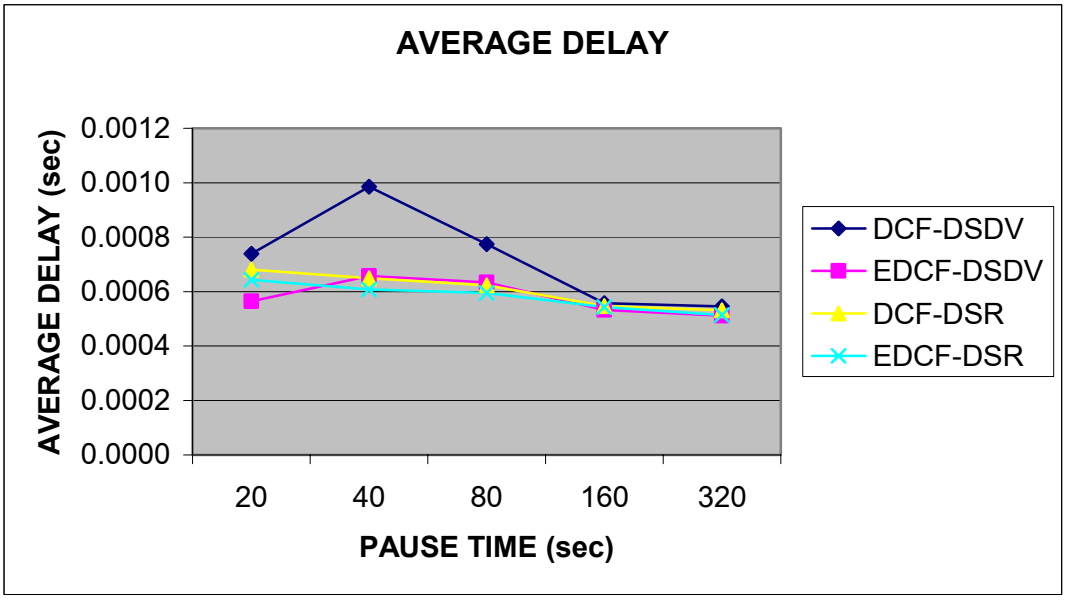


Figure 5-9 Average Delay Characterized by Pause Time for Comparison of Routing Protocols

If the mobility is high in the network, drop rate is higher for DSDV than DSR with the same reason as in section 5.5.1. When the pause time increases (mobility is less), DSDV drop rate decreases and closes to DSR drop rate, because mobility is less between the update periods.

For the delay, DCF DSR is better than DCF DSDV with the same reason in results for data rate in section 5.5.1. For EDCF, at some pause times, EDCF DSDV is better and at some pause time EDCF DSR better. In Figure 5-9, DSR generally performs better delay characteristics than DSDV, but for example at 20msec, DSDV delay characteristic is better than DSR when EDCF is used. This is because of the mobility path in the topology. Simulation duration is 500 seconds for this evaluation. When the pause time changes, the location of node 1 is different at the end of simulation for each pause time. When the end location of node 1 is different for each pause time, delay characteristics for DSDV and DSR may alter for different pause times.

In all results DSR almost gives a zero drop rate, so we cannot see the performance of EDCF when DSR is used. To analyze the performance of EDCF when DSR routing protocol is used, we have increased the data rate from 32Kbps to 300Kbps for both traffic flows from node 0 and node 1 and decrease the simulation time from 500 seconds to 100 seconds for the topology in Figure 5-2. We decreased simulation time, because for higher data rates if we set the simulation time longer, trace file uses too much space on disc (hundreds of megabytes) and this makes the file difficult to open in the analyzer program. We have analyzed the performance of DSR routing protocol when DCF and EDCF are used for the pause times of 20, 40 and 80 seconds. The results are shown in Figure 5-10 and Figure 5-11.

CWmin is set to 31 slots for both traffic flows when DCF is used, and is set to 7 slots for high priority traffic flow which is from node 1 to node 0 and 31 slots for low priority traffic flow which is from node 0 to node 1 when EDCF is used.

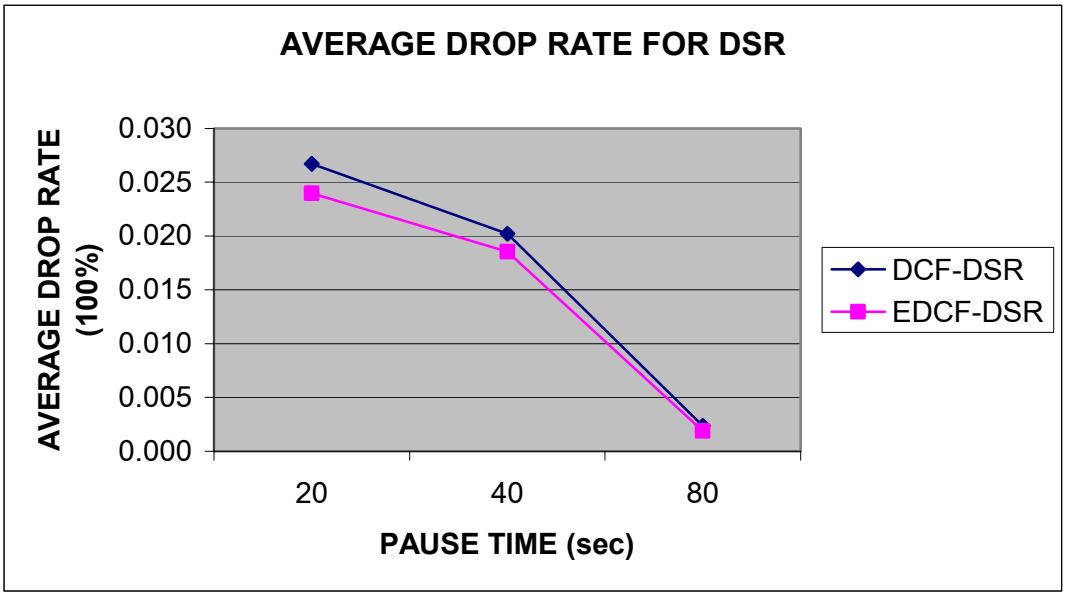


Figure 5-10 Average Drop Rate Characterized by Pause Time for DSR

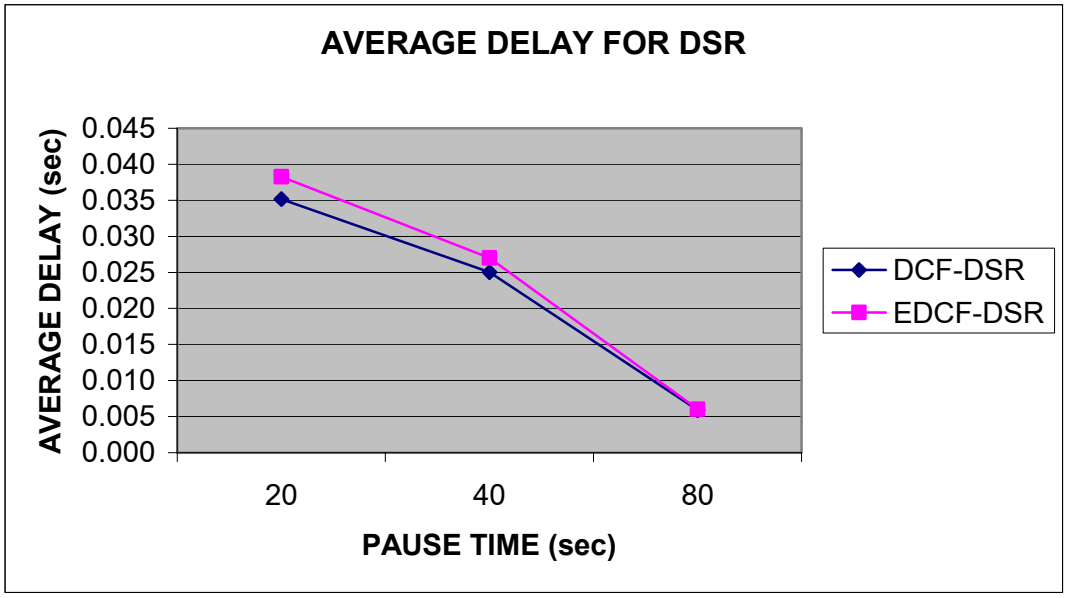


Figure 5-11 Average Delay Characterized by Pause Time for DSR

For the evaluation of DSR with DCF and EDCF, average drop rate of the network decreases when EDCF is used. This is because; we have differentiated the service by giving smaller CW_{min} to access the medium. For delay characteristics of DSR, DCF performs better than EDCF. This is because; there are less packets to drop and more packets to send when EDCF is used, that causes the packets waiting more than when DCF is used.

