A
Major Project Report
On

ENHANCEMENT OF IEEE 802.11 WLAN MAC USING EDCF PROTOCOL

Submitted in partial fulfillment of the requirements
for the award of the degree of
MASTER OF ENGINEERING
(Computer Technology & Applications)

By

Pulgari Praneeth Reddy
College Roll No. 03/CTA/04
Delhi University Roll No. 8512

Under the guidance of
Prof. Asok De

Department Of Computer Engineering

Delhi College of Engineering

Bawana Road, Delhi-110042
(University of Delhi)

CERTIFICATE

It is to certify that the work that is being presented in this Project entitled " **ENHANCEMENT OF IEEE 802.11 WLAN MAC USING EDCF PROTOCOL**", in partial fulfillment of the requirement for the award of the degree of Master of Engineering in Computer Technology and Application submitted by ***Pulgari Praneeth Reddy*** (03/CTA/04) to the Department of Computer Engineering, Delhi College of Engineering, is the record of the student's own work that is carried out under my supervision and guidance.

Dr. Asok De
Professor
Dept. of Computer Engineering
Delhi College of Engineering

ACKNOWLEDGEMENTS

It is a great pleasure to have the opportunity to extend my heartiest felt gratitude to everybody who helped me throughout the course of this project.

I would like to express my heartiest felt regards to **Dr. Asok De**, Professor, Department of Computer Engineering for his constant motivation and support for the entire duration of this project. It is my privilege and honor to have worked under his supervision. His invaluable guidance and helpful discussions in every stage of this project really helped me in materializing the project.

I would like to thank **Dr. Goldie Gabrani**, Head of the department, Computer Engineering, for providing facilities in doing this project.

Finally, I would like to take this opportunity to present my sincere regards to my teachers viz. Professor D. Roy Choudhury, Dr S. K. Saxena, Mr. Rajeev Kumar and Mrs. Rajni Jindal for their support and encouragement.

I am thankful to my friends and classmates for their unconditional support and motivation during this project.

Pulgari Praneeth Reddy

M.E. (Computer Technology & Applications)

College Roll No. 03/CTA/04

Delhi University Roll No. 8512

ABSTRACT

IEEE 802.11 Wireless LAN is one of the most deployed wireless technologies all over the world and likely to play a major role in the next generation wireless communication networks. In the present 802.11 WLAN two basic co-ordination functions exists. DCF (Distributed Coordination function) and PCF (Point Coordination Function). DCF is responsible for Asynchronous data services and the PCF was developed for time bounded services. Although IEEE 802.11 is most widely used WLAN today, it cannot provide QoS(Quality of Service) Support for the increasing number of the multimedia applications typically for time bounded service such as video/audio conferencing which require some level of specified bandwidth and delay. So a new protocol called EDCF (Enhanced Distribution Coordination function) is being introduced in IEEE 802.11e standard which supports the QoS requirements to a certain level. Even though EDCF protocol supports many QoS requirements, it has some drawbacks like static values assignment of CW size. In this work I had analyzed the various QoS limitations of the DCF and QoS Support of EDCF. Then I had enhanced the EDCF protocol to support the Adaptive technique in assigning the values of Contention Window size. The simulational analysis of these protocols is done in NS2 (network simulator 2).

The NS2 is a open source software embedded with rich set of network protocols at different layers and is used for simulation purposes with real time environment. The basic structure of ns2 and the networking protocols are realized in C++ programming language where as the simulations are controlled by script language TCL (Tool Command Language).

ACRONYMS

AC	Access Category
ACK	Acknowledgement
AIFS	Arbitration Inter Frame Spacing
AP	Access Point
CA	Collision Avoidance
CAP	Controlled Access Period
CFB	Contention Free Burst
CFP	Contention Free Period
CF-Poll	Contention Free - Poll
CF-End	Contention Free - End
CP	Contention Period
CSMA	Carrier Sense Multiple Access
CW	Contention Window
CWmax	Contention Window Maximum
CWmin	Contention Window Minimum
DCF	Distributed Coordination Function
EDCF	Enhanced Distributed Coordination Function
HC	Hybrid Coordinator
HCF	Hybrid Coordination Function
HCCA	HCF Controlled Channel Access
IEEE	Institute of Electrical and Electronics Engineers
ISM	Industrial, Science, and Medical
MAC	Medium Access Control
MSDU	MAC Service Data Unit
NAV	Network Allocation Vector
PC	Point Coordinator
PCF	Point Coordination Function
PF	Persistence Factor
PIFS	PCF Inter Frame Space

QAP	QoS Access Point
QBSS	Quality Of Service Basic Service Set
QoS	Quality Of Service
QSTA	QoS station
RTS/CTS	Request to Send/Clear to Send
SIFS	Short Inter Frame Space
TBTT	Target Beacon Transmission Time
TXOP	Transmission Opportunity
UP	User Priorities
WLAN	Wireless Local Area Network
WSTA	Wireless Station

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 Motivation

Wireless computing is a rapidly emerging technology providing users with network connectivity without being tethered off of a wired network. Wireless local area networks (WLANs), like their wired counterparts, are being developed to provide high bandwidth to users in a limited geographical area. WLANs are being studied as an alternative to the high installation and maintenance costs incurred by traditional additions, deletions, and changes experienced in wired LAN infrastructures. Physical and environmental necessity is another driving factor in favor of WLANs. The operational environment may not accommodate a wired network, or the network may be temporary and operational for a very short time, making the installation of a wired network impractical. Examples where this is true include ad hoc networking needs such as conference registration centers, campus classrooms, emergency relief centers, and tactical military environments. However, to meet these objectives, the wireless community faces certain challenges and constraints that are not imposed on their wired counterparts.

1.2 Need for Specialized Wireless MAC

Existing MAC schemes from wired networks like, CSMA/CD are not directly applicable to wireless medium. **Carrier Sense** describes the fact that a transmitter listens for carrier wave before trying to send. That is, it tries to detect the presence of an encoded signal from another station before attempting to transmit. In CSMA/CD sender senses the medium to see if it is free. If medium is busy, the sender waits until it is free. If the medium is free, sender starts transmitting data and also continues to listen into the medium. It stops transmission as soon as it detects collision and sends a jam signal. In wired medium, this works because more or less the same signal strength can be assumed all over the wire. If collision occurs somewhere in the wire, everybody will notice it. This assumption gets invalidated in wireless medium, as the signal strength decreases

proportionally to the square of distance to the sender. In wireless medium, sender may apply carrier sense and detect an idle medium. Thus, the sender starts sending, but a collision happens at the receiver due to a second sender. Second sender may or may not be audible to first sender. Hence the sender detects no collision, assumes that data has been transmitted without errors, but actually a collision might have destroyed the data at the receiver. Besides that, wireless devices are half duplex and battery operated. They are unable to listen to the channel for collision while transmitting data.

1.3 Hidden and Exposed Node Problem

The transmission range of stations in wireless network is limited by the transmission power, therefore all the station in a LAN can not listen to each other. This gives rise to hidden node and exposed node problem. Consider a scenario shown in Figure 1.1. Transmission range of A reaches B, but not C. The transmission range of C reaches B, but not A. Finally, the transmission range of B reaches both A and C.

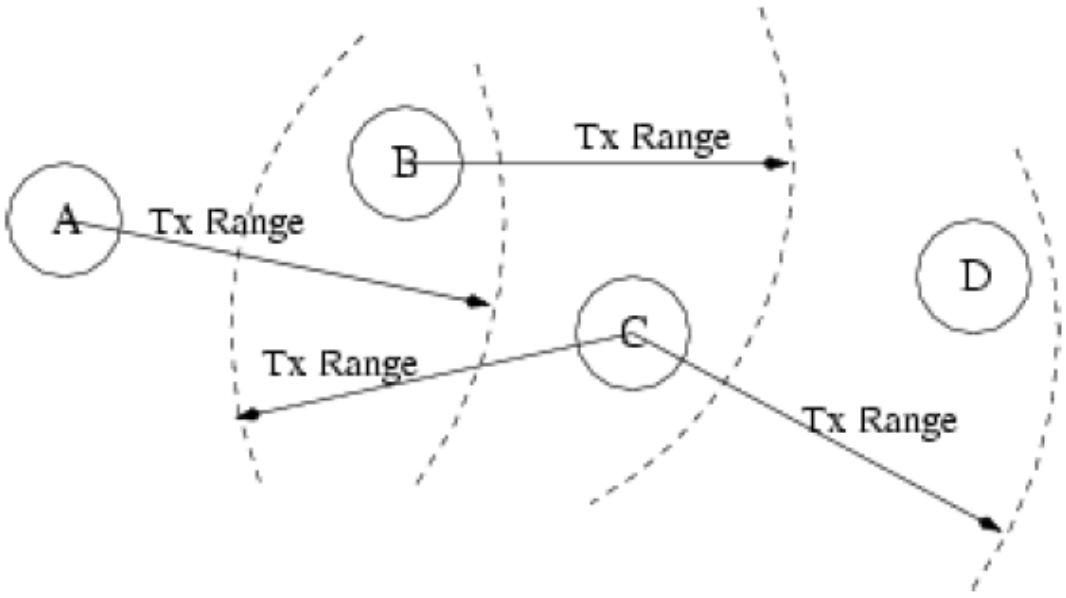


Figure 1.1: Hidden and Exposed Node Scenario

Hence C can listen to B but not A. A start sending to B, C does not hear this

transmission, and also wants to send something to B. C senses the medium, medium appears to be free and it starts sending. Hence collision occurs at B. A cannot detect this collision and continues with its transmission. A and C are hidden to each other. This problem is termed as Hidden Node problem.

Hidden terminals cause collision, and Exposed terminals suffer unnecessary delays. Consider the situation that B sends something to A and C wants to transmit data to D. D is not in transmission or interference range of A and B. C senses the medium and finds it busy. Thus, C postpones its transmission. But as A is outside the interference range of C, waiting is not necessary. Collision at B due to C's transmission does not matter as it is too weak to propagate to A. This termed as Exposed Node problem. Node C is exposed to B.

1.4 Challenges in Wireless LANs

Many different and sometimes competing design goals have to be taken into account for WLANs to ensure their commercial success.

- **Global operation**: WLAN products should sell in all countries, therefore, many national and international frequency regulations have to be considered.
- **Low Power**: Devices communicating via a WLAN are typically also wireless devices running on battery power. Hence, WLAN must implement special power saving modes and power management functions.
- **License-free operation**: LAN operators do not want to apply for a special license in order to be able to use the product. Thus, the equipment must operate in a license-free band, such as the 2.4 GHz ISM band.
- **Bandwidth**: Bandwidth is the one of the scarcest resource in wireless networks. The available bandwidth in wireless networks is far less than the wired links.
- **Link Errors**: Channel fading and interference cause link errors and these errors may sometimes be very severe.
- **Robust transmission technology**: Compared to wired counterparts, WLANs operate under difficult conditions. If they use radio transmission, many other electrical devices may interfere.

- **Simplified spontaneous co-operation**: To be useful in practice, WLANs should not require complicated setup routines but should operate spontaneously after power up. Otherwise these LANs would not be useful for supporting e.g., ad hoc meetings, etc.
- **Easy to use**: LANs should not require complex management but rather work on a plug-and-play basis.
- **Protection of investment**: A lot of money has already been invested into wired LANs. Hence new WLANs must protect this investment by being inter operable with the existing networks.
- **Safety and security**: Most important concern is of safety and security. WLANs should be safe to operate, especially regarding low radiation. Furthermore, no users should be able to read personal data during transmission i.e., encryption mechanism should be integrated. The network should also take into account user privacy.
- **Transparency for application**: Existing applications should continue to run over WLANs. The fact of wireless access and mobility should be hidden if not relevant.

1.5 Problem Statement

IEEE 802.11 WLAN supports two basic protocols named DCF and PCF. These protocols even though provides various services, they lack quality of service support to the different data streams. They give same priority to transmit all types of data streams where as the streams like voice and video should have higher priority to support QOS.

QoS is the ability of a network element (e.g. an application, a host or a router) to provide some levels of assurance for consistent network data delivery. Parameterized QoS is a strict QoS requirement which is expressed in terms of quantitative values, such as data rate, delay bound, and jitter bound.

IEEE 802.11e is a new standard proposed which overcomes the drawback of these protocols by introducing new protocols EDCF(Enhanced Distributed Coordination function) and HCCA(HCF controlled channel access). EDCF guarantees the QOS support pretty well when compared against the DCF. But with the increase of number of nodes with a particular access point the delay increases. In EDCF scheme, the size of the contention window is a static value.

This work aims to introduce an adaptive technique in the EDCF protocol so that the relative priorities are provisioned by adjusting the size of the Contention Window (CW) of each traffic class taking into account both applications requirements and network conditions i.e. the value of the contention window size is dynamic in nature. The simulations are performed using Network simulator(ns2) where the results of both protocols are analyzed and compared.

1.6 Thesis Outline

In this thesis the chapter 2 gives the brief overview of the IEEE 802.11 Wireless Lan standard and its activities. Chapter 3 summarizes the IEEE 802.11 MAC layer with its protocols and the QoS support required for Real time applications. A detailed description of Enhancement schemes proposed for the MAC layer is given in the following chapter. It also describes the new adaptive technique which proves to be more efficient than EDCF. Chapter 5 gives the brief tutorial of the network simulator (ns2) which is used in this work for simulation purpose. The next chapter presents the Post simulation analysis which involves the analysis of trace files and the simulation results obtained. Finally, I have given the Future work and conclusion of my work.

Chapter 2

IEEE 802.11 WLAN

2.1 Why IEEE 802.11 WLAN

IEEE 802.11 standard is one of the prominent wireless local area network standards being adopted as a mature technology. The success of the IEEE 802.11 standard has resulted in the easy availability of commercial hardware and a proliferation of wireless network deployment, in wireless LANs as well as in mobile ad hoc networks. Although IEEE 802.11 is not designed for multi-hop ad hoc networks, the easy availability has made it, most chosen MAC

2.2 Standard Activities of IEEE 802.11

The IEEE 802.11 WLAN standard covers the MAC sub-layer and the physical (PHY) layer of the open system interconnection (OSI) network reference model. **Logical link control (LLC)** sub-layer is specified in the IEEE 802.2 standard. This architecture provides a transparent interface to the higher layer users: stations (STAs) may move, roam through an 802.11 WLAN and still appear as stationary to 802.2 LLC sub-layer and above. This allows existing TCP/IP protocols to run over IEEE 802.11 WLAN just like wired Ethernet.