CHAPTER 6

CONCLUSION AND FUTURE WORK

In IEEE 802.11, basic mechanism for medium access is DCF. In DCF, all traffic flows have same right to access the medium. That is, this mechanism provides a fair sharing of medium by multiple stations. But in some cases, there can be a need that traffic flows access the medium in unfair conditions. Especially in disaster areas or military operations, service differentiation takes an important role in the communication. In EDCF, traffic flows are given different priorities to access the medium in unfair conditions. That provides higher priority traffic not to access the medium with same opportunities as lower priority traffics. Eight different priorities can be set to different traffic flows. Lower priority has higher right for medium access.

Priority mechanism can be achieved by setting different CW_{min} or using different AIFS time. If AIFS is used for priority mechanism, different waiting times are given to different traffic flows to access the medium unfairly. DIFS is the time that terminals wait before transmitting while the medium is free and AIFS is the time to wait DIFS duration plus some slot time after DIFS. So, terminals should wait for time duration of AIFS before transmitting, after sensing the medium as free. Priority access is achieved by setting lower AIFS time for higher priority traffic, since higher priority traffic can access the medium earlier than with lower priority traffic that has higher AIFS time.

CW is related to random backoff time that the terminal has to wait after DIFS time if the medium is free to transmit. The initial value of the CW is CW_{min} and

maximum value of CW is $CW_{max.}$ If CW_{min} is set different values for different traffic flow, the traffic flow that has the lowest CW_{min} has higher access right for the medium. That means the terminal that has the lowest CW_{min} will probably get the smallest random backoff time to wait. This provides the terminal with lowest CW_{min} to wait less than other terminals, so transmit earlier than other terminals.

In this study, we have investigated service differentiation by using EDCF and setting different CW_{min} for different traffic flows. EDCF was investigated for only single hop network topologies. In the nature of Ad Hoc networks, there can be multiple hops to reach the destination host, so each host acts as a router. To evaluate the EDCF performance, we have designed a topology such that there is a possibility of multihop communication between hosts but the hop count may not be more than two. If there are more than two hops, EDCF does not guarantee the QoS in the network. Because traffic is prioritized and not the packets, intermediate nodes process the packets as there is no priority mechanism. Intermediate nodes send packets by using DCF not EDCF, and DCF provides purely fair access to medium for all traffic flows. As a result, EDCF improves overall performance of the network for maximum two hops of routing.

Our second evaluation is to compare routing protocols for fair and unfair cases. For fair conditions there are some previous studies (such as [19], [20]) that analyze routing protocols in mobile environments. In this study, we have compared DSDV and DSR routing protocols for fair conditions and also evaluated DSDV and DSR behavior in unfair conditions. Because mobility is an important issue for comparison of routing protocols, we have designed our topology with different pause times that shows the degree of mobility in the network. Besides pause time we have also changed data rates for purely mobile networks which is the same topology with nearly zero pause time. So, we got two different evaluations for routing protocols comparison; one is based on pause times and other is based on data rates with pure mobile network. When DCF is used, DSR figures out better drop rate and delay characteristics than DSDV, because network changes are updated dynamically in DSR. Whereas DSR routing table is updated when there is

any change in network, DSDV routing table is updated in a period of time. If there is any change in network between periods, nodes recognize any change just in periodic update time. When EDCF is used, we have observed that DSR achieve still better performance for drop rate and delay when network is characterized by data rates. When the network is characterized by pause time, DSR figures still better characteristics than DSDV for drop rate, but there are some variations in delay. At some pause times DSDV is better, but at some pause times DSR is better because of the mobility path in the topology.

6.1. Future Work

We have investigated EDCF performance by setting different CW_{min} for different traffic flows. EDCF performance may be evaluated by using AIFS time. Another service differentiation method, such as Adaptive Enhanced Distributed Coordination Function (AEDCF) [3] can be compared with DCF. EDCF and AEDCF comparison with DCF may be done in future studies.

We have analyzed the performance of routing protocols by using the topology in Figure 5-2, since it can be a suitable topology in disaster area or in military operations. There may be some other topologies to compare routing protocols. With different topologies, various service differentiation methods such as EDCF and AEDCF may be compared with different routing protocols.

We have evaluated the performance of routing protocols with and without service differentiation under CBR traffic. The evaluations may also be done by using non-CBR traffic source, such as TCP, in future study.

REFERENCES

[1] S. Chen, "Routing Support for Providing Guaranteed End-To-End Quality Of Service", University of Illinois at Urbana-Champaign, Report UIUCDCS-R-99-2090, July 1999

[2] T. Chen, M. Gerla, J.T. Tsai, "QoS Routing Performance in Multihop, Multimedia, Wireless Network", In Proceedings of ICUPC '97

[3] L. Romdhani, N. Qiang, T. Turletti, "AEDCF: Enhanced Service Differentiaiton for IEEE 802.11 Wireless Ad Hoc Networks", Research Report, No 4544, September 2002

[4] N. Qiang, L. Romdhani, T. Turletti, I. Aad, "QoS Issues and Enhancements for IEEE 802.11 Wireless LAN", Research Report, No 4612, ISSN 0249-6399, November 2002

[5] S. Xu, T. Saadawi, "Does the IEEE 802.11 MAC Protocol Work Well in Multihop Wireless Ad Hoc Networks?", IEEE Communication Magazine, pp. 130-137, September 2001

[6] A. Lindgren, A. Almquist, O. Schel´en, "Evaluation of Quality of Service Schemes for IEEE 802.11 Wireless LANs", Proceedings of the 26th Annual IEEE Conference on Local Computer Networks (LCN 2001), November 2001

[7] S.J. Lee, W. Suu, M. Gerla, "Wireless Ad Hoc Multicast Routing with Mobility Prediction", ACM Mobile Networks and Applications, Volume 6, pp. 351-360, 2001

[8] H. Hassanein, A. Zhou, "Routing with Load Balancing in Wireless Ad Hoc Networks", Proceedings of the 4th ACM international workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems, pp. 89-96, Rome, Italy, 2001

[9] C.K. Toh, "Maximum Battery Life Routing to Support Ubiquitous Mobile Computing in Wireless Ad Hoc Networks", IEEE Communication Magazine, pp. 138-147, September 2001

[10] I. Aad, C. Castelluccia, “Introducing service differentiation into IEEE 802.11”, IEEE Symposium on Computers and Communications (ISCC), Antibes - France, July 4th-7th 2000

[11] B.P. Crow, I. Widjaja, J.G. Kim, P.T. Sakai, “IEEE 802.11 Wireless Local Area Networks”, IEEE Communication Magazine, pp. 116-126, September 1997

[12] B.P. Crow, I. Widjaja, J.G. Kim, P.T. Sakai, “Investigation of the IEEE 802.11 Medium Access Control (MAC) Sublayer Functions”, INFO-COM'97, Vol. 1 (1997) pp. 126-133

[13] K. Tang, M. Gerla, “Fair Sharing of MAC under TCP in Wireless Ad Hoc Networks”, Proceedings of IEEE MMT'99, Venice, Italy, Oct. 1999

[14] D.B. Johnson, “Routing in Ad Hoc Networks of Mobile Hosts”, Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications, Dec. 1994

[15] J. Broch, D.A. Maltz, D.B. Johnson, Y.C. Hu, J. Jetcheva, “A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols”, Proceedings of the fourth annual ACM/IEEE international conference on Mobile computing and networking, pp.85-97, October 25-30, 1998, Dallas, Texas, United States

[16] S. Mangold, S. Choi, P. May, O. Klein, G. Hiertz, L. Stibor, “IEEE 802.11e Wireless LAN for Quality of Service”, In Proceedings European Wireless (EW’ 2002), Florence, Italy, February 2002

[17] C.E. Perkins, P. Bhagwat, “Highly Dynamic Destination-Sequenced Distance-Vector Routing for Mobile Computers”, Proceedings of the conference on Communications architectures, protocols and applications, pp.234-244, August 31-September 02, 1994, London, United Kingdom

[18] M. Maleki, K. Dantu, M. Pedram, “Power-aware Source Routing Protocol for Mobile Ad Hoc Networks”, pp. 72-75, Proceedings of the 2002 International Symposium on Low Power Electronics and Design

[19] A. Patil, A. Sahoo, “Routing Protocols for Ad-Hoc Wireless Networks”

http://www.egr.msu.edu/~patilabh/CSE824_Final_Report.PDF (Last Visited)

[20] E.M. Royer, C.K. Toh, “A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks”, IEEE Personal Communications, pp 46-55, 1999

[21] A.S. Tanenbaum, “Computer Networks”, pp. 292-302, Chapter 4, Fourth Edition

[22] ANSI/IEEE Std 802.11 1999 Edition

[23] I. Mahadevan, K.M. Sivalingam, "An architecture for QoS Guarantees and Routing in Wireless/Mobile Networks", Proceedings of the first ACM international workshop on Wireless mobile multimedia, pp.11-20, October 25-30, 1998, Dallas, Texas, United States

[24] W. Stallings, "High-Speed Networks and Internets: Performance and Quality of Service", pp. 467-477, 492-500, Chapter 17, Second Edition

[25] L.M. Feeney, B. Ahlgren, A. Westerlund, "Spontaneous Networking: An Application Oriented Approach to Ad Hoc Networking", IEEE Communications Magazine, pp. 176-181, June 2001

[26] S. Chakrabarti, A. Mishra, "Quality of Service Issue in Ad Hoc Wireless Networks", IEEE Communications Magazine, pp. 142-148, February 2001

[27] D. Qiao, K.G. Shin, "A Simple Enhancement to the IEEE 802.11 DCF", 36th Hawaii International Conference on System Sciences (HICSS-36), Hawaii, January 06-09, 2003

[28] T. Franklin, "Wireless Local Area Networks", Technical Report

http://www.techlearn.ac.uk/NewDocs/Wireless%20LAN%20Tech%20Rep.pdf (Last Visited)

[29] N. Qiang Web Page http://www-sop.inria.fr/planete/qni/Research.html (Last Visited)

[30] Ns-2 Network Simulator Web Page http://www.isi.edu/nsnam/ns/ (Last Visited)

[31] Ns Manual, 2001 at http://www.isi.edu/nsnam/ns/ (Last Visited)

[32] John Ousterhout Web Page http://home.pacbell.net/ouster/ (Last Visited)

APPENDICES

A. NETWORK SIMULATOR

Ns2 is an object-oriented simulator, written in C++, with an OTcl interpreter as a front-end.

Network protocols are complex and thus mathematical analysis of their performance is not always feasible. Network simulators provide a flexible way of studying network behaviors and performance. Simulation is often the method of choice for evaluating the performance implications of policies and mechanisms on hosts and routers.

Followings are the reasons why Ns2 was chosen as the base simulator:

- Ns2 source code is available on the Internet [30]
- A lot of notes and documentation is available.
- There are mailing lists for discussions among Ns2-users, for announcements about Ns2.
- Widespread use in the academic community.

The first step in the simulation is to acquire an instance of the Simulator class. Instances of objects in classes are created in NS2 using the new methods. For example, an instance of the Simulator object is created by the following command:

set ns [new Simulator]

A network topology is realized using three primitive building blocks: nodes, links, and agents. The Simulator class has methods to configure each of these building blocks.

Nodes are created with the node Simulator method that automatically assigns a unique address to each node. Links for wired networks are created between nodes to form a network topology with the simplex-link and duplex-link methods that set up unidirectional and bidirectional links respectively.

$ns simplex-link *node1 node2 bw delay type*

will create a simplex link between node 1 and node 2 with bandwidth *bw*, delay *delay* and type *type*. *Type* may be which may be DropTail, FQ, SFQ, RED, CBQ, or CBQ/WRR etc.

Agents are the objects that actively drive the simulation. Agents can be thought of as the processes and/or transport entities that run on nodes that may be end hosts or routers. Traffic sources and sinks, and the various protocol modules are all examples of agents. Agents are created by instantiating objects in the subclass of class Agent i.e., Agent/type where type specifies the nature of the agent. For example, a TCP agent is created using the command:

set tcp [new Agent/TCP]

Once the agents are created, they are attached to nodes with the attach-agent Simulator method. Each agent is automatically assigned a port number unique across all agents on a given node. Some types of agents may have sources attached to them while others may generate their own data. For example, you can attach ftp and telnet sources to tcp agents but constant bit-rate agents generate their own data. Sources are attached to agents using the attach-source and attach-traffic agent methods.

Each object has some configuration parameters associated with it that can be modified. Configuration parameters are instance variables of the object. These parameters are initialized during startup to default values that can simply be read from the instance variables of the object. For example, $tcp set window_ returns the default window size for the tcp object. The default values for that object can be explicitly overridden by simple assignment either before a simulation begins. For example the window-size for a particular TCP session can be changed in the following manner:

$tcp set window_ 25

The simulation is started via the run method and continues until there are no more events to be processed. At this time, the original invocation of the *run* command returns and the Tcl script can exit or invoke another simulation run after possible reconfiguration. Alternatively, the simulation can be prematurely halted by invoking the *stop* command or by exiting the script with Tcl's standard exit command. For example, the commands below are used to start and stop the simulation, respectively.

$ns_ run

$ns_ at $endtime stop

Node Movements

Irrespective of the methods used to generate node movement, the topography for mobile nodes needs to be defined. It should be defined before creating mobile nodes. Normally flat topology is created by specifying the length and width of the topography using the following command:

set topo [new Topography]

$topo load_flatgrid $opt(x) $opt(y)

Where opt(x) and opt(y) are the boundaries used in simulation.

The mobile node is designed to move in a three dimensional topology. However the third dimension (Z) is not used. That is the mobile node is assumed to move always on a flat terrain with Z always equal to 0. Thus the mobile node has X, Y, Z(=0) co-ordinates that is continually adjusted as the node moves. There are two mechanisms to induce movement in mobile nodes. In the first method, starting position of the node and its future destinations may be set explicitly. These directives are normally included in a separate movement scenario file.

The start-position and future destinations for a mobile node may be set by using the following commands:

$node set X_ <x1>
$node set Y_ <y1>
$node set Z_ <z1>

$ns at $time $node setdest <x2> <y2> <speed>

At $time sec, the node would start moving from its initial position of (x1,y1) towards a destination (x2,y2) at the defined speed.

Trace Format

The trace support for wireless simulations can be old trace or revised new trace format. Because the new trace format is easier to understand, we have used new trace format. Currently new trace support is available for wireless simulations only and can be enabled by the command:

$ns use-newtrace

An example of the new trace format is shown below:

s -t 0.267662078 -Hs 0 -Hd -1 -Ni 0 -Nx 5.00 -Ny 2.00 -Nz 0.00 -Ne-1.000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 0.255 -Id -1.255 –It message -Il 32 -If 0 -Ii 0 -Iv 32

s -t 1.511681090 -Hs 1 -Hd -1 -Ni 1 -Nx 390.00 -Ny 385.00 -Nz 0.00 –Ne -1.000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 1.255 -Id -1.255 –It message -Il 32 -If 0 -Ii 1 -Iv 32

s -t 10.000000000 -Hs 0 -Hd -2 -Ni 0 -Nx 5.00 -Ny 2.00 -Nz 0.00 –Ne -1.000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 0.0 -Id 1.0 -It tcp -Il 1000 –If 2 -Ii 2 -Iv 32 -Pn tcp -Ps 0 -Pa 0 -Pf 0 -Po 0

r -t 10.000000000 -Hs 0 -Hd -2 -Ni 0 -Nx 5.00 -Ny 2.00 -Nz 0.00 –Ne -1.000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 0.0 -Id 1.0 -It tcp -Il 1000 –If 2 -Ii 2 -Iv 32 -Pn tcp -Ps 0 -Pa 0 -Pf 0 -Po 0

r -t 100.004776054 -Hs 1 -Hd 1 -Ni 1 -Nx 25.05 -Ny 20.05 -Nz 0.00 –Ne -1.000 -Nl AGT -Nw --- -Ma a2 -Md 1 -Ms 0 -Mt 800 -Is 0.0 -Id 1.0 –It tcp -Il 1020 -If 2 -Ii 21 -Iv 32 -Pn tcp -Ps 0 -Pa 0 -Pf 1 -Po 0

s -t 100.004776054 -Hs 1 -Hd -2 -Ni 1 -Nx 25.05 -Ny 20.05 -Nz 0.00 –Ne -1.000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 1.0 -Id 0.0 -It ack -Il 40 -If 2 -Ii 22 -Iv 32 -Pn tcp -Ps 0 -Pa 0 -Pf 0 -Po 0