In 1997, IEEE provided three kinds of options in the PHY layer, which are:-

- Infrared (IR) base-band PHY
- Frequency hopping spread spectrum (FHSS) radio
- Direct sequence spread spectrum (DSSS) radio

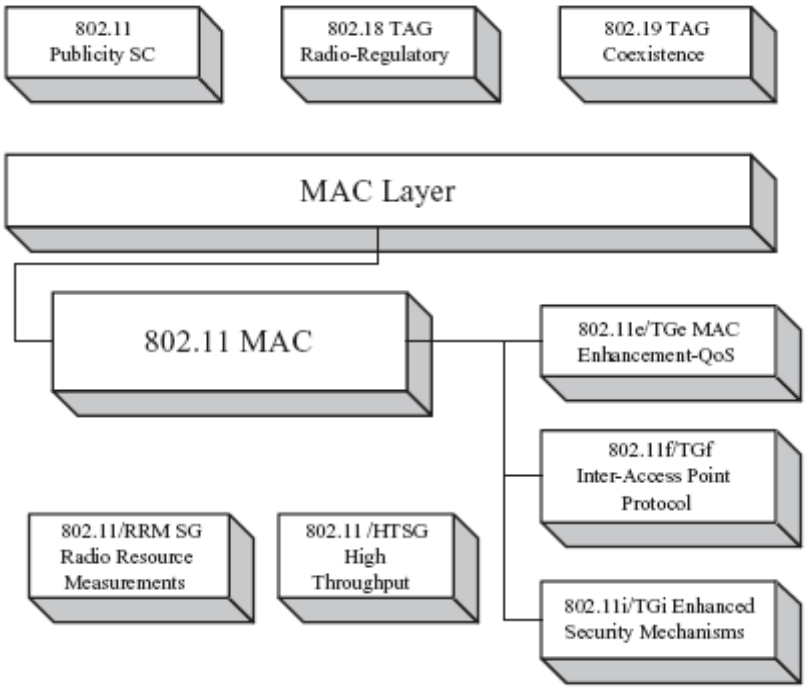
All these options support both 1 and 2 Mbps PHY rate.

In 1999, the IEEE defined two high rate extensions:-

- 802.11b in the 2.4GHz band with data rates up to 11Mbps, based on DSSS technology

- 802.11a in the 5GHz band with data rates up to 54Mbps, based on orthogonal frequency division multiplexing (OFDM) technology

Recently, 802.11g is finalized. It extends 802.11b PHY layer to support data rates up to 54Mbps in the 2.4GHz band. Moreover, ongoing 802.11h will enhance 802.11a with adding indoor and outdoor license regulations for the 5GHz band in Europe. At the MAC layer, 802.11e is the first supplement to enhance the QoS performance of 802.11 WLAN; an Inter-Access Point protocol is defined in 802.11f to allow STAs roaming between multi-vendor access points; 802.11i aims to enhance security and authentication mechanisms for 802.11 MAC.



(a) MAC layer activities

Fig 2.1: MAC layer activities

Thus the 802.11 family can be summarized as follows:-

802.11**a**	54 Mbps	5GHz (OFDM)
802.11**b**	11 Mbps	2.4GHz (DSSS)

802.11**c**	Bridge Operation Procedures
802.11**d**	Global Harmonization
802.11e	*MAC enhancements for QoS*
802.11**f**	Inter Access Point Protocol (roaming)
802.11**g**	Extends 802.11b PHY layer
802.11**h**	Dynamic Frequency Selection
802.11**i**	Enhance security and authorization

In this project, 802.11e is the key area of my study.
At the MAC layer 802.11e is the first support to enhance the QoS performance of 802.11e WLAN an Inter access point protocol is defined in 802.11f to allow STAs roaming between multi-vendor access points.

2.3 Architecture Components

An 802.11 LAN is based on a cellular architecture where the system is subdivided into cells, where each cell (called Basic Service Set or BSS, in the 802.11 nomenclature) is controlled by a Base Station (called Access Point, or in short AP).
Even though that a wireless LAN may be formed by a single cell, with a single Access Point, (it can also work without an Access Point) most installations will be formed by several cells, where the Access Points are connected through some kind of backbone (called Distribution System or DS), typically Ethernet, and in some cases wireless itself.

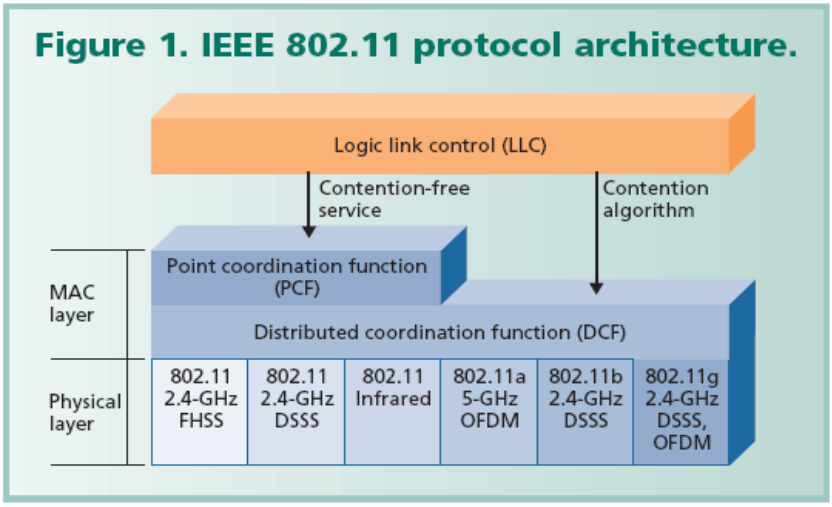


Fig 2. 2. Snapshot of 802.11 PHY and MAC standard activities.

2.4 IEEE 802.11 MAC- Terminology and Concepts

IEEE 802.11 MAC features two mode of operations:

Distributed Coordinating Function(DCF) and Point Coordinating Function (PCF).

DCF is CSMA/CA based random access protocol that uses random backoff to avoid collision. It uses RTS/CTS exchange mechanism to reserve channel, when packet size is above the RTSthreshold. It overcomes the hidden terminal problem.

PCF provides centralized scheduled access to channel. It comprises of chain of contention free period (CFP) and contention period(CP). DCF rules are followed in the CP. In the CFP point coordinator (PC) polls the node one by one and grant access to channel. New stations that need to get enrolled in poll list, send request in CP.

CSMA/CA

CSMA/CA is a modification of pure Carrier Sense Multiple Access (CSMA). Collision avoidance is used to improve the performance of CSMA by attempting to reserve the network for a single transmitter. This is the function of the jamming signal in CSMA/CA. The performance improvement is achieved by reducing the probability of collision and retry. Extra overhead is added due to the jamming signal wait time, so other techniques give better performance. Collision avoidance is particularly useful in media such as radio, where reliable collision detection is not possible.

RTS/CTS

A sending node wishing to send data sends a Request to Send (RTS) frame. The destination node replies with a Clear To Send (CTS) frame. Any other node receiving the CTS frame should refrain from sending data for a given time (solving the Hidden node problem). Any other node receiving the RTS frame but not the CTS frame should not refrain from sending data to another node (solving the Exposed node problem). The amount of time any other node should wait before trying to get access to the medium is included in both the RTS and the CTS frame. The RTS/CTS mechanism is a way to solve the called Hidden node problem and Exposed node problem.

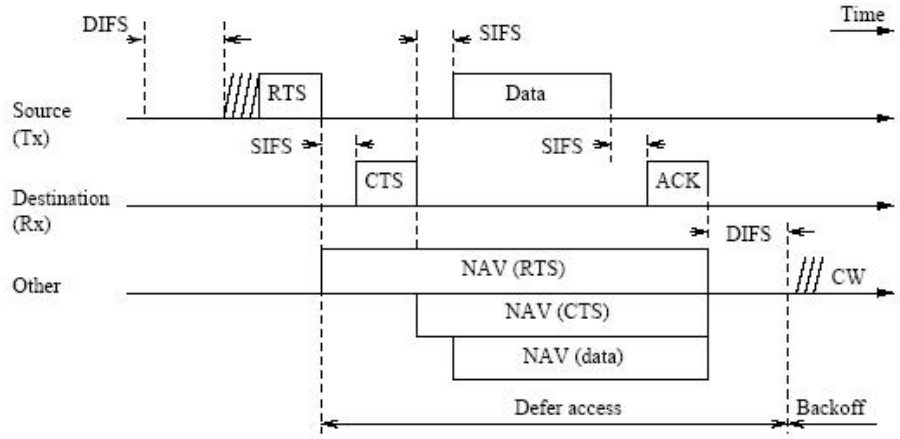


Fig 2. 3. RTS, CTS and NAV representation

Inter Frame Spaces

The Standard defines 4 types of Inter Frame Spaces, which are use to provide different priorities:

- **SIFS** - Which stands for **Short Inter Frame Space,** is used to separate transmissions belonging to a single dialog (e.g. Fragment-Ack), and is the minimum Inter Frame Space, and there is always at most one single station to transmit at this given time,

hence having priority over all other stations. This value is a fixed value per PRY and is calculated in such a way that the transmitting station will be able to switch back to receive mode and be capable of decoding the incoming packet, on the 802.11 FR PRY this value is set to 28 microseconds

- **PIFS - Point Co-ordination IFS,** is used by the Access Point (or Point Coordinator, as called in this case), to gain access to the medium before any other station.
  This value is SIFS plus a Slot Time (defined in the following paragraph), i.e. 78 microseconds.
- **DIFS - Distributed IFS,** is the Inter Frame Space used for a station willing to start a new transmission, which is calculated as PIFS plus one slot time, i.e. 128 microseconds and
- **EIFS - Extended IFS,** which is a longer IFS used by a station that has received a packet that could not understand, this is needed to prevent the station (who could not understand the duration information for the Virtual Carrier Sense) from colliding with a future packet belonging to the current dialog.

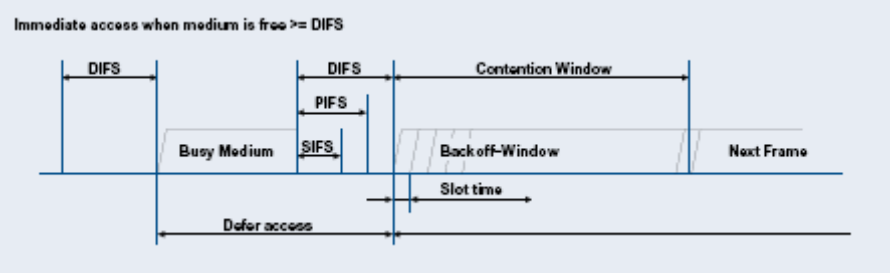


Fig 2.4: Timing Intervals

Exponential Backoff Algorithm

Backoff is a well known method to resolve contention between different stations willing to access the medium, the method requires each station to choose a Random Number (n) with in an interval, and wait for this number of Slots before accessing the medium, always checking whether a different station has accessed the medium before.
The **Slot Time** is defined in such a way that a station will always be capable of determining if other station has accessed the medium at the beginning of the previous slot. This reduces the collision probability by half. Exponential Backoff means that each

time the station chooses a slot and happens to collide, it will increase the maximum number for the random selection exponentially.

The 802.11 standard defines an **Exponential Backoff Algorithm,** which must be executed in the following cases:

- If when the station senses the medium before the first transmission of a packet, and the medium is busy,
- After each retransmission, and
- After a successful transmission

The only case when this mechanism is not used is when the station decides to transmit a new packet and the medium has been free for more than DIFS

Network Allocation Vector(NAV):

It is the duration of time in which all the other stations which sensed the ongoing RTS/CTS of a communication, remain silent without sensing the channel for the duration learnt by the RTS/CTS frames.

BSS

A group of STAs coordinated by DCF or PCF is formally called a **basic service set (BSS)**. The area covered by the BSS is known as the **basic service area (BSA)**, which is similar to a cell in a cellular mobile network.

There are two different modes to configure an 802.11 wireless network:-

- Ad-hoc mode

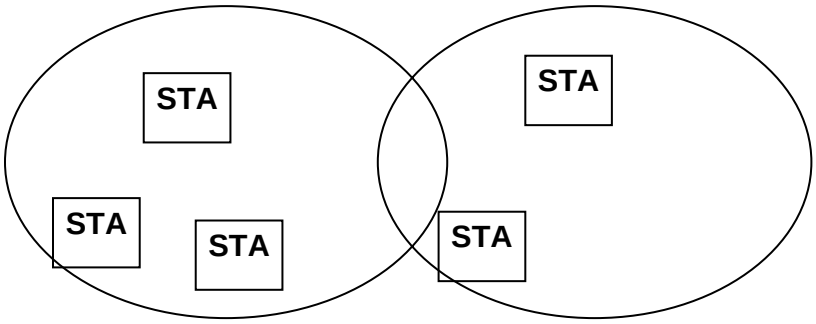


Fig: 2.5: Stations of Ad hoc mode

The mobile STAs can directly communicate with each other to form an **Independent BSS (IBSS)** without connectivity to any wired backbone.

- Infrastructure mode

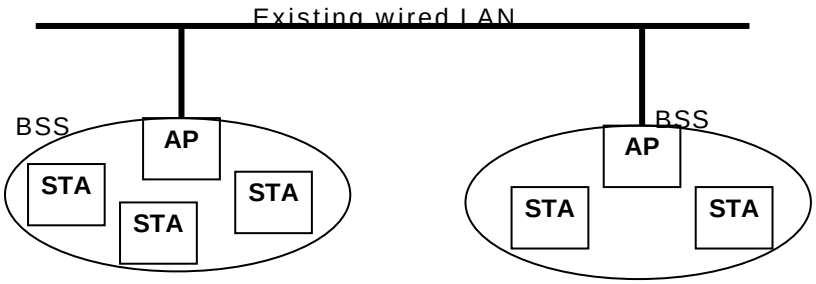


Fig 2.6: Stations of Infrastructure mode

The mobile STAs can communicate with the wired backbone through the bridge of access point (AP).

DCF can be used both in ad-hoc and infrastructure modes, while PCF is only used in infrastructure mode.

Chapter 3

MAC LAYER AND ITS QOS SUPPORT

The IEEE 802.11 MAC sub-layer defines two relative medium access coordination functions, the Distributed Coordination Function (DCF) and the optional Point Coordination Function (PCF) The transmission medium can operate both in contention mode (DCF) and contention-free mode (PCF). The IEEE 802.11 MAC protocol provides two types of transmission: asynchronous and synchronous.

The asynchronous type of transmission is provided by DCF which implements the basic access method of the 802.11 MAC protocol. DCF is based on the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) protocol, and should be implemented in all the stations. The synchronous service (also called contention free service) is provided by PCF which basically implements a polling-based access method. The PCF uses a centralized polling approach which requires an Access Point (AP) that acts as a Point Coordinator (PC). The AP cyclically polls stations to give them the opportunity to transmit the packets. Unlike the DCF, the implementation of the PCF is not mandatory. Furthermore, the PCF itself relies on the asynchronous service provided by the DCF.

3.1 DCF: Distributed Coordination Function

DCF is a distributed medium access scheme. In this mode, a station must sense the medium before initiating a packet transmission by PHY and MAC layer virtual carrier sensing. If the medium is found idle for a time interval longer than Distributed InterFrame Space (DIFS), then the station can transmit the packet directly. Otherwise, the transmission is deferred and the backoff process is started. Specifically, the station computes a random time interval named Backoff time, uniformly distributed between zero and the current Contention Window size (CW),

Backoff_time = rand[0; CW], where CWmin < CW < CWmax and Slot time depends on the PHY layer type. The backoff timer is decreased only when the medium is idle, whereas it is frozen when another station is transmitting. Each time the medium becomes idle, the station waits for a DIFS and then continuously decrements the backoff timer. As soon as the backoff timer expires, the station is authorized to access the medium.

Obviously, a collision occurs if two or more stations start transmission simultaneously. Unlike wired networks (e.g. CSMA/CD), in wireless environment collision detection is impossible due to the significant difference between transmitted and received power levels. Hence, a positive acknowledgement is used to notify the sender that the transmitted frame has been successfully received. The transmission of the acknowledgement is initiated at a time interval equal to the Short InterFrame Space (SIFS) after the end of the reception of the previous frame. Since the SIFS is
smaller than the DIFS, the receiving station does not need to sense the medium before transmitting an acknowledgement. If the acknowledgement is not received, the sender assumes that the transmitted frame was lost and schedules a retransmission and then enters the backoff process again.

To reduce the probability of collisions, after each unsuccessful transmission attempt, the contention window is doubled until a predefined maximum value CWmax is reached. To improve the channel utilization, after each successful transmission, the contention window is reset to a fixed minimum value CWmin. The Network Allocation Vector (NAV) is used for MAC virtual carrier sensing, by updating the local NAV with the value of other station's transmission duration. By using NAV, a station can know when the current transmission ends and channel is idle.

In order to solve the so-called hidden terminal problem, an optional RTS/CTS (RequestToSend and ClearToSend) scheme is introduced. The transmitter sends a short RTS frame (20 octets) before each data frame transmission. Note that a collision of the short RTS frames is less severe and probable than a collision of data frames (up to 2346 octets). The receiver replies with a CTS frame if it is ready to receive and the channel is reserved for the duration of packet transmission. When the source receives the CTS, it

starts transmitting its frame, being sure that the channel has been reserved for it during the entire frame transmission duration. All other stations in the Basic Service Set (BSS) update their Network Allocation Vectors (NAVs) whenever they hear a RTS, a CTS or a data frame.

The overhead of sending RTS/CTS frames becomes considerable when data frame sizes are small, thus the channel is used sub-optimally. Moreover, very large frames may reduce transmission reliability, e.g. an uncorrectable error in a large frame wastes more bandwidth and transmission time than an error in a shorter frame. So another optimization parameter of fragmentation threshold is used. That means, when data frame size is larger than this threshold, the data frame will be partitioned into several smaller MAC level frames.

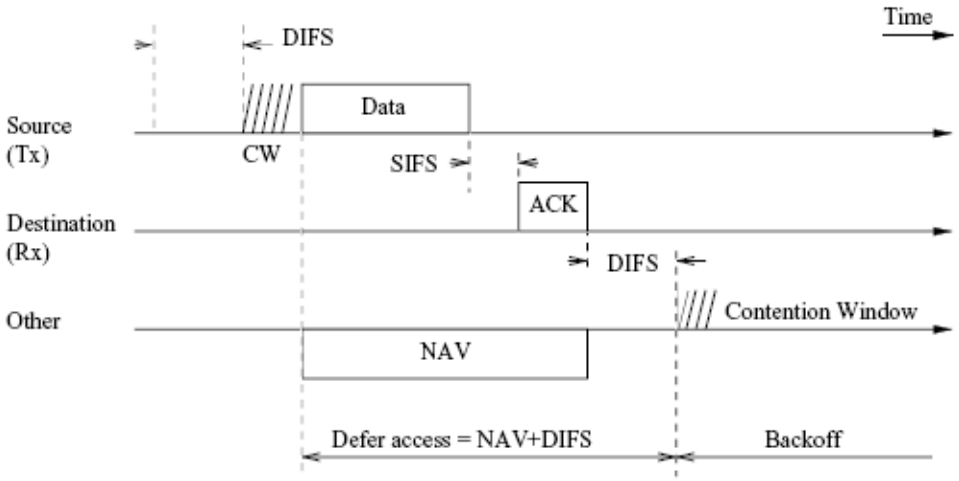


Fig 3.1: DCF access mechanism

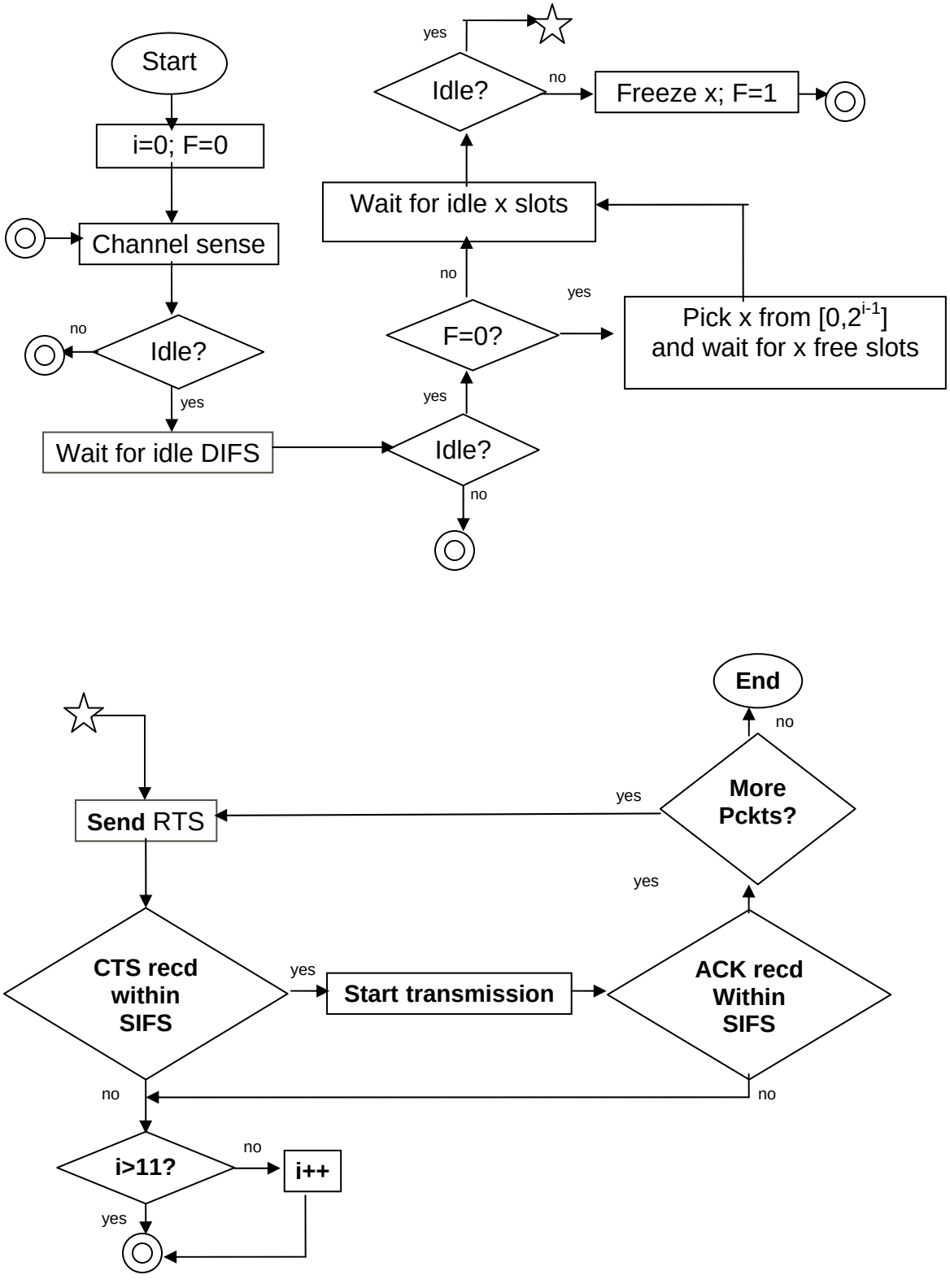


Fig 3.2: Flow Chart for the DCF protocol

3.2 PCF: Point Coordination Function

PCF uses a centralized polling scheme, which requires the AP as a point coordinator (PC). If a BSS is set up with PCF-enabled, the channel access time is divided into periodic intervals named beacon intervals. The beacon interval is composed of a contention-free period (CFP) and a contention period (CP). During the CFP, the PC maintains a list of registered STAs and polls each STA according to its list. Then, when an STA is polled, it gets the permission to transmit data frame. Since every STA is permitted a maximum length of frame to transmit, the maximum CFP duration for all the STAs can be known and decided by the PC, which is called *CFP_max_duration*. The time used by the PC to generate beacon frames is called target beacon transmission time (TBTT). In the beacon, the PC denotes the next TBTT and broadcasts it to all the other STAs in the BSS. In order to ensure that no DCF STAs are able to interrupt the operation of the PCF, a PC waits for a PCF InterFrame Space (PIFS), which is shorter than DIFS, to start the PCF. Then, all the other STAs set their NAVs to the values of *CFP_max_duration* time, or the remaining duration of CFP in case of delayed beacon. During the CP, the DCF scheme is used, and the beacon interval must allow at least one DCF data frame to be transmitted.

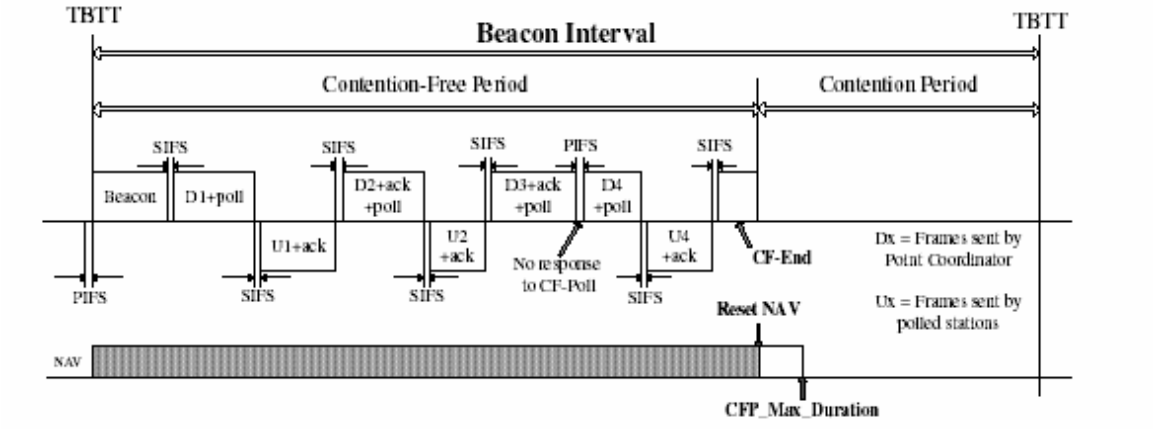


Fig 3.3 PCF and DCF Cycle.

A typical medium access sequence during PCF is shown in Figure 3.3 When a PC polls an STA, it can piggyback the data frames to the STA together with the CF-Poll, and then the STA sends back data frame piggybacked with an ACK after a SIFS interval. When the PC polls the next STA, it piggybacks not only the data frame to the destination, but also an ACK to the previous successful transmission. Note that almost all packet transmissions are separated by the SIFS except for one scenario: if the polled STA does not respond the PC within a PIFS period, the PC will poll the following STA. Silent STAs are removed from the polling list after several periods and may be polled again at the beginning of the next CFP. At any time, the PC can terminate the CFP by transmitting a CF-End packet, then all the STAs in the BSS should reset their NAVs and attempt to transmit during the CP. Normally, PCF uses a round-robin scheduler to poll each STA sequentially in the order of polling list, but priority-based polling mechanisms can also be used if different QoS levels are requested by different STAs.

3.3 QoS (Quality of Service) – What and Why?

In general terms, QoS is the ability of a network element (e.g. an application, a host or a router) to provide some levels of assurance for consistent network data delivery. Good QoS services include guaranteed features of

- Required Bandwidth
- Faster Response Time
- Minimal Error Rate
- Consistent connectivity

In the recent past, web services using multimedia applications have grown fast as a necessity. They include the services such as transmission of high speed video, audio, graphic files, animated files, 3D games, etc…These services require guaranteed QoS support from the network.

3.4 IEEE 802.11 QoS Limitation

3.4.1 QoS Limitation of DCF (Distributed Coordination Function):

1. DCF Supports only the best-effort service, not any QoS guarantees. Typically, time-bounded services such as voice over IP or audio/video conferencing require specified bandwidth, delay, and jitter, but can tolerate some losses.
2. In DCF mode, all the STA's in one BSS compete for the resources and channel with same priorities where as priorities should be assigned depending on the type of data flow.
3. There is no differentiation mechanism to guarantee bandwidth, packet delay and jitter for high priority STAs or Multimedia flows.

3.4.2 QoS Limitation of PCF(Point Coordination Function)

Although PCF has been designed to support time bounded multimedia applications, this mode has three main problems that lead to poor QoS performances

1) Central Polling Scheme. All the communication between two STA's in the same BSS has to go through the AP(Access Point), thus some of the channel bandwidth is wasted. As traffic increases a lot of channel resources are wasted.
2) The cooperation between CP and CFP modes may lead to unpredictable beacon delays.

Chapter 4

NEW ENHANCED SCHEMES FOR QOS GUARANTEES

In IEEE 802.11e standard which was finalized recently, they had proposed two new protocols to overcome the drawbacks of DCF and PCF. They are EDCF and HCCA which guarantees the quality of service up to a certain level. The following sections describe each of them briefly.