Explanation of new trace format

The new trace format as seen above can be can be divided into the following fields :

Event type: In the traces above, the first field (as in the older trace format) describes the type of event taking place at the node and can be one of the four types:

s send
r receive
d drop
f forward

General tag: The second field starting with "-t" may stand for time or global setting

-t time
-t * (global setting)

Node property tags: This field denotes the node properties like node-id, the level at which tracing is being done like agent, router or MAC. The tags start with a leading "-N" and are listed as below:

-Ni: node id
-Nx: node’s x-coordinate
-Ny: node’s y-coordinate
-Nz: node’s z-coordinate

-Ne: node energy level

-Nl: trace level, such as AGT, RTR, MAC

-Nw: reason for the event. The different reasons for dropping a packet are given below:

"END" DROP_END_OF_SIMULATION

"COL" DROP_MAC_COLLISION

"DUP" DROP_MAC_DUPLICATE

"ERR" DROP_MAC_PACKET_ERROR

"RET" DROP_MAC_RETRY_COUNT_EXCEEDED

"STA" DROP_MAC_INVALID_STATE

"BSY" DROP_MAC_BUSY

"NRTE" DROP_RTR_NO_ROUTE i.e no route is available.

"LOOP" DROP_RTR_ROUTE_LOOP i.e there is a routing loop

"TTL" DROP_RTR_TTL i.e TTL has reached zero.

"TOUT" DROP_RTR_QTIMEOUT i.e packet has expired.

"CBK" DROP_RTR_MAC_CALLBACK

"IFQ" DROP_IFQ_QFULL i.e no buffer space in IFQ.

"ARP" DROP_IFQ_ARP_FULL i.e dropped by ARP

"OUT" DROP_OUTSIDE_SUBNET i.e dropped by base stations on receiving routing updates from nodes out-side its domain.

Packet information at IP level: The tags for this field start with a leading "-I" and are listed along with their explanations as following:

-Is: source address.source port number

-Id: dest address.dest port number

-It: packet type

-Il: packet size

-If: flow id

-Ii: unique id

-Iv: ttl value

Next hop info: This field provides next hop info and the tag starts with a leading "-H".

-Hs: id for this node

-Hd: id for next hop towards the destination.

Packet info at MAC level: This field gives MAC layer information and starts with a leading "-M" as shown below:

-Ma: duration
-Md: dst's ethernet address
-Ms: src's ethernet address
-Mt: ethernet type

Packet info at Application level: The packet information at application level consists of the type of application like ARP, TCP, the type of adhoc routing protocol like DSDV, DSR, AODV etc being traced. This field consists of a leading "-P" and list of tags for different application is listed as below:

-P arp: Address Resolution Protocol. Details for ARP are given by the following tags:

-Po: ARP Request/Reply
-Pm: src mac address
-Ps: src address
-Pa: dst mac address
-Pd: dst address

-P dsr: This denotes the adhoc routing protocol called Dynamic source routing. Information on DSR is represented by the following tags:

-Pn: how many nodes traversed
-Pq: routing request flag
-Pi: route request sequence number
-Pp: routing reply flag
-Pl: reply length
-Pe: src of srcrouting->dst of the source routing
-Pw: error report flag?
-Pm: number of errors
-Pc: report to whom

-Pb: link error from linka->linkb

-P cbr: Constant bit rate. Information about the CBR application is represented by the following tags:

-Pi: sequence number
-Pf: how many times this pkt was forwarded
-Po: optimal number of forwards

-P tcp: Information about TCP flow is given by the following subtags:

-Ps: seq number
-Pa: ack number
-Pf: how many times this pkt was forwarded
-Po: optimal number of forwards

NETWORK ANIMATOR

Nam is an animation tool for viewing network simulation traces and real world packet trace data. It supports topology layout, packet level animation, and various data inspection tools. The first step to use nam is to produce a trace file. The trace file should contain topology information, for example nodes, links and packet traces. During an NS2 simulation, user can produce topology configurations, layout information and packet traces using tracing events in NS2. When the trace file is generated, it is ready to be animated by nam. Upon startup, nam will read the trace file and create topology. Nam provides control over many aspects of animation through its user interface.

NAM User Interface

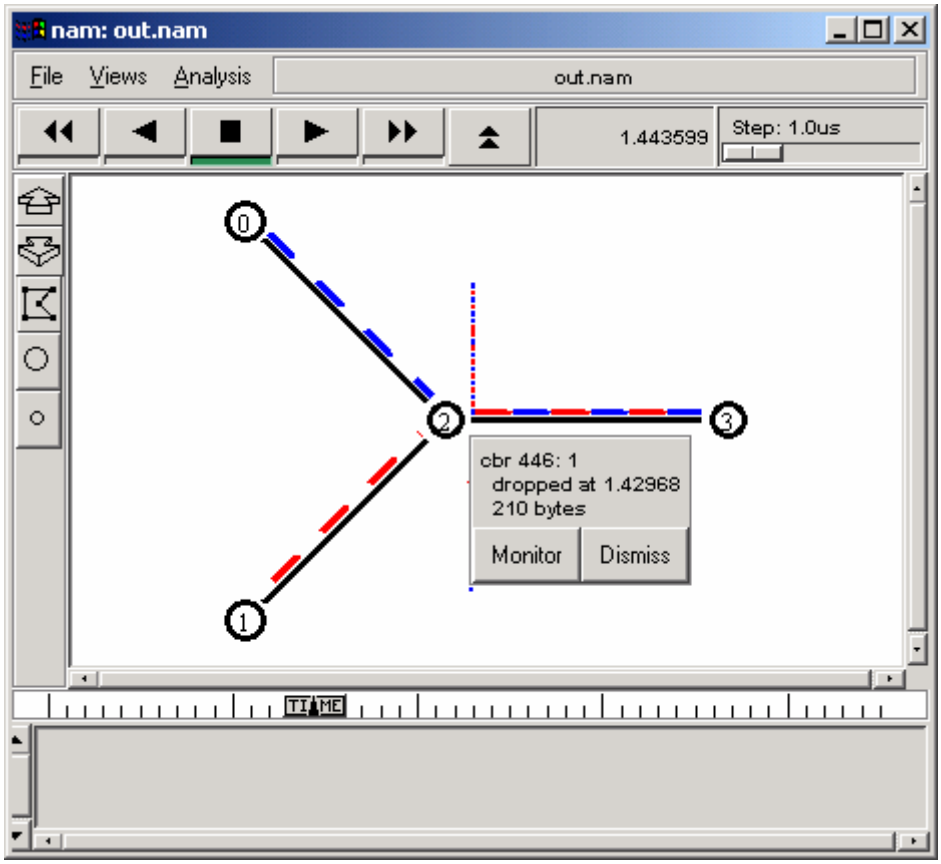


Figure A-1 Nam Animation Window

Starting up nam will create the nam console window.

For the nam console there are 'File' and 'Help' menus. Under the 'File' there is a 'New' command for creating a ns topology using the nam editor, an 'Open' command which allows you to open existing trace file, a 'WinList' command that popup a window which shows the names of all currently opened trace files, and a 'Quit' command which exits nam. The 'Help' menu contains a very limited popup help screen and a command to show version and copyright information.

When a trace file is tried to be opened, an animation window will appear as shown in Figure A-1. By pressing the forward play button (>), animation will be played. Current time for the simulation can be followed in the time scroll bar just under the animation area. Time is also viewed as numerical on the time label just above the animation area. To pause the animation, it is enough to press the pause button (Square).

By clicking the left button on any object (node, packet ...), an information windows will appear related with the object.

B. TOOL COMMAND LANGUAGE

Tool Command Language (Tcl) is an interpreted script language developed by Dr. John Ousterhout at the University of California, Berkeley, [32].