4.1 Enhanced Distributed Coordination Function (EDCF)

The EDCF is designed for the contention-based prioritized QoS support. Figure 4.1 shows that in EDCF, each QoS-enhanced STA (QSTA) has 4 queues (ACs), to support 8 user priorities (UPs) as defined in IEEE 802.11e. Therefore, one or more UPs are mapped to the same AC queue, see Table 4.1. This comes from the observation that usually eight kinds of applications do not transmit frames simultaneously, and using less ACs than UPs reduces the MAC layer overheads. Each AC queue works as an independent DCF STA and uses its own backoff parameters

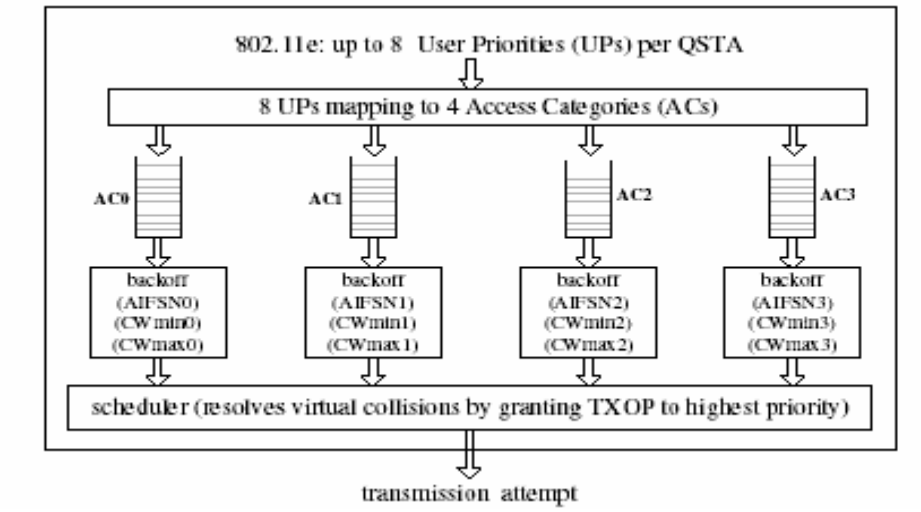


Fig 4.1: EDCF is proposed in IEEE 802.11e

UP, User Priority (Same as 802.1D)	802.1D Designation	802.11e AC (Access Category)	Service type
2	Not defined	0	Best Effort
1	Background (BK)	0	Best Effort
0	Best Effort (BE)	0	Best Effort
3	Excellent Effort (EE)	1	Video Probe
4	Controlled Load (CL)	2	Video
5	VI (Video < 100ms latency and jitter)	2	Video
6	VO (Video < 10ms latency and jitter)	3	Voice
7	Network Control (NC)	3	Voice

Table 4.1: Mapping between User Priorities (UP) and Access Category (AC).

In EDCF, two main methods are introduced to support service differentiation: The first one is to use different InterFrame Space (IFS) sizes for different ACs. Figure 4.2 Shows the detailed timing diagram of the EDCF scheme. A new kind of IFS called Arbitration IFS (AIFS) is used in EDCF, instead of DIFS in DCF. The AIFS [AC] is determined by

$$\mathbf{AIFS\ [AC] = AIFSN\ [AC] \cdot SlotTime + SIFS}$$

Where the default values of the arbitration inter frame spacing number (AIFSN) is defined as either 1 or 2. When AIFSN = 1, high priority queues AC1, AC2 and AC3 have AIFS value equal to PIFS. When AIFSN = 2, the low priority queue AC0 has AIFS value of DIFS. When a frame arrives at an empty AC queue and the medium has been idle longer than AIFS [AC]+SlotTime, the frame is transmitted immediately. If the channel is busy, the arriving packet in each AC has to wait until the medium becomes idle and then defer for AIFS+SlotTime. So the AC with the smaller AIFS has the higher priority. For example, the earliest transmission time for high priority queue is to wait for PIFS+SlotTime = DIFS, while the earliest transmission time for best effort queue is to wait for DIFS + SlotTime.

The second method consists in allocating different CW sizes for different ACs. Assigning a short CW size to a high priority AC ensures that in most cases, high-priority AC is able to transmit packets ahead of low-priority one. If the backoff counters of two or more parallel ACs in one QSTA reach zero at the same time, a scheduler inside the QSTA will avoid the virtual collision by granting the EDCF-TXOP to the highest priority AC. At the same time, the other colliding ACs will enter a backoff process and double the CW sizes

as if there is an external collision. In this way, EDCF is supposed to improve the performance .

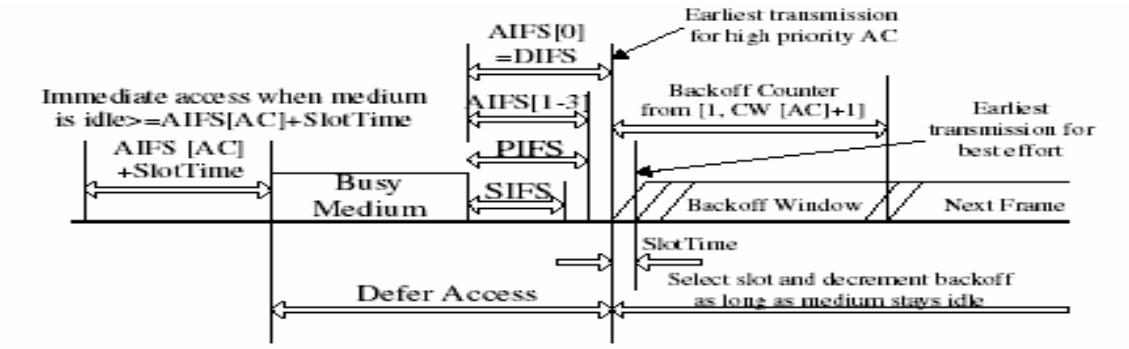
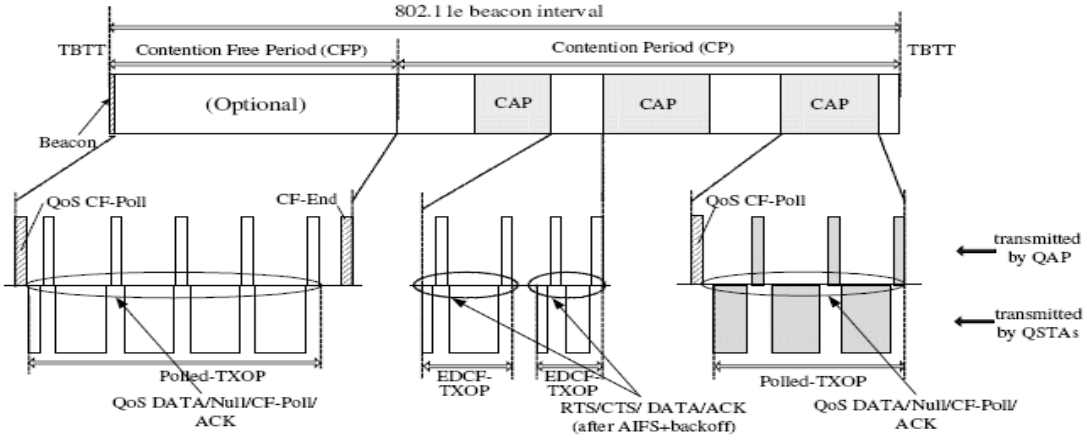


Fig 4.2: EDCF Channel Access and IFS relationships.

4.2 Hybrid Co-ordination Function:

In order to support both IntServ and DiffServ QoS approaches in 802.11 a new mechanism called HCF. This mechanism is backwardly compatible with legacy DCF and PCF. It has both polling based and contention based channel access mechanisms in a single channel access protocol. HCF consists of two access methods, **EDCF** and **HCF** controlled channel access (HCCA) mechanisms. EDCF is already discussed in previous section. In HCCA there may are two phases of operations within the super frames, the CFP and CP, which alternate over time. The EDCF is used in the CP only, while the HCF controlled channel access method is use in both phases, which makes this new coordination function hybrid.



.

Fig.4.3. IEEE 802.11e HCF Beacon interval.

A station (STA) that implements both the QoS facility and the HCF function is denoted by QSTA (QoS station)in IEEE 802.11e. A Hybrid Coordinator (HC) is defined as a centralized controller in one QBSS (QoS Basic Service Set), which implements the frame exchange sequences and MSDU handling rules defined by the HCF. The HC, which is collocated within the QAP (QoS Access Point), operates during both the Contention Free Period (CFP) and the CP (Contention Period). The HC uses the point coordination's higher priority of access to the medium to initiate frame exchange sequences and to allocate Transmission Opportunities (TXOP) to QSTAs as to provide Controlled Access Periods (CAPs) to transfer QoS data. TXOP may be allocated at appropriate times during both the CFP and CP, in order to meet predefined delivery priority, service rate, delay and jitter requirements of particular traffic streams. The HC may initiate Controlled Contention Intervals (CCIs) during which contention occurs only between QSTAs which need to request new TXOPs . The HCF protects the transmissions during each CAP using the virtual carrier sense mechanism (i.e., NAV) which provides improved protection of the CFP.

There are two main architectural approaches to add QoS support in the Internet: ***integrated services (IntServ)*** and ***differentiated services (DiffServ)***.

IntServ provides fine-grained service guarantees to individual flows. It requires a module in every hop IP router along the path that reserves resources for each session. However, *IntServ* is not deployed since its requirement of setting states in all routers along a path is not scalable. (eg. HCCA)

Differentiated Services, or DiffServ, is an IP QoS architecture based on packet marking that allows packets to be prioritized according to user requirements. During the time of congestion, more low priority packets are discarded than high priority packets. (eg. EDCF)

4.3 Adaptive EDCF (AEDCF)

AEDCF is an adaptive service differentiation scheme for QoS enhancement in IEEE 802.11 wireless ad-hoc networks. It is derived from the concept of EDCF introduced in the IEEE 802.11e standard. This method aims to share the transmission channel efficiently. The priorities of different classes are provisioned by adjusting the size of the Contention Window (CW) of each traffic class taking into account both applications requirements and network conditions.

Let us assume that 'n' stations are sending packets through the wireless media. The flows sent by each station may belong to different classes of service with various priority levels. In each station and for each class 'i', the scheme maintains the following

1. current contention window value (CW[i])
2. the minimum contention window value (CWmin[i])
3. the maximum contention window value (CWmax[i]) .

OTHER DIFFERENTIATION SERVICE BASED ENHANCEMENT SCHEMES

First of all, QoS enhancement can be supported by adding service differentiation into the MAC layer. This can be achieved by modifying the parameters that define how an STA or a flow should access the wireless medium. Current service differentiation based schemes can be classified with respect to a various characteristics like whether the schemes is based on per-STA or per-queue (per-priority) parameters.

Classification of Service Differentiation schemes:-

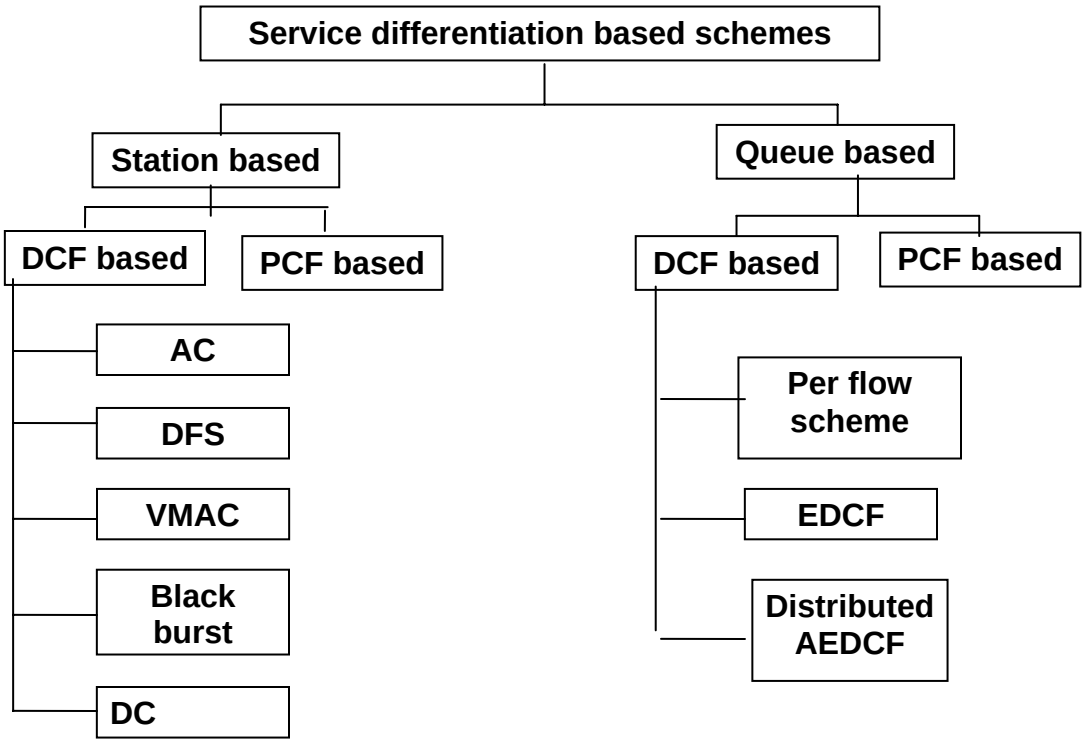


Fig.4.4 Service Differentiation Schemes

4.3.1 Station-based Service Differentiation with DCF Enhancement

a) Aad and Castelluccia (AC) Scheme

For the IEEE 802.11 standard DCF access method, it proposes three techniques to provide priorities:-

i. Different Back off increase function
ii. Different DIFS
iii. Different Maximum frame length

- ***Different Back off increase function***: Each priority level has a different backoff increment function. Assigning a short contention window to those higher priority STAs ensures that in most cases, high-priority STAs are more likely to access the channel than low-priority one.

Drawbacks:

- All TCP ACKs are set the same priorities which affect the differentiation mechanism.
- A high priority station can send a low priority data at the cost of real time traffic at another station.

Suggestions:

- All these drawbacks can be removed by using Queue-based enhancements.

- ***Different DIFS***: Each priority level has a different DIFS.

$DIFS_{j+1} < DIFS_j$,

where (priority of station j-1)>(priority of station j).

No STA of priority *j* has queued frames when STA of priority *j*-1 starts transmission.

Drawbacks:

- Low priority traffic suffers as long as high priority frames are queued.

- ***Different maximum frame lengths***: Each STA has a different maximum frame length according to its priority level. Therefore, a high priority STA can transmit more information per medium access than a low priority STA.

Drawbacks:

- In a noisy environment, long packets are more likely to be corrupted than short ones, which decrease the service differentiation efficiency.

b. Distributed Fair Scheduling (DFS)

The backoff interval is computed as a function of packet size and weight of the station. Thus STAs with low weights generate longer backoff intervals than those with high weights, thus getting lower priority.

Fairness is achieved by considering the packet size in the calculation of the backoff interval, causing flows with smaller packet size to be sent more often.