Tcl operates within its own shell, the command for invoking the Tcl shell is *tclsh*.

A significant advantage to the Tcl language and its applications is the fact that it is fully compatible with the C programming language, and Tcl libraries can be incorporated directly into C programs.

Within the Tcl shell, Tcl commands can be executed interactively, or commands can be placed in a script file and executed all at once.

Tcl script can be run by the Tcl shell command tclsh. Eg:

```
% tclsh myscript.tcl
```

The standard tcl shell offers a broad set of commands (flow control, arithmetic and logical expressions, string manipulation, file, network and serial port io, ...). At the same time, each application can add its own specific features to the shell which makes tcl highly reusable for many different purposes in many different applications.

Shell Basics

A string is delimited by double quotes, braces or square brackets, but only strings delimited by double quotes or square brackets exhibit variable and command substitution. When a string consists of a single word, i.e. when it contains no white-space, the double quotes can be omitted.

```
puts "i equals $i"
puts {i equals $i}
put [format "%s equals %d" I $i]
```

Variable substitution is obtained by placing a dollar sign in front of the variable's name.

```
puts $i
```

Command substitution is obtained by placing the command between square brackets.

```
puts [format %04d $i]
```

The following rules define the syntax and semantics of the Tcl language:

[1] A Tcl script is a string containing one or more commands. Semi-colons and newlines are command separators unless quoted as described below. Close brackets are command terminators during command substitution (see below) unless quoted.

[2] A command is evaluated in two steps. First, the Tcl interpreter breaks the command into words and performs substitutions as described below. These substitutions are performed in the same way for all commands. The first word is used to locate a command procedure to carry out the command, then all of the words of the command are passed to the command procedure. The command procedure is free to interpret each of its words in any way it likes, such as an integer, variable name, list, or Tcl script. Different commands interpret their words differently.

[3] Words of a command are separated by white space (except for newlines, which are command separators).

[4] If the first character of a word is double-quote (``"'') then the word is terminated by the next double-quote character. If semi-colons, close brackets, or white space characters (including newlines) appear between the quotes then they are treated as ordinary characters and included in the word. Command substitution, variable substitution, and backslash substitution are performed on the characters between the quotes as described below. The double-quotes are not retained as part of the word.

[5] If the first character of a word is an open brace (``{'') then the word is terminated by the matching close brace (``}''). Braces nest within the word: for each additional open brace there must be an additional close brace (however, if an open brace or close brace within the word is quoted with a backslash then it is not counted in locating the matching close brace). No substitutions are performed on the characters between the braces except for backslash-newline substitutions described below, nor do semi-colons, newlines, close brackets, or white space receive any special interpretation. The word will consist of exactly the characters between the outer braces, not including the braces themselves.

[6] If a word contains an open bracket (``['') then Tcl performs command substitution. To do this it invokes the Tcl interpreter recursively to process the characters following the open bracket as a Tcl script. The script may contain any number of commands and must be terminated by a close bracket (``]''). The result of the script (i.e. the result of its last command) is substituted into the word in place of the brackets and all of the characters between them. There may be any number of command substitutions in a single word. Command substitution is not performed on words enclosed in braces.

[7] If a word contains a dollar-sign (``$'') then Tcl performs variable substitution: the dollar-sign and the following characters are replaced in the word by the value of a variable. Variable substitution may take any of the following forms:

```
$name
```

Name is the name of a scalar variable; the name is a sequence of one or more characters that are a letter, digit, or underscore.

```
$name(index)
```

Name gives the name of an array variable and index gives the name of an element within that array. Name must contain only letters, digits, and underscores, but may be an empty string. Command substitutions, variable substitutions, and backslash substitutions are performed on the characters of index.

```
${name}
```

Name is the name of a scalar variable. It may contain any characters whatsoever except for close braces.

There may be any number of variable substitutions in a single word. Variable substitution is not performed on words enclosed in braces.

[8] If a hash character (``#'') appears at a point where Tcl is expecting the first character of the first word of a command, then the hash character and the characters that follow it, up through the next newline, are treated as a comment and ignored. The comment character only has significance when it appears at the beginning of a command.

[9] Each character is processed exactly once by the Tcl interpreter as part of creating the words of a command. For example, if variable substitution occurs then no further substitutions are performed on the value of the variable; the value is inserted into the word verbatim. If command substitution occurs then the nested command is processed entirely by the recursive call to the Tcl interpreter; no substitutions are performed before making the recursive call and no additional substitutions are performed on the result of the nested script.

[10] Substitutions do not affect the word boundaries of a command. For example, during variable substitution the entire value of the variable becomes part of a single word, even if the variable's value contains spaces.

DCF MOBILITY SCRIPT

```
# ======================================================================
#
#
# Main tcl code to access various wireless lan scenarios
#
# ======================================================================
#


# ======================================================================
#
# Set defaults; its important to set defaults before parsing
# command line arguments, so that defaults can be overridden

set mode          DCF                     ;# mode for this test
DCF/PCF
set pcf_period    0                       ;# CFP period for pcf mode
set pcf_duration  0                       ;# CFP duration for pcf
mode
|set endtime           150.0                    ;# duration of the
simulation
set title_        ""                      ;# default title for the
scenario
set idea_mac      0                       ;# default cleverness of
mac
                                          ;# 0 - normal
                                          ;# 1 - persistent slotting
                                          ;# 0 - PDCF

set val(chan)     Channel/WirelessChannel   ;# Channel Type
set val(prop)     Propagation/TwoRayGround  ;# radio-prop. 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(rp)           DSDV                      ;# routing
protocol
|Agent/DSDV set min_update_periods_ 1.5
|set val(x)       600
|set val(y)       600

Phy/WirelessPhy set CPThresh_ 2000
Phy/WirelessPhy set per_      0.0
Mac/802_11 set bandwidth_     36Mb
MAC_MIB set RTSThreshold_     3000
MAC_MIB set ShortRetryLimit_  7
MAC_MIB set LongRetryLimit_   4
PHY_MIB set MinimumBandwidth_ 6Mb
```

```
#########################################################################
#
# Usage

if {$argc < 1} {
      puts stderr "usage: ns $argv0 <scenario> \[list of options\]"
      puts stderr "       options:"
      puts stderr "             varName=varValue"
      puts stderr "             vectorName=value0,value1,value2..."
      puts stderr "       option example:"
      puts stderr "             num_nodes=17"
      exit 1;
}


#########################################################################
#
# Load up the scenario; allow it to override the above defaults

set scenario [lindex $argv 0]
puts "% set scenario $scenario"
puts "Running scenario $scenario"
source $scenario-scenario.tcl


#########################################################################
#
# Parse remaining command line arguments to override any defaults

for {set i 1} {$i < $argc} {incr i} {
      scan [lindex $argv $i] {%[a-zA-Z0-9/_:]=%[a-zA-Z0-9_.,]} var
value
      if {![info exists value]} {
            puts stderr "BAD ARGUMENTS to main.tcl!!"
            exit 1;
      }

      if {[string match *::* $var]} {
            scan $var {%[a-zA-Z0-9/_]::%[a-zA-Z0-9/_]} class var
      } else {
            if {[string match CW* $var] || [string match S* $var]
|| [string match MinimumBandwidth_ $var]} {
                  set class PHY_MIB
            }
            if {[string match RTSThreshold_ $var] || [string match
*RetryLimit_ $var]} {
                  set class MAC_MIB
            }
            if {[string match *levels_ $var]} {
                  set class PLevels
            }
            if {[string match bandwidth_ $var]} {
                  set class Mac/802_11
            }
            if {[string match CPThresh_ $var] || [string match per_
$var]} {
                  set class Phy/WirelessPhy
            }
```

```
	}

	scan $value {%[a-zA-Z0-9_.],%[a-zA-Z0-9_.,]} value tail
	if {[info exists tail]} {
		for {set n 0} {[info exists tail]} {incr n} {
			set var_n [format "%s_%d" $var $n]
			if {[info exists class]} {
				puts "% $class set $var_n $value"
				$class set $var_n $value
			} else {
				puts "% set $var_n $value"
				set $var_n $value
			}

			set value $tail
			unset tail

			scan $value {%[a-zA-Z0-9_.],%[a-zA-Z0-9_.,]}
value tail
		}

		set var_n [format "%s_%d" $var $n]
		if {[info exists class]} {
			puts "% $class set $var_n $value"
			$class set $var_n $value
		} else {
			puts "% set $var_n $value"
			set $var_n $value
		}

		unset var_n
	} else {
		if {[info exists class]} {
			puts "% $class set $var $value"
			$class set $var $value
		} else {
			if {$var == "phy"} {
				if {$value == "11b"} {
					puts "% setting PHY parameters to
802.11b"
					Mac/802_11 set bandwidth_     11Mb
					PHY_MIB set MinimumBandwidth_ 1Mb
					PHY_MIB set SlotTime_         0.000020
					PHY_MIB set SIFS_       0.000010
					PHY_MIB set CWMin_0           31
				} elseif {$value != "11a"} {
					puts stderr "BAD ARGUMENTS to
main.tcl!!"
					exit 1;
				}
			} else {
				puts "% set $var $value"
				set $var $value
			}
		}
	}

	if {[info exists class]} {
```

```
            unset class
      }
      unset var value
}


#############################################################################
#
# Setup other vars

puts "Creating $num_nodes nodes"

# cookie appended to scenario name for filenames that are created
if {[info exists cookie]} {
      set outfile_ $scenario.$cookie
} else {
      set outfile_ $scenario
}

# num_bss_nodes describes the number of nodes in the BSS; the
# remaining nodes are ADHOC; default: all nodes are in the BSS
if {![info exists num_bss_nodes]} {
      set num_bss_nodes $num_nodes
}


#############################################################################
#
# Primary simulation objects

set ns_            [new Simulator]
$ns_ use-newtrace

|ns-random 0

set tracefd [open $outfile_.tr w]
$ns_ trace-all $tracefd

set namtrace      [open $outfile_.nam w]
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)

# set up topography object
set topo       [new Topography]

$topo load_flatgrid $val(x) $val(y)

# Create God
create-god $num_nodes


#############################################################################
#
# Generic WLAN simulation objects

# Create channel #1
set chan_1_ [new $val(chan)]

# Configure nodes to be "attached" to channel #1
```

```
$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) \
             -topoInstance $topo \
             -agentTrace ON \
             -routerTrace ON \
             -macTrace ON \
             -movementTrace OFF \
             -channel $chan_1_

# Create nodes with IFQ send tracing enabled,
# random motion disabled
for {set i 0} {$i < $num_nodes} {incr i} {

      set node_($i) [$ns_ node]
      $node_($i) add-ifq-send-trace
      $node_($i) random-motion 0

      $ns_ initial_node_pos $node_($i) 20

}

|        $node_(0) set X_ 100
|        $node_(0) set Y_ 400
|        $node_(0) set Z_ 0.0

|        $node_(1) set X_ 500
|        $node_(1) set Y_ 100
|        $node_(1) set Z_ 0.0

|        $node_(2) set X_ 100
|        $node_(2) set Y_ 100
|        $node_(2) set Z_ 0.0

|        $node_(3) set X_ 200
|        $node_(3) set Y_ 100
|        $node_(3) set Z_ 0.0

|        $node_(4) set X_ 300
|        $node_(4) set Y_ 100
|        $node_(4) set Z_ 0.0

|        $node_(5) set X_ 400
|        $node_(5) set Y_ 100
|        $node_(5) set Z_ 0.0

|        $node_(6) set X_ 100
|        $node_(6) set Y_ 200
|        $node_(6) set Z_ 0.0

|        $node_(7) set X_ 200
|        $node_(7) set Y_ 200
|        $node_(7) set Z_ 0.0
```

```
|        $node_(8) set X_ 300
|        $node_(8) set Y_ 200
|        $node_(8) set Z_ 0.0

|        $node_(9) set X_ 400
|        $node_(9) set Y_ 200
|        $node_(9) set Z_ 0.0

|        $node_(10) set X_ 100
|        $node_(10) set Y_ 300
|        $node_(10) set Z_ 0.0

|        $node_(11) set X_ 200
|        $node_(11) set Y_ 300
|        $node_(11) set Z_ 0.0

|        $node_(12) set X_ 300
|        $node_(12) set Y_ 300
|        $node_(12) set Z_ 0.0

|        $node_(13) set X_ 400
|        $node_(13) set Y_ 300
|        $node_(13) set Z_ 0.0

|        $node_(14) set X_ 200
|        $node_(14) set Y_ 400
|        $node_(14) set Z_ 0.0

|        $node_(15) set X_ 300
|        $node_(15) set Y_ 400
|        $node_(15) set Z_ 0.0

|        $node_(16) set X_ 400
|        $node_(16) set Y_ 400
|        $node_(16) set Z_ 0.0

|$ns_ at 0.01000 "$node_(0) setdest 100 400 20"
|$ns_ at 0.01000 "$node_(1) setdest 500 100 20"
|$ns_ at 0.01000 "$node_(2) setdest 100 100 20"
|$ns_ at 0.01000 "$node_(3) setdest 200 100 20"
|$ns_ at 0.01000 "$node_(4) setdest 300 100 20"
|$ns_ at 0.01000 "$node_(5) setdest 400 100 20"
|$ns_ at 0.01000 "$node_(6) setdest 100 200 20"
|$ns_ at 0.01000 "$node_(7) setdest 200 200 20"
|$ns_ at 0.01000 "$node_(8) setdest 300 200 20"
|$ns_ at 0.01000 "$node_(9) setdest 400 200 20"
|$ns_ at 0.01000 "$node_(10) setdest 100 300 20"
|$ns_ at 0.01000 "$node_(11) setdest 200 300 20"
|$ns_ at 0.01000 "$node_(12) setdest 300 300 20"
|$ns_ at 0.01000 "$node_(13) setdest 400 300 20"
|$ns_ at 0.01000 "$node_(14) setdest 200 400 20"
|$ns_ at 0.01000 "$node_(15) setdest 300 400 20"
|$ns_ at 0.01000 "$node_(16) setdest 400 400 20"

|$ns_ at 6.00000 "$node_(1) setdest 250 250 10"
|$ns_ at 35.00000 "$node_(1) setdest 350 350 10"
|$ns_ at 45.00000 "$node_(1) setdest 250 350 10"
```

```
|$ns_ at 50.00000 "$node_(1) setdest 250 150 10"
|$ns_ at 75.00000 "$node_(1) setdest 150 150 10"
|$ns_ at 81.00000 "$node_(1) setdest 150 350 10"
|$ns_ at 103.00000 "$node_(1) setdest 350 350 10"
|$ns_ at 118.00000 "$node_(1) setdest 350 200 10"

#####################################################################
#
# Scenario specific simulation objects (agents etc.)

create_scenario


#####################################################################
#
# Modifications to the mac and/or other stuff dictated by the
scenario!

for {set i 0} {$i < $num_nodes} {incr i} {
      if {$idea_mac} {
            $node_($i) setMac IdeaMac $idea_mac
            puts "% node_($i) setMac IdeaMac $idea_mac"
      }
      for {set p 0} {$p < [PLevels set plevels_]} {incr p} {
            set difs_p [format "difs_%d" $p]
            if {[info exists $difs_p]} {
                  set v [set $difs_p]
                  $node_($i) setMac difs $p $v
                  puts "% node_($i) setMac difs $p $v"
            }
      }
      if {$mode == "PCF"} {
            $node_($i) CFP $pcf_period $pcf_duration
            if {$i == 0} {
                  puts "% node_($i) CFP $pcf_period $pcf_duration"
            }
      }
}

#####################################################################
#
# End of simulation conditions/operations

for {set i 0} {$i < $num_nodes } {incr i} {
    $ns_ at $endtime "$node_($i) reset";
}
$ns_ at $endtime "do_stop"

set delta 0.01
set endtime [expr $endtime + $delta]
$ns_ at $endtime "do_halt"

proc do_stop {} {
      global ns_ tracefd
|     global ns_ namtrace
      $ns_ flush-trace
      close $tracefd
|     close $namtrace
```

```
}

proc do_halt {} {
      global ns_
      puts "NS EXITING..."
      $ns_ halt
}


#########################################################################
#
# Start up the simulation

puts "Starting Simulation..."
$ns_ run
```