Drawbacks:

iv. Implementation complexity.

v. Smaller the packet size, higher the priority of the station since backoff is directly proportional to the packet size. Whereas, higher priority stations should be allowed to send bigger (or more no. of) packets.

c. Virtual MAC (VMAC)

A VMAC algorithm monitors the radio channel and MAC level statistics related to service quality such as delay, jitter, packet collision, and packet loss. The VMAC algorithm operates in parallel to the MAC in the mobile host but does not handle real packet transmission like in MAC. Moreover, a **virtual source (VS)** algorithm can estimate application-level service quality and allow application parameters to be tuned in response to dynamic channel conditions. This mechanism is relevant for real-time services since relative service differentiation is not enough for them. Moreover, this scheme uses the following backoff timer differentiation:-

CW_{min} (high priority) < CW_{min} (low priority)

And CW_{max} (high priority) < CW_{max} (low priority).

Drawbacks:

- The interactions between application and MAC layers introduce complexities.
- Since CW_{max} (high priority) < CW_{max} (low priority), high priority data gets lesser no. of attempts.

Suggestions:

- The statistics collected at regular intervals can be communicated among the STAs and the priorities of the stations can be assigned dynamically. The priority of the stations can be a function of the data transmitted by them in the past and other parameters like delay, jitter, etc.

- Based on the statistics collected, admission control algorithms can be run to improve system throughput.

d. Black burst

When a high priority STA wants to send a frame, it senses the medium to verify if it has been idle for PIFS time units and then sends its frame. If the medium is busy, the STA waits for the medium to be idle for a PIFS and then enters a black burst contention period: the STA sends a so-called **black burst** to jam the channel. The length of the black burst is determined by the time the STA has waited to access the medium. After this, the STA listens to the medium for a short period of time to see if some other STA is sending a longer black burst which would imply that the other STA has waited longer and thus should access the medium first. If the medium is idle, the STA will send its frame, otherwise it will wait until the medium becomes idle again and enters another black burst contention period. After the successful transmission of a frame, the STA schedules the next transmission attempt t_{sch} seconds in the future. In Black burst scheme, low priority STAs use the ordinary CSMA/CA access method of IEEE 802.11.

Drawbacks:

- It requires constant access intervals for high-priority traffic, otherwise the performance degrades considerably.
- Starvation of low priority stations.

Suggestions:

- The jam signals received by all STAs can be used to collect information/statistics about the whole system, and estimate when it would get an opportunity to transmit without re-transmission of jam signals by the contending STAs.
- Since the low priority STAs starve, we can introduce the concept of *ageing*, where the priority of STAs having low priority is increased gradually if they don't get channel access. This increase in priority is exhibited from the fact that the low priority STAs get the power of sending a jam signal like the high

priority STAs which too is proportional to the no. of time units it has waited to access the channel.

e. Deng and Chang (DC) Scheme

The DC scheme uses two parameters of IEEE 802.11 MAC, the backoff interval and IFS between each data transmission, to provide the differentiation.

Priority	IFS	Backoff algorithm
0	DIFS	$B=2^{2+i}/2+ \lfloor rd \times 2^{2+i}/2 \rfloor$
1	DIFS	$B= \lfloor rd \times 2^{2+i}/2 \rfloor$
2	PIFS	$B=2^{2+i}/2+ \lfloor rd \times 2^{2+i}/2 \rfloor$
3	PIFS	$B= \lfloor rd \times 2^{2+i}/2 \rfloor$

Table 2: DC scheme's priority classes

(*rd* is a uniform random variable in (0,1))

The higher priority STAs have shorter waiting time when accessing the medium. Moreover, when a collision occurs, higher priorities STAs have more chances to access the medium than the lower priority ones.

Drawbacks:

- When there are no high priority STAs which want to transmit packets, the low priority ones still generate a long backoff time.

f. Early Backoff Algorithm (EBA)

This enhancement is motivated by the fact that collisions occur because stations in a network do not know when other stations in the system will start their transmission. If stations knew the backoff durations of other stations, they could set their backoff duration so as not to conflict with each other, which in turn will prevent collisions. By adding their backoff value information into the MAC header of each transmitted data frame, stations

exchange their future backoff duration information. A key advantage of EBA is its backward compatibility with the legacy 802.11 DCF

Under DCF, after a successful transmission of the frame, the station resets its contention window size CW to CW_{min}, and determines the next backoff period *bnext randomly* at the end of the corresponding ACK frame reception. Whereas, under EBA, a station determines its next backoff period before transmitting the frame and piggybacks this future backoff information into the MAC frame header. This information is referred as **EBA information**.

All stations maintain a circular Reservation Window of size 1024 (CW_{max}) to keep a track of time slots when other stations will start transmission.

Chapter 5

NETWORK SIMULATOR

5.1 INTRODUCTION

NS (version 2) is an object-oriented, discrete event driven network simulator developed at UC Berkeley written in C++ and OTcl. NS is primarily useful for simulating local and wide area networks. It implements network protocols such as TCP and UPD, traffic source behavior such as FTP, Telnet and CBR, router queue management mechanism such as Drop Tail, RED and CBQ, routing algorithms such as AODV, DSR, and more. NS also implements multicasting and some of the MAC layer protocols for LAN simulations. The purpose of this section is to give some basic idea of how the simulator works and how to setup a simulation network.

5.2 GENERAL STRUCTURE AND ARCHITECTURE OF NS

NS is a discrete event simulator written in C++, with an OTcl interpreter as a front-end. The simulator supports a class hierarchy in C++ (we also call it the compiled hierarchy), and a similar class hierarchy within the OTcl interpreter (we also call it the interpreted hierarchy). The two hierarchies are closely related to each other. From the user's perspective, there is a one-to-one correspondence between a class in the interpreted hierarchy and one in the compiled hierarchy. The root of this hierarchy is the class TclObject. Users create new simulator objects through the interpreter; these objects are instantiated within the interpreter, and are closely mirrored by a corresponding object in the compiled hierarchy. The interpreted class hierarchy is automatically established through methods defined in the class TclClass. User instantiated objects are mirrored through methods defined in the class TclObject.

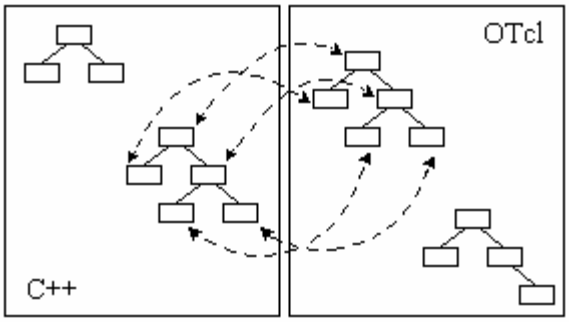


Figure 5.1: C++/OTcl Duality

NS uses two languages because the simulator has two different kinds of things it needs to do. On the one hand, detailed simulations of protocols require a systems programming language which can efficiently manipulate bytes, packet headers, and implement algorithms that run over large data sets. For these tasks, the run-time speed is important and the turn-around time is less important. On the other hand, a large part of network research involves slightly varying parameters or configurations, or quickly exploring a number of scenarios. In these cases, the iteration time is more important. Since configuration runs once (at the beginning of the simulation), the run-time of this part of the task is less important. C++ is fast to run but slower to change, making it suitable for detailed protocol implementation. OTcl runs much slower but can be changed very quickly (and interactively), making it ideal for simulation configuration.

When a simulation is finished, NS produces one or more text-based output files that contain detailed simulation data, if specified to do so in the input OTcl script. The data can be used for simulation analysis or as an input to a graphical simulation display tool called Network Animator (NAM). NAM has a nice graphical user interface that can graphically present information such as throughput and number of packet drops at each link, although the graphical information cannot be used for accurate simulation analysis.

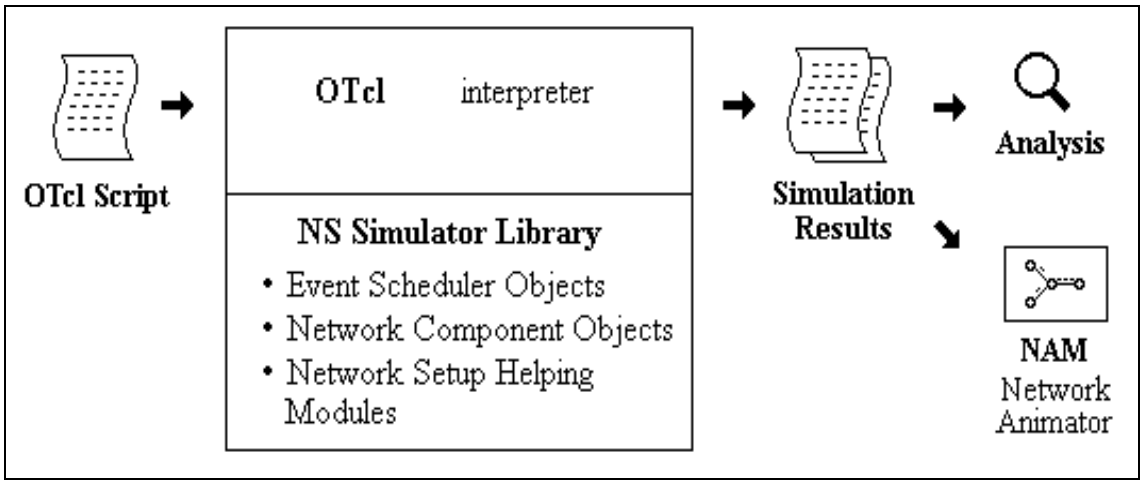


Figure 5.2: Simplified View of NS

We now briefly examine what information is stored in which directory or file in ns-2.

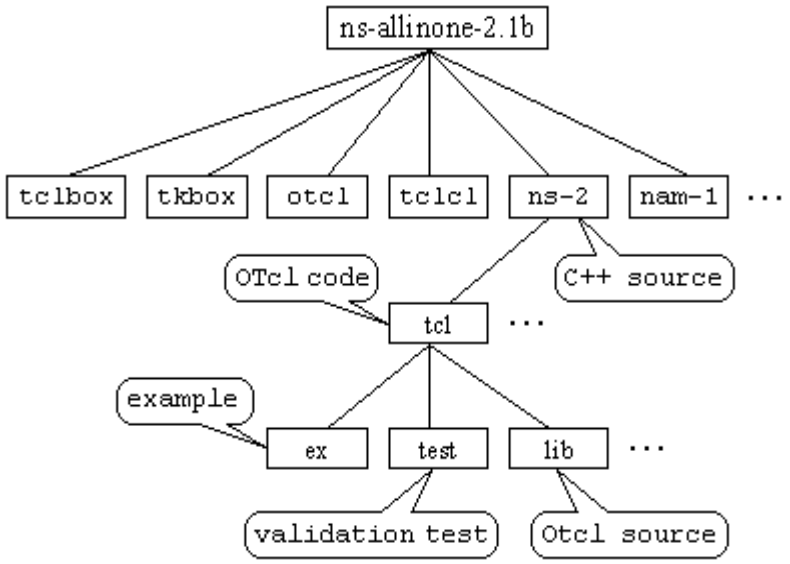


Figure 5.3: Directory Structure of NS

Among the sub-directories of ns-allinone-2.27, ns-2 is the place that has all of the simulator implementations (either in C++ or in OTcl), validation test OTcl scripts and example OTcl scripts. Within this directory, all OTcl codes located under a sub-directory called tcl, and most of C++ code, which implements event scheduler and basic network component object classes are located in the main level.

5.3 SAMPLE SIMULATION SCRIPT

We now present a simple simulation script and explain what each line means. Consider the following network topology of Figure 5.4.

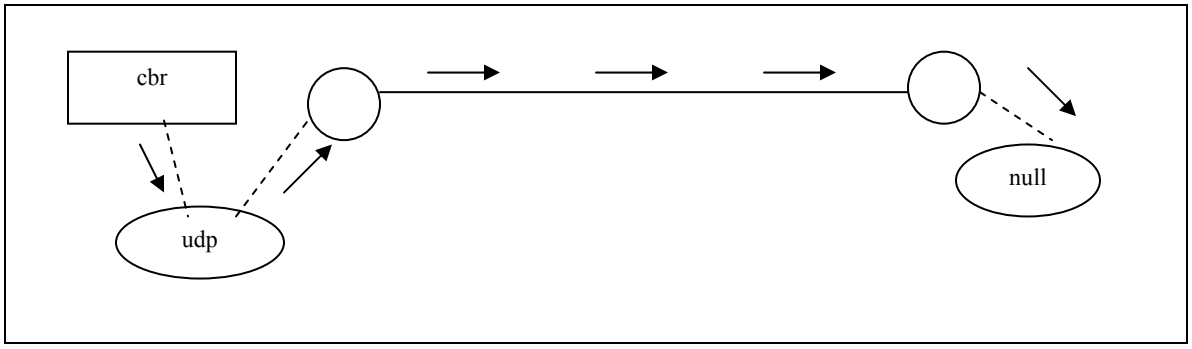


Figure 5.4: Scenario for a simple Simulation Script

We have two nodes; n0 and n1.The duplex link between n0 and n1 has 2 Mbps of bandwidth and 10 ms of delay. Each node uses a DropTail queue, of which the maximum size is 10. A "udp" agent that is attached to n0 is connected to a "null" agent attached to n1. A "null" agent just frees the packets received. A "cbr" traffic generator is attached to the "udp" agent, and the "cbr" is configured to generate a 500 byte packet every 1 second. The "cbr" is set to start at 0.5 second and stop at 4.5 second.

A simple NS simulation tcl script

```
# Create simulator object
set ns [new Simulator]

# Open the NAM trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#Define a finish procedure
proc finish {} {
global ns nf
$ns flush-trace
close $nf
exec nam out.nam &
exit 0
}

#Create 2 nodes
set n0 [$ns node]
set n1 [$ns node]

#Create link between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail

#Setup  a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set null0 [new Agent/Null]
$ns attach-agent $n1 $null0
$ns connect $udp0 $null0
#Setup a CBR over UDP connection
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 1.0
$cbr0 attach-agent $udp0

#Schedule events for CBR agent
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"

#Call finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"

#Run the simulation
$ns run
```

The following is the explanation of the script above.

- ***set ns [new Simulator]*** generates a new NS simulator object instance, and assigns it to a variable ns. This line creates an event scheduler for the simulation, initializes the packet format and selects the default address format.

- ***$ns namtrace-all nf*** tells the simulator to record simulation traces in NAM input format. *nf* is the file name that the trace will be written to later by the command *$ns flush-trace*.

- ***proc finish {}*** closes the trace file and starts nam.

- ***set n0 [$ns node]*** creates a node. A node in NS is compound object made of address and port classifiers (described in a later section).