EDCF MOBILITY SCRIPT

```
#####################################################################
#
#
# Main tcl code to access various wireless lan scenarios
#
#####################################################################
#


#####################################################################
#
# Set defaults; its important to set defaults before parsing
# command line arguments, so that defaults can be overridden

set mode          DCF               ;# mode for this test DCF/PCF
set pcf_period    0                 ;# CFP period for pcf mode
set pcf_duration  0                 ;# CFP duration for pcf mode
|set endtime            150.0             ;# duration of the
simulation
set title_        ""                ;# default title for the
scenario
set idea_mac      1                 ;# default cleverness of mac
                                    ;#    0 - normal
                                    ;#    1 - persistent slotting
                                    ;#    0 - PDCF

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(rp)            DSDV                      ;# routing
protocol
|Agent/DSDV set min_update_periods_ 1.5
|set val(x)       600
|set val(y)       600

Phy/WirelessPhy set CPThresh_ 2000
Mac/802_11 set bandwidth_     36Mb
MAC_MIB set RTSThreshold_     3000
MAC_MIB set ShortRetryLimit_  7
MAC_MIB set LongRetryLimit_   4
PHY_MIB set MinimumBandwidth_ 6Mb

#####################################################################
#
# Usage

if {$argc < 1} {
```

```
	puts stderr "usage: ns $argv0 <scenario> \[list of options\]"
	puts stderr "	options:"
	puts stderr "		varName=varValue"
	puts stderr "		vectorName=value0,value1,value2..."
	puts stderr "	option example:"
	puts stderr "		num_nodes=17"
	exit 1;
}


#########################################################################
#
# Load up the scenario; allow it to override the above defaults

set scenario [lindex $argv 0]
puts "% set scenario $scenario"
puts "Running scenario $scenario"
source $scenario-scenario.tcl


#########################################################################
#
# Parse remaining command line arguments to override any defaults

for {set i 1} {$i < $argc} {incr i} {
	scan [lindex $argv $i] {%[a-zA-Z0-9/_:]=%[a-zA-Z0-9_.,]} var
value
	if {![info exists value]} {
		puts stderr "BAD ARGUMENTS to main.tcl!!"
		exit 1;
	}

	if {[string match *::* $var]} {
		scan $var {%[a-zA-Z0-9/_]::%[a-zA-Z0-9/_]} class var
	} else {
		if {[string match CW* $var] || [string match S* $var]
|| [string match MinimumBandwidth_ $var]} {
			set class PHY_MIB
		}
		if {[string match RTSThreshold_ $var] || [string match
*RetryLimit_ $var]} {
			set class MAC_MIB
		}
		if {[string match *levels_ $var]} {
			set class PLevels
		}
		if {[string match bandwidth_ $var]} {
			set class Mac/802_11
		}
		if {[string match CPThresh_ $var] || [string match per_
$var]} {
			set class Phy/WirelessPhy
		}
	}

	scan $value {%[a-zA-Z0-9_.],%[a-zA-Z0-9_.,]} value tail
	if {[info exists tail]} {
		for {set n 0} {[info exists tail]} {incr n} {
```

```
                set var_n [format "%s_%d" $var $n]
                if {[info exists class]} {
                      puts "% $class set $var_n $value"
                      $class set $var_n $value
                } else {
                      puts "% set $var_n $value"
                      set $var_n $value
                }

                set value $tail
                unset tail

                scan $value {%[a-zA-Z0-9_.],%[a-zA-Z0-9_.,]}
value tail
          }

          set var_n [format "%s_%d" $var $n]
          if {[info exists class]} {
                puts "% $class set $var_n $value"
                $class set $var_n $value
          } else {
                puts "% set $var_n $value"
                set $var_n $value
          }

          unset var_n
    } else {
          if {[info exists class]} {
                puts "% $class set $var $value"
                $class set $var $value
          } else {
                if {$var == "phy"} {
                      if {$value == "11b"} {
                            puts "% setting PHY parameters to
802.11b"
                            Mac/802_11 set bandwidth_     11Mb
                            PHY_MIB set MinimumBandwidth_ 1Mb
                            PHY_MIB set SlotTime_         0.000020
                            PHY_MIB set SIFS_       0.000010
                            PHY_MIB set CWMin_0           31
                      } elseif {$value != "11a"} {
                            puts stderr "BAD ARGUMENTS to
main.tcl!!"
                            exit 1;
                      }
                } else {
                      puts "% set $var $value"
                      set $var $value
                }
          }
    }

    if {[info exists class]} {
          unset class
    }
    unset var value
}
```

```
#############################################################
#
# Setup other vars

puts "Creating $num_nodes nodes"

# cookie appended to scenario name for filenames that are created
if {[info exists cookie]} {
      set outfile_ $scenario.$cookie
} else {
      set outfile_ $scenario
}

# num_bss_nodes describes the number of nodes in the BSS; the
# remaining nodes are ADHOC; default: all nodes are in the BSS
if {![info exists num_bss_nodes]} {
      set num_bss_nodes $num_nodes
}


#############################################################
#
# Primary simulation objects

set ns_           [new Simulator]
$ns_ use-newtrace

|ns-random 0

set tracefd [open $outfile_.tr w]
$ns_ trace-all $tracefd

set namtrace      [open $outfile_.nam w]
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)

# set up topography object
set topo       [new Topography]

$topo load_flatgrid $val(x) $val(y)

# Create God
create-god $num_nodes


#############################################################
#
# Generic WLAN simulation objects

# Create channel #1
set chan_1_ [new $val(chan)]

# Configure nodes to be "attached" to channel #1
$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) \
            -topoInstance $topo \
            -agentTrace ON \
            -routerTrace ON \
            -macTrace ON \
            -movementTrace OFF \
            -channel $chan_1_

# Create nodes with IFQ send tracing enabled,
# random motion disabled
for {set i 0} {$i < $num_nodes} {incr i} {

      set node_($i) [$ns_ node]
      $node_($i) add-ifq-send-trace
      $node_($i) random-motion 0

      $ns_ initial_node_pos $node_($i) 20

}

|        $node_(0) set X_ 100
|        $node_(0) set Y_ 400
|        $node_(0) set Z_ 0.0

|        $node_(1) set X_ 500
|        $node_(1) set Y_ 100
|        $node_(1) set Z_ 0.0

|        $node_(2) set X_ 100
|        $node_(2) set Y_ 100
|        $node_(2) set Z_ 0.0

|        $node_(3) set X_ 200
|        $node_(3) set Y_ 100
|        $node_(3) set Z_ 0.0

|        $node_(4) set X_ 300
|        $node_(4) set Y_ 100
|        $node_(4) set Z_ 0.0

|        $node_(5) set X_ 400
|        $node_(5) set Y_ 100
|        $node_(5) set Z_ 0.0

|        $node_(6) set X_ 100
|        $node_(6) set Y_ 200
|        $node_(6) set Z_ 0.0

|        $node_(7) set X_ 200
|        $node_(7) set Y_ 200
|        $node_(7) set Z_ 0.0

|        $node_(8) set X_ 300
|        $node_(8) set Y_ 200
|        $node_(8) set Z_ 0.0
```

```
|        $node_(9) set X_ 400
|        $node_(9) set Y_ 200
|        $node_(9) set Z_ 0.0

|        $node_(10) set X_ 100
|        $node_(10) set Y_ 300
|        $node_(10) set Z_ 0.0

|        $node_(11) set X_ 200
|        $node_(11) set Y_ 300
|        $node_(11) set Z_ 0.0

|        $node_(12) set X_ 300
|        $node_(12) set Y_ 300
|        $node_(12) set Z_ 0.0

|        $node_(13) set X_ 400
|        $node_(13) set Y_ 300
|        $node_(13) set Z_ 0.0

|        $node_(14) set X_ 200
|        $node_(14) set Y_ 400
|        $node_(14) set Z_ 0.0

|        $node_(15) set X_ 300
|        $node_(15) set Y_ 400
|        $node_(15) set Z_ 0.0

|        $node_(16) set X_ 400
|        $node_(16) set Y_ 400
|        $node_(16) set Z_ 0.0

|$ns_ at 0.01000 "$node_(0) setdest 100 400 20"
|$ns_ at 0.01000 "$node_(1) setdest 500 100 20"
|$ns_ at 0.01000 "$node_(2) setdest 100 100 20"
|$ns_ at 0.01000 "$node_(3) setdest 200 100 20"
|$ns_ at 0.01000 "$node_(4) setdest 300 100 20"
|$ns_ at 0.01000 "$node_(5) setdest 400 100 20"
|$ns_ at 0.01000 "$node_(6) setdest 100 200 20"
|$ns_ at 0.01000 "$node_(7) setdest 200 200 20"
|$ns_ at 0.01000 "$node_(8) setdest 300 200 20"
|$ns_ at 0.01000 "$node_(9) setdest 400 200 20"
|$ns_ at 0.01000 "$node_(10) setdest 100 300 20"
|$ns_ at 0.01000 "$node_(11) setdest 200 300 20"
|$ns_ at 0.01000 "$node_(12) setdest 300 300 20"
|$ns_ at 0.01000 "$node_(13) setdest 400 300 20"
|$ns_ at 0.01000 "$node_(14) setdest 200 400 20"
|$ns_ at 0.01000 "$node_(15) setdest 300 400 20"
|$ns_ at 0.01000 "$node_(16) setdest 400 400 20"

|$ns_ at 6.00000 "$node_(1) setdest 250 250 10"
|$ns_ at 35.00000 "$node_(1) setdest 350 350 10"
|$ns_ at 45.00000 "$node_(1) setdest 250 350 10"
|$ns_ at 50.00000 "$node_(1) setdest 250 150 10"
|$ns_ at 75.00000 "$node_(1) setdest 150 150 10"
|$ns_ at 81.00000 "$node_(1) setdest 150 350 10"
|$ns_ at 103.00000 "$node_(1) setdest 350 350 10"
|$ns_ at 118.00000 "$node_(1) setdest 350 200 10"
```

```
#############################################################
#
# Scenario specific simulation objects (agents etc.)

create_scenario


#############################################################
#
# Modifications to the mac and/or other stuff dictated by the
scenario!

for {set i 0} {$i < $num_nodes} {incr i} {
      if {$idea_mac} {
            $node_($i) setMac IdeaMac $idea_mac
            #puts "% node_($i) setMac IdeaMac $idea_mac"
      }
      for {set p 0} {$p < [PLevels set plevels_]} {incr p} {
            set difs_p [format "difs_%d" $p]
            if {[info exists $difs_p]} {
                  set v [set $difs_p]
                  $node_($i) setMac difs $p $v
                  puts "% node_($i) setMac difs $p $v"
            }
      }
      if {$mode == "PCF"} {
            $node_($i) CFP $pcf_period $pcf_duration
            if {$i == 0} {
                  puts "% node_($i) CFP $pcf_period $pcf_duration"
            }
      }
}


#############################################################
#
# End of simulation conditions/operations

for {set i 0} {$i < $num_nodes } {incr i} {
    $ns_ at $endtime "$node_($i) reset";
}
$ns_ at $endtime "do_stop"

set delta 0.01
set endtime [expr $endtime + $delta]
$ns_ at $endtime "do_halt"

proc do_stop {} {
      global ns_ tracefd
|     global ns_ namtrace
      $ns_ flush-trace
      close $tracefd
|     close $namtrace
}

proc do_halt {} {
      global ns_
```

```
      puts "NS EXITING..."
      $ns_ halt
}


#############################################################################
#
# Start up the simulation

puts "Starting Simulation..."
$ns_ run
```

DCF SCENARIO SCRIPT

```
#
# A wireless lan scenario traffic over udp
#

# defaults
|set num_nodes 17       ;# number of mobile nodes in the scenario
|set cbr_nodes 17
|set cbrpktsize 1000
|set cbrrate 384000
|set cbrpktsize1 160
|set cbrrate1 64000

set num_bss_nodes 0     ;# adhoc mode

proc create_scenario { } {
      global ns_ node_ num_nodes endtime
|     global cbr_nodes cbrpktsize cbrinterval cbrrate cbrpktsize1
cbrrate1

|     set endtime 150

      ###########################################################
      # Setup CBR flows

|     for {set i 1} {$i < $cbr_nodes - 15} {incr i} {
|           set cbr           [new Application/Traffic/CBR]
|           set udp_src [new Agent/UDP]
|           set udp_sink      [new Agent/Null]

|           $cbr set    packetSize_ $cbrpktsize
|           $cbr set    rate_ $cbrrate
|           $udp_src set      class_ [expr $i + 1]

|           set node_src      $i
|           set node_sink     0

|           $ns_ attach-agent $node_($node_src) $udp_src
|           $ns_ attach-agent $node_($node_sink) $udp_sink
|           $ns_ connect $udp_src $udp_sink
|           $cbr attach-agent $udp_src

|           $ns_ at [expr 23.77 + ($i-1)*0.2] "$cbr start"
      }
|      for {set i 1} {$i < $cbr_nodes - 15} {incr i} {
|                set cbr1         [new Application/Traffic/CBR]
|                set udp_src1     [new Agent/UDP]
|                set udp_sink1    [new Agent/Null]

|                $cbr1 set        packetSize_ $cbrpktsize1
|                $cbr1 set        rate_ $cbrrate1
|                $udp_src1 set    class_ [expr $i + 15]

|                set node_src1    0
|                set node_sink1   $i
```

```
|               $ns_ attach-agent $node_($node_src1) $udp_src1
|               $ns_ attach-agent $node_($node_sink1) $udp_sink1
|               $ns_ connect $udp_src1 $udp_sink1
|               $cbr1 attach-agent $udp_src1
|               $ns_ at [expr 23.22 + ($i-1)*0.15] "$cbr1 start"
        }
}
```

EDCF SCENARIO SCRIPT

```
#
# A wireless lan scenario traffic over udp
#

# defaults
|set num_nodes 17       ;# number of mobilenodes in the scenario
|set cbr_nodes 17
|set cbrpktsize 1000
|set cbrrate 384000
|set cbrpktsize1 160
|set cbrrate1 64000

set num_bss_nodes 0     ;# adhoc mode

|PHY_MIB set CWMin_0          7
|PHY_MIB set CWMin_5          31

proc create_scenario { } {
      global ns_ node_ num_nodes endtime
|     global cbr_nodes cbrpktsize cbrinterval cbrrate cbrpktsize1
cbrrate1

|     set endtime 150

      ###########################################################
      # Setup CBR flows

|     for {set i 1} {$i < $cbr_nodes - 15} {incr i} {
|           set cbr           [new Application/Traffic/CBR]
|           set udp_src [new Agent/UDP]
|           set udp_sink      [new Agent/Null]

|           $cbr set    random_ 0
|           $cbr set    packetSize_ $cbrpktsize
|           $cbr set    rate_ $cbrrate
|           $udp_src set      class_ [expr $i + 1]

|                 set pri 0
|                 $udp_src set      prio_ $pri

|           set node_src      $i
|           set node_sink     0

|           $ns_ attach-agent $node_($node_src) $udp_src
|           $ns_ attach-agent $node_($node_sink) $udp_sink
|           $ns_ connect $udp_src $udp_sink

|           $cbr attach-agent $udp_src
|           puts "cbr$i with pktsize $cbrpktsize flows from
Node$node_src to Node$node_sink with priority $pri"

|           $ns_ at [expr 23.7 + ($i-1)*0.25] "$cbr start"
      }
|      for {set i 1} {$i < $cbr_nodes - 15} {incr i} {
```

```
|                set cbr1          [new Application/Traffic/CBR]
|                set udp_src1      [new Agent/UDP]
|                set udp_sink1     [new Agent/Null]

|                $cbr1 set         random_ 0
|                $cbr1 set         packetSize_ $cbrpktsize1
|                $cbr1 set         rate_ $cbrrate1
|                $udp_src1 set     class_ [expr $i + 15]

|                 set pri 5
|                 $udp_src set    prio_ $pri

|                set node_src1   0
|                set node_sink1   $i
|                $ns_ attach-agent $node_($node_src1) $udp_src1
|                $ns_ attach-agent $node_($node_sink1) $udp_sink1
|                $ns_ connect $udp_src1 $udp_sink1
|                $cbr1 attach-agent $udp_src1
|                puts "cbr1-$i with pktsize $cbrpktsize1 flows from
Node$node_src1 to Node$node_sink1 with priority $pri"
|                $ns_ at [expr 23.2+ ($i-1)*0.15] "$cbr1 start"
        }
}
```