- ***$ns duplex-link node1 node2 bandwidth delay queue-type*** creates two simplex links of specified bandwidth and delay, and connects the two specified nodes.

- ***set udp [new Agent/UDP]*** creates a udp agent. Users can create any agent or traffic sources in this way.

- ***$ns attach-agent node agent*** attaches an agent object created to a node object.

- ***$ns connect agent1 agent2*** connects the two agents specified. After two agents that will communicate with each other are created, the next thing is to establish a logical network connection between them. This line establishes a network connection by setting the destination address to each others' network and port address pair.

- ***$ns at 4.5 "$cbr start"*** tells the CBR agent when to start sending data.

- ***$ns at 0.5 "$cbr stop"*** tells the CBR agent when to stop sending data.

- ***$ns at 5.0 "finish"*** tells the simulator object to execute the 'finish' procedure after 5.0 seconds of simulation time

- ***$ns run*** starts the simulation.

5.4. SIMULATOR BASICS

5.4.1 Event Scheduler

NS is an event driven simulator. There are presently four schedulers available in the simulator, each of which is implemented using a different data structure: a simple linked-list, heap, calendar queue (default), and a special type called ``real-time''. The real-time scheduler is for emulation, which allows the simulator to interact with a real network. Currently, emulation is under development although an experimental version is available. Event schedulers are used to schedule events such as when to start a cbr agent, when to send /receive/drop a packet, etc. They are also used to simulate delay.

Event schedulers run by selecting the next earliest event, executing it to completion (by invoking appropriate network components and letting them do the appropriate action associated with the event), and returning to execute the next event. The simulator is single-threaded, and there is only one event in execution at any given time.

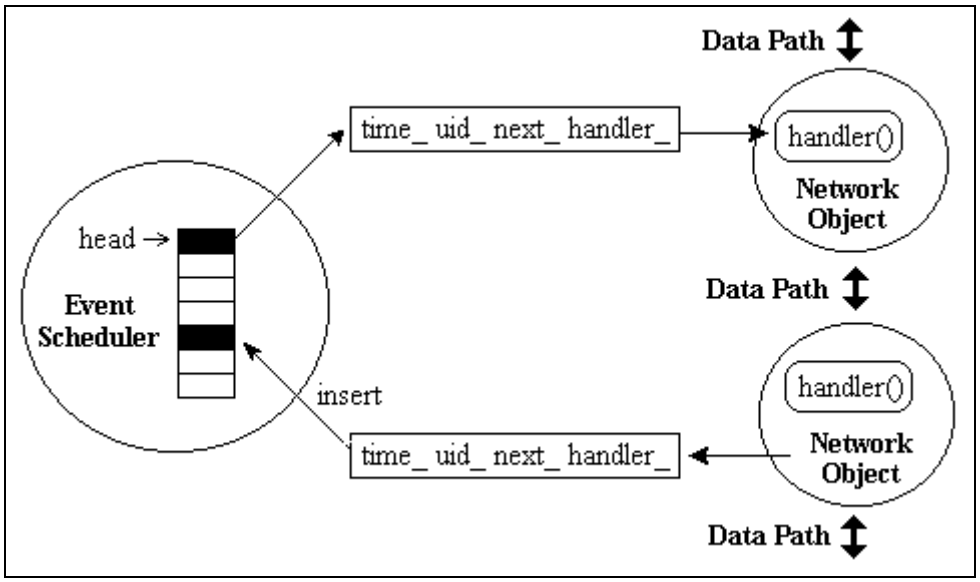


Figure 5.5: Event Scheduler

5.4.2 Basic Node

A node is a compound object. It is composed of a node entry object and classifiers. There are two types of nodes in NS. A unicast node has an address classifier that does unicast routing and a port classifier (agents are attached to ports). A multicast node, in addition, has a classifier that classify multicast packets from unicast packets and a multicast classifier that performs multicast routing.

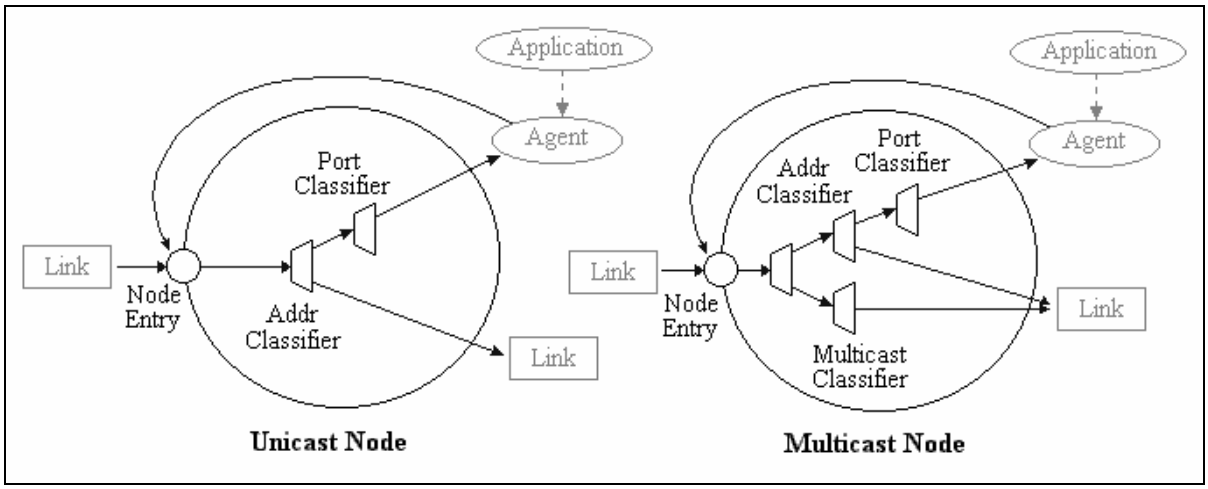


Figure 5.6: Basic Structure of Node in NS

5.4.3 Link

Link is a compound object in NS. It connects two nodes. We can create both simplex and duplex links in NS. A duplex link is nothing but two simplex links in both directions.

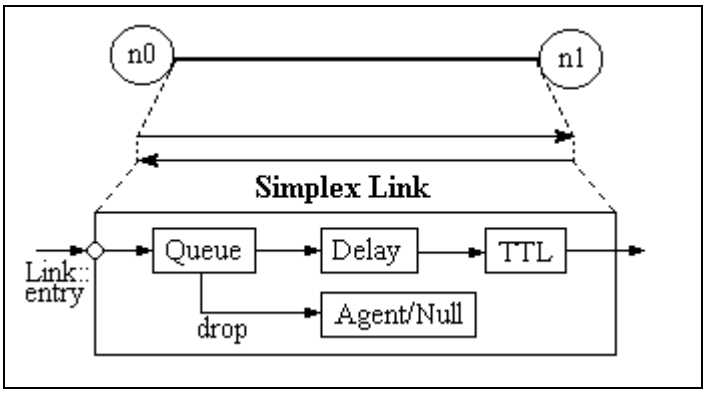


Figure 5.7: Simplex Link

When a node wants to send a packet through a link, it puts the packet on the Queue object of the link. Packets de-queued from the queue object are passed to the delay object. The delay object simulates link delay. Sending a packet to a null agent from the

queue object simulates the dropping of the packet. The TTL object calculates the time to live of each packet received and updates the TTL field of the packet.

5.4.4 Packet

Packets are the fundamental units of exchange between objects. A packet is composed of a stack of headers and an optional data space.

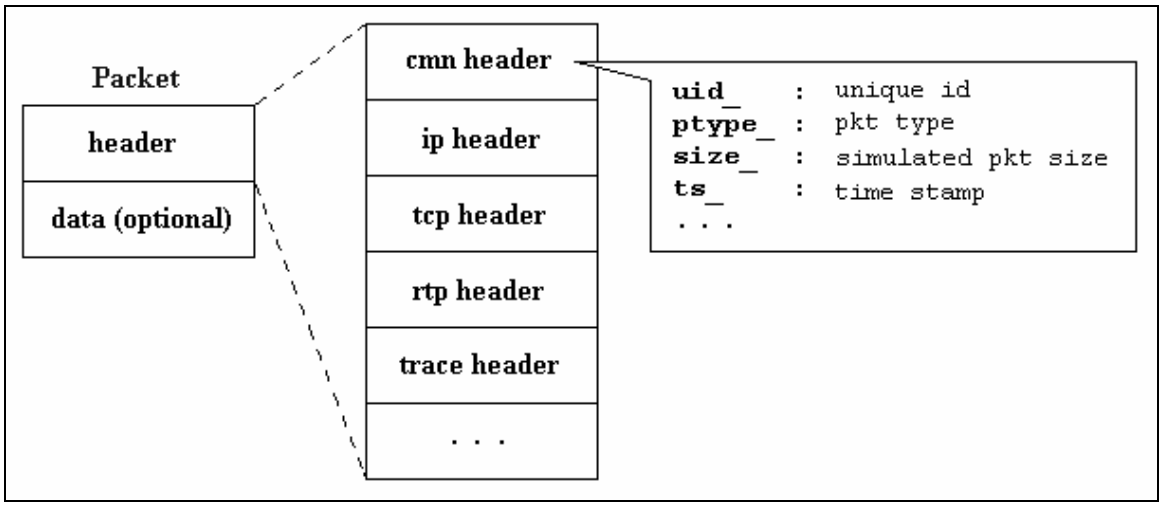


Figure 5.8: Packet Structure

Each header within the stack corresponds to a particular layer. In addition, there is a common header which can be accessed by all the layers, and a trace header which contains information for trace support. New protocols may define their own packet headers or may extend existing headers with additional fields.

5.4.5 Agent

Agents are used in the implementation of protocols at various layers. They represent endpoints where network-layer packets are constructed or consumed. The table below gives the list of the different agents currently supported by NS at the transport layer. NS also has routing agents implementing the different routing protocols like

DSDV, TORA, AODV and DSR. There routing protocols will be discussed in a later section.

TCP	a Tahoe TCP sender (cwnd = 1 on any loss)
TCP/Reno	a Reno TCP sender (with fast recovery)
TCP/Newreno	a modified Reno TCP sender (changes fast recovery)
TCP/Sack1	a SACK TCP sender
TCP/Fack	a forward SACK sender TCP
TCP/FullTcp	a more full-functioned TCP with 2-way traffic
TCP/Vegas	a Vegas TCP sender
TCP/Vegas/RBP	a Vegas TCP with .rate based pacing
TCP/Vegas/RBP	a Reno TCP with .rate based pacing
TCP/Asym	an experimental Tahoe TCP for asymmetric links
TCP/Reno/Asym	an experimental Reno TCP for asymmetric links
TCP/Newreno/Asym	an experimental Newreno TCP for asymmetric links
TCPSink	a Reno or Tahoe TCP receiver (not used for FullTcp)
TCPSink/DelAck	a TCP delayed-ACK receiver
TCPSink/Asym	an experimental TCP sink for asymmetric links
TCPSink/Sack1	a SACK TCP receiver
TCPSink/Sack1/DelAck	a delayed-ACK SACK TCP receiver
UDP	a basic UDP agent
RTP	an RTP sender and receiver

RTCP	an RTCP sender and receiver
LossMonitor	a packet sink which checks for losses
IVS/Source	an IVS source
IVS/Receiver	an IVS receiver
CtrMcast/Encap	a centralized multicast encapsulator
CtrMcast/Decap	a centralized multicast de-encapsulator
Message	a protocol to carry textual messages
Message/Prune	processes multicast routing prune messages
SRM	an SRM agent with non-adaptive timers
SRM/Adaptive	an SRM agent with adaptive timers
Tap	interfaces the simulator to a live network
Null	a degenerate agent which discards packets
rtProto/DV	distance-vector routing protocol agent

TABLE: 5.1 : List of Agents supported by NS

All agents are derived from the C++ Agent class. The following member functions are implemented by the C++ Agent class, and are generally *not* over-ridden by derived classes:

[]Packet* allocpkt	allocate new packet and assign its fields
[int]Packet* allocpkt	allocate new packet with a data payload of n bytes and assign its fields

The following member functions are also defined by the class Agent, but *are* intended to be over-ridden by classes deriving from Agent:

[timeout number]void timeout	subclass-specific time out method
[Packet*, Handler*]void recv	receiving agent main receive path

The allocpkt method is used by derived classes to create packets to send. The recv method is the main entry point for an Agent which receives packets, and is invoked by upstream nodes when sending a packet. The timeout method is used to periodically send request packets.

5.4.5.1 Adding a new Agent to NS simulator

NS allows users to extend the existing protocol stack by adding their own agents. We show how this can be done by adding a simple agent - MyAgent to NS. We create a new network object class in C++, "MyAgent", derived from the "Agent" class. To make it possible to create an instance of this object in OTcl, we define a linkage object, say "MyAgentClass", derived from "TclClass" which creates an OTcl object of specified name "Agent/MyAgentOtcl".

```
class MyAgent : public Agent {
public:
        MyAgent();
protected:
        int command(int argc, const char*const* argv);
private:
        int    my_var1;
        double my_var2;
        void   MyPrivFunc(void);
};
```

```
static class MyAgentClass : public TclClass {
public:
        MyAgentClass() : TclClass("Agent/MyAgentOtcl") {}
        TclObject* create(int, const char*const*) {
                return(new MyAgent());
        }
} class_my_agent;
```

Example C++ Network Component and Linkage object

When NS is first started, it creates an instance of “MyAgentClass”. In this process, the “Agent/MyAgentOtcl” class and its appropriate methods are created in OTcl space. Whenever a user in OTcl space tries to create an instance of this object using the command “new Agent/MyAgentOtcl”, it invokes “MyAgentClass::create” that creates an instance of “MyAgent” and returns the address.

We can access the member variables of the C++ object, “MyAgent” from the OTcl input script. To do this we use a binding function for each of the C++ class variables we want to export. A binding function takes two arguments. It creates a new member variable of the first argument in the matching OTcl object class ("Agent/MyAgentOtcl"), and creates bi-directonal bindings between the OTcl class variable and the C++ variable whose address is specified as the second variable. Suppose "MyAgent", has two parameter variables, say "my_var1" and "my_var2",that we want to access from OTcl using the input simulation script. The bindings for "my_var1" and "my_var2" are shown below. The binding functions are placed in the "MyAgent" constructor function to establish the bindings when an instance of this object is created.

```
MyAgent::MyAgent() : Agent(PT_UDP) {
        bind("my_var1_otcl", &my_var1);
        bind("my_var2_otcl", &my_var2);
}
```

Variable binding creation example

We then define a “command” member function of the C++ object ("MyAgent"). This function works as an OTcl command interpreter. An OTcl command defined in a "command" member function of a C++ object looks the same as a member function of the matching OTcl object to a user. When the user tries to call a member function of an OTcl object, OTcl searches for the given member function name in that OTcl object. If the given member function name cannot be found, then it invokes the “MyAgent::command” passing the invoked OTcl member function name and arguments

in argc/argv format. If there is an action defined for the invoked OTcl member function name in the "command" member function, it carries out what is asked and returns the result. If not, the "command" function for its ancestor object is recursively called until the name is found. If the name cannot be found in any of the ancestors, an error message is return to the OTcl object, and the OTcl object gives an error message to the user. In this way, a user in OTcl space can control a C++ object's behavior.

```
int MyAgent::command(int argc, const char*const* argv) {
      if(argc == 2) {
            if(strcmp(argv[1], "call-my-priv-func") == 0) {
                  MyPrivFunc();
                  return(TCL_OK);
            }
      }
      return(Agent::command(argc, argv));
}
```

Example OTcl command interpreter

5.5 MOBILE NETWORKING

The wireless model essentially consists of the MobileNode at the core, with additional supporting features. It is derived from the basic Node class. It is the basic Node object with added mobility features like node movement, periodic position updates, maintaining topology boundary etc.

5.5.1 Mobile Node

MobileNode is a split object. In addition to the basic node model, it consists of a network stack. The network stack for a mobile node consists of a link layer (LL), an ARP module connected to LL, an interface priority queue (IFq), a mac layer(MAC), a network interface (netIF), all connected to a common wireless channel. These network components are created and plumbed together in OTcl. A packet sent down the stack flows through the link layer (and ARP),the Interface queue, the MAC layer, and the

physical layer. At the receiving node, the packet then makes its way up the stack through the Mac, and the LL.

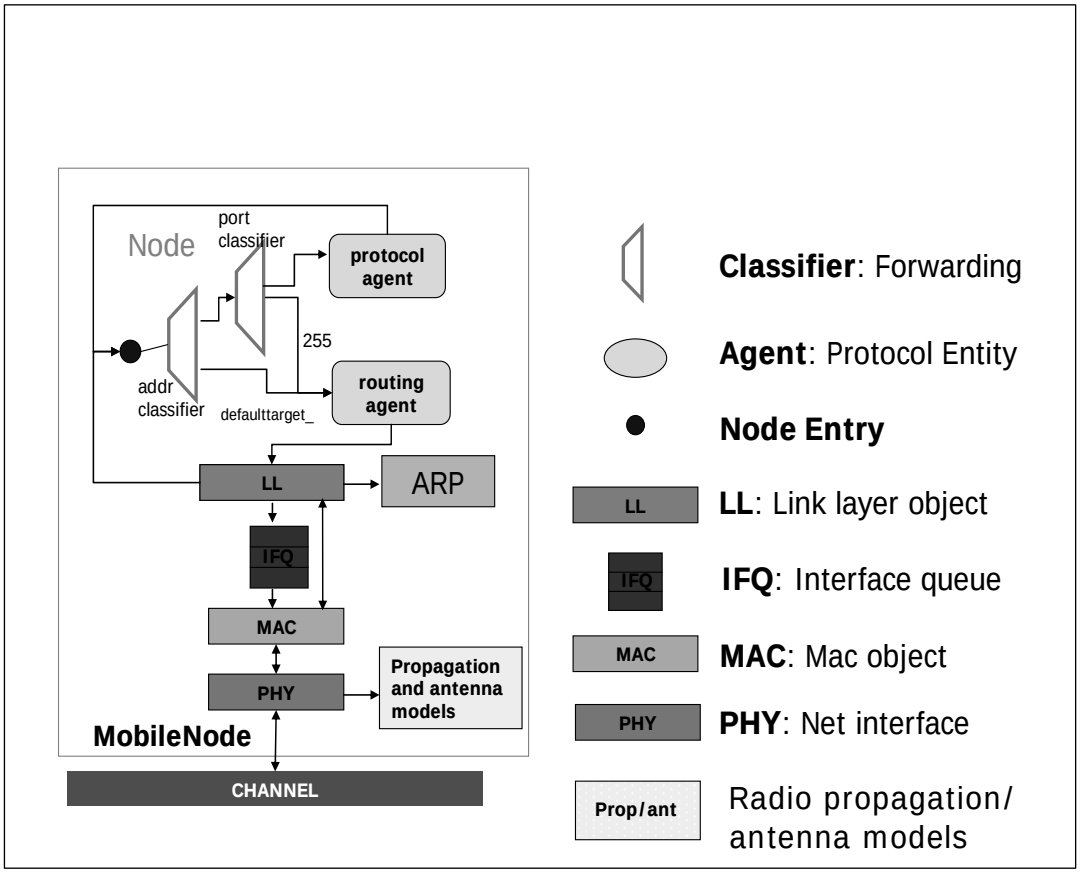


Figure 5.9: Mobile Node

Each component is briefly described here.

- **Link Layer**

The link layer can potentially have many functionalities such as queuing and link-level retransmission. The LL object implements a particular data link protocol, such as ARQ. By combining both the sending and receiving functionalities into one module, the LL object can also support other mechanisms such as piggybacking.

The link layer for mobile node has an ARP module connected to it which resolves all IP to hardware (Mac) address conversions. Normally for all outgoing (into the

channel) packets, the packets are handed down to the LL by the Routing Agent. The LL hands down packets to the interface queue. For all incoming packets (out of the channel), the mac layer hands up packets to the LL which is then handed off at the node_entry_ point.

- **ARP**

The Address Resolution Protocol (implemented in BSD style) module receives queries from Link layer. If ARP has the hardware address for destination, it writes it into the mac header of the packet. Otherwise it broadcasts an ARP query, and caches the packet temporarily. For each unknown destination hardware address, there is a buffer for a single packet. Incase additional packets to the same destination is sent to ARP, the earlier buffered packet is dropped. Once the hardware address of a packet's next hop is known, the packet is inserted into the interface queue.

- **Interface Queue**

The Interface queue is implemented as a priority queue, which gives priority to routing protocol packets, inserting them at the head of the queue. It supports running a filter over all packets in the queue and removes those with a specified destination address.

- **Mac Layer**

Depending on the type of physical layer, the MAC layer must contain a certain set of functionalities such as: carrier sense, collision detection, collision avoidance, etc. Since these functionalities affect both the sending and receiving sides, they are implemented in a single Mac object. For sending, the Mac object must follow a certain medium access protocol before transmitting the packet on the channel. For receiving, the MAC layer is responsible for delivering the packet to the link layer.

The IEEE 802.11 distributed coordination function (DCF) Mac protocol has been implemented by CMU. It uses a RTS/CTS/DATA/ACK pattern for all unicast

packets and simply sends out DATA for all broadcast packets. The implementation uses both physical and virtual carrier sense.

- **Network Interfaces (Physical layer)**

The Network Interface layer serves as a hardware interface which is used by mobile node to access the channel. This interface subject to collisions and the radio propagation model receives packets transmitted by other node interfaces to the channel. The interface stamps each transmitted packet with the meta-data related to the transmitting interface like the transmission power, wavelength etc. This meta-data in packet header is used by the propagation model in receiving network interface to determine if the packet has minimum power to be received and/or captured and/or detected (carrier sense) by the receiving node. The Network interface in NS approximates the DSSS radio interface (Lucent WaveLan direct-sequence spread-spectrum).

- **Radio Propagation Model**

These models are used to predict the received signal power of each packet. At the physical layer of each wireless node, there is a receiving threshold. When a packet is received, if its signal power is below the receiving threshold, it is marked as error and dropped by the MAC layer. NS supports the free space model (at near distances), two-ray ground reflection model (at far distances) and the shadowing model (includes fading)

- **Antenna**

An omni-directional antenna having unity gain is used by mobile nodes.

The following API configures for a mobile node with all the given values of adhoc-routing protocol, network stack, channel, topography, propagation model, with wired routing turned on or off (required for wired-cum-wireless scenarios) and tracing turned on or off at different levels (router, mac, agent).

$ns_ node-config -adhocRouting $opt(adhocRouting) *# dsdv/dsr/aodv/tora*

-llType $opt(ll) *# specifies link layer object*

-macType $opt(mac) *# specifies mac object*

-ifqType $opt(ifq) *# specifies ifq object*

-ifqLen $opt(ifqlen) *# specifies length of ifq*

-antType $opt(ant) *# specifies antenna object*

-propInstance [new $opt(prop)]*# propagation object*

-phyType $opt(netif) *# specifies physical layer object*

-channel [new $opt(chan)] *# specifies channel object*

-topoInstance $topo *# specifies topography*

-wiredRouting OFF *# for wired cum wireless simulations*

-agentTrace ON *# specifies agent level trace ON/OFF*

-routerTrace OFF *# specifies router level trace ON/OFF*

-macTrace OFF *# specifies mac level trace ON/OFF*

A mobile node is created using the following procedure:

```
for { set j 0 } { $j \< $opt(nn)} {incr j} {
   set node_($j) [ $ns_ node ]
   $node_($i) random-motion 0       ;#disable random motion
}
```

This procedure creates a mobile node (split)object, creates an adhoc-routing routing agent as specified, creates the network stack consisting of a link layer, interface queue, mac layer, and a network interface with an antenna, uses the defined propagation model, interconnects these components and connects the stack to the channel.

5.5.2 Creating Node Movements

The MobileNode is designed to move in a three dimensional topology. However, the third dimension (Z) is not used. That is the MobileNode is assumed to

move always on a flat terrain with Z always equal to 0. Thus the MobileNode has X, Y, Z(=0) co-ordinates that is continually adjusted as the node moves.

We first need to define the topography creating the mobile nodes. Normally flat topology is created by specifying the length and width of the topography using the following primitive:

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

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

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 mobilenode may be set by using the following APIs:

$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.

In this method the node-movement-updates are triggered whenever the position of the node at a given time is required to be known. This may be triggered by a query from a neighboring node seeking to know the distance between them, or the setdest directive described above that changes the direction and speed of the node.

The second method employs random movement of the node. The primitive to be used is:

$mobilenode start

This starts the mobile node with a random position and have routine updates to change the direction and speed of the node. The destination and speed values are generated in a random fashion.

5.6 TRACE SUPPORT

In order to obtain results from simulations, we need to know what exactly happens during a simulation run. NS realizes this by generation of event logs that can be analyzed offline, after a simulation. These event log files however, contain only events of packets being sent, received or dropped, so called Packet Traces. NS does not support the logging of more abstract events, i.e. related to connection establishment.

We now give an overview of the new wireless trace format generated by NS simulations.

Sample Wireless Trace file

s -t 0.267662078 -Hs 0 -Hd -1 -Ni 0 -Nx 5.00 -Ny 2.00 -Nz 0.00 -Ne
-1.000000 -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
-1.000000 -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
-1.000000 -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
-1.000000 -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 -1.000000 -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

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 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 outside 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 ad-hoc 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 these 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: Number of 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

We use these trace file to calculate measures such as throughput, end-to-end delay, packet drop ratio etc.

CHAPTER 6

POST SIMULATION ANALYSIS AND RESULTS

The trace files that are generated after the simulation are further processed to get the numeric statistics. These files are analyzed using perl scripts or awk scripts to calculate different QoS parameters like bandwidth, throughput, channel utilization, latency.

6.1 PERL (Practical Extraction and Report Language)

The Benefits of Using Perl

- Availability
- Interpreted Language
- Practical
- Complete

Perl combines some of the best features of 'C', 'SED', 'AWK', and 'Sh'.

grep/awk	General pattern-matching languages for selecting elements from a file.
C	A general-purpose compiled programming language. (Perl is written in C.)
sh	A control language generally used for running programs and scripts written in other languages.
sed	A stream editor for processing text streams (STDIN/STDOUT).

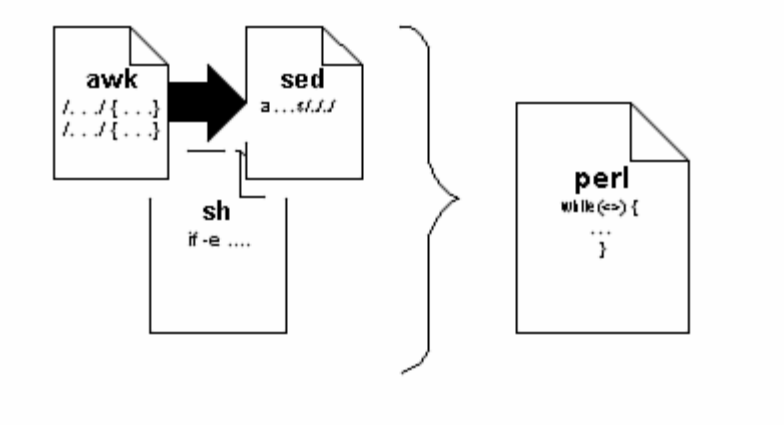


Fig: 6.1 Parts of a Perl Script

The trace files generated by ns2 normally contains huge amount of data with fixed trace format. These files are analyzed by shell, awk or perl scripts to extract the required data and statistics. The syntax of these languages used is similar to the 'C' language syntax and is easier to understand. The Values obtained are used for comparison of different protocols in different scenarios.

6.2 Simulation Scenario

The scenario script is given in the Appendix 1. In this scenario the I have used three flows, simultaneously to be sent over a network.

The flows used in this scenario are Background flows, Phone flows and Video flows.

The specifications used for these flows are

Packet size

Background Streams	:	200
Phone streams	:	160
Video Streams	:	1280

Time interval

Background Streams	:	12.5 ms
Phone streams	:	20 ms
Video Streams	:	10 ms

CW min and CW max

Background Streams	:	31 and 1023
Phone streams	:	15 and 600
Video Streams	:	0 and 200

DIFS

Background Streams	:	3
Phone streams	:	2
Video Streams	:	1

6.3 Results

The following simulational results are obtained from the simulation of above said flows using EDCF and AEDCF protocol separately.

CASE 1

The simulations are done with 20 stations, with alpha value (soothing value) as 2, and the updating period value the of collision rate parameter as 3.
The simulation is carried out for a period of 15 time slots.

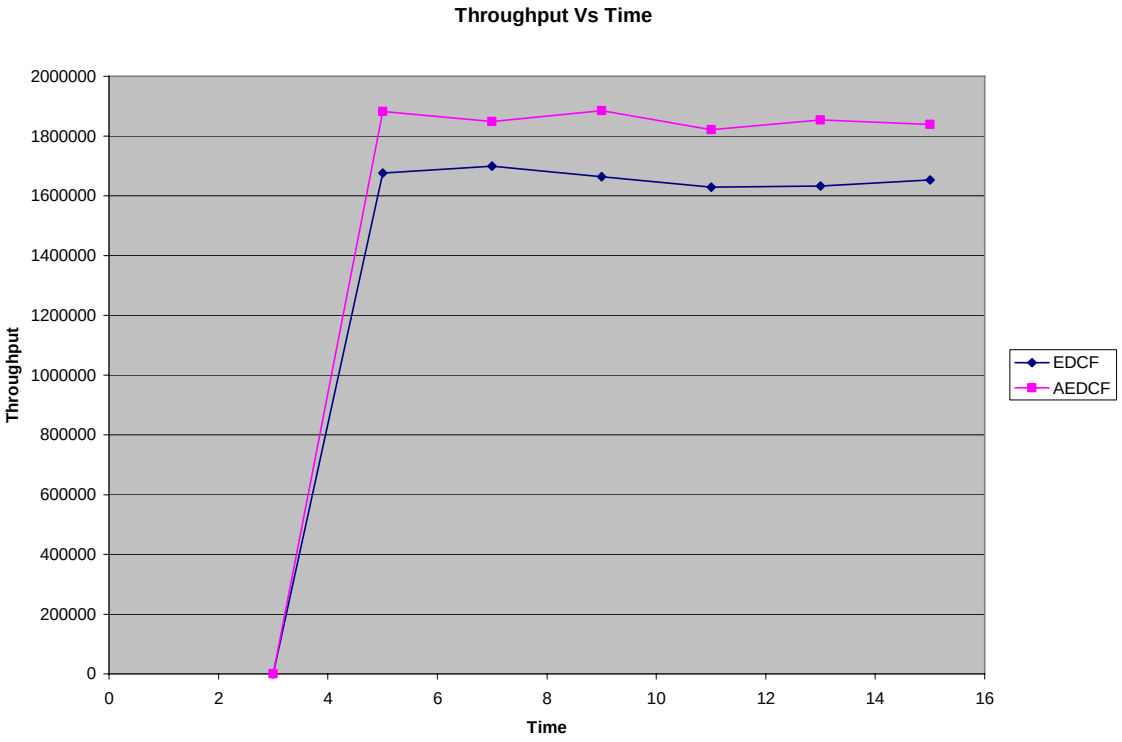


Fig 6.2: Graph of EDCF Vs AEDCF for throughput

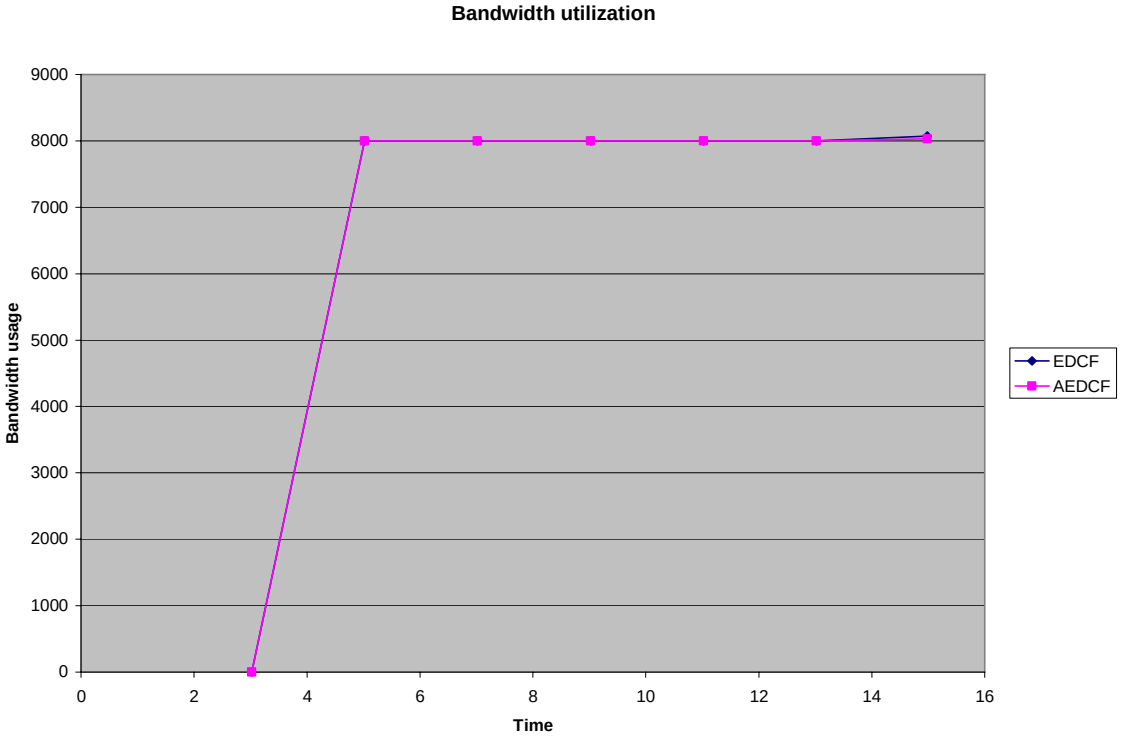


Fig 6.2: Graph of EDCF Vs AEDCF for bandwidth utilization

Numeric Results

The following are the different values obtained when these two protocols are compared. These values shows the performance of the AEDCF which are much better comparatively

PARAMETER	EDCF	AEDCF
Throughput	1619.771	1811.281
Mean Bandwidth Utilization	7.824	7.817
Channel Utilization (idle time/Collision time)	58.20296/859.24385	64.48211/346.20213
Delay (Max delay /Mean delay)	77.591/4.671	51.988/4.362

CASE 2

These simulations are done with 30 stations. The simulations are done with alpha value as 1 and update period of 2 and made to run using both the protocols.

The results are shown as under.

Here the simulation is carried out for a period of 30 time slots.

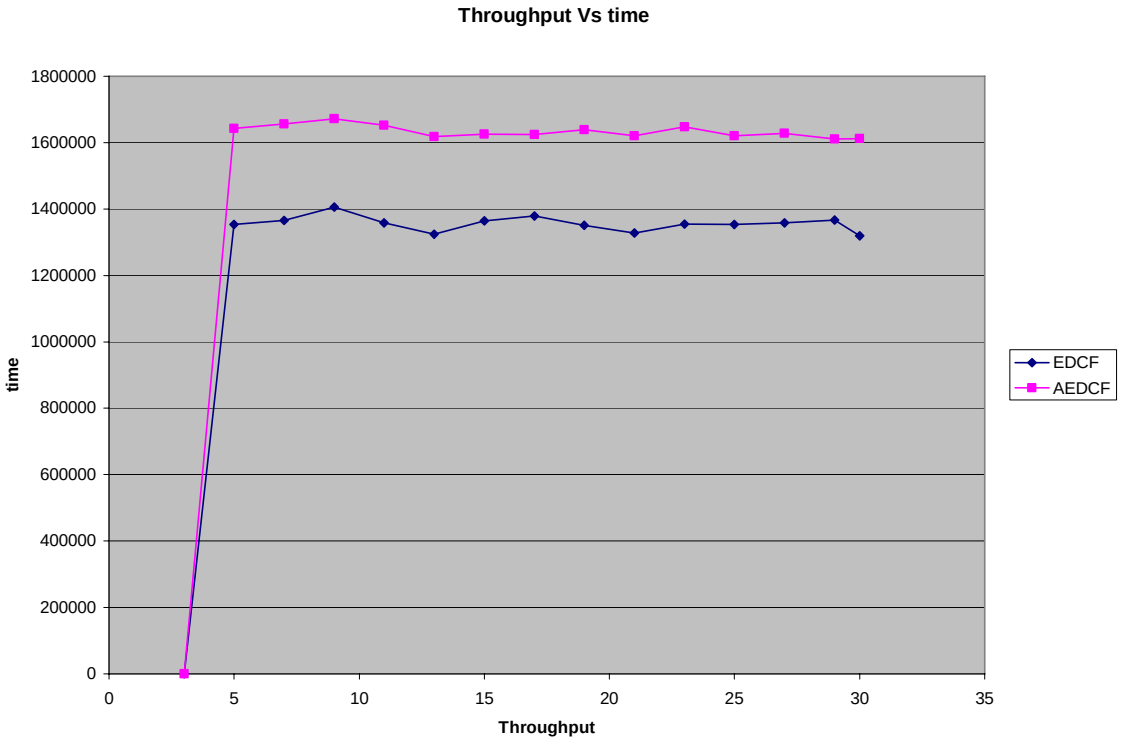


Fig 6.4: Graph of EDCF Vs AEDCF for throughput

Bandwidth utilization

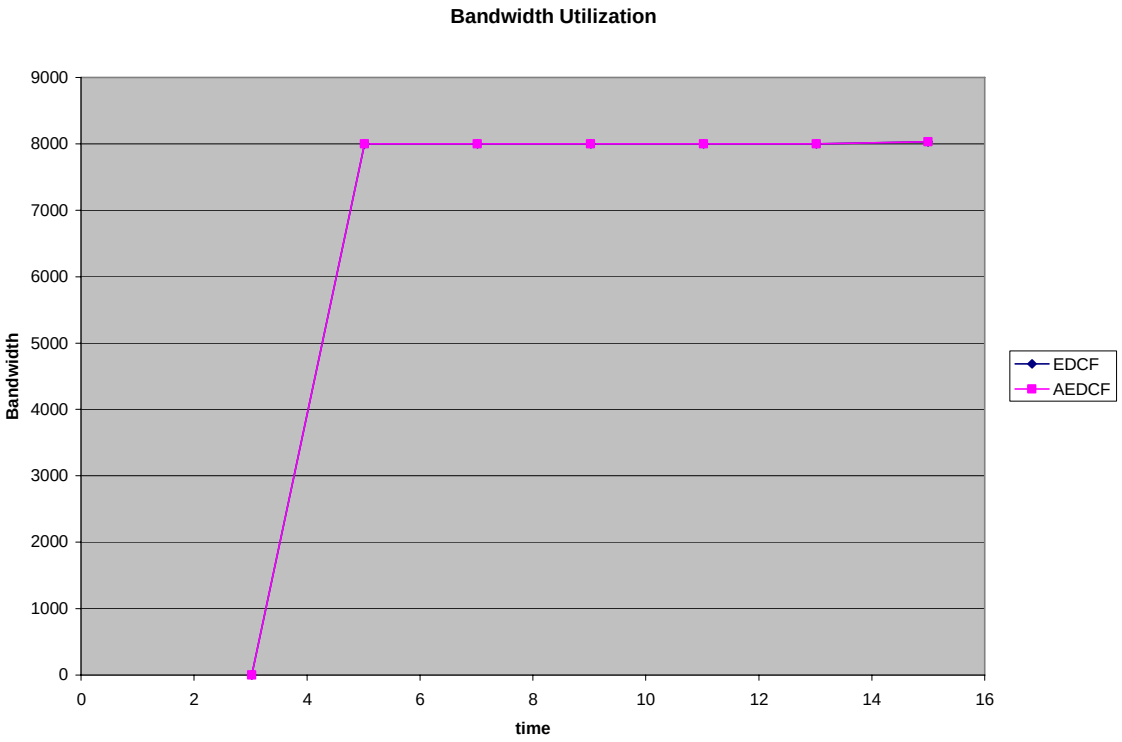


Fig 6.5: Graph of EDCF Vs AEDCF for BW utilization

Numeric Results:

The values of different parameters for this scenario are given in the table below.

PARAMETER	EDCF	AEDCF
Throughput	1325.334	1596.084
Mean Bandwidth Utilization	7.802	7.814
Channel Utilization (idle time/Collision time)	53.74866/1118.12828	61.87065/479.83562
Delay (Max delay /Mean delay)	137.213/6.293	52.861/6.022

The above values also shows that at heavy traffic or loads AEDCF outperforms EDCF.

CONCLUSION AND FUTURE WORK

The applications like video conferencing, voice applications in wireless networks, are becoming more common these days. Therefore there arises a need for finite delay and dedicated bandwidth. The user needs more quality of service for these applications. The current schemes provide some quality of service but better utilization can be done and better QoS can be provided. The Adaptive technique of the EDCF protocol makes the multimedia applications flow more efficient than before, and reduces the latency. The dynamic assignment of CW size makes it more suitable for reducing collisions and thus increasing throughput.

These two protocols are simulated using network simulator under different scenarios and the performance characteristics are figured out from the trace files using perl scripts. The different parameters considered for performance evaluation are throughput, Band width, channel Utilization, and Latency. On analyzing the results of both these protocols it is found that under heavy loads AEDCF outperforms the EDCF in many cases.

Future Work

In this work only adaptable parameter present is the contention window size. But this can be extended to different other parameters so that they can be assigned dynamically depending on network conditions. The other parameters that can be made adaptive are CWmax, size of the packet, retransmission counter etc…

Bibliography

Research Papers:

1. **Analysis of IEEE 802.11e For Qos Support in Wireless Lans**
   Stephan Mangold, Philips Research, Sunghyun Choi, Seol National University.
   IEEE Wireless Communications • December 2003
2. **QoS Issues and Enhancements for IEEE 802.11 Wireless LAN**
   Qiang Ni, Lamia Ramdhoani, Thierry Turletti and Imad Aad – Nov 2002
3. **A Survey of QoS Enhancements for IEEE 802.11 Wireless LAN**
   Qiang Ni*, Lamia Romdhani, Thierry Turletti Planete –. Journal of Wireless Communications and Mobile Computing, Wiley. 2004
4. **Using IEEE 802.11e MAC for QoS over Wireless**
   Priyank Garg, Rushabh Doshi, Russell Greene, Mary Baker, Majid Malek, Xiaoyan Cheng- 2003 IEEE
5. **Protection and Guarantee for Voice and Video Traffic in IEEE 802.11e Wireless LANs**
   Yang Xiao, Haizhon Li, Sunghyun Choi.
6. **The *ns* Manual Notes and Documentation**
   Kevin Fall and Kannan Varadhan
7. **A Survey of QoS Techniques and Enhancements for IEEE 802.11 Wireless LANs**
   Lassaâd Gannoune, Stephan Robert and Daniel Rodellar
8. **A technical tutorial of IEEE 802.11**
   Pablo Brenner-2001
9. **Differentiation mechanisms for IEEE 802.11**
   Imad Aad and Claude Castelluccia

Website References

1. **www.isi.edu\nsnam\ns\index.html** (ns2 official website)

2. **http://www.tkn.tu-berlin.de/research/802.11e_ns2/**

3. **www.ieee.org/**

4. **http://mosquitonet.stanford.edu**

5. **www.isi.edu\nsnam\ns\tutorial\index.html** (Marc Greis's Tutorial)

6. **www.nile.wpi.edu/NS/index.html** (ns by example tutorial